

CLOSER 2015

5th INTERNATIONAL CONFERENCE ON CLOUD
COMPUTING AND SERVICES SCIENCE

PROCEEDINGS

Lisbon, Portugal

20 - 22 May, 2015

► www.closer.scitevents.org

SPONSORED BY:

The logo for INSTICC, featuring the word "INSTICC" in a bold, sans-serif font with a stylized yellow and orange graphic element to the left.

LOGISTICS PARTNER:

The logo for SCITEVENTS, featuring the word "SCITEVENTS" in a bold, sans-serif font with a red dotted line graphic element to the left.

PAPERS AVAILABLE AT:

The logo for SCITEPRESS, featuring the word "SCITEPRESS" in a bold, sans-serif font with a green graphic element to the left.

CLOSER 2015

Proceedings of the
5th International Conference on
Cloud Computing and Services Science

Lisbon, Portugal

20 - 22 May, 2015

Sponsored by
INSTICC – Institute for Systems and Technologies of Information, Control and Communication

In Cooperation with
ACM SIGMIS – ACM Special Interest Group on Management Information Systems

Copyright © 2015 SCITEPRESS – Science and Technology Publications
All rights reserved

Edited by Markus Helfert, Donald Ferguson and Víctor Méndez Muñoz

Printed in Portugal
ISBN: 978-989-758-104-5
Depósito Legal: 391663/15

<http://closer.scitevents.org>
closer.secretariat@insticc.org

BRIEF CONTENTS

INVITED SPEAKERS IV

ORGANIZING AND STEERING COMMITTEES V

PROGRAM COMMITTEE VI

AUXILIARY REVIEWERS X

SELECTED PAPERS BOOK X

FOREWORD XI

CONTENTS XIII

INVITED SPEAKERS

Victor Chang

Leeds Beckett University

U.K.

Paolo Traverso

Center for Information Technology - IRST (FBK-ICT)

Italy

Omer Rana

Cardiff University

U.K.

Chung-Sheng Li

IBM

U.S.A.

Cornel Klein

Siemens AG

Germany

ORGANIZING AND STEERING COMMITTEES

CONFERENCE CHAIR

Markus Helfert, Dublin City University, Ireland

PROGRAM CO-CHAIRS

Víctor Méndez Muñoz, Universitat Autònoma de Barcelona, UAB, Spain

Donald Ferguson, Dell, U.S.A.

PROCEEDINGS PRODUCTION

Bruno Encarnação, INSTICC, Portugal

Lúcia Gomes, INSTICC, Portugal

Ana Guerreiro, INSTICC, Portugal

Raquel Pedrosa, INSTICC, Portugal

Vitor Pedrosa, INSTICC, Portugal

Sara Santiago, INSTICC, Portugal

José Varela, INSTICC, Portugal

CD-ROM PRODUCTION

Pedro Varela, INSTICC, Portugal

GRAPHICS PRODUCTION AND WEBDESIGNER

André Lista, INSTICC, Portugal

Mara Silva, INSTICC, Portugal

SECRETARIAT

Carla Mota, INSTICC, Portugal

WEBMASTER

Susana Ribeiro, INSTICC, Portugal

PROGRAM COMMITTEE

Antonia Albani, University of St. Gallen, Switzerland

Vasilios Andrikopoulos, University of Stuttgart, Germany

Claudio Ardagna, Università degli Studi di Milano, Italy

Alvaro Arenas, Instituto de Empresa Business School, Spain

José Enrique Armendáriz-Iñigo, Universidad Pública de Navarra, Spain

Muhammad Atif, Australian National University, Australia

Amelia Badica, Faculty of Economics and Business Administration, University of Craiova, Romania

Costin Badica, University of Craiova, Romania

Costas Bekas, IBM Zurich Research Lab, Switzerland

Adam S. Z. Belloum, University of Amsterdam, The Netherlands

Simona Bernardi, Centro Universitario de la Defensa - Academia General Militar, Spain

Karin Bernsmed, SINTEF ICT, Norway

Nik Bessis, University of Derby, U.K.

Luiz F. Bittencourt, IC/UNICAMP, Brazil

Stefano Bocconi, TU Delft, The Netherlands

Anne Boyer, Loria - Inria Lorraine, France

Ivona Brandić, Vienna UT, Austria

Iris Braun, Dresden Technical University, Germany

Andrey Brito, Universidade Federal de Campina Grande, Brazil

John Brooke, University of Manchester, U.K.

Anna Brunstrom, Karlstad University, Sweden

Rebecca Bulander, Pforzheim University of Applied Science, Germany

Tomas Bures, Charles University in Prague, Czech Republic

Massimo Cafaro, University of Salento, Italy

Manuel Isidoro Capel-Tuñón, University of Granada, Spain

Miriam Capretz, University of Western Ontario, Canada

Noel Carroll, National University of Ireland, Galway, Ireland, Ireland

John Cartlidge, University of Bristol, U.K.

Valentina Casola, University of Naples Federico II, Italy

Rong N. Chang, IBM T. J. Watson Research Center, U.S.A.

Augusto Ciuffoletti, Università di Pisa, Italy

Daniela Barreiro Claro, Universidade Federal da Bahia (UFBA), Brazil

José Cordeiro, E.S.T. Setúbal, I.P.S., Portugal

Thierry Coupaye, Orange, France

António Miguel Rosado da Cruz, Instituto Politécnico de Viana do Castelo, Portugal

Eduardo Huedo Cuesta, Universidad Complutense de Madrid, Spain

Eliezer Dekel, IBM Research Haifa, Israel

Yuri Demchenko, University of Amsterdam, The Netherlands

Frédéric Desprez, LIP / INRIA, France

Ake Edlund, Royal Institute of Technology, Sweden

Erik Elmroth, Umeå University and Elasticsys AB, Sweden

Robert van Engelen, Florida State University, U.S.A.

Antonio Espinosa, Universitat Autònoma de Barcelona, Spain

Donald Ferguson, Dell, U.S.A.

Tomás Fernández, Univ. Santiago de Compostela, Spain

Stefano Ferretti, University of Bologna, Italy

PROGRAM COMMITTEE (CONT.)

Mike Fisher, BT, U.K.

Geoffrey Charles Fox, Indiana University, U.S.A.

Chiara Francalanci, Politecnico di Milano, Italy

David Genest, Université d'Angers, France

Chirine Ghedira, IAE - University Jean Moulin
Lyon 3, France

Lee Gillam, University of Surrey, U.K.

Katja Gilly, Miguel Hernandez University, Spain

Tristan Glatard, CNRS, Canada

Andrzej Goscinski, Deakin University, Australia

Patrizia Grifoni, CNR, Italy

Stephan Groß, T-Systems Multimedia Solutions,
Germany

Dirk Habich, Dresden University of Technology,
Germany

Benjamin Heckmann, University of Applied
Sciences Darmstadt, Germany

Dong Huang, Chinese Academy of Sciences, China

Marianne Huchard, Lirimm Université Montpellier
2, France

Mohamed Hussien, Suez Canal University, Egypt

Sorin M. Iacob, Thales Nederland B.V., The
Netherlands

Ilian Ilkov, IBM Nederland B.V., The Netherlands

Anca Daniela Ionita, University Politehnica of
Bucharest, Romania

Fuyuki Ishikawa, National Institute of Informatics,
Japan

Hiroshi Ishikawa, Tokyo Metropolitan University,
Japan

Ivan Ivanov, SUNY Empire State College, U.S.A.

Martin Gilje Jaatun, SINTEF ICT, Norway

Keith Jeffery, Independent Consultant (previously
Science and Technology Facilities Council), U.K.

Meiko Jensen, Independent Centre for Privacy
Protection Schleswig-Holstein, Germany

Yiming Ji, University of South Carolina Beaufort,
U.S.A.

Ming Jiang, University of Leeds, U.K.

Xiaolong Jin, Chinese Academy of Sciences, China

Carlos Juiz, Universitat de les Illes Balears, Spain

David R. Kaeli, Northeastern University, U.S.A.

Yücel Karabulut, VMware, U.S.A.

Gabor Kecskemeti, MTA SZTAKI, Hungary

Attila Kertesz, MTA SZTAKI, Hungary

Claus-Peter Klas, GESIS Leibniz Institute for the
Social Sciences, Germany

Carsten Kleiner, University of Applied Sciences &
Arts Hannover, Germany

Geir M. Kjøien, University of Agder, Norway

Dimitri Konstantas, University of Geneva,
Switzerland

George Kousiouris, National Technical University
of Athens, Greece

László Kovács, MTA SZTAKI, Hungary

Marcel Kunze, Karlsruhe Institute of Technology,
Germany

Young Choon Lee, University of Sydney, Australia

Miguel Leitão, ISEP, Portugal

Wilfried Lemahieu, KU Leuven, Belgium

Fei Li, Siemens AG, Austria, Austria

Donghui Lin, Kyoto University, Japan

Shijun Liu, School of computer science and
technology, Shandong University, China

Xumin Liu, Rochester Institute of Technology,
U.S.A.

Francesco Longo, Università degli Studi di
Messina, Italy

Pedro Garcia Lopez, University Rovira i Virgili,
Spain

Suksant Sae Lor, HP Labs, U.K.

Joseph P. Loyall, BBN Technologies, U.S.A.

PROGRAM COMMITTEE (CONT.)

Simone Ludwig, North Dakota State University, U.S.A.

Glenn Luecke, Iowa State University, U.S.A.

Hanan Lutfiyya, University of Western Ontario, Canada

Theo Lynn, Dublin City University, Ireland

Shikharesh Majumdar, Carleton University, Canada

Elisa Marengo, Free University of Bozen-Bolzano, Italy

Ioannis Mavridis, University of Macedonia, Greece

Jose Ramon Gonzalez de Mendivil, Universidad Publica de Navarra, Spain

Mohamed Mohamed, IBM Research, Almaden, U.S.A.

Owen Molloy, National University of Ireland, Galway, Ireland

Marco Casassa Mont, Hewlett-Packard Laboratories, U.K.

Kamran Munir, University of the West of England (Bristol, UK), U.K.

Víctor Méndez Muñoz, Universitat Autònoma de Barcelona, UAB, Spain

Hidemoto Nakada, National Institute of Advanced Industrial Science and Technology (AIST), Japan

Philippe Navaux, UFRGS - Federal University of Rio Grande Do Sul, Brazil

Jean-Marc Nicod, Institut FEMTO-ST, France

Bogdan Nicolae, IBM Research, Ireland

Karsten Oberle, Alcatel-Lucent Bell Labs, Germany

Sebastian Obermeier, ABB Corporate Research, Switzerland

David Padua, University of Illinois at Urbana-Champaign, U.S.A.

Federica Paganelli, CNIT - National Interuniversity Consortium for Telecommunications, Italy

Claus Pahl, Dublin City University, Ireland

Michael A. Palis, Rutgers University, U.S.A.

Nikos Parlavantzas, IRISA, France

David Paul, The University of Newcastle, Australia

Siani Pearson, HP Labs, Bristol, U.K.

Mikhail Pereplechikov, RMIT University, Australia

Juan Fernando Perez, Imperial College London, U.K.

Giovanna Petrone, University of Torino, Italy

Agostino Poggi, University of Parma, Italy

Antonio Puliafito, Università degli Studi di Messina, Italy

Francesco Quaglia, Sapienza Università di Roma, Italy

Cauligi (Raghu) Raghavendra, University of Southern California, Los Angeles, U.S.A.

Rajendra Raj, Rochester Institute of Technology, U.S.A.

Arcot Rajasekar, University of North Carolina at Chapel Hill, U.S.A.

Arkalgud Ramaprasad, University of Illinois at Chicago, U.S.A.

Manuel Ramos-Cabrer, University of Vigo, Spain

Nadia Ranaldo, University of Sannio, Italy

Andrew Rau-Chaplin, Dalhousie University, Canada

Christoph Reich, Hochschule Furtwangen University, Germany

Norbert Ritter, University of Hamburg, Germany

Luis Roderio-Merino, Spain

Jerome Rolia, Hewlett Packard Labs, Canada

Pedro Frosi Rosa, UFU - Federal University of Uberlandia, Brazil

Elena Sanchez-Nielsen, Universidad De La Laguna, Spain

Patrizia Scandurra, University of Bergamo, Italy

Erich Schikuta, Universität Wien, Austria

PROGRAM COMMITTEE (CONT.)

Lutz Schubert, Ulm University, Germany

Uwe Schwiegelshohn, TU Dortmund University, Germany

Wael Sellami, Faculty of Sciences Economics and management, Sfax, Tunisia

Giovanni Semeraro, University of Bari Aldo Moro, Italy

Carlos Serrao, ISCTE-IUL, Portugal

Armin Shams, Sharif University of Technology, Iran, Islamic Republic of

Keiichi Shima, IJ Innovation Institute, Japan

Marten van Sinderen, University of Twente, The Netherlands

Richard O. Sinnott, University of Melbourne, Australia

Frank Siqueira, Universidade Federal de Santa Catarina, Brazil

Cosmin Stoica Spahiu, University of Craiova - Faculty of Automation, Computers and Electronics, Romania

Josef Spillner, Technische Universität Dresden, Germany

Ralf Steinmetz, Technische Universität Darmstadt, Germany

Burkhard Stiller, University of Zürich, Switzerland

Yasuyuki Tahara, The University of Electro-Communications, Japan

Cedric Tedeschi, IRISA - University of Rennes 1, France

Gilbert Tekli, Nobatek, France, Lebanon

Joe Tekli, Antonine University (UPA), Lebanon

Patricia J. Teller, University of Texas at El Paso, U.S.A.

Guy Tel-Zur, Ben-Gurion University of the Negev (BGU), Israel

Orazio Tomarchio, University of Catania, Italy

Johan Tordsson, Umea University and Elasticsys, Sweden

Francesco Tusa, University College London, U.K.

Astrid Undheim, Telenor ASA, Norway

Geoffroy Vallee, Oak Ridge National Laboratory, U.S.A.

Luis M. Vaquero, HP Labs, U.K.

Sabrina de Capitani di Vimercati, Università degli Studi di Milano, Italy

Bruno Volckaert, Ghent University, Belgium

Hiroshi Wada, NICTA, Australia

Maria Emilia M. T. Walter, University of Brasilia, Brazil

Chen Wang, CSIRO ICT Centre, Australia

Dadong Wang, CSIRO, Australia

Martijn Warnier, Delft University of Technology, The Netherlands

Hany F. El Yamany, Suez Canal University, Egypt

Bo Yang, University of Electronic Science and Technology of China, China

Zhifeng Yun, University of Houston, U.S.A.

Michael Zapf, Georg Simon Ohm University of Applied Sciences, Germany

Wolfgang Ziegler, Fraunhofer Institute SCAI, Germany

AUXILIARY REVIEWERS

Márcio Assis, Unicamp, Brazil

Maria Estrela Ferreira da Cruz, Instituto Politécnico de Viana do Castelo, Portugal

Fernando Gómez-Folgar, Centro Singular de Investigación en Tecnoloxías da Información, Spain

Mehdi Khouja, University of Gabes, Tunisia

Amardeep Mehta, Umeå University, Sweden

Athina Provataki, University of Macedonia, Greece

Eduardo Roloff, UFRGS, Brazil

SELECTED PAPERS BOOK

A number of selected papers presented at CLOSER 2015 will be published by Springer-Verlag in a CCIS Series book. This selection will be done by the Conference Chair and Program Co-chairs, among the papers actually presented at the conference, based on a rigorous review by the CLOSER 2015 Program Committee members.

FOREWORD

This book contains the proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER 2015) which was organized and sponsored by the Institute for Systems and Technologies of Information, Control and Communication (INSTICC). This conference was held in Cooperation with the Association for Computing Machinery - Special Interest Group on Management Information Systems.

The technical program of CLOSER 2015 covered areas like “Cloud Computing Fundamentals”, “Services Science Foundation for Cloud Computing”, “Cloud Computing Platforms and Applications”, “Cloud Computing Enabling Technology” and “Mobile Cloud Computing and Services”. We expect that these proceedings will appeal to a broad audience of engineers, scientists and business people interested in cloud computing and service systems. We further believe that the papers in these proceedings demonstrate new and innovative solutions, and highlight technical problems in the mentioned areas that are challenging and worthwhile.

The conference was also complemented with the second edition of the Emerging Software as a Service and Analytics Workshop – EsaaSA 2015 (co-chaired by Victor Chang, Muthu Ramachandran, Gary Wills, Robert Walters, Verena Kantere and Chung-Sheng Li). CLOSER 2015 received 146 paper submissions from all continents. From these, 23 papers were published and presented as full papers, 64 were accepted as short papers. These numbers, leading to a full-paper acceptance ratio of 15,8 % and an oral paper acceptance ratio of 60%, show the intention of preserving a high quality forum for the next editions of this conference.

The high quality of the CLOSER 2015 programme was enhanced by five keynote lectures, delivered by experts in their fields, including: Victor Chang (Leeds Beckett University, United Kingdom), Paolo Traverso (Center for Information Technology - IRST (FBK-ICT), Italy), Omer Rana (Cardiff University, United Kingdom), Chung-Sheng Li (IBM, United States) and Cornel Klein (Siemens AG, Germany).

The meeting was also complemented with a Doctoral Consortium on “Cloud Computing and Services Science” chaired by Prof. Paulo Novais.

The high number and high quality of the papers received imposed difficult choices in the review process. Each paper was reviewed by at least two experts. Those papers that, according to the reviews, were considered adequately balanced in terms of quality, originality and relevance to the conference themes were selected. We hope that these Conference Proceedings, submitted for indexation to Thomson Reuters Conference Proceedings Citation Index, INSPEC, DBLP, EI and Scopus, may help the Cloud Computing community to find interesting research work. All presented papers will be available at the SCITEPRESS digital library. Furthermore, a short list of presented papers will be selected and their authors invited to submit an extended and revised version of their paper to be included in a forthcoming

book of CLOSER Selected Papers to be published in CCIS Series by Springer during 2015. A short list of papers will also be selected for publication in a special issue of the Springer's Computing journal and Grid Computing journal.

At the closing session, awards based on the best combined marks of paper reviewing, as assessed by the Program Committee, and the quality of the presentation, as evaluated by session chairs at the conference venue, were conferred to the best papers and the best student papers submitted to the conference. The best paper award for this year's edition was sponsored by Dell.

As a final point, we would like to express our thanks, first of all, to the authors of the technical papers, whose work and dedication make possible to put together a program that we believe is very exciting and of high technical quality. Next, we would like to thank all the members of the program committee and auxiliary reviewers, who helped us with their expertise and time. We would also like to thank the invited speakers for their invaluable contribution and for sharing their vision with the CLOSER 2015 audience. Special thanks should be addressed to the INSTICC Steering Committee whose invaluable work made this event possible and finally, a word of appreciation for the hard work of the secretariat: organizing a conference of this level is a task that can only be achieved by the collaborative effort of a dedicated and highly capable team.

Markus Helfert

Dublin City University, Ireland

Víctor Méndez Muñoz

Universitat Autònoma de Barcelona, UAB, Spain

Donald Ferguson

Dell, U.S.A.

CONTENTS

INVITED SPEAKERS

KEYNOTE SPEAKERS

Cloud Computing and Big Data Can Improve the Quality of Our Life <i>Victor Chang</i>	IS-5
Change Alone is Unchanging - Continuous Context-aware Adaptation of Service-based Systems for Smart Cities and Communities <i>Paolo Traverso</i>	IS-7
In-transit Analytics on Distributed Clouds - Applications and Architecture <i>Omer Rana</i>	IS-9
At Scale Enterprise Computing <i>Chung-Sheng Li</i>	IS-11
Software- and Systems Architecture for Smart Vehicles <i>Cornel Klein</i>	IS-13

CLOUD COMPUTING FUNDAMENTALS

FULL PAPERS

Effects of Active Cooling on Workload Management in High Performance Processors <i>Won Ho Park and C. K. Ken Yang</i>	5
A Mathematical Programming Approach to Multi-cloud Storage <i>Makhlouf Hadji</i>	17
Cloud Provider Transparency - A View from Cloud Customers <i>Daniela Cruzes and Martin Gilje Jaatun</i>	30
OCCI and TTCN-3 - Towards a Standardized Cloud Quality Assessment Framework <i>Yongzheng Liang</i>	40
Using Cloud-Aware Provenance to Reproduce Scientific Workflow Execution on Cloud <i>Khawar Hasham, Kamran Munir and Richard McClatchey</i>	49
Addressing Issues of Cloud Resilience, Security and Performance through Simple Detection of Co-locating Sibling Virtual Machine Instances <i>John O'Loughlin and Lee Gillam</i>	60

SHORT PAPERS

P-TOSCA Portability of SOA Applications <i>Marjan Gusev, Magdalena Kostoska, Sasko Ristov and Aleksandar Donevski</i>	71
A Cloud-based Data Analysis Framework for Object Recognition <i>Rezvan Pakdel and John Herbert</i>	79
Factors Affecting Cloud Adoption and Their Interrelations <i>Radhika Garg and Burkhard Stiller</i>	87

A Comparative Study of Current Open-source Infrastructure as a Service Frameworks <i>Theo Lynn, Graham Hunt, David Corcoran, John Morrison and Philip Healy</i>	95
CSP Formulation for Scheduling Independent Jobs in Cloud Computing <i>M'hamed Mataoui, Faouzi Sebbak, Kadda Beghdad Bey and Farid Benhammadi</i>	105
Quality of Service Trade-offs between Central Data Centers and Nano Data Centers <i>Farzaneh Akhbar and Tolga Ovatman</i>	113
Cloud Readiness Assessment of Legacy Application <i>Flavio Corradini, Francesco De Angelis, Andrea Polini and Samuele Sabbatini</i>	119
Development of an Anything Relationship Management Prototype for the Smart Factory <i>Jonathan Knoblauch, Rebecca Bulander and Thomas Greiner</i>	127
Redefining the Cloud based on Beneficial Service Characteristics - A New Cloud Taxonomy Leads to Economically Reasonable Semi-cloudification <i>Bastian Kemmler and Dieter Kranzlmüller</i>	135
CoMA: Resource Monitoring of Docker Containers <i>Lara Lorna Jiménez, Miguel Gómez Simón, Olov Schelén, Johan Kristiansson, Kåre Synnes and Christer Åhlund</i>	145
A Survey of Trust Management Models for Cloud Computing <i>Flavio Corradini, Francesco De Angelis, Fabrizio Ippoliti and Fausto Marcantoni</i>	155
Towards Dynamic QoS Monitoring in Service Oriented Architectures <i>Norman Ahmed and Bharat Bhargava</i>	163
Reality Vs Hype - Does Cloud Computing Meet the Expectations of SMEs? <i>Katie Wood and Kevan Buckley</i>	172
Offline Scheduling of Map and Reduce Tasks on Hadoop Systems <i>Aymen Jlassi, Patrick Martineau and Vincent Tkindt</i>	178
A Generalized Service Replication Process in Distributed Environments <i>Hany F. El Yamany, Marwa F. Mohamed, Katarina Grolinger and Miriam A. M. Capretz</i>	186
Implementation of Cloud ERP - Moderating Effect of Compliance on the Organizational Factors <i>Shivam Gupta and Subhas C. Misra</i>	194
User Requirement and Behavioral Aspects in Web Service Discovery <i>Wala Ben messaoud, Khaled Ghédira and Youssef Ben Halima</i>	199
PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services <i>Yiannis Verginadis, Antonis Michalas, Panagiotis Gouvas, Gunther Schiefer, Gerald Hübsch and Iraklis Paraskakis</i>	206
Classifying Security Threats in Cloud Networking <i>Bruno M. Barros, Leonardo H. Iwaya, Marcos A. Simplício Jr., Tereza C. M. B. Carvalho, Andrés Méhes and Mats Näslund</i>	214
Setting Priorities - A Heuristic Approach for Cloud Data Center Selection <i>Ronny Hans, David Steffen, Ulrich Lampe, Björn Richerzhagen and Ralf Steinmetz</i>	221

SERVICES SCIENCE FOUNDATION FOR CLOUD COMPUTING

SHORT PAPERS

Business Process Generation by Leveraging Complete Search over a Space of Activities and Process Goals <i>Dipankar Deb, Nabendu Chaki and Aditya Ghose</i>	233
“BPELanon” - Protect Business Processes on the Cloud <i>Marigianna Skouradaki, Vincenzo Ferme, Frank Leymann, Cesare Pautasso and Dieter H. Roller</i>	241
Automated Mapping of Business Process Execution Language to Diagnostics Models <i>Hamza Ghandorh and Hanan Lutfiyya</i>	251
Cross-layer Service Adaptation - State-of-the-Art, Shortcoming Analysis, and Future Research Directions <i>Ameni Meskini, Yehia Taher, Rafiqul Haque and Yahya Slimani</i>	260
The Influence of the Provider’s Service Fairness on the Customer’s Service Recovery Satisfaction and on Positive Behavioral Intentions in Cloud Computing <i>Montri Lawkobkit and Roland Blomer</i>	268
Context-aware Security@run.time Deployment <i>Wendpanga Francis Ouedraogo, Frederique Biennier, Catarina Ferreira Da Silva and Parisa Ghodous</i>	276
Choreography-based Consolidation of Interacting Processes Having Activity-based Loops <i>Sebastian Wagner, Oliver Kopp and Frank Leymann</i>	284
Key Requirements for Predictive Analytical IT Service Management - Architectural Key Characteristics for a Cloud based Realization <i>Christopher Schwarz, Hans-Peter Bauer, Lukas Blödorn and Erwin Zinser</i>	297
BPMN Extensions for Decentralized Execution and Monitoring of Business Processes <i>Jonas Anseeuw, Gregory Van Seghbroeck, Bruno Volckaert and Filip De Turck</i>	304
A Smart Decisional Cognitive System based on Self-adaptability of Web Services to the Context <i>Faïçal Felhi, Marwa Ayadi and Jalel Akaichi</i>	310

CLOUD COMPUTING PLATFORMS AND APPLICATIONS

FULL PAPERS

Secure Evidence Collection and Storage for Cloud Accountability Audits <i>Thomas Ruebsamen, Tobias Pulls and Christoph Reich</i>	321
Supporting Multiple Persistence Models for PaaS Applications using MDE - Issues on Cloud Portability <i>Elias Adriano Nogueira da Silva, Daniel Lucrédio, Ana Moreira and Renata Fortes</i>	331
A Lightweight Tool for Anomaly Detection in Cloud Data Centres <i>Sakil Barbhuiya, Zafeirios Papazachos, Peter Kilpatrick and Dimitrios S. Nikolopoulos</i>	343
Performance and Cost Evaluation for the Migration of a Scientific Workflow Infrastructure to the Cloud <i>Santiago Gómez Sáez, Vasilios Andrikopoulos, Michael Hahn, Dimka Karastoyanova, Frank Leymann, Marigianna Skouradaki and Karolina Vukojevic-Haupt</i>	352

SHORT PAPERS

An Approach in the Design of Common Authentication Solution for a Multi-Platform Cloud Environment <i>Primož Cigoj, Borka Jerman Blažič and Tomaž Klobučar</i>	365
Executing Bag of Distributed Tasks on Virtually Unlimited Cloud Resources <i>Long Thai, Blesson Varghese and Adam Barker</i>	373
Automatic Abstraction of Flow of Control in a System of Distributed Software Components <i>Nima Kaviani, Michael Maximilien, Ignacio Silva-Lepe and Isabelle Rouvellou</i>	381
Towards Cross-layer Monitoring of Cloud Workflows <i>Eric Kübler and Mirjam Minor</i>	389
Automating Resources Discovery for Multiple Data Stores Cloud Applications <i>Rami Sellami, Michel Veldrine, Sami Bhiri and Bruno Defude</i>	397
MusicBeetle - Intelligent Music Royalties Collection and Distribution System <i>Carlos Serrão, Hélder Carvalho and Nelson Carvalho</i>	406
Context-aware MapReduce for Geo-distributed Big Data <i>Marco Cavallo, Giuseppe Di Modica, Carmelo Polito and Orazio Tomarchio</i>	414

CLOUD COMPUTING ENABLING TECHNOLOGY

FULL PAPERS

Secure Keyword Search over Data Archives in the Cloud - Performance and Security Aspects of Searchable Encryption <i>Christian Neuhaus, Frank Feinbube, Daniel Janusz and Andreas Polze</i>	427
A Many-objective Optimization Framework for Virtualized Datacenters <i>Fabio López Pires and Benjamín Barán</i>	439
CloudMPL: A Domain Specific Language for Describing Management Policies for an Autonomic Cloud Infrastructure <i>Marwah M. Alansari, Andre Almeida, Nelly Bencomo and Behzad Bordbar</i>	451
Dynamic Testing and Deployment of a Contract Monitoring Service <i>Ellis Solaiman, Ioannis Sfyrakis and Carlos Molina-Jimenez</i>	463
ANY2API – Automated APIfication - Generating APIs for Executables to Ease their Integration and Orchestration for Cloud Application Deployment Automation <i>Johannes Wettinger, Uwe Breitenbücher and Frank Leymann</i>	475
A Modelling Concept to Integrate Declarative and Imperative Cloud Application Provisioning Technologies <i>Uwe Breitenbücher, Tobias Binz, Oliver Kopp, Frank Leymann and Johannes Wettinger</i>	487

SHORT PAPERS

A Hedonic Price Index for Cloud Computing Services <i>Persefoni Mitropoulou, Evangelia Filiopoulou, Stavroula Tsaroucha, Christos Michalakelis and Mara Nikolaidou</i>	499
---	-----

New Approach to Partitioning Confidential Resources in Hybrid Clouds <i>Kaouthar Samet, Samir Moalla and Mahdi Khemakhem</i>	506
Cloud Spreadsheets Supporting Data Processing in the Encrypted Domain <i>D. A. Rodríguez-Silva, L. Adkinson-Orellana, B. Pedrero-López and F. J. González-Castaño</i>	514
SLAFM - A Service Level Agreement Formal Model for Cloud Computing <i>Lucia De Marco, Filomena Ferrucci and M-Tahar Kechadi</i>	521
Towards High Performance Big Data Processing by Making Use of Non-volatile Memory <i>Shuichi Oikawa</i>	529
The Docker Ecosystem Needs Consolidation <i>René Peinl and Florian Holzschuher</i>	535
Container-based Virtualization for HPC <i>Holger Gantikow, Sebastian Klingberg and Christoph Reich</i>	543
Towards Self-Protective Multi-Cloud Applications - MUSA – a Holistic Framework to Support the Security-Intelligent Lifecycle Management of Multi-Cloud Applications <i>Erkuden Rios, Eider Iturbe, Leire Orue-Echevarria, Massimiliano Rak and Valentina Casola</i>	551
High Performance Virtual Machine Recovery in the Cloud <i>Valentina Salapura and Richard Harper</i>	559
Adopting an Agent and Event Driven Approach for Enabling Mutual Auditability and Security Transparency in Cloud based Services <i>Moussa Ouedraogo, Eric Dubois, Djamel Khadraoui, Sebastien Poggi and Benoit Chenal</i>	565
MOBILE CLOUD COMPUTING AND SERVICES	
FULL PAPERS	
The Case for Visualization as a Service - Mobile Cloud Gaming as an Example <i>Abdelmounaam Rezgui and Zaki Malik</i>	577
Cloud-side Execution of Database Queries for Mobile Applications <i>Robert Pettersen, Steffen Viken Valvåg, Åge Kvalnes and Dag Johansen</i>	586
SHORT PAPER	
Telco Clouds - Modelling and Simulation <i>Jakub Krzywda, William Tärneberg, Per-Olov Östberg, Maria Kihl and Erik Elmroth</i>	597
AUTHOR INDEX	611

INVITED SPEAKERS

KEYNOTE SPEAKERS

Cloud Computing and Big Data Can Improve the Quality of Our Life

Victor Chang

Leeds Beckett University, U.K.

Abstract: The rise of Cloud Computing and Big Data has played influential roles in the evolution of IT services and has made significant contributions to different disciplines. For example, there are ten services that cannot be achieved without the combined effort from Cloud Computing and Big Data techniques: They are Storage as a Service, Health Informatics as a Service, Financial Software as a Service, Business Intelligence as a Service, Education as a Service, Big Data Processing as a Service, Integration as a Service, Security as a Service, Social Network as a Service and Data Visualization as a Service (Weather Science) respectively, in which the keynote speaker will summarize the motivation, methods, results and contributions in each service. He will explain how the unique services can improve the quality of our life by understanding the complex biological and physiological science and ensuring the best approaches of treatments and actions can be adopted. These include development projects and successful deliveries in brain segmentation and learning, proteins and body defense mechanisms, tumor studies and DNA sequencing. Research and enterprise contributions to other disciplines are available which include Business Intelligence as a Service to provide accurate and up-to-date tracking of risk and prices with regard to the investment, as well as contributions for weather data visualization and forecasting to inform the general public about the consequences of the extreme weather.

BRIEF BIOGRAPHY

Dr. Victor Chang is a Senior Lecturer in the School of Computing, Creative Technologies at Leeds Beckett University, UK and a visiting Researcher at the University of Southampton, UK. He is an expert on Cloud Computing and Big Data in both academia and industry with extensive experience in related areas since 1998. Dr Chang completed a PGCert (Higher Education) and PhD (Computer Science) within four years while working full-time. He has over 70 peer-reviewed published papers. He won £20,000 funding in 2001 and £81,000 funding in 2009. He was involved in part of the £6.5 million project in 2004, part of the £5.6 million project in 2006 and part of a £300,000 project in 2013. Dr Chang won a 2011 European Identity Award in Cloud Migration. He was selected to present his research in the House of Commons in 2011 and won the best student paper in CLOSER 2012. He has demonstrated Storage as a Service, Health Informatics as a Service, Financial Software as a Service, Education as a Service, Big Data Processing as a Service, Integration as a Service, Security as a Service, Social Network as a Service, Data Visualization as a Service (Weather Science) and Consulting as Service in Cloud Computing and Big Data services in both of his practitioner and

academic experience. His proposed frameworks have been adopted by several organizations. He is the founding chair of international workshops in Emerging Software as a Service and Analytics and Enterprise Security. He is a joint Editor-in-Chief (EIC) in International Journal of Organizational and Collective Intelligence and a founding EIC in Open Journal of Big Data. He is the Editor of a highly prestigious journal, Future Generation Computer Systems (FGCS). He is a reviewer of numerous well-known journals. He has 27 certifications with 97% on average. He is a keynote speaker of CLOSER/WEBIST/ICT4AgeingWell 2015. Dr Chang has published three books on Cloud Computing which are available on Amazon website.

Change Alone Is Unchanging

Continuous Context-aware Adaptation of Service-based Systems for Smart Cities and Communities

Paolo Traverso

Center for Information Technology - IRST (FBK-ICT), Italy

Abstract: Service-based systems have to deal with highly dynamic environments in which they must often operate. Consider for instance the case of smart cities and communities, i.e., communities of people who actively participate to the creation and use of ICT-based solutions to improve their quality of life within their own city or region. Within a smart city and community, the context in which applications must operate continuously changes, as well as the situation, the accessibility to (ICT-based) services, the people, their interactions, requirements, and preferences. Moreover, most often, the only way applications can react to such changing environment is at run-time, since we cannot predict a priori different situations, requirements, interactions, and availability of (ICT) services. Continuous context-aware and incremental adaptation becomes therefore the key enabling property for the delivery of ICT based value added services to cope with the dynamics of the continuously changing environment.

In my talk, I will present some of the compelling needs for context aware incremental adaptation in the case of service-based applications for smart cities and communities. I will discuss some alternative approaches, some lessons learned from applications we have been working with in this field, and the still many related open research challenges.

BRIEF BIOGRAPHY

Paolo Traverso is the Director of FBK ICT first, Centre for Information Technology at FBK (Fondazione Bruno Kessler) since 2007. The Centre counts about 200 people working on software and services, cloud computing, embedded systems, content and semantics, perception and interaction.

He was also CEO of Trento RISE (the Trento Research, Innovation, and Education System) from 2011 until June 2014, the association between FBK and the University of Trento, which is part of the European Institution of Innovation and Technology (EIT) in ICT, the EIT ICT Labs.

Paolo joined IRST after working in the advanced technology groups of companies for management information consulting in Chicago, London, and Milan, where he led projects for the development of safety critical systems, data and knowledge management, and service oriented applications. He contributed to research in automated planning and service oriented computing.

He was Program Chair of the International Conference on Automated Planning and Scheduling (ICAPS), General and Program Chair of the International Conference on Service-Oriented

Computing (ICSOC), and Program Chair of the Extended Semantic Web Conference (ESWC). His recent research interests are in the monitoring, adaptation, evolution of service oriented applications, and in the development of new-generation services delivery platforms for improving individual and societal quality of life.

In-transit Analytics on Distributed Clouds

Applications and Architecture

Omer Rana
Cardiff University, U.K.

Abstract: The increasing deployment of sensor network infrastructures (in a variety of applications, ranging from environmental monitoring, "Smart Cities", energy demand forecasting, social media analysis to emergency response) has led to large volumes of data becoming available, leading to new challenges in storing, processing, analysing and transferring such data. This is especially true when data from multiple sensors is pre-processed prior to delivery to users. Where such data is processed in-transit (i.e. from data capture to delivery to a user) over a shared distributed computing infrastructure, and due to the increasing availability of software defined networks, it is necessary to provide some Quality of Service (QoS) guarantees to each user. This talk provides: (i) scenarios of applications that have these types of characteristics; (ii) a computational architecture for supporting QoS for multiple concurrent scientific workflow data streams being processed (prior to delivery to a user) over a shared infrastructure. The architecture is used to demonstrate how a streaming pipeline, with intermediate data size variation (inflation/deflation), can be supported and managed using a dynamic control strategy at each node. Such a strategy supports end-to-end QoS with variations in data size between the various nodes involved in the workflow enactment process.

BRIEF BIOGRAPHY

Omer Rana is Professor of Performance Engineering at Cardiff School of Computer Science & Informatics. He was formerly the deputy director of the Welsh eScience Centre at Cardiff University -- where he had an opportunity to collaborate with a number of scientists working in computational science and engineering. He holds a PhD in "Neural Computing and Parallel Architectures" from Imperial College (University of London, UK). His research interests are in the areas of high performance distributed computing, data mining and analysis and multi-agent systems. Prior to joining Cardiff University he worked as a software developer with Marshall BioTechnology Limited in London, working on projects with a number of international biotech companies, such as Merck, Hybaid and Amersham International. He has been involved in the Distributed Programming Abstractions and the 3DPAS themes at the UK National eScience Institute. He is an associate editor of the ACM Transactions on Autonomous and Adaptive Systems, IEEE Transactions on Cloud Computing, series co-editor of the book series on "Autonomic Systems" by Birkhauser publishers, and on the editorial boards of "Concurrency and Computation: Practice and Experience" (John

Wiley), the Journal of Computational Science (Elsevier) and the recently launched IEEE Cloud Computing magazine. Along with his co-researchers, he was recipient of the best paper award at CLOSER 2013 (Aachen, Germany).

At Scale Enterprise Computing

Chung-Sheng Li
IBM, U.S.A.

Abstract: At scale computing” is becoming one of the most disruptive trends in recent technology history, and is becoming the central theme for the post distributed computing era and likely to become the primary driver for the innovation and IT transformation during the coming decade.

At scale computing describes a computing environment that may involve very large amount of computation, transactions, large amount of data, large number of users, or any combinations of the above. Recent examples of at scale computing include AWS/Google datacenters, Amazon/Alibaba e-Commerce activities, Facebook, Google search, Netflix, etc. These are in direct contrast with more traditional at scale enterprise such as Walmart, UPS, VISA/Mastercard/Amex.

At scale computing is a paradigm shift from the traditional distributed computing. It is both a blessing and a curse: Very large scale of computation offers new opportunities to rethink how to achieve resiliency without having to pay for redundancy. It definitely promotes the possibility of “fail in place” as opposed to having to perform field service of a failed component in the environment as soon as it happens. It also offers great opportunity to amortize the potential investment involved in optimization and customization. At scale does impose severe challenge on just about every aspect of a large scale system, including system architecture, hardware, software and the continuous operations. It stresses the importance of continuous availability, extreme scalability, and maniac focus on cost efficiency (capex, opex) as every penny counts in this at scale environment.

In this keynote, we will discuss this new “at scale” era - which we believed has started, and perhaps well under way. Nearly all exciting innovations for enterprise computing during the past decade originated from this environment. These innovative technologies were often motivated by at scale business models. All of these at scale business models started small. They invariably found the recipe for creating a virtuous cycle among the “addictive” services or content they provided, and the community and ecosystem created on top of it. In such a virtuous cycle, addictive content or services attracted new members into the community (either as direct consumers or developers), which in turn generate more content or develop more services. This virtuous cycle often leads to winner takes all in nearly all case studies.

BRIEF BIOGRAPHY

Dr. Chung-Sheng Li is currently the director of the Commercial Systems Department, PI for the IBM Research Cloud Initiatives, and the executive sponsor of the Security 2.0 strategic initiative. He has been with IBM T.J. Watson Research Center since May 1990. His research interests include cloud computing, security and compliance, digital library and multimedia databases, knowledge discovery and data mining, and data center networking. He has authored or coauthored more than 130 journal and conference papers and received the best paper award from IEEE Transactions on Multimedia in 2003. He is both a member of IBM Academy of Technology and a Fellow of the IEEE. He received BSEE from National Taiwan University, Taiwan, R.O.C., in 1984, and the MS and PhD degrees in electrical

engineering and computer science from the University of California, Berkeley, in 1989 and 1991, respectively.

Software- and Systems Architecture for Smart Vehicles

Cornel Klein
Siemens AG, Germany

Abstract: Both fully automated driving and electromobility are promising approaches to address the challenges of mobility with regards to sustainability, urbanization and demographic change. Moreover, they also change the usage patterns and concepts for future passenger vehicles and enable new kinds of applications for special purpose vehicles, for instance in logistics. Recently, many projects and demonstrators have shown the feasibility and tremendous potential of driving automation for building such “Smart vehicles”. However, we are convinced that for the cost-effective development, validation and deployment of automation functions current vehicle architectures are insufficient. Therefore, we present results and research directions in software- and systems architectures. Moreover, we discuss their relevance for the efficient implementation of new vehicle functions and innovative applications.

BRIEF BIOGRAPHY

Cornel Klein is Software Architect and Project Manager for the Technology & Innovation Project “eCar” at Siemens Corporate Technologies in Munich. He is project manager and coordinator for RACE (Robust and reliable Automotive Computing Environment for future eCars) which aims at the development of an advanced automotive E/E architecture. In various positions at Siemens, he has been responsible for software technologies and SW based innovations. Starting his career 1998 at Siemens Public Networks, he has gained an extensive knowledge in communication networks, embedded systems, IT platforms and SW architecture as well as in application domains like eCars and smart environments. He holds a master and a PhD degree in Computer Science from the Technical University of Munich.

CLOUD COMPUTING FUNDAMENTALS

FULL PAPERS

Effects of Active Cooling on Workload Management in High Performance Processors

Won Ho Park and C. K. Ken Yang

Department of Electrical Engineering, UCLA, Los Angeles, U.S.A.
wonhopark80@gmail.com, yang@ee.ucla.edu

Keywords: Power-aware Systems, Workload Scheduling, Server Provisioning, Electronic Cooling, Data Centers, Thermal Management.

Abstract: This paper presents an energy-efficient workload scheduling methodology for multi-core multi-processor systems under actively cooled environment that improves overall system power performance with minimal response time degradation. Using a highly efficient miniature-scale refrigeration system, we show that active-cooling by refrigeration on a per-server basis not only leads to substantial power-performance improvement, but also improves the overall system performance without increasing the overall system power including the cost of cooling. Based on the measured results, we present a model that captures different relations and parameters of multi-core processor and the refrigeration system. This model is extended to illustrate the potential of power optimization of multi-core multi-processor systems and to investigate different methodologies of workload scheduling under the actively cooled environment to maximize power efficiency while minimizing response time. We propose an energy-efficient workload scheduling methodology that results in total consumption comparable to the *spatial subsetting* scheme but with faster response time under the actively cooled environment. The actively cooled system results in $\geq 29\%$ of power reduction over the non-refrigerated design across the entire range of utilization levels. The proposed methodology is further combined with the G/G/m-model to investigate the trade-off between the total power and target SLA requirements.

1 INTRODUCTION

Increase in energy consumption due to the tremendous growth in the number and size of data centers presents a whole new set of challenges in maintaining energy-efficient infrastructure. While data centers' energy consumption had accounted for 2% of the total energy budget of the USA in 2007, it is expected to reach 4% by the year 2011. This number is equivalent to \$7.4 billion per year on electric power where this number has changed by 60% since 2006 (U.S. EPA, 2007).

Worldwide trend of energy consumption in data centers tracks the US trend (Rajamani, 2008). Fig. 1 shows the number of data center installations, worldwide new server spending, and electric power and cooling costs. Despite the steady increase of installed base of data centers over the last decade, new server spending has stayed relatively constant due to the decrease in electronic costs. As the data center infrastructure becomes denser, power density has been increasing by approximately 15% annually

(Humphreys, 2006), hence increasing electricity consumption for operating servers and cooling. It is likely that IT operating cost will soon outweigh the initial capital investment.

Detailed energy breakdown of different types of data centers can be found in (Rajamani, 2008), (Tschudi, 2003), (Lawrence Berkeley National Labs, 2007), (Patterson, 2008). A data center can consist of hundreds or even thousands of server racks where each rack can draw more than 20kW of power. Relative percentage of various contributors to energy usage varies considerably among data centers, but up to 90% of the total energy is attributed to the energy dissipated by the computer load and the energy required by the Computer Room Air Conditioning unit (CRAC). Additionally, there is a strong relation between the energy consumed by the computer load and the CRAC units since any reduction in electronic heat can be compounded in the cooling system. For example, CRAC energy efficiency of data centers can increase by 1% per degree Celsius.

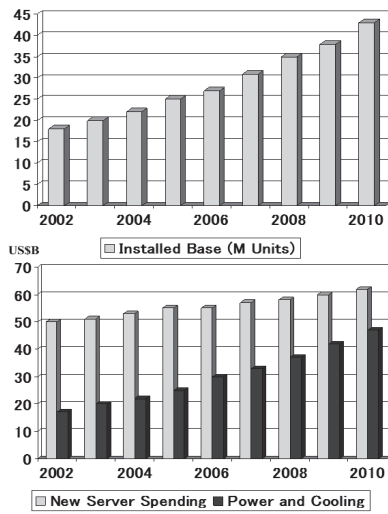


Figure 1: (a) Number of worldwide installed bases of data center. (b) Worldwide spending on new servers and operation cost. Adopted from (Rajamani, 2008).

Reducing the overall energy usage is an area of interest across multiple disciplines. The focus of most efforts on energy/power saving in server systems is on processor elements (Jing, 2011), (Chaparro, 2007), (Ma, 2003), (Tschanz, 2003), (Brooks, 2000), (Sato, 2007), (Ghosh, 2011), (Rabaey, 2003). Approaches for power saving of processors often adopt both the software-based energy-aware workload scheduling (Jing, 2011), (Lin, 2011), (Luo, 2013) and hardware-based circuit and architectural power management techniques to effectively optimize energy usage. A typical software-based workload scheduling algorithm controls energy by distributing workloads to processors in a way to reduce both the electric and cooling costs. The basis of these approaches relies on powering-off servers that are not utilized by concentrating the workload on a subset of the servers. This method is known as *spatial subsetting*, and has been shown to successfully tackle the issue of idle server power consumption (Jing, 2011), (Pinheiro, 2001), (Chase, 2001).

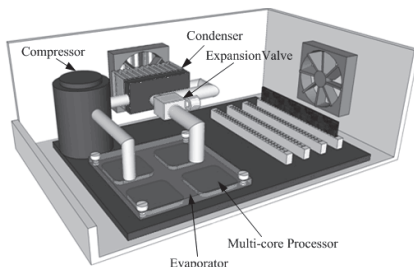


Figure 2: Configuration of a multi-processor computing server unit with a refrigerated-cooling.

Moreover, energy savings from the off-power servers is compounded in the cooling systems that consume power to remove the heat dissipated in the servers. While this approach significantly reduces idle power, it raises a concern of degraded response time in computing systems, due to the power-latency trade off.

To address the problem of degraded response time in *spatial subsetting*, one solution is to employ an *over-provisioning* scheme (Chen, 2005), (Ahmad, 2010). The over-provisioning algorithm can be considered as a power and response time optimization problem. By predicting how many servers are required to service the requested workload, the workload management software assigns a subset of processors to remain at idle state to absorb sudden increases in the load. Determining the number of server to be held at idle state often relies on a good model that successfully plans capacity depending on the upcoming workloads. For instance, G/G/m-models from queueing theory have been used to obtain useful measures like average execution velocity and average wait time to support capacity and workload planning of multi-processor systems in order to satisfy target SLA requirements (Chen, 2005), (Ahmad, 2010), (Müller-Colstermann, 2007).

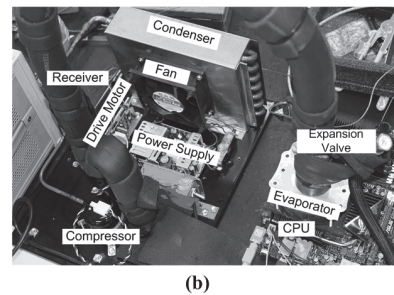
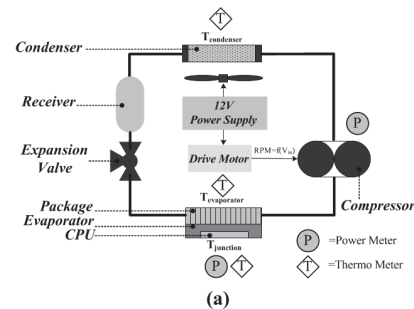


Figure 3: (a) Layout and (b) photograph of the refrigeration system for electronic cooling.

Often the software utilizes special hardware supports (Jing, 2011), such as dynamic voltage frequency scaling (McGowen, 2006), (Burd, 2000),

(Nowka, 2002), or thread migration (Zhang, 2005), stopping a processor through power gating (Tschanz, 2003), (Zhang, 2005), (Henzler, 2005), or body biasing (Tschanz, 2003). However, these techniques not only induces area penalty but also require some transition time in and out of the low-power state and imply performance degradation.

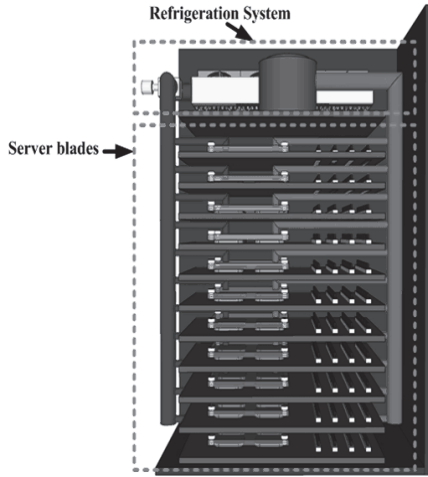


Figure 4: Configuration of a multi-processor computing server unit with a refrigerated-cooling.

Along with these techniques, actively cooling the processors using refrigeration has attracted recent interest as a practical option to ease the power problems in high performance computing units (Copeland 2005), (Mahajan, 2006), (Nnanna, 2006), (Chu, 2004), (Trutassanawin, 2006). Operating CMOS circuitry at sub-ambient temperatures for higher performance has been shown over the past few decades (Carson, 1989), (Aller, 2000). While the speed improvement can be traded for lower power dissipation of the electronics, the cost of cooling can limit the overall system power performance. Recent work (Park, 2010), (Park, 2010), has shown that active cooling not only can lead to overall power improvement that includes the cost of cooling power without performance degradation; the results show that the amount of power savings is roughly proportional to the ratio of leakage power to total power due to the exponential sensitivity of leakage power to temperature, irrespective of type of workload. For instance, cooling a processor that dissipates 175.4W of power with 30% electronic leakage power resulted in a total system power consumption of 133W. This performance is 25% better than the non-cooled reference design (Park, 2010). Focus of this paper is to explore the effectiveness of workload scheduling to improve power efficiency of multi-core multi-

processor systems in an actively cooled environment using a highly efficient refrigeration system. Results presented in this paper suggest that there exists a methodology under actively cooled environment that optimizes power efficiency while minimizing response time in and out of the low-power state. Furthermore, we combine our proposed methodology with the G/G/m-models to reduce both total power and response time degradation while meeting target SLA requirements.

2 MULTI-CORE PROCESSOR UNDER THE ACTIVELY-COOLED ENVIRONMENT

A miniature-scale refrigeration system for electronic cooling that is capable of operating at a reduced temperature with high efficiency has been developed and experimentally tested in (Park, 2010). The compressor used in our miniature refrigeration system has cooling capacity in the several hundred-watt ranges, indicating that this refrigeration system can potentially be configured to simultaneously cool multi-processor servers. We envision a possible configuration of the HPC server unit as illustrated in Fig. 2.

A layout and photograph of the refrigeration system for electronic cooling is shown in Fig. 3. A configuration of the refrigeration system charged with R-134a refrigerant consists of a compressor, condenser, an expansion valve, a cold plate, evaporator, and a cooling fan. A 12V power supply provided the required power. Additionally, a motor drive board is installed to control the compressor speed and modulate the refrigeration capacity at different loads. K-type bead probes are taped to the evaporator and the condenser for temperature measurements. Power meters are used to measure power consumptions of the cooler and the heat source. By controlling the speed of the compressor, we cool the microprocessor at different heat loads and temperatures in order to obtain minimal total system power. Specific chip junction temperature would be the temperature that resulted in the lowest system power. The detailed description of the experimental setup and performance of our miniature refrigeration system for electronic cooling can be found in (Park, 2010). We characterize the power performance of a 4-core processor at different operating conditions using this refrigeration system. It is important to mention that while our analysis

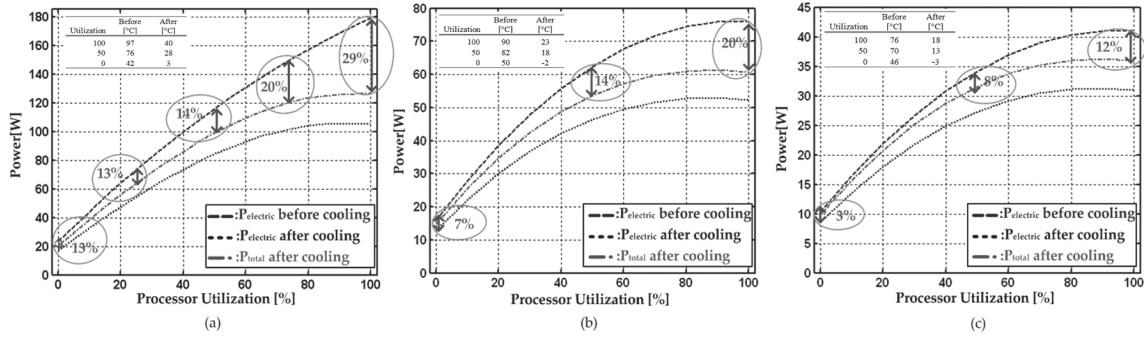


Figure 5: Power consumption before and after cooling across different processor utilization levels when (a) 4, (b) 2, or (c) 1 core out of the 4-core processor is powered up based on the model and measured data. The associated power savings after electronic cooling is also shown.

uses a system that can be enclosed in a server chassis, and vapor compression refrigeration systems can achieve considerably higher efficiency with larger cooling capacity at the expense of larger volume. Such systems can potentially cool entire racks of servers with the coolant distributed with parallel flow through the server blades as shown in Fig. 4. The results discussed in this paper can be directly applied.

The mechanisms for power dissipation of digital CMOS ICs are well understood. The total power dissipation can be estimated by the sum of the active power and leakage power (Rabaey, 2003), (Chandrakasan, 1992).

$$P_{electric} = P_{active} + P_{leakage} \quad (1)$$

$$P_{electric} = \alpha * C_{switched} * f_{clk} * V_{dd}^2 \quad (2)$$

$$P_{leakage} = V_{dd} * I_0 \exp\left(\frac{-V_{th}}{kT_{junction}/q}\right) \quad (3)$$

The active power, P_{active} depends on the activity factor, α , and the amount of power that dissipates charge/discharge capacitive nodes between the supply voltage (V_{dd}) and ground when executing the logic, $C_{switched}f_{clk}V_{dd}^2$. At nano-meter scale technology, the switches that implement the logic results in a leakage current to flow through each logic gate even when the logic is not active. This leakage becomes a significant component of total chip power in modern era processors. The leakage power, $P_{leakage}$, has an exponential relation with the degree that a transistor's ON/OFF threshold, V_{th} , exceeds the thermal voltage, $KT_{junction}/q$. The $P_{leakage}$ equation simplifies the dependence of leakage power by lumping (1) the number and size of logical switching paths in a computational unit, (2) the carrier properties in the transistor, and (3)

dependence of leakage on the logical structure of each logic gate of a digital processor into a single constant I_0 .

For a digital processor, power dissipation and computing performance are closely related. The Equation (4) shows this relationship for the delay of a logic gate. The current is a function of temperature and primarily depends on the carrier mobility. A designer can typically trade-off any improved speed performance by reducing the supply voltage, V_{dd} .

$$Delay \propto RC \propto \frac{V_{dd}}{I_{logic}} \quad (4)$$

Lower temperatures lead to improved performance of electronic devices. Lower power and higher speed results from (1) an increase in carrier mobility and saturation velocity, (2) an exponential reduction in sub-threshold currents from a steeper sub-threshold slope (KT/q), (3) an improved metal conductivity for lower delay, and (4) better threshold voltage control enabling to lower V_{th} .

For coolers and refrigerators, the efficiency is represented in terms of COP defined by

$$COP = \frac{P_{electric}}{P_{cooling}} \quad (5)$$

where $P_{cooling}$ represents the cooling power of the refrigeration system required to lift the total amount of heat ($P_{electric}$) generated by the processor.

Furthermore, the cooling power can be expressed in terms of COP and the COP of the Carnot cycle with Eq. (6) and (7) where T_{evap} is the cold-end temperature of the evaporator, T_{cond} is the temperature at the condenser, and η is the second law of efficiency.

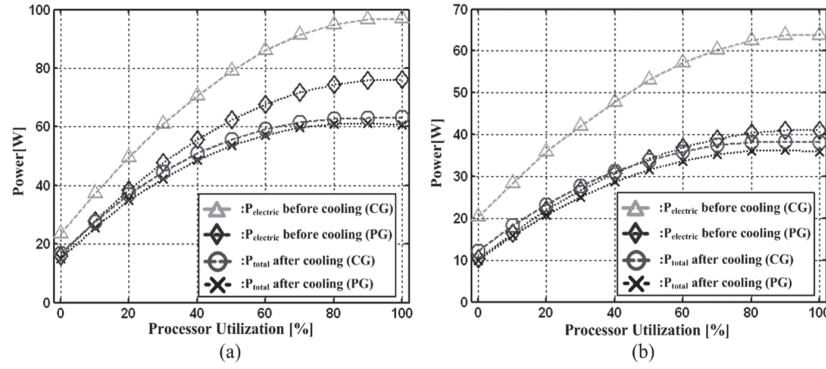


Figure 6: Power consumption across different utilization level before and after cooling using PG and CG as the core stopping techniques for (a) 2-core and (b) 1-core processor.

$$\frac{P_{electric}}{P_{cooling}} = \eta COP_{carnot} = \eta \left(\frac{T_{evap}}{T_{cond} - T_{evap}} \right) \quad (6)$$

$$P_{cooling} = \frac{P_{electric}}{\eta} \left(\frac{T_{cond}}{T_{evap}} - 1 \right) \quad (7)$$

Equating Eq. (1) and (7) results in total system power of

$$P_{total} = P_{electric} + P_{cooling} \quad (8)$$

that includes the cooling power consumption in order to quantify whether the system offers an overall power reduction at different operating temperature. The model serves as a useful tool to evaluate overall system performance including optimal operating temperatures and the amount of total power reduction.

Using this approach, we explore the optimal operating conditions of the system across different processor utilization. In order to experimentally quantify the power consumption of compute-intensive processors, the workload used in all our experiments is Intel's LINPACK, workload, which is CPU bound. Note that our model tracks well with measured data (Park, 2010), (Park, 2010). It is also important to emphasize that our experiment not only uses voltage scaling to trade-off the improved speed performance into a power reduction but also controls refrigeration system to modulate the cooling capacity in order to obtain minimal system power. The speed performance of the processor is kept constant across utilization level. The results are used to build a model of the 4-core processor operating at reduced temperatures and applied to multi-core multi-processors in later sections.

The 4-core processor can be configured such that 1, 2 or 4 cores are active while unused cores are completely turned off to address the problem of idle

power consumption. The refrigeration system is used to cool the microprocessor at different configurations. The amount of total power before and after cooling and the associated power saving across different process utilization levels for different number of cores is shown in Fig. 5. Here, total power before cooling represents forced air cooling that includes the fan power. As can be seen, the result shows that the total power savings of at least 3, 7 and 13 percent can be obtained across the entire range of processor utilization for 1, 2, and 4 cores respectively. The detailed temperature and voltage operating points and the breakdown of the total system power in terms of active, leakage, and cooling components at different utilization with and without active cooling components are shown in (Park, 2010). Effectiveness of cooling is proportional to utilization level. This result suggests that the energy-conscious provisioning would need to concentrate the workload on a minimal active set of cores that run near a maximum utilization level, while other excess cores transition to low-power states to reduce the energy cost. However, using power gating (PG) technique to power on/off cores comes at a price of response time degradation since powering up a core that is completely shut down requires up to 1000 cycles (Kumar, 2003).

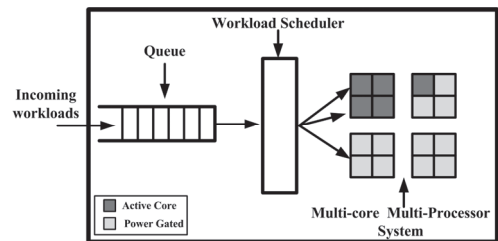


Figure 7: Generic workload scheduling management for multi-core multi-processor computing system.

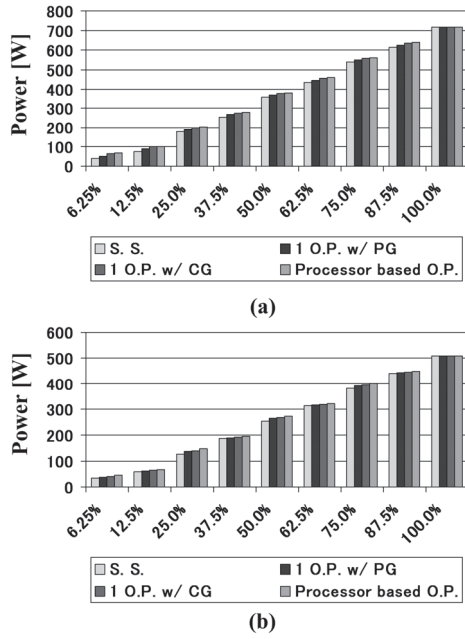


Figure 8: Power at different utilization (a) before and (b) after electronic cooling for different methodologies

On the other hand, a simpler way to stop a core with minimal response time degradation is to clock gate (CG) the core (Tschanz, 2003), (Kurd, 2001). Main advantage of this power saving technique is the state of the processor can be preserved since supply voltage is not cut. This provides a response time which is orders of magnitude faster than waking from power gating or powering up a processor. However, in terms of power consumption, this technique stops dynamic power dissipation, but since power is not entirely cut-off, the core continues dissipating leakage power. Operating CMOS circuitry at reduced temperatures substantially reduces the power since leakage power depends exponentially on temperature. The result that captures the impact of CG at reduced temperatures is shown in Fig. 6.

Before cooling, the CG processor consumes considerably higher power as compared to the PG processor, due to the increase in leakage power. As expected, lowering the temperature of the CG processor exponentially reduces leakage power and results in total power that is comparable to PG processors. At 100% utilization level, power savings from cooling with CG and PG are 36% and 20%, respectively, for a 2-core processor. Results are more significant for a 1-core processor where power savings from cooling with CG and PG are 40% and 12%, respectively. For both cases, CG appears to be a better core stopping technique under the actively

cooled environment. In this way, response time significantly improves at the expense of negligible (~2.5W) power penalty.

The model that captures different relations and parameters of our 4-core processor and the refrigeration system is extended to illustrate the potential of power optimization of multi-core multi-processor systems and investigate different methodologies of workload scheduling under the actively cooled environment.

3 WORKLOAD SCHEDULING METHODOLOGY

With our model derived in Section 2, energy-aware workload scheduling algorithms assign incoming workload to available processors such that power consumption is minimized as constrained by response time requirements.

The server platform we analyze consists of a 4-processor server system with 4-cores per processor under the actively cooled environment. In particular, we are interested in aspects where the effects of electronic cooling change the conventional way of assigning workloads. Detailed results and discussions are presented in this section.

Fig. 7 provides generic management architecture for multi-core multi-processor computing systems where 5 out of 16 cores are utilized. This particular HPC server unit has total of 100% utilization level where each core is responsible for 6.25%. The methodologies we evaluate are the following:

- *Spatial Subsetting (S.S.)*: We assume that unused cores power off by PG. The next core can turn up upon arrival of the workload when the current core is fully occupied.
- *1 Core Over-provision with PG (1 O.P. w/ PG)*: Similar to *spatial subsetting* but one core remains at idle state to absorb sudden peaks in loading.
- *1 Core Over-provision with CG (1 O.P. w/ CG)*: Similar to *1 Core over-provision with PG* but uses CG for the core stopping mechanism.
- *Processor based Over-provision (Processor based O.P.)*: Neither PG nor CG is employed and unused cores remain at idle state.

For all cases, the next processor powers up after all four cores within the active processor are fully utilized to prevent idle power consumption.

For comparison purposes, we show the amount of total power consumption before and after cooling for different types of methodologies across varying

utilization levels in Fig. 8 (a) and (b). Note that the total power after cooling includes the cost of cooling. The impact of cooling on different schemes can be seen through the associated power savings as illustrated in Fig. 9. Several observations can be made based on the results. First, *spatial subsetting* clearly consumes the least amount of power, but the advantage diminishes under the cooled environment. Second, the *processor based over-provision* scheme dissipates the largest amount of power but has no response time degradation. Third, the *1 core over-provision with CG* scheme achieves an excellent compromise that provides the largest amount of power reduction from cooling. Finally, since the next processor powers up after all four cores within the active processor are fully utilized, three power-up transition delays are unavoidable for all cases. They occur from 25% to 31.25%, 50% to 56.25%, and 75% to 81.25%.

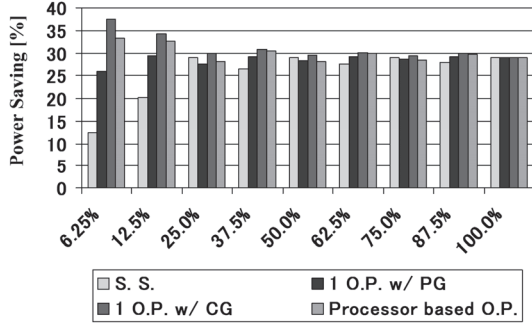


Figure 9: Associated power savings at different utilization level from electronic cooling for different workload assignment methodologies.

Next, we show a new way of assigning workloads under refrigerated cooling and the approach is described in Fig. 10. We demonstrate that the proposed way reduces both the power consumption and the response time requirements at reduced temperature, resulting in power comparable to spatial subsetting but provides a similar response time as 1 core over-provision with CG. Example of the approach is shown in Fig. 10.a; given a workload that requires 4 cores at 100% utilization, the workload scheduling is such that 4 cores are assigned equally to 2 processors. Total power consumption of 196W and 127W is measured, before and after cooling, resulting in a 35% power reduction. On the other hand, the system employing (b) the spatial subsetting scheme and (c) the 1 core over-provision with CG scheme consumes 179W and 199W and 127W and 140W before and after cooling, respectively. The amount of total power saving of the proposed approach is considerably

higher compared to (b) and (c), which has 29% and 30% of power savings.

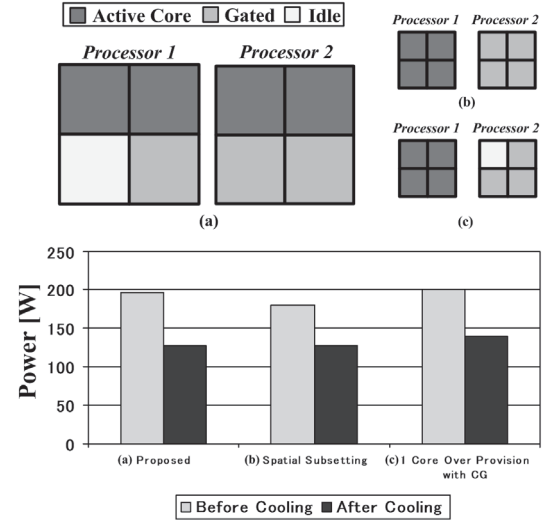


Figure 10: (a) Proposed methodology compared with (b) spatial subsetting and (c) 1 core over-provision with CG.

To be complete, we show the proposed workload scheduling methodology for different utilization levels in Fig. 11. Since power-up events are necessary when a new processor is brought online, three power-up transition delays are unpreventable. These events occur when (B) transitions to (C), (D) transitions to (E), and (F) transitions to (G). In between these transitions and at higher utilizations beyond (G), performance does not degrade with increasing utilization besides the response delay of a few cycles due to CG. Fig. 12 plots the power dissipation and the percentage savings before and after cooling for each of the conditions shown in Fig. 11.

Although conclusions in this section are drawn from a given platform, the intent is not to restrict to a particular platform. The absolute amount of power saving number would be different as different type of systems would have different electronic profile. However, we suggest applying the idea to larger systems where the proposed workload scheduling methodology is applied after cooling. Leveraging the benefits of clock gating at reduced temperatures, our methodology reduces both the power consumption and the response time requirements at reduced temperatures, resulting in power comparable to the *spatial subsetting* scheme but provides a faster response time since our scheme does not power-off processor cores.

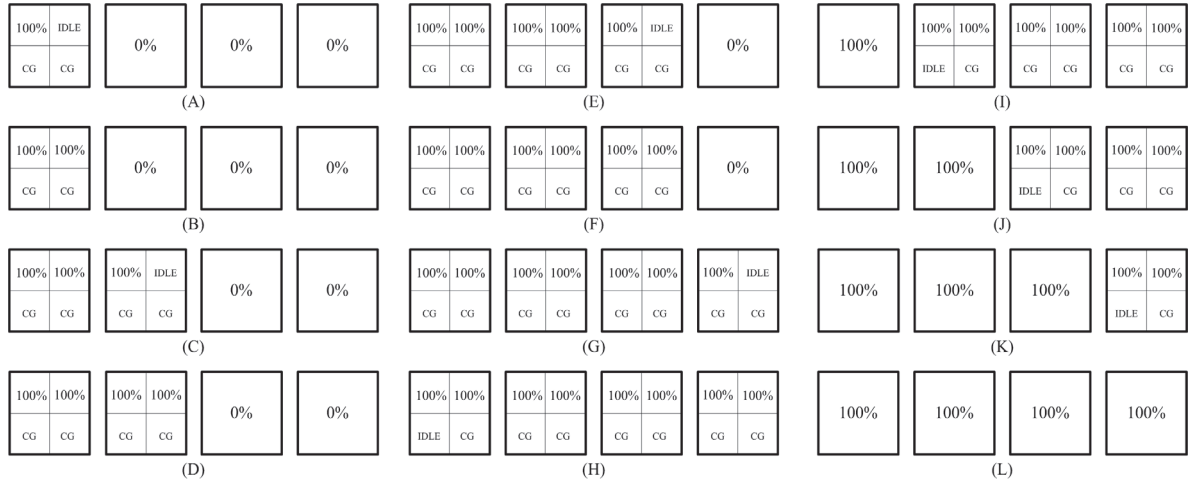


Figure 11: Proposed methodology across different utilization levels.

4 ASSIGNMENT OF WORLOAD BASED ON SLA

As an extension to the proposed methodology, we combine it with the G/G/m-model to reduce both the total power consumption and the response time degradation while meeting specific SLA requirements. Results from the queuing theory have been used to obtain measures like average execution velocity and average wait time to support capacity and workload planning of multi-processor systems for different workload variability (z). Using the approximation formulas for a G/G/m-model, we can reach an optimal agreement between high utilization of the processors (energy-conscious provisioning) and the target SLA requirements. For simplicity, consider a scenario where a specific workload requires 8 cores at 98% of utilization level, and assume that this workload can be linearly mapped to 9 and 10 cores at 87% and 78%, respectively as shown in Fig. 13. Fig. 14 shows the execution velocity for each of these 3 workload scenarios.

Execution velocity is the average ratio for the total amount of workload units that are served without any delay. The value ranges from 0 to 100 where the value 100 means that the workload does not encounter any wait delays for the system resources while the value 0 means that all work is delayed. Fig. 14 is derived using the formula given in (Müller-Colstermann, 2007). When setting the SLA for execution velocity of $>60\%$, using 10 processors to a utilization of 78% satisfies the requirement. On the other hand, using 8 processors result in an unacceptable execution velocity of 6.5%. Moreover, it is important to note that by increasing the number

of processors, there is no transition delay due to powering up a processor, and the only performance degradation results from the response delay of CG.

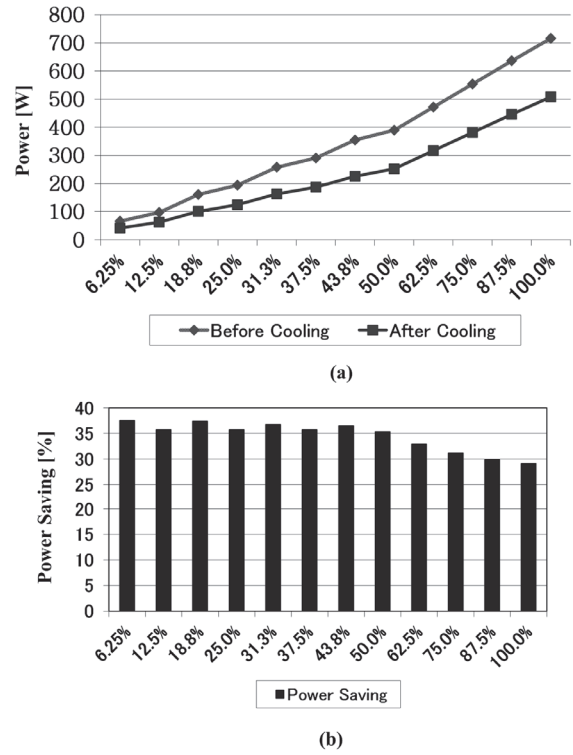


Figure 12: Power consumption at corresponding utilization levels of Fig. 10. Number in the figure represents the associated power saving from electronic cooling.

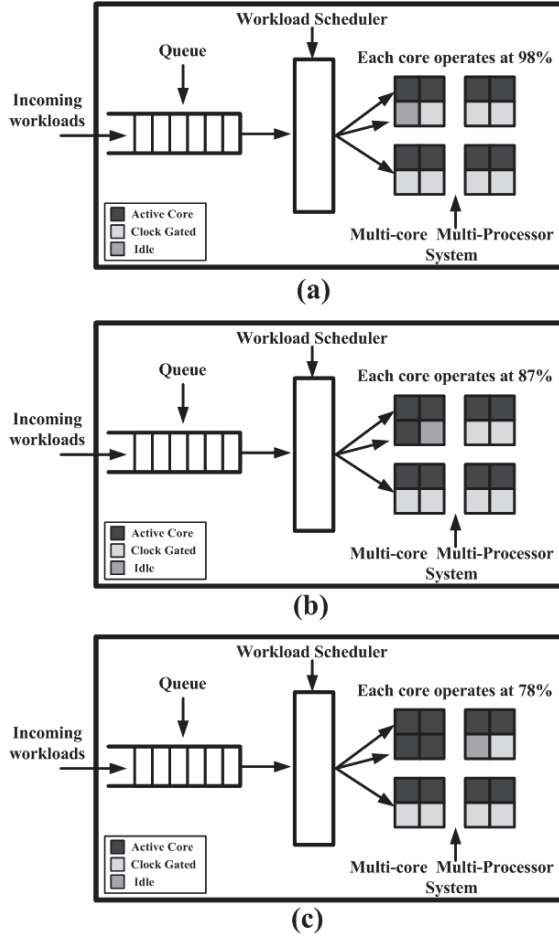


Figure 13: Required utilization level across different number of processors for the proposed methodology.

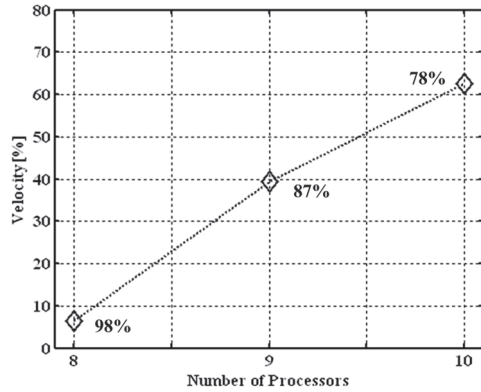


Figure 14: Execution velocity vs. number of processors. Number in the figure represents required utilization level.

Next, we evaluate the normalized average wait time, $E[W]$, for different values of workload

variability, z , where the normalization is performed with respect to the service time to the length of one unit. Here, workload variation represents the variation of request inter-arrival times and request sizes. We consider $0 \leq E[W] \leq 0.5$ for the good quality of service level. Similarly, notice how the system requires 10 processors at 78% of utilization to meet the average wait time requirements for $z \leq 5$ (see Fig.15).

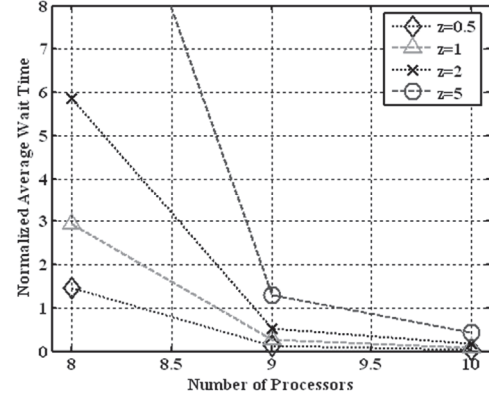


Figure 15: Normalized average wait time vs. number of processors as function of workload variability $z=0.5, 1.0, 2.0, 5.0$.

Finally, we summarize the results of our proposed methodology by comparing with *spatial subsetting*. The total amount of power consumption before and after cooling for the two schemes is shown in Fig. 16. As expected, the actively cooled system with the proposed methodology dissipates power that is comparable to the *spatial subsetting* scheme but enables superior response time for different levels of SLA. Analysis also shows that the overall system power savings of 35, 30, and 29% are obtained when using 8, 9, and 10 cores, respectively. It is worth noting that the amounts of saving decreases as we increase the number of cores as the cores now operate at lower utilization levels. Using a larger number of cores at lower utilization levels inevitably increases the total power consumption, but the system operates with much improved SLA. For instance, using 10 processors instead of 8 increase the total power consumption by 25%, but the system now operates at execution velocity of $>60\%$ and normalized wait time of ≤ 0.5 .

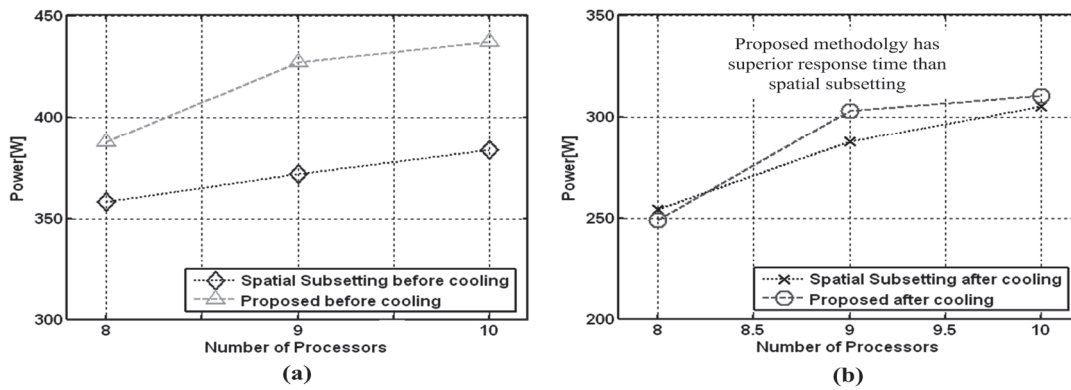


Figure 16: Power consumption vs. number of processors (a) before and (b) after cooling.

5 CONCLUSIONS

An energy-efficient workload scheduling methodology for HPC servers is presented using a highly efficient miniature scale refrigeration system for electronic cooling. By leveraging the benefits of clock gating at reduced temperatures, our proposed methodology results in total power consumption that is comparable to the *spatial subsetting* scheme. Moreover, it provides a response time of disabling clock gating which is orders of magnitude faster than waking from power gating or powering up a processor. Our actively cooled system results in $\geq 29\%$ power reduction over the non-refrigerated design across the entire range of utilization levels. Furthermore, combining our proposed methodology with the G/G/m-model, we show the trade-off between power and SLA requirements. Setting the target SLA requirement to execution velocity of $>60\%$ and normalized wait time of ≤ 0.5 , the number of required processors to execute a particular workload inevitably increased, leading to the 25% increase in total power consumption. Nevertheless, this still maintains 29% of power reduction, compared to non-cooled design.

While the results discussed in this paper can be directly applied to large-scale multi-server systems, overall system realization is still a big challenge and some important design issues of building such systems are overall power consumption, reliability, and cost. Furthermore, thorough understanding of the strong coupling between refrigerated server racks and CRAC units (cascaded cooling system) is needed for future research. Nevertheless, current data centers can consume up to 90% of the total energy from computer load and the energy required by the CRAC units. Decreasing the power dissipated or by the computer load is imperative as any

reduction in electronic heat can be compounded in the cooling system.

Finally, it would be interesting to explore different feedback-driven control solutions that provide capability to adapt to diverse environment, workload, and user constraints. This is relegated to future work. A model-based software framework that predicts and senses upcoming workloads and provides real-time information to refrigeration and electronic systems to tune compressor speed, temperature, and supply voltage are worthy of being studied in order to achieve optimal power performance.

REFERENCES

- U.S. EPA. Report to congress on server and data center energy efficiency. In *U.S. Environmental Protection Agency, Tech Report*, 2007.
- K. Rajamani, C. Lefurgy, J. Rubio, S. Ghiasi, H. Hanson, and T. Keller, "Power management for computer systems and data centers", Tutorial presented at the *2008 International and Symposium on Low Power Electronics and Design*, August, 2008.
- J. Humphreys and J. Scaramella, "The impact of power and cooling on data center infrastructure," Market Research Report, IDC, 2006.
- Tschudi, et al., "Data Centers and Energy Use – Let's Look at the Data", *ACEEE*, 2003.
- Lawrence Berkeley National Labs, Benchmarking: Data Centers, Dec 2007.
- M. Patterson, "The effect of data center temperature on energy efficiency," *Proceedings ITherm*, pp. 1167-1174, 2008.
- Jing, S., Ali, S., She, K., Zhong, Y., State-of-the-art Research Study for Green Cloud Computing. *Journal of Supercomputing*, Special Issue on Cloud Computing, 2011
- M. Lin, A. Wierman, L. Andrew, and E. Thereska,

- "Dynamic Right-Sizing for Power-Proportional Data Centers," *Proc. IEEE INFOCOM*, 2011.
- J. Luo, L. Rao, and X. L. Liu, "Data center energy cost minimization: a spatio-temporal scheduling approach," in *Proceedings of the INFOCOM 2013*.
- P. Chaparro, et al., "Understanding the Thermal Implications of Multicore Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1055-1065, August 2007.
- M. Ma, S. Gunther, B. Greiner, N. Wolff, C. Deutschle, and Tawfik Arabi, "Enhanced Thermal Management for Future Processors," *IEEE Symposium on VLSI Circuits of Technical Papers*, pp. 201-204, June 2003.
- J. Tschanz, S. Narendra, Y. Ye, B. Bloechel, S. Borkar, and V. De, "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 1838-1845, November 2003.
- D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *International Symposium on Computer Architecture*, June 2000.
- T. Sato and T. Funaki, "Power-Performance Trade-off of a Dependable Multicore Processor," in 13th Pacific Rim International Symposium on Dependable Computing (PRDC), 2007.
- R. Ghosh, V. K. Naik, and K. S. Trivedi, "Power-Performance Trade-offs in IaaS Cloud: A Scalable Analytic Approach," in IEEE/IFIP DSN Workshop on Dependability of Clouds, Data Centers and Virtual Computing Environments (DCDV), 2011.
- E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," *Workshop on Compilers and Operating Systems for Low Power*, 2001.
- J. Chase, D. Anderson, P. Thakur, and A. Vahdat, "Managing Energy and Server Resources in Hosting Centers," *Proceedings of the 18th Symposium on Operating systems Principles SOSP'01*, October 2001.
- Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee, "Managing Server Energy and Operational Costs in Hosting Centers," *SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 303-314, 2005.
- F. Ahmad and T. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," *Architectural Support for Programming Languages and Operating Systems*, 2010.
- B. Müller-Clostermann, "Using G/G/m-Models for Multi-Server and Mainframe Capacity Planning," *ICB Research Report*, no. 16, May 2007.
- R. McGowen, C. A. Poirier, C. Bostak, J. Igonowski, M. Millican, W. H. Parks, and S. Naffziger, "Power and temperature control on a 90-nm Itanium family processor," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 228-236, January 2006.
- T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571-1580, November 2000.
- K. J. Nowka, G. D. Carpenter, E. W. MacDonald, H. C. Ngo, B. C. Brock, K. I. Ishii, T. Y. Nguyen, and J. L. Burns, "A 32-bit PowerPC System-on-a-Chip With Support for Dynamic Voltage Scaling and Dynamic Frequency Scaling," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, November 2002.
- S. Heo, K. Barr, and K. Asanovic, "Reducing power density through activity migration," *International Symposium on Low Power Electronics and Design*, Aug. 2003.
- K. Zhang et al., "SRAM design on 65-nm CMOS technology with dynamic sleep transistor for leakage reduction," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 4, April 2005, pp. 895-901.
- S. Henzler, T. Nirschl, S. Skiathitis, J. Berthold, J. Fischer, P. Teichmann, F. Bauer, G. Georgakos, and D. Schmitt-Landsiedel, "Sleep transistor circuits for fine-grained power switch-off with short power-down times," in *Proc. Int. Solid-State Circuits Conf.*, 2005, pp. 302-303.
- D. Copeland, "64-bit Server Cooling Requirements," *IEEE SEMI-THERM Symposium*, 2005.
- R. Mahajan, C. P. Chiu, and G. Ghysler, "Cooling a Microprocessor Chip," in *Proceedings of the IEEE*, vol. 94, no. 8, Aug. 2006.
- A. G. Agwu Nnanna, "Application of refrigeration system in electronics cooling," *Applied Thermal Engineering*, vol. 26, pp. 18-27, 2006.
- R. C. Chu, R. E. Simons, M. J. Ellsworth, R. R. Schmidt, and V. Cozzolino, "Review of Cooling Technologies for Computer Products," *IEEE Trans. on Device and Material Reliability*, vol. 4, no. 4, pp. 568-585, Dec. 2004.
- P. E. Phelan, V. A. Chiriac, and T. T. Lee, "Current and Future Miniature Cooling Technologies for High Power Microelectronics," *IEEE Trans. on Components and Packaging Technologies*, vol. 25, no. 3, pp. 356-365, Sep. 2002.
- S. Trutassanawin, E. Groll, V. Garimella, and L. Cremaschi, "Experimental Investigation of a Miniature-Scale Refrigeration System for Electronics Cooling," *IEEE Trans. on Components and Packaging Technologies*, vol. 29, no. 3, pp. 678-687, Sep. 2006.
- D. M. Carson, D. C. Sullivan, R. E. Bach, and D. R. Resnick, "The ETA-10 liquid-nitrogen-cooled supercomputer system," *IEEE Trans. Electron. Devices*, vol. 36, no. 8, pp. 1404-1413, Aug. 1989.
- Won Ho Park, Tamer Ali, and C.K. Ken Yang, "Analysis of Refrigeration Requirements of Digital Processors in Sub-ambient Temperatures," *Journal of Microelectronics and Electronic Packaging*, vol. 7, no. 4, 4th Qtr 2010.
- Won Ho Park and C.K. Ken Yang, "Effects of Using Advanced Cooling Systems on the Overall Power Consumption of Processors," accepted for *IEEE*

Transactions on Very Large Scale Integration Systems.

- I. Aller et al., "CMOS Circuit Technology for Sub-Ambient Temperature Operation," Proc. Int. Solid-State Circuits Conf., 2000, pp. 214-215, Feb. 2000.
- J. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective; 2nd ed., 2003.
- A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, April 1992.
- R. Kumar, K. Farkas, N.P. Jouppi, P. Ranganathan, and D.M. Tullsen, "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction," *Proc. Int'l Symp. Microarchitecture*, Dec. 2003.
- N. A. Kurd, J. S. Barkatullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland, "A multigigahertz clocking scheme for Pentium 4 microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 1647-1653, November 2001.

A Mathematical Programming Approach to Multi-cloud Storage

Makhlouf Hadji

Technological Research Institute SystemX, Palaiseau, Saclay, France
{makhlouf.hadji}@irt-systemx.fr

Keywords: Cloud Computing, Distributed Storage, Data Replication, Encryption, Broker, Optimization.

Abstract: This paper addresses encrypted data storage in multi-cloud environments. New mathematical models and algorithms are introduced to place and replicate encrypted data chunks and ensure high availability of the data. To enhance data availability, we present two cost-efficient algorithms based on a complete description of a linear programming approach of the multi-cloud storage problem. Performance assessment results, using simulations, show the scalability and cost-efficiency of the proposed multi-cloud distributed storage solutions.

1 INTRODUCTION

Cloud storage has emerged as a new paradigm to host user and enterprise data in cloud providers and data centers. Cloud storage providers (such as Amazon, Google, etc.) store large amounts of data and various distributed applications (AWS, 2014) with differentiated prices. Amazon provides for example storage services at a fraction of a dollar per Terabyte per month (AWS, 2014)). Cloud service providers propose also different SLAs in their storage offers. These SLAs reflect the different cost of proposed availability guarantees. End-users interested in more reliable SLAs, must pay more, and this leads to cause high costs when storing large amounts of data. The cloud storage providers to attract users do not charge for initial storage or put operations. Retrieval becomes unfortunately a hurdle, a costly process and users are likely to experience data availability problems. A way to avoid unavailability of data is to rely on multiple providers by replicating the data and actually chunk the data and distribute it across the providers so none of them can actually reconstruct the data to protect it from any misuse. This paper aims at improving this type of distributed storage across multiple providers to achieve high availability at reasonable (minimum) storage service costs by proposing new scalable and efficient algorithms to select providers for distributed storage. The objective is to optimally replicate data chunks and store the replicates in a distributed fashion across the providers. In order to protect the data even further, the chunks are encrypted.

1.1 Paper Contributions and Structure

We propose data chunk placement algorithms to tradeoff data availability and storage cost and provide some guarantees on the performance of the distributed storage. We assume end-users involved in PUT (write) and GET (read) operations of data objects stored in an encrypted manner and distributed optimally in different data centers require a specified level of data availability during data retrieval. More specifically, after data encryption and partition operations which consist to split the data into encrypted chunks to be distributed across multiple data centers, our main work focuses on improving and optimizing two operations:

- **Data Chunks Placement Optimization:** through a novel, efficient, scalable algorithm that minimizes placement cost and meets data availability requirements given probabilities of failure (or unavailability) of the storage systems and hence the stored data.
- **Chunk Replication:** to meet a required high level of availability of the data using optimal replication of chunks to reduce the risk of inaccessibility of the data due to data center failures (or storage service degradations).

To realize these objectives, we derive a number of mathematical models to be used by a broker (real or logical broker) to select the storage service providers leading to cost-efficient and reliable data storage. The proposed broker collaborates with the providers having different storage costs and reliability (storage service availability), as depicted in detail in Figure 1.

We assume that the providers propose storage services to the broker and to end-users with same reliability but with different prices (prices for a real broker for instance will be lower than those proposed to end-users).

It is consequently assumed that there exist benefits for a storage service brokerage that optimally distributes encrypted data across the most appropriate providers. Thus, the aim of this paper consists to propose a scalable and polynomial algorithm spanning a cost efficient chunk placement model that can achieve optimal solutions, when guaranteeing high data availability to end-users.

Section 2 presents related work on cloud storage and optimization. In Section 3, we use the well known Advanced Encryption Standard (AES) algorithm (Seungmin et al., 2014) to encrypt end-user data and divide them into $|\mathcal{N}|$ chunks. In the same section, we propose mathematical models to deal with chunk placement and replication in an optimal manner for given server costs and availabilities. Performance assessments and results are reported in Section 4. Conclusion and future work are reported in Section 5.

2 RELATED WORK

Data storage and data replication has received a lot of attention at the data management, distribution and application level since the distribution of original data objects and their replicas is crucial to overall system performance, especially in the cloud environment where data are supposed to be protected and highly available in different data centers. The current literature concerns essentially the cloud storage problem in tandem with replication techniques to improve data availability, but to our knowledge, does not consider data transfer in/out costs, or migration costs, etc. We will nevertheless cite some of the related work even if it can not be directly compared to the proposed algorithms in this paper.

In (Mansouri et al., 2013), authors dealt with the problem of multi-cloud storage with a focus on availability and cost criteria. The authors proposed a first algorithm to minimize replication cost and maximize expected availability of objects. The second algorithm has the same objective subject to budget constraints. However, this paper did not embed security aspects apart from dividing the data into chunks or objects. In our work, we propose to divide data into encrypted chunks, that will be optimally stored and distributed through various data centers with minimum costs while satisfying the QoS required by end-users. Moreover, the proposed algorithm in (Mansouri et al.,

2013) is a simple heuristic without any convergence guarantee to the optimal solution. Our proposed algorithm converges in few seconds to optimal solutions benchmarked by the Bin-Packing algorithm.

In (Thanasis et al., 2012), authors present Scalia, a system to deal with the problem of multi cloud data storage under availability and durability requirements and constraints. The authors note the NP-Hardness of the considered problem, and propose algorithms to solve small instances of the problem. In our work, we propose a new efficient and scalable solution capable of handling large instances in a few seconds. Clearly, the proposed solution in (Thanasis et al., 2012) suffers from scalability challenges to handle on with larger instances, when our algorithms are able to quickly solve large instances of the defined problem.

To avoid failure and achieve higher availability when storing data in the cloud, reference (Yanzhen and Naixue, 2012) proposes a distributed algorithm to better replicate data objects in different virtual nodes instantiated in physical servers. According to the traffic load of all considered nodes, the authors considered three decisions or actions as replicate, migrate, or suicide to better meet end-user requirements and requests. However, the proposed approach consists only in checking the feasibility of migrating a virtual node, performs suicide actions or replicating a copy of a virtual node, without optimizing the system. In our work, we propose optimization algorithms based on a complete description of the convex hull of the defined problem, leading to reach optimal solutions even for large instances.

Reference (Srivastava et al., 2012) proposes a simple heuristic to give stored data greater protection and higher availability by splitting a file (data) into subfiles to be placed in different virtual machines belonging to the physical resources (data centers for example) of one provider or different providers. The paper dealt with PUT and GET operations to distribute and retrieve the required subfiles (data) without encrypting them. The proposed heuristic in (Srivastava et al., 2012) can only reach suboptimal solutions, leading to considerable gaps compared to the optimal solutions. We propose a new scalable and cost efficient solution to deal with the multi-cloud storage problem.

Aiming to provide cost-effective availability and improve performance and load balancing of cloud storage, the authors of reference (Qingsong et al., 2010) propose CDRM as a cost-effective dynamic replication management scheme. CDRM consists in maintaining a minimal number of replica for a given availability requirement, and proposes a replica placement based on the blocking probability of data nodes. Moreover, CDRM allows us to dynamically adjust the

replica number according to changing workload and node capacities. However, the paper focuses only on the relationship between availability and replica number, and there is no proposal to deal with the optimal placement of replicas.

To achieve high performance and reduce data loss when we require storage services in the cloud, different papers in the literature propose various algorithms that are useful only for small instances due to the NP-Hardness of the problem. In (Bonvin et al., 2010), the authors propose a key-value store named Skute, which consists in dynamically allocating the resources of a data cloud to several applications in a cost effective and fair way using game theoretical models. To guarantee cloud object storage performance, the authors of (Jindarak and Uthayopas, 2012) propose a dynamic replication scheme to enhance the workload distribution of cloud storage systems. The authors of (Chia-Wei et al., 2012) conduct a study based on a dynamic programming approach, to deal with the problem of selecting cloud providers offering storage services with different costs and failure probabilities.

Reference (Abu-Libdeh et al., 2010) proposes a distributed storage solution named RACS, to avoid vendor lock-in, reduce the cost of switching providers, and better tolerate provider outages. The authors applied erasure coding (see references (Weatherspoon and Kubiatowicz, 2002), (Li and Li, 2013) and (Rodrigo and Liskov, 2005)) to design the proposed solution RACS. In the same spirit, references (Ford et al., 2010), (Myint and Thu, 2011), (Negru et al., 2013) and (Zhang et al., 2012) addressed the cloud storage problem described above, under different constraints including energy consumption, budget limitation, limited storage capacities, and the availability of the stored data.

In (Varghese and Bose, 2013), authors propose a new solution to guarantee the data integrity when stored in a cloud data center. The proposed solution is based on homomorphic verifiable response and hash index hierarchy. This kind of solutions can be integrated to our work to reinforce data security and privacy for reticent users. An other reference on secured multi cloud storage can be found in (Balasaraswathi and Manikandan, 2014). Authors presented a cryptographic data splitting with dynamic approach for securing information. The splitting approach of the proposed solution is not deeply studied. This may lead to not select cost efficient providers.

3 SYSTEM MODEL

To store encrypted data in multiple DCs belonging to various cloud providers system, while optimizing storage costs and failure probabilities, we separate the global problem into a number of combinatorial optimization sub-problems. To derive the model we make a simplifying assumption regarding the pricing scheme between cloud service providers, the broker and end-users. We assume that the proposed storage price by a service cloud provider to end-users is higher than that proposed to the broker. This can be explained by the large amount of demands that will be required by the broker aggregating the demands of a finite set of end-users seeking to avoid vendor lock-in and higher availability. One can assume that prices proposed by cloud providers are smaller as the volume of data is larger. Note that the broker will guarantee a minimum storage cost meeting end-users requirements, ensuring that the proposed cost to end-users can never exceed a certain threshold.

We first propose to use the well known AES (Advanced Encryption Standard) algorithm (Seungmin et al., 2014) for efficient data encryption. This will generate different encrypted chunks to be distributed in the available storage nodes or data centers. This encryption ensures the confidentiality of the stored data. Moreover, the used solution permits to construct diverse chunks (with small sizes) to facilitate PUT and GET requests as is shown in Figures 1 and 2.

We derive two algorithms to handle encrypted data chunk placement and replication to guarantee data high availability, and storage cost efficiency. This can be summarized as follows:

- **Data Chunk Placement:** The first important objective of our paper consists in guaranteeing the availability of all chunks of stored data by optimally distributing them to a cost-efficient set of selected data centers (see Figure 1). This avoids user lock-in, and reduces the total cost of the storage service. This optimization is performed under end-user or data owner constraints and requirements such as the choice of a minimum number of data centers to be involved in storing the chunks of the data. This can reinforce the availability of data for given data centers failure probabilities.
- **Data Chunk Replication:** After optimally storing the encrypted chunks of a data, we determine a replication algorithm based on bipartite graph theory, to derive optimal solutions of the problem of storing replica chunks. This ensures high data availability since content can be retrieved even if some servers or data centers are not available.

Once all data chunk are placed in different data

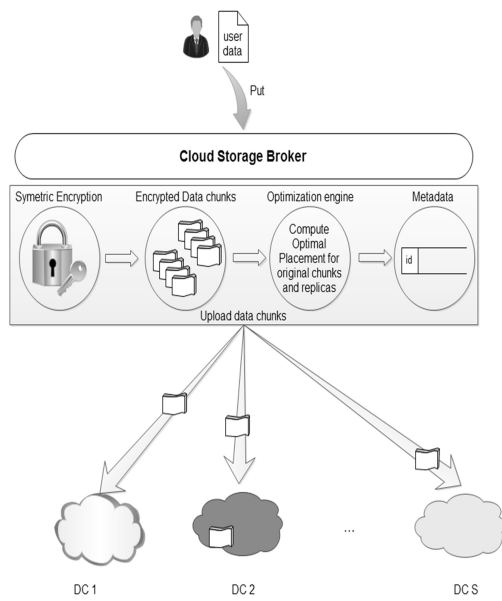


Figure 1: The system model: PUT requests.

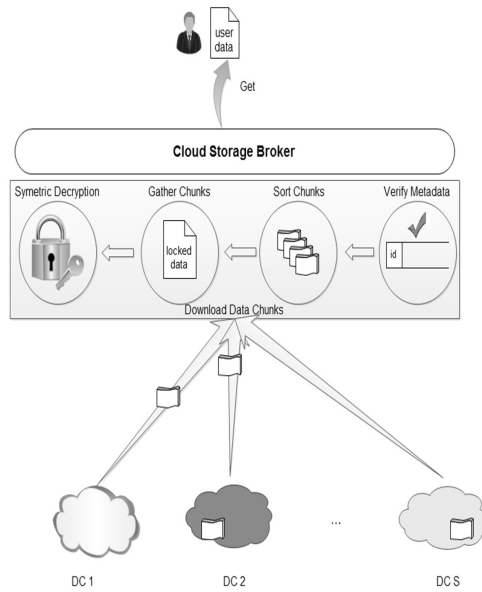


Figure 2: The system model: GET requests.

centers, end-users may solicit the data by GET requests (download data). The broker needs to gather all the data chunks, sort them, decrypt them, and finally deliver the entire data to the end-user. Figure 2 gives more details on GET operations.

In the following, we suppose that each data object (chunk) has r replicas. Finding the optimal number of replicas of each chunk, is not in the scope of this paper. A well-known example on the choice of r is the Google storage solution based on $r = 3$ replicas of each stored data chunk (Ghemawat et al., 2003).

3.1 Data Encryption Algorithm

While consumers have been willing to trade privacy for the convenience of cloud storage services, this is not the case for enterprises and government organizations. To achieve high data security and privacy, we propose to divide the requested user data to store into encrypted chunks. This facilitates PUT and GET requests by considering small files (chunks), and reinforces the security of data (thanks to the encryption) in the same time.

To preserve the confidentiality of data, we seek algorithms that can encrypt and decrypt multiple chunks in a small time. To deal with this problem, we propose to use the symmetric encryption algorithm noted AES for Advanced Encryption Standard (Seungmin et al., 2014). The AES algorithm is a fast solution to handle with large amount of data as it is shown in Figure 3 where three different keys (128 bits, 192 bits and 256 bits) are used to encrypt and decrypt data sizes ranging from 1 Megabyte to 4 Gigabytes in a time interval ranging from 200 seconds to 800 seconds.

The key sizes are chosen by end-users depending on the privacy level of their data. In our proposal, we suppose that the broker proposes an encryption solution in which generated private keys are well kept within end-users with a key size of 128 bits.

Note that more details on the encryption/decryption algorithms used in this paper, can be found in the literature (see for example (Seungmin et al., 2014) and (NIST, 2014)). A deep study of these solutions is not in the scope of this paper.

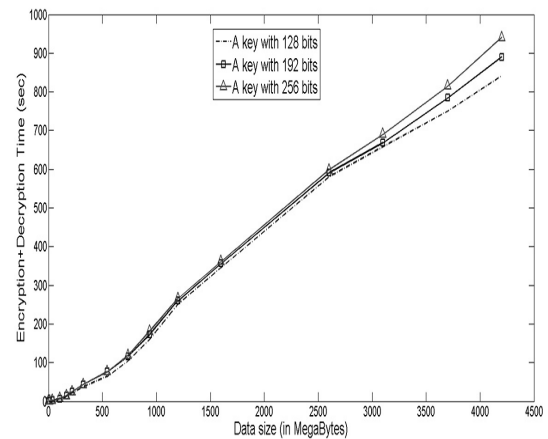


Figure 3: Encryption and decryption's time evolution with data size.

3.2 Data Placement Cost Minimization: b-Matching Formulation

We start data chunks placement model by considering each data D of a user u , as a set of chunks (noted by \mathcal{N}), resulting from the AES algorithm. Let S be the set of all available data centers able to host and store end-user data. We investigate an optimal placement by storing all the chunks in the "best" available data centers. Each cloud provider with a data center $s \in S$ proposes a storage cost per Gigabyte and per month noted by μ_s . This price varies for different reasons: varying demands and workloads, data center reliability, geographical constraints, etc. End-user requests are submitted to the broker which will relay them to cloud service providers, in an encrypted form with optimized storage costs. The broker guarantees end-users high data availability with minimum cost by choosing a set of cloud providers (or DCs) meeting the requirements (see Figure 1 for more details).

In the following, we will address chunks placement optimization model based on different constraints as the probability of failure of a data center or a provider, and a limited storage capacity. Each data center (or provider) has a probability of data availability (according to the number of nines in the proposed SLA), and a failure probability (f) is then equal to $1 - \text{probability of data availability}$. Moreover, the limited storage capacity is given by a storage quota proposed by the provider to the broker according to a negotiated pricing menu.

Our optimization problem is similar to a classical Bin-Packing formulation, in which bins can be represented by the different Data Centers, and the items can be seen as the data chunks. Reference (Korte and Vygen, 2001) has shown a while ago the NP-Hardness of the Bin-Packing problem. Thus, we deduce the complexity (NP-Hardness) of our chunks' placement problem.

For this reason, and the fact that workloads and requests to store data arrive overtime, the broker seeks a dynamic chunk placement solution that will be regularly and rapidly updated to remain cost-effective and ensure data high availability.

Each data chunk $i \in \mathcal{N}$ has a certain volume noted by v_i . We graphically represent the storage of a chunk i in a data center k as an edge $e = (i, k)$ (with the initial extremity ($i = I(e)$) of e corresponding to a chunk, and the terminal extremity ($k = T(e)$) of e representing the data center (see Figure 4).

Based on this configuration, one can construct a new weighted bipartite graph $G = (\mathcal{N} \cup S, E)$, where \mathcal{N} is the set of vertices representing encrypted chunks to be stored, and S is the set of all available data cen-

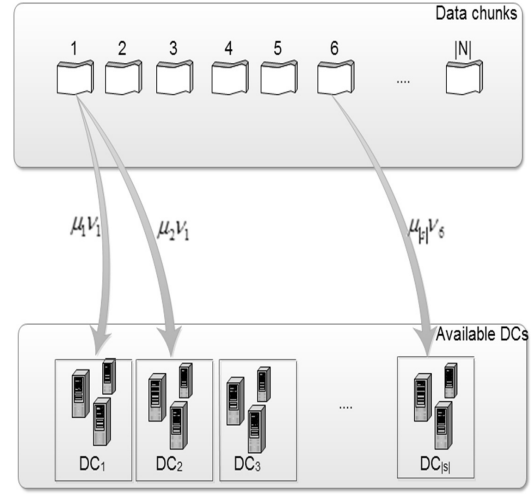


Figure 4: Complete bipartite graph construction.

ters (see Figure 4). E is the set of weighted edges between \mathcal{N} and S constructed as described: there is an edge $e = (i, k)$ between each encrypted chunk i and each available data center k , and the weight of e is given by $\mu_k v_i$.

We now introduce the well known "minimum weight b-matching problem" to build a combinatorial optimization solution. The b-matching is a generalization of the minimum weight matching problem and can be defined as follows (see (Korte and Vygen, 2001) for more details):

Definition Let G be an undirected graph with integral edge capacities: $u : E(G) \rightarrow \mathbb{N} \cup \infty$ and numbers $b : V(G) \rightarrow \mathbb{N}$. Then a b-matching in G is a function $f : E(G) \rightarrow \mathbb{N}$ with $f(e) \leq u(e)$, $\forall e \in E(G)$, and $\sum_{e \in \delta(v)} f(e) \leq b(v)$ for all $v \in V(G)$.

In the above, $\delta(v)$ represents the set of incident edges of v . To simplify notation, with no loss in generality, we use E and V for the edges and vertices of G . That is we drop the G in $E(G)$ and $V(G)$.

From the definition, finding a minimum weight b-matching in a graph G consists in identifying f such that $\sum_{e \in E} \gamma_e f(e)$ is minimum, where γ_e is an associated cost to edge e . This problem can be solved in polynomial time since the full description of its convex hull is given in (Korte and Vygen, 2001).

Proposition 3.1. *Let $G = (\mathcal{N} \cup S, E)$ be a weighted complete bipartite graph built as described in Figure 4. Then, finding an optimal chunk placement solution is equivalent to an uncapacitated ($u \equiv \infty$) minimum weight b-matching solution, where $b(v) = 1$ if $v \in \mathcal{N}$ (v is a chunk) and for all vertices $v \in S$, we put $b(0) = 0$, and for $v \geq 1$, we have*

$$b(v) = \left\lceil \frac{|\mathcal{N}| - \sum_{k=0}^{v-1} b(k)}{\beta} \right\rceil \quad (1)$$

where β is the minimum number of data centers to be used to store the data chunks. This parameter is required by end-users to avoid vendor lock-in.

To mathematically formulate our model, we associate a real decision variable x_e to each edge e in the bipartite graph. As shown in Figure 4, each edge links a chunk to a data center. After optimization, if the decision is $x_e = 1$ then chunk i ($i = I(e)$ initial extremity) will be stored in data center j ($j = T(e)$ terminal extremity). Since the solution of a b-matching problem is based on solving a linear program, an integer solution of the minimum weight b-matching is found in polynomial time. This is equivalent to the optimal solution of the chunk placement problem described in this section.

According to the storage costs listed previously and by defining the probability of failure of a data center (or a provider) noted by f , we assign each chunk to the best data center with minimum cost. We note by $Cost_{plac}$ the total cost of placing $|\mathcal{N}|$ chunks in an optimal manner. We can formulate the objective function as follows:

$$\min Cost_{plac} = \sum_{e \in E, e=(i,j)} \left(\frac{\mu_j}{1-f_j} v_i \right) x_e \quad (2)$$

where v_i is the volume of chunk i , and $(1-f)$ is the probability of data center availability (or provider availability).

This optimization is subject to a number of linear constraints. For instance, the broker has to consider the placement of all data chunks, and each chunk will be assigned to one and only one data center (the chunk replication problem will be discussed in the next section). This is represented by (3):

$$\sum_{e \in \delta(v)} x_e = 1, \forall v \in \mathcal{N} \quad (3)$$

Each data center s has a capacity Q_s . This leads to the following constraints:

$$\sum_{C=1}^{|\mathcal{N}|} v_C x_{Cs} \leq Q_s, \forall s \in \mathcal{S} \quad (4)$$

According to end-user requirements and to guarantee high data availability, chunks will be deployed in different data centers to avoid vendor lock-in. This is given by the following inequality:

$$\sum_{C=1}^{|\mathcal{N}|} x_{Cs} \leq b(s), \forall s \in \mathcal{S} \quad (5)$$

Using the b-matching model with constraints (4), enables the use of the complete convex hull of b-matching and makes the problem easy in terms of combinatorial complexity theory.

Reference (Korte and Vygen, 2001) gives a complete description of the b-matching convex hull expressed in constraints (3), (4) and (5). These families of constraints are reinforced by *blossom inequalities* to get integer optimal solutions with continuous variables:

$$\sum_{e \in E(G(A))} x_e + x(F) \leq \left\lfloor \frac{\sum_{v \in A} b_v + |F|}{2} \right\rfloor, \forall A \in \mathcal{N} \cup \mathcal{S}, \quad (6)$$

where $F \subseteq \delta(A)$ and $\sum_{v \in A} b_v + |F|$ is odd, and $\delta(A) = \sum_{i \in A, j \in A} x_{(ij)} \cdot E(G(A))$ represents a subset of edges of the subgraph $G(A)$ generated by a subset of vertices A . An in depth study of blossom constraints (6) is out of the scope of this paper, but more details can be found in (Grotschel et al., 1985).

Based on the bipartite graph G , we constructed a polynomial time approximation scheme of the data chunks placement problem by identifying the b-matching formulation. The blossom constraints (6) are added to our model to get *optimal integer solutions* of the *placement problem* whose model is finally given by:

$$\begin{aligned} \min Cost_{plac} &= \sum_{s=1}^{|\mathcal{S}|} \sum_{C=1}^{|\mathcal{N}|} \frac{\mu_s}{1-f_s} v_C x_{Cs} \\ S.T. : & \begin{cases} \sum_{s=1}^{|\mathcal{S}|} x_{Cs} = 1, \forall C \in \mathcal{N} \\ \sum_{C=1}^{|\mathcal{N}|} v_C x_{Cs} \leq Q_s, \forall s \in \mathcal{S} \\ \sum_{C=1}^{|\mathcal{N}|} x_{Cs} \leq b(s), \forall s \in \mathcal{S} \\ \sum_{e \in E(G(A))} x_e + x(F) \leq \left\lfloor \frac{\sum_{v \in A} b_v + |F|}{2} \right\rfloor, \forall A \in \mathcal{N} \cup \mathcal{S} \\ F \subseteq \delta(A), \sum_{v \in A} b_v + |F| \text{ is odd} \\ x_{Cs} \in \mathbb{R}^+, \forall C \in \mathcal{N}, \forall s \in \mathcal{S} \end{cases} \end{aligned} \quad (7)$$

The variables and constants used in the final model are summarized as follows:

3.3 Data Replication Algorithm

To enhance performance and availability of end-user stored data, we propose a replication model of data chunks depending on data center failure probabilities, and expected availability (noted by A_{expec}) required by each user. The objective consists in finding the optimal trade-off between data center availability and storage costs. This leads to avoiding expensive data centers with high failure probability.

We assume that each data chunk is replicated r times, and reconstituting a file data needs to get one

Table 1: Variables and constants of the model.

Variables	Meaning
\mathcal{N}	set of data chunks
\mathcal{S}	set of data centers
v_C	volume of a data chunk C
μ_j	storage cost per Gigabyte/month of provider j
x_e	real variable indicating if e is solicited or not
b_v	upper bound of the degree of v
$\delta(A) =$	$\sum_{i \in A, j \in A} x_{(ij)}$
$\delta(v)$	set of incident edges to v
β	minimum number of providers

copy of all chunks (i.e. $|\mathcal{N}|$ chunks among $r \times |\mathcal{N}|$ are necessary to reconstruct a data). Figure 5 gives more details and shows chunks replication procedure.

It is important to note that initially, each encrypted chunk will be replicated by the selected hosting providers within their data centers, and the broker can reinforce this mechanism by proposing to add more replicas guaranteeing higher data availability.

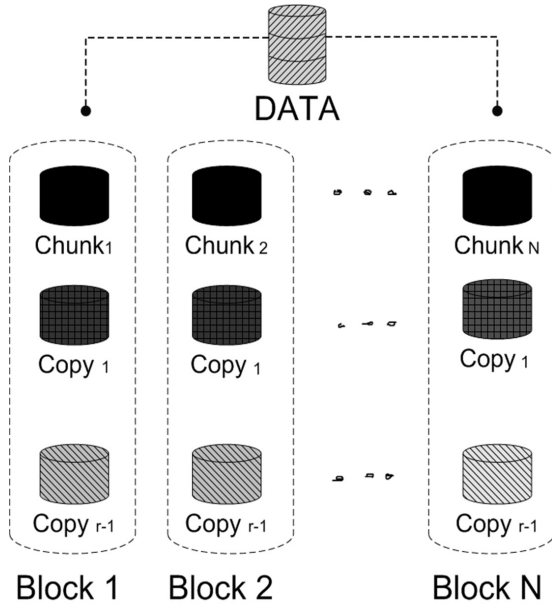


Figure 5: Data replication.

In the following, we would like to replicate $|\mathcal{N}|$ chunks into $|\mathcal{S}|$ data centers according to various costs (storage costs) and performance requirements such as the data availability. We suppose that $\mathcal{S} = \{s_1, s_2, \dots, s_{|\mathcal{S}|}\}$ and for the sake of simplicity (due to the problem NP-Hardness), we suppose w.l.o.g. each data center has a large amount of storage resources able to host data chunks and replicas. We associate

each data center $s \in \mathcal{S}$ with a probability of failure f_s .

We suppose (as cited above) that each data chunk C ($C = 1, |\mathcal{N}|$) has r replicas to place in r data centers that do not contain the chunk C . Thus we ask the following question: *How do we replicate data chunks through available data centers so that the total cost of storage is optimal (minimal) and data availability is maximal?*

Thus, for each chunk C , the problem consists in selecting a subset Φ_C of r available data centers that do not contain C , leading to a minimum storage cost and a high probability of data availability.

We note by $P(C)$ the probability of chunk C availability (respect. $P(\bar{C})$ is the probability of non-availability of a chunk C). $P(D)$ is the probability of data availability (respect. $P(\bar{D})$ is the probability of non-availability of data D). Note that a chunk C is not available if all of its copies are not available (see Figure 5). In other words, a block in Figure 5 with r replicas is non available if all of the data centers storing this block are non available. By supposing the data centers are independent, we get the following proposition:

Proposition 3.2. $P(\bar{C}) = \prod_{s \in \Phi_C} f_s$

Proof.

$$\begin{aligned} P(\bar{C}) &= P(\bar{C}_1 \text{ and } \bar{C}_2 \text{ and } \dots \text{ and } \bar{C}_r) \\ &= P(\bar{C}_1) \times P(\bar{C}_2) \times \dots \times P(\bar{C}_r) \\ &= \prod_{s \in \Phi_C} f_s \end{aligned}$$

Proposition 3.3. $P(D) = \prod_{C=1}^{|\mathcal{N}|} (1 - \prod_{s \in \Phi_C} f_s)$

Proof. A data D with $r \times |\mathcal{N}|$ chunks, is entirely available if all chunks are available. According to Proposition (3.2), the probability of data file availability (i.e. $P(D)$) is then given by:

$$\begin{aligned} P(D) &= \prod_{C=1}^{|\mathcal{N}|} P(C) \\ &= \prod_{C=1}^{|\mathcal{N}|} \left(1 - \prod_{s \in \Phi_C} f_s \right) \end{aligned}$$

The QoS requirement for end-users is presented by the data availability. This is noted by A_{expect} (as used in (Mansouri et al., 2013) for example). Thus, to meet end-user QoS requirement, the broker should replicate each D in a selected sub-set of data centers that satisfies:

$$\prod_{C=1}^{|\mathcal{N}|} \left(1 - \prod_{s \in \Phi_C} f_s \right) \geq A_{expect} \quad (8)$$

We derive a mathematical model to efficiently reduce the replication costs noted by $Cost_{rep}$, under the QoS requirements described by the inequality (8). As the number of replicas of each chunk is supposed to be r , we seek an optimal sub-set of data centers of size r to store the replicas of each chunk. Moreover, our solution should not put all the chunks within the same data center to avoid vendor lock-in. Thus, in the following, we address a mathematical optimization model to efficiently replicate all the chunks of a data D .

$$\begin{aligned} \min_{\Phi_C} Cost_{rep} &= \sum_{C=1}^{|\mathcal{N}|} \sum_{s \in \Phi_C} \mu_s v_C \\ S.T. : & \\ \begin{cases} \prod_{C=1}^{|\mathcal{N}|} (1 - \prod_{s \in \Phi_C} f_s) \geq A_{expect}, & ; \\ |\Phi_C| = r, & \forall C = 1, |\mathcal{N}|; \end{cases} \end{aligned} \quad (9)$$

To solve the model (9), we can resort to dynamic programming approach as the objective function of (9) is separable and monotone. As these methods resort to recursion technique, they can prove to be expensive in certain cases due to the exponential number of data centers subsets to enumerate. For this reason, and for the sake of scalability, we prefer to address a simple, scalable and succinct algorithm to reach near optimal solutions for large instances in few seconds.

Solving the model (9) is equivalent to find a subset of data centers able to host chunks in a cost efficient manner, and that satisfies the requirement (8). We propose a simple and scalable algorithm to solve (9) in few seconds for large number of data centers and data chunks. Without loss of generality, we assume that minimizing a function Z is approximately equivalent to minimize $\ln(Z)$. Thus, for each chunk C , we seek a subset of data centers that minimizes $\ln(\prod_{s \in \Phi_C} f_s)$. This is equivalent to minimize $\sum_{s \in \Phi_C} \ln(f_s)$. Moreover, We construct a new bipartite graph $G_2 = (V_2 \cup S_2, E_2)$, where V_2 is the set of chunks to be stored and S_2 is the set of all available data centers (see Figure 6). E_2 is the set of weighted edges between the two parts of vertices of G_2 . There is an edge between each chunk C and each data center s (not containing a copy of chunk C) with a weight given by $\ln(f_s)$. If a data center s has already stored a copy of chunk C , then the weight of the edge (C, s) is equal to 2. Figure 6 gives more details.

From graph G_2 , we identify a minimum weight b-matching with a given vector b as follows :

- for each $v \in V_2$, degree of v is equal to $b(v) = r - 1$,
- the degree of each vertex $v \in S_2$ is equal to $b(v)$ given by (1).

To summarize, we give the following algorithm,

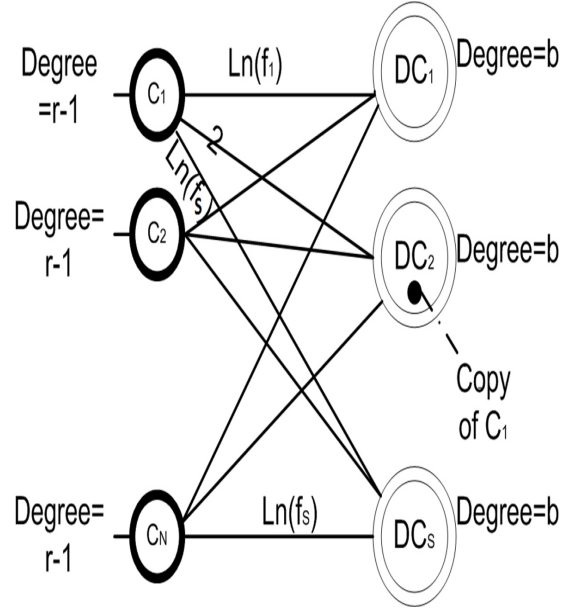


Figure 6: New bipartite graph G_2 to replicate chunks.

leading to find the best subset of data centers to replicate all the chunks in a cost efficient manner, verifying condition (8).

Algorithm 1: Data replication algorithm.

- Step 0:** Construct the bipartite graph G_2 (see Figure 6);
- Step 1:** Compute a b-Matching with a minimum cost solution using the vector b ;
- Step 2:** Check if (8) is satisfied;
- Step 3:** If (8) is not satisfied, GOTO Step 0, by incrementing the degrees of vertices in S_2 ;
-

The algorithm 1 is deployed to replicate efficiently $r - 1$ copies of each chunk C of a data D .

3.4 Data Chunk Splitting

In this section, we discuss the rational number of chunks ($|\mathcal{N}^*|$) to be used to split the data according to data center failure probabilities (f_s for a DC s), number of replicas (r) of each chunk, and the data availability expected by end-users (A_{expect}).

According to Proposition (3.3), we seek a rational number of encrypted chunks to get after splitting the data when satisfying end-users QoS represented by data availability A_{expect} . We get the following inequality :

$$P_D = \prod_{C=1}^{|\mathcal{N}|} P_C = \prod_{C=1}^{|\mathcal{N}|} \left(1 - \prod_{s \in \Phi_C} f_s \right) \geq A_{expect} \quad (10)$$

As $A_{expect} < 1$ and $\prod_{C=1}^{|\mathcal{N}|} (1 - \prod_{s \in \Phi_C} f_s) < 1$, inequality (10) leads to the following one:

$$\ln \left(\prod_{C=1}^{|\mathcal{N}|} \left(1 - \prod_{s \in \Phi_C} f_s \right) \right) \leq \ln(A_{expect}) \quad (11)$$

We also note that for each chunk indexed by C , we have r replicas and then $|\Phi_C| = r$. For the sake of simplicity, we also suppose that the failure probability of each data center is close to the average failure probability given by \bar{f} . This allows us to deduce :

$$\prod_{s \in \Phi_C} f_s = (\bar{f})^r \quad (12)$$

And following inequality (11), we get:

$$|\mathcal{N}| \times \ln(1 - \bar{f}^r) \leq \ln(A_{expect}) \quad (13)$$

According to (13), we deduce the number of data chunks as follows :

$$|\mathcal{N}^*| \geq \frac{\ln(A_{expect})}{\ln(1 - \bar{f}^r)} \quad (14)$$

4 NUMERICAL RESULTS

To evaluate and assess performance, our algorithms have been implemented and evaluated using simulations and an experimental platform managed by an instance of OpenStack (Openstack, 2014). The linear programming solver CPLEX (CPLEX, 2014) was used to derive the b-matching solution and the Bin-Packing solution used to benchmark our heuristic.

As our goal in this paper is to analyze and discuss the applicability and the interest of storage brokering services in interaction with multiple data centers or cloud providers, we devote some numerical results to cross validating our proposed algorithms and assessing their cost efficiency and scalability for large data sizes. It is obvious to remark that the Bin-Packing model used to place data chunks invokes a branch and bound approach leading to explore the entire space of all the existing solutions. This leads to find "optimal" solutions for small data sizes serving as a benchmark for other approaches and algorithms. As the data size increases, the optimal solution for data chunk placement can only be found in exponential time. Thus, for large data, we resort to our heuristic solution based on graph theory and the b-matching approach.

In addition, our performance evaluation seeks to identify the limits of the discussed problem in terms of algorithmic complexity, and its suitability for optimizing real life instances. We will also determine

the gap between the suboptimal heuristic solutions and the optimal solution provided by the Branch and Bound model when it can be reached in acceptable times.

4.1 Simulation Environment

The proposed algorithms in this paper were evaluated using a 1.70 GHz server with 6 GBytes of available RAM. We used data files with sizes ranging from 100 Megabytes to 4 Gigabytes. These data were stored in a distributed manner over a number of available data centers or providers ranging from 10 to 50. We associate with each data center, a data price per Gigabyte and per month, uniformly generated between 0 \$ and 1 \$. Each data is splitting multiple chunks and each chunk size is equal to 1 Megabyte. This configuration leads to construct a full mesh bipartite graph as described above. The number of generated bipartite graphs was set to 100 in our simulations yielding an average value reported in the following curves and tables. Without loss of generality, we suppose that each data center has an unlimited storage capacity.

In addition, we also used a platform of 20 servers running a Havana instance of OpenStack (Openstack, 2014) in a multi-node architecture. Each server (assimilated to a data center in real life) proposes Swift containers (Swift, 2014) to store data chunks. We associate a storage cost to each container (or DC) as described above. It is important to note that we used Swift API only to guarantee PUT and GET operations from and to the broker by intercepting and hosting encrypted chunks, without considering Swift replication policy. To improve our broker functionalities, we will add an S3 compatible interface allowing end-users to request the broker storing their data within Amazon S3.

4.2 Performance Evaluation

The first experiment consists in comparing the Bin-Packing and b-Matching (heuristic) approaches in terms of delay to derive the optimal and suboptimal solutions, respectively. We report different scenarios in Table 2, varying the number of data centers able to store end-users data (from 12 to 700 DCs), and the number of chunks ranging from 50 chunks to 2000 chunks, which is equivalent to store data size of 50 Megabytes to 2000 Megabytes, as each chunk is of 1 Megabyte.

To get a better grasp of the relative performance of the two algorithms, we generate 100 runs and take the average value of each instance, as reported.

The performance of the heuristic algorithm com-

pared to the optimal solution is represented by a gap defined as the percentage difference between the cost of the optimal and the heuristic solutions:

$$Gap(\%) = 100 \times \frac{bM_{sol} - BP_{sol}}{BP_{sol}} \quad (15)$$

where BP_{sol} is the cost of the exact solution provided by the Bin-Packing algorithm (to use as a reference or benchmark) and bM_{sol} is the cost of the b-Matching solution.

Table 2 reports the results of the evaluation and clearly shows the difficulty to reach optimal solutions using the Bin-Packing (Branch and Bound) algorithm whose resolution times become prohibitive for the scenarios of a data file of 2 Gigabytes to be distributed on a selected set of data centers among 300, 500 and 700 providers or data centers. Our heuristic solution performs close to optimal with *Gap* not exceeding 6% for the evaluated cases. More specifically the gap is in the interval [0.65%; 5.93%].

The results shown in Table 2 illustrate the difficulty to optimally solve the data chunks placement problem (see case of a data of 50 Mb with 25 DCs). At the same time, they demonstrate that the heuristic approach can find good and near-optimal solutions whose cost is quite close to the optimum (see case of data with 2000 MB and 700 DCs). Our algorithm provides an excellent trade-off between convergence time, optimality, scalability and cost. With respect to convergence time as seen in the third column of Table 2, it converges in a few seconds for the scenario with 2000 chunks and 700 DCs (54 secs compared to more than 3 hours for Bin-Packing).

To get a better grasp of the relative performance of the two algorithms used in this paper, a data file of 100 Megabytes is used and split into 100 encrypted chunks to be stored in a number of data centers ranging from 20 to 200. Figure 7 shows the characteristics of the algorithms. The b-matching algorithm achieves the best cost performance since it has consistently incurred the smallest cost, very close to the Bin-Packing which does not scale (as seen in Table 2). Exceptionally, one can remark in Figure 7 (the scenario with 20 to 40 available DCs), the cost found by the b-Matching is slightly lower than the cost of the Bin-Packing leading to negligible SLA violations caused by the quality of the upper bound given by equation (1) which should be enhanced in a future work. This is explained by the difficulty to optimally store and place all the data chunks in different data centers.

Another experiment consists in evaluating the proposed heuristic solution to determine the trade-off between storage cost and data availability. We associate with each user a required percentage of its data availability, denoted by A_{expect} . We reformulate A_{expect} in

Table 2: Encrypted data chunks placement: b-Matching algorithm performances.

$ \mathcal{N} $	$ \mathcal{S} $	b-Matching Time (sec)	Bin-Packing Time (sec)	Gap (%)
50	12	0.15	0.16	2.24
	25	0.15	0.16	5.93
	40	0.17	0.18	2.06
100	25	0.17	0.20	3.08
	50	0.18	0.20	0.65
	75	0.20	0.22	2.98
500	100	1.10	2.11	1.94
	250	1.27	3.68	4.37
	350	1.33	4.20	0.97
1000	200	7.22	12.7	5.36
	400	8.5	17.5	1.37
	700	10.4	22.6	3.66
2000	300	30.7	> 3H	1.45
	500	45.2	> 3H	4.3
	700	54.8	> 3H	0.81

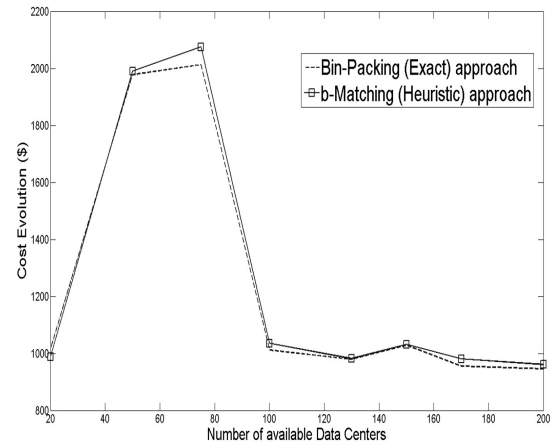


Figure 7: Storage cost gap.

terms of the number of nines required by a user. We simulated a cloud storage market of 15 data centers belonging to different providers having different failure rates. For example, Amazon S3 (AWS, 2014) offers two levels of services: "Standard Storage" which has 11 nines of storage availability for 0.03\$ per Gigabyte per month, while "Amazon S3 Reduced Redundancy Storage (RRS)" has 4 nines of data availability for 0.024\$ per GB per month. The simulated market is summarized in Table 3.

We consider a user data of 100 Gigabytes, and we investigate four methods to find the trade-off between a maximum data availability and a minimum price (cost). We use the following scenarios:

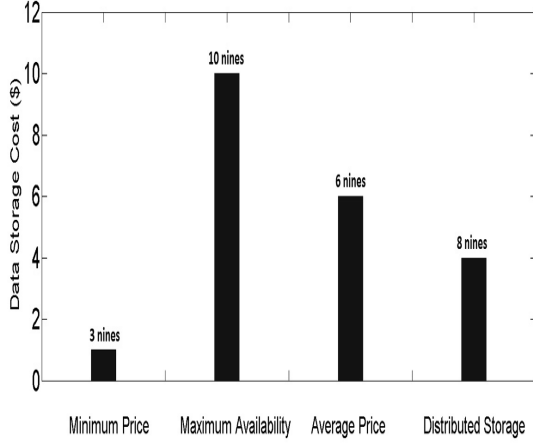


Figure 8: Data storage cost and availability trade-off.

- **Minimum Price:** A user selects simply the cheapest provider in the market (Provider 15 proposing a price of \$0.01 per Gigabyte and per month in Table 3) without concerns on data availability (3 nines). Following this approach, the data will be stored with a total minimum costs of 1\$ and a weak data availability (3 nines in Figure 8). Moreover, the user is locked-in within one cloud provider with a weak data availability. This can lead to disrupting services and loss of data.
- **Maximum Availability:** A user selects the provider with high availability in the simulated market (Provider 1 with 10 nines in Table 3). According to pricing proposal of Provider 2, the total storage cost is higher than the cost of the first scenario (10 \$ in Figure 8). This may also lead to users' lock-in within the same provider.
- **Average Price:** In this case, we use the average price of the market, and we store the data within the provider with equivalent price (Provider 9 with 0.06\$ per Gigabyte per month in Table 3). The total data cost in this case is equal to 6\$ with 6 nines of data availability (according to the proposal of Provider 6). This scenario presents higher data availability than scenario 1 with a considerable cost increase. In this case, we also solicited one provider to store the data, which may cause user lock-in.
- **Distributed Storage:** We used our proposed approach (Algorithm 1) to find the trade-off between data availability and price. As depicted in Figure 8, our solution reaches a maximum availability of 8 nines with a minimum cost of 4\$. This is due to data distribution over a set of selected providers with high availability and reasonable prices, avoiding user lock-in at the same time.

Table 3: Storage market costs and data availability.

Providers	Price (\$/GB/month)	Data Availability
Prov 1	0.1	99.99999999%
Prov 2	0.095	99.99999995%
Prov 3	0.09	99.9999999%
Prov 4	0.085	99.9999995%
Prov 5	0.08	99.999999%
Prov 6	0.075	99.999995%
Prov 7	0.07	99.99999%
Prov 8	0.065	99.99995%
Prov 9	0.06	99.9999%
Prov 10	0.055	99.9995%
Prov 11	0.05	99.999%
Prov 12	0.04	99.995%
Prov 13	0.03	99.99%
Prov 14	0.02	99.95%
Prov 15	0.01	99.9%

A last experiment consists in evaluating the behavior of the number of replicas (noted by r) of each chunk with the evolution of the number of data chunks ($|\mathcal{X}^*|$) identified in (14) for example. We supposed that the average value of data centers failure probability is equal to 10^{-3} , when the expected data availability required by cloud consumers is equal to 99.9999%.

Figure 9 depicts the evolution of r for different chunks number ranging from 1 to 60. Thus, we remark that for a number of chunks $|\mathcal{X}^*| \leq 43$, the number of required replicas is equal to 2, and for $|\mathcal{X}^*| \geq 44$ chunks, the number of replicas converges to 3 and there is no need to replicate more even for larger number of chunks. This may lead to store large data volumes with reduced costs when satisfying the required QoS (data availability). Note that this result is very similar than that determined by the Google File System solution (Ghemawat et al., 2003).

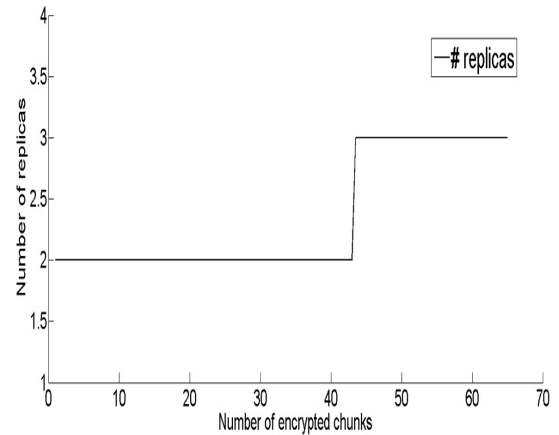


Figure 9: Data chunks replication behavior.

5 SUMMARY AND FUTURE WORK

In this paper we considered an encrypted and distributed solution allowing to store users' data in different providers' data centers offering storage services with different prices and SLAs. To eliminate user lock-in and to liberate user data from a unique provider, we proposed a new efficient and scalable solution based on b-Matching theory to optimize the storage cost and the data failure at the same time. The b-Matching algorithm works in tandem with a replication solution allowing to efficiently increase the data availability of end-users. This replication algorithm is based on a simple and fast approach giving near optimal solutions even for large problem instances.

In future work, we will reinforce our mathematical model of data chunk placement based on b-Matching theory, to consider network constraints when users are involved in PUT and GET operations. This may lead cloud consumers to combine requests of compute (as EC2 instances (EC2, 2014)) services with storage services (as Google Drive (Google, 2014)) at the same time. Thus, we will reinforce our broker's functionalities to give cloud consumers various means to consume proposed cloud resources in a more secure manner with reduced cost.

REFERENCES

- Abu-Libdeh, H., Princehouse, L., and Weatherspoon, H. (2010). Racs: A case for cloud storage diversity. In *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC '10*, pages 229–240, New York, NY, USA. ACM.
- AWS (2014). <http://aws.amazon.com/fr/s3/pricing/>.
- Balasaraswathi, V. and Manikandan, S. (2014). Enhanced security for multi-cloud storage using cryptographic data splitting with dynamic approach. In *Advanced Communication Control and Computing Technologies (ICACCT)*, 2014 International Conference on, pages 1190–1194.
- Bonvin, N., Papaioannou, T., and Aberer, K. (2010). A self-organized, fault-tolerant and scalable replication scheme for cloud storage. In *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC '10*, pages 205–216, New York, NY, USA. ACM.
- Chia-Wei, C., Pangfeng, L., and Jan-Jan, W. (2012). Probability-based cloud storage providers selection algorithms with maximum availability. In *Parallel Processing (ICPP)*, 2012 41st International Conference on, pages 199–208.
- CPLEX (2014). <http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>.
- EC2 (2014). <http://aws.amazon.com/fr/ec2/>.
- Ford, D., Labelle, F., Popovici, F., Stokely, M., Truong, V., Barroso, L., Grimes, C., and Quinlan, S. (2010). Availability in globally distributed storage systems. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*.
- Ghemawat, S., Gobioff, H., and Leung, S. (2003). The google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43.
- Google (2014). drive.google.com/.
- Grotschel, M., Lovsz, L., and Shrijver, A. (1985). Geometric algorithms and combinatorial optimization. *Springer*.
- Jindarak, K. and Uthayopas, P. (2012). Enhancing cloud object storage performance using dynamic replication approach. In *Parallel and Distributed Systems (ICPADS)*, 2012 IEEE 18th International Conference on, pages 800–803.
- Korte, B. and Vygen, J. (2001). Combinatorial optimization: theory and algorithms. *Springer*.
- Li, J. and Li, B. (2013). Erasure coding for cloud storage systems: A survey. *Tsinghua Science and Technology*, 18(3):259–272.
- Mansouri, Y., Toosi, A., and Buyya, R. (2013). Brokering algorithms for optimizing the availability and cost of cloud storage services. In *Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science - Volume 01, CLOUDCOM '13*, pages 581–589, Washington, DC, USA. IEEE Computer Society.
- Myint, J. and Thu, N. T. (2011). A data placement algorithm with binary weighted tree on pc cluster-based cloud storage system. In *Cloud and Service Computing (CSC)*, 2011 International Conference on, pages 315–320.
- Negru, C., Pop, F., Cristea, V., Bessisy, N., and Jing, L. (2013). Energy efficient cloud storage service: Key issues and challenges. In *Emerging Intelligent Data and Web Technologies (EIDWT)*, 2013 Fourth International Conference on, pages 763–766.
- NIST (2014). Announcing the advanced encryption standard (aes).
- Openstack (2014). <https://www.openstack.org/>.
- Qingsong, W., Veeravalli, B., Bozhao, G., Lingfang, Z., and Dan, F. (2010). Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster. In *Cluster Computing (CLUSTER)*, 2010 IEEE International Conference on, pages 188–196.
- Rodrigo, R. and Liskov, B. (2005). High availability in dhds: Erasure coding vs. replication. In *Peer-to-Peer Systems IV 4th International Workshop IPTPS 2005*, Ithaca, New York.
- Seungmin, K., Bharadwaj, V., and KhinMiMi, A. (2014). Espresso: An encryption as a service for cloud storage systems. *Monitoring and Securing Virtualized Networks and Services*, pages 15–28.
- Srivastava, S., Gupta, V., Yadav, R., and Kant, K. (2012). Enhanced distributed storage on the cloud. In *Computer and Communication Technology (ICCCCT)*, 2012 Third International Conference on, pages 321–325.
- Swift (2014). <http://docs.openstack.org/developer/swift/>.
- Thanasis, G. P., Bonvin, N., and Aberer, K. (2012). Scalia: An adaptive scheme for efficient multi-cloud storage.

- In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 20:1–20:10, Los Alamitos, CA, USA. IEEE Computer Society Press.
- Varghese, L. and Bose, S. (2013). Integrity verification in multi cloud storage. In *Proceedings of International Conference on Advanced Computing*.
- Weatherspoon, H. and Kubiatowicz, J. (2002). Erasure coding vs. replication: A quantitative comparison. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 328–338, London, UK, UK. Springer-Verlag.
- Yanzhen, Q. and Naixue, X. (2012). Rfh: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage. In *Parallel Processing (ICPP), 2012 41st International Conference on*, pages 520–529.
- Zhang, Q., Xue-zeng, P., Yan, S., and Wen-juan, L. (2012). A novel scalable architecture of cloud storage system for small files based on p2p. In *Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on*, pages 41–47.

Cloud Provider Transparency

A View from Cloud Customers

Daniela S. Cruzes and Martin Gilje Jaatun

SINTEF – ICT, Postboks 4760 Sluppen, 7465 Trondheim, Norway
{danielac, martin.g.jaatun}@sintef.no

Keywords: Cloud, Provider, Customer, Security, Privacy, Accountability, Transparency.

Abstract: A major feature of public cloud services is that data are processed remotely in unknown systems that the users do not own or operate. This context creates a number of challenges related to data privacy and security and may hinder the adoption of cloud technology. One of these challenges is how to maintain transparency of the processes and procedures while at the same time providing services that are secure and cost effective. This paper presents results from an empirical study in which the cloud customers identified a number of transparency requirements to the adoption of cloud providers. We have compared our results with previous studies, and have found that in general, customers are in synchrony with research criteria for cloud service provider transparency, but there are also some extra pieces of information that customers are looking for.

1 INTRODUCTION

Cloud computing, which allows for highly scalable computing and storage, is increasing in importance throughout information technology (IT). Cloud computing providers offer a variety of services to individuals, companies, and government agencies, with users employing cloud computing for storing and sharing information, database management and mining, and deploying web services, which can range from processing vast datasets for complicated scientific problems to using clouds to manage and provide access to medical records (Paquette, 2010).

Several existing studies emphasize the way technology plays a role in the adoption of cloud services, and most of these studies conclude that the most important challenges are related to security, privacy and compliance (Kuo, 2011), (Gavrilov and Trajkovik, 2012), (AbuKhoua et al., 2012), (Rodrigues et al. 2013), (Ahuja et al. 2012). Cloud service users may hand over valuable and sensitive information to cloud service providers without an awareness of what they are committing to or understanding of the risks, with no control over what the service does with the data, no knowledge of the potential consequences, or means for redress in the event of a problem.

In the European A4Cloud research project (<http://a4cloud.eu>), our focus is on accountability as the most critical prerequisite for effective governance and control of corporate and private data processed by

cloud-based IT services. We want to make it possible to hold cloud service providers accountable for how they manage personal, sensitive and confidential information in the cloud, and for how they deliver services. This will be achieved by an orchestrated set of mechanisms: preventive (mitigating risk), detective (monitoring and identifying risk and policy violation) and corrective (managing incidents and providing redress). Used individually or collectively, they will make the cloud services in the short- and longer-term more transparent and trustworthy for:

- users of cloud services who are currently not convinced by the balance of risk against opportunity
- their customers, especially end-users who do not understand the need to control access to personal information
- suppliers within the cloud eco-system, who need to be able to differentiate themselves in the ultimate commodity market.

In this paper we report on the results of an elicitation activity related to transparency requirements from the perspective of cloud customers. A Cloud Customer in our context is an entity that (a) maintains a business relationship with, and (b) uses services from a Cloud Provider; correspondingly, a Cloud Provider is an entity responsible for making a [cloud] service available to Cloud Customers.

Transparency is the property of an accountable system that is capable of ‘giving account’ of, or

providing visibility of, how it conforms to its governing rules and commitments (Felici et. al, 2013). Transparency involves operating in such a way as to maximize the amount of and ease-of-access to information which may be obtained about the structure and behavior of a system or process. An accountable organization is transparent in the sense that it makes the policies on treatment of personal and confidential data known to relevant stakeholders, can demonstrate how these are implemented, provides appropriate notifications in case of policy violation, and responds adequately to data subject access requests. In an ideal scenario, the user knows the information requirements and is able to communicate that clearly to the provider, and in return, the provider is transparent and thus willing to address the regulatory and legislative obligations required with regard to the assets.

The rest of the paper is organized as follows. Section 2 presents some background from the literature. Section 3 explains the methodology that we used to elicit the views of the stakeholders. In section 4 we present the results, and in section 5 we discuss our findings compared to related work. We draw our conclusions in section 6.

2 RELATED WORK

Transparency is closely connected to trust (Yang and Tate, 2012). Onwubiko (2010) affirms that trust is a major issue with cloud computing irrespective of the cloud model being deployed. He says that cloud users must be open-minded and must not whole-heartedly trust a provider just because of the written-down service offerings without carrying out appropriate due diligence on the provider; where certain policies are not explicit, users should ensure that missing policies are included in the service contract. By understanding the different trust boundaries, each cloud computing model assists users when making decision as to which cloud model they can adopt or deploy.

Khorshed et al. highlight the gaps between cloud customers' expectations and the actually delivered services, as shown in Figure 1 (Khorshed et al., 2012). They affirm that cloud customers may form their expectations based on their past experiences and organizations' needs. They are likely to conduct some sort of survey before choosing a cloud service provider similar to what people do before choosing an Internet Service Provider (ISP). Customers are

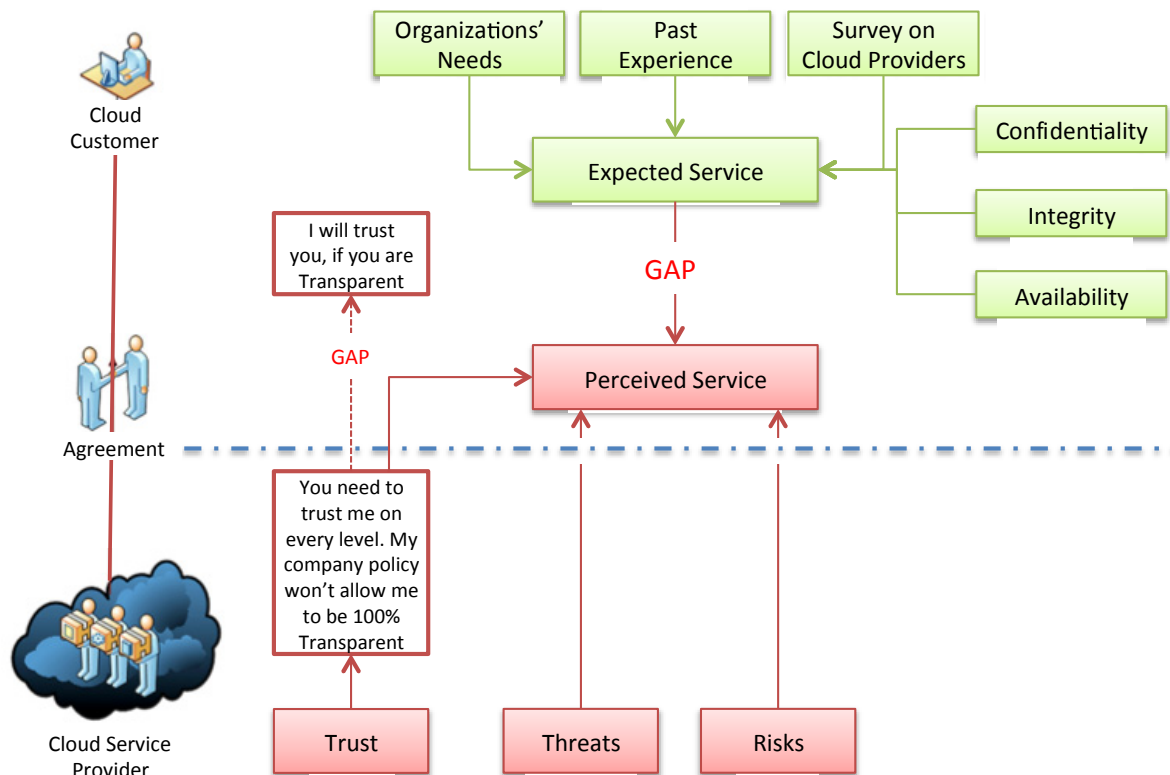


Figure 1: Understanding Cloud Computing Gaps adapted from Khorshed et al. (2012).

expected to also establish to what extent providers satisfy confidentiality, integrity and availability requirements. On the other hand, cloud service providers may promise a lot to entice a customer to sign a deal, but harsh reality is frequently accompanied by insurmountable barriers to keeping some of their promises. Many potential cloud customers are well aware of this, and are consequentially still sitting on the sidelines. They will not venture into cloud computing unless they get a clear indication that all gaps are within acceptable limits.

Durkee (2010) says that transparency is one of the first steps to developing trust in a relationship, and that the end customer must have a quantitative model of the cloud's behavior. The cloud provider must provide details, under NDA if necessary, of the inner

workings of their cloud architecture as part of developing a closer relationship with the customer. Durkee also says that this transparency can only be achieved if the billing models for the cloud clearly communicate the value (and avoided costs) of using the service. To achieve such clarity, the cloud vendor has to be able to measure the true cost of computing operations that the customer executes and bill for them.

Pauley (2010) proposed an instrument for evaluating the transparency of a cloud provider. It is the only empirical evaluation that we found that focuses on transparency in the cloud as a subject of study. The study aims to help businesses assess the transparency of a cloud provider's security, privacy, auditability, and service-level agreements the

Table 1: Pauley's Cloud Provider Transparency Scorecard.

Aspect	Criteria	Mentioned in Interviews?
Business factors	1. Length in years in business > 5?	No
	2. Published security or privacy breaches?	Yes
	3. Published outages?	Yes
	4. Published data loss?	Yes
	5. Similar customers?	Yes
	6. Member of ENISA, CSA, CloudAudit, OCCI, or other cloud standards groups?	No
	7. Profitable or public?	No
Security	8. Portal area for security information?	Yes
	9. Published security policy?	Yes
	10. White paper on security standards?	Yes
	11. Does the policy specifically address multi-tenancy issues?	Yes
	12. Email or online chat for questions?	No
	13. ISO/IEC 27000 certified?	Partially
	14. COBIT, NIST SP800-53 security certified?	Partially
	15. Offer security professional services (assessment)?	No
	16. Employees CISSP, CISM, or other security certified?	Partially
Privacy	17. Portal area for privacy information?	Yes
	18. Published privacy policy?	Yes
	19. White paper on privacy standards?	Yes
	20. Email or online chat for questions?	No
	21. Offer privacy professional services (assessment)?	No
	22. Employees CIPP or other privacy certified?	Partially
External audits or certifications	23. SAS 70 Type II	No
	24. PCI-DSS	No
	25. SOX	No
	26. HIPAA	No
Service-level agreements	27. Does it offer an SLA?	Yes
	28. Does the SLA apply to all services	No
	29. ITIL-certified employees?	No
	30. Publish outage and remediation?	Yes

the transparency of a cloud provider's security, privacy, auditability, and service-level agreements via self-service Web portals and publications. Pauley designed a scorecard (Table 1) to cover the assessment areas frequently raised in his research, and to begin to establish high-level criteria for assessing provider transparency. He concludes that further research is needed to determine the standard for measuring provider transparency. In our research we used a different strategy than Pauley; we have interviewed customers of cloud services to see what kind of information they would like to get from the cloud providers.

3 METHODOLOGY

As part of the project, we were responsible for running a set of stakeholder workshops for eliciting requirements for accountability tools. In total, our elicitation effort has involved more than 300 stakeholders, resulting in 149 stakeholder requirements. The first workshop dealt with eliciting initial accountability requirements, serving as a reality-check on the three selected business use cases we had constructed (Bernsmed et al., 2014). The second workshop dealt with risk perception. The aim was to focus on the notion of risk and trust assessment of cloud services, future Internet services and dynamic combinations of such services (mashups). After the first two workshops, we decided to organize multiple smaller, local workshops on each theme to ease participation of cloud customers and end users. The third set of workshops presented stakeholders with accountability mechanisms to gather their operational experiences and expectations about accountability in the cloud.

Of particular importance to this study was the risk workshop, where 15 tentative requirements related to transparency were identified. This workshop comprised 20 international stakeholders from the manufacturing industry, telecom, service providers, banking industry and academia, and the tentative transparency requirements were subsequently presented to our interviewees as a starting point for the discussion.

In addition to the stakeholder requirements, we have devised a set of high-level requirements which, from an organizational perspective, set out what it takes to be an accountable cloud provider (Jaaton et al., 2014). These requirements intend to supplement the requirements elicitation process by providing a set of high-level "guiding light" requirements, formulated as requirements that accountable

organizations should meet. In short, these requirements state that an accountable organization that processes personal and/or business confidential data must 1) demonstrate willingness and capacity to be responsible and answerable for its data practices 2) define policies regarding their data practices, 3) monitor their data practices, 4) correct policy violations, and 5) demonstrate policy compliance.

From these activities we have created a repository with requirements from all elicitation workshops, the guiding lights requirements as well as a number of more technical requirements that have originating from the conceptual work and technical packages in the project. These have been classified in terms of whether they are functional requirements, which are directly related to the actors involved in the cloud service delivery chain, or requirements for accountability mechanisms, which are related to the tools and technologies that are being developed in the project.

For refining and confirming the elicited requirements of transparency, we have performed an interview study with eight interviewees, followed by an in-depth analysis of the collected information.

Invitations were sent to our list of contacts in Norwegian software companies. Participation was voluntary. Eight people accepted to participate in the interviews. The participants were all IT security experts working with cloud related projects. The participants represented six different organizations: a consultancy, 2 cloud service providers (1 public, 1 private), an application service provider, a distribution service provider, and a tertiary education institution.

The interviews were performed on Skype and lasted about one hour. The main questions of the interview were:

1. What is the most important information you think should be provided to the cloud customer when buying services from cloud service providers?
2. In which parts would you like to be involved in making the decisions? In which parts would you like just to be informed of the decisions?
3. What would increase your trust that the data is secure in this scenario?
4. What do you want to know about how the provider corrects data security problems?

The eight interviews for this study were transcribed into text documents based on the audio recordings. For further analysis of the transcription, we followed the Thematic Synthesis recommended steps proposed by Cruzes and Dybå (2011). Thematic synthesis is a method for identifying, analyzing, and

reporting patterns (themes) within data. It comprises the identification of the main, recurrent or most important (based on the specific question being answered or the theoretical position of the reviewer) issues or themes arising from a body of evidence. The level of sophistication achieved by this method can vary; ranging from simple description of all the themes identified, through to analyses of how the different themes relate to one another in a conceptual map. Five steps were performed in this research: initial reading of data/text (extraction), identification of specific segments of text, labeling of segments of text (coding), translation of codes into themes, creation of the model and assessment of the trustworthiness of the model.

4 RESULTS

For the question "What is the most important information you think should be provided to the cloud customer in this scenario?" the participants talked mostly about nine themes (Figure 2):

1. clear statements of what is possible to do with the data,
2. conformance to data agreements,
3. how the provider handles data,
4. location,
5. who else other than the provider is participant of the value chain,
6. multi-tenant situations,
7. what the provider does with the data,
8. procedures to leave the service
9. assurance that the user still owns the right to the data.

One respondent commented that even though he would like to have clear statements of what is possible to do with the data: "100 pages document could be written about this, but for some non-technical people it would not help at all". Another one said: "I would like to have a [web] page where they could tell me about security mechanisms, for example, firewalls, backup etc."

On the conformance to data agreements, the respondents agree that having Data Agreements helps, but it is mainly for technicians, not for non-technical people. On how the provider handles data, the respondents said that they would like to have functional, technical and security related information about how the providers handle the data. On location, the respondents are concerned about where the data is physically stored, and the legal jurisdiction of the services. Another important piece of information is about sub-providers, if there are any; where they are

located and whether they meet legal requirements of the customer's location. Multi-tenant situations are a concern of the customers, and they would like to have this information transparent. Also, information on how the providers ensure that data from one customer will not be accessed by another customer.

It is also important for transparency to know what the provider does to protect customers' data. One respondent said that he would like to have information on: "How to protect the information or how the information is protected; not much in detail for the end-user, but only for enterprises." It was also highlighted that they would like to have the procedures to leave the service and on how to move data from one service to another transparent. Besides, they would like to have the assurance that they still own the rights to their data.

On the question "What would increase your trust that the data is secure in this scenario?" the participants mentioned eight different themes: 1) upfront transparency; 2) community discussions, 3) customer awareness; 4) way out; 5) reputation; 6) encryption; 7) data processor agreements; and 8) location.

Some answers were overlapping towards the answers from the first question: upfront transparency, location and conformance to data processor agreement. Interesting answers for this question were related to community discussions, customer awareness and reputation. The respondents said that it increases their trust in a cloud provider if they know that the provider has an active security research team, or participates in security communities. The respondents also said that for security: "Customers should be proactive and make sure that all the documentation is there". And another one commented on the importance of having webpages telling what customers could do to keep the data safe. Two participants also mentioned "Way out", meaning that they would like to have webpages telling them what to do to remove the data from the service provider.

On the questions: "In which parts would you like to be involved in making the decisions? In which parts would you like just to be informed of the decisions?" it was surprising that the participants mostly answered that they would like to be informed but not really taking part of every decision (Figure 4); the exceptions were when the provider was moving data to another country, other parties are introduced in the service provider value chain, or there are significant changes in the initial terms of contract.

One participant said: "Some customers sometimes have some requests, but in general they do not care about taking part in the decisions", and another one

said: “there are some decisions that we don't need to explicitly know about, but it has to be regulated by some other agreement about the responsibility of each one towards the data”. One respondent also said: “I would like to be involved in decisions on moving my data to another country in most situations. Unless for

example a disaster and there is the need to move to another country.” Some respondents said that they would like to be informed when the data is transferred from one actor to the next, one of them added: “For example if calling to the call center your data will be transferred to another country then the customers has

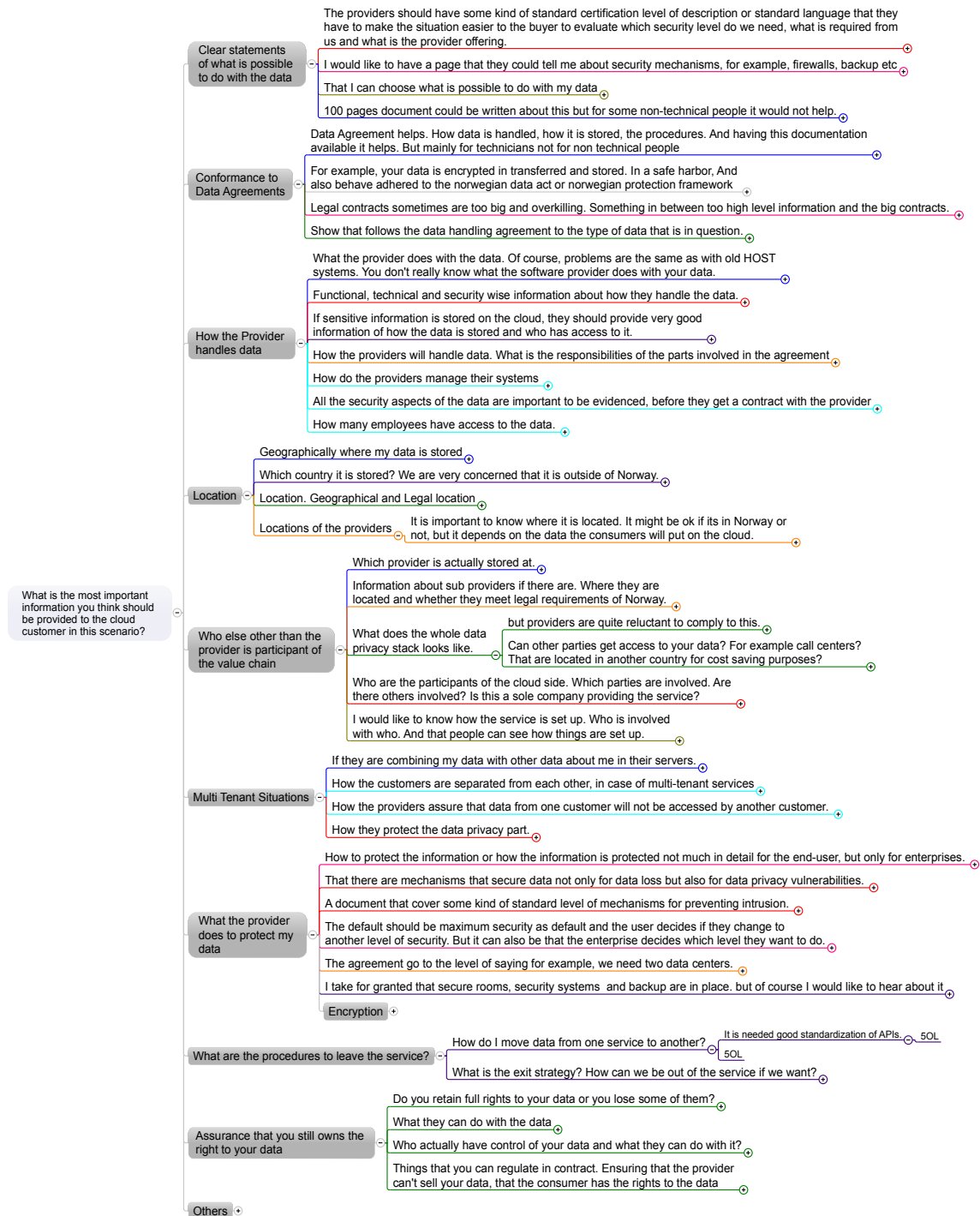


Figure 2: Important Upfront Information for Transparent Services.

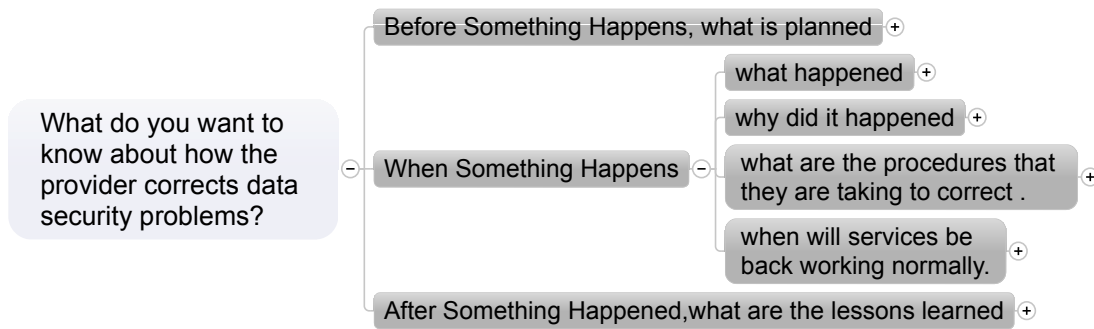


Figure 3: Transparency on Correction of Data Security Problems.

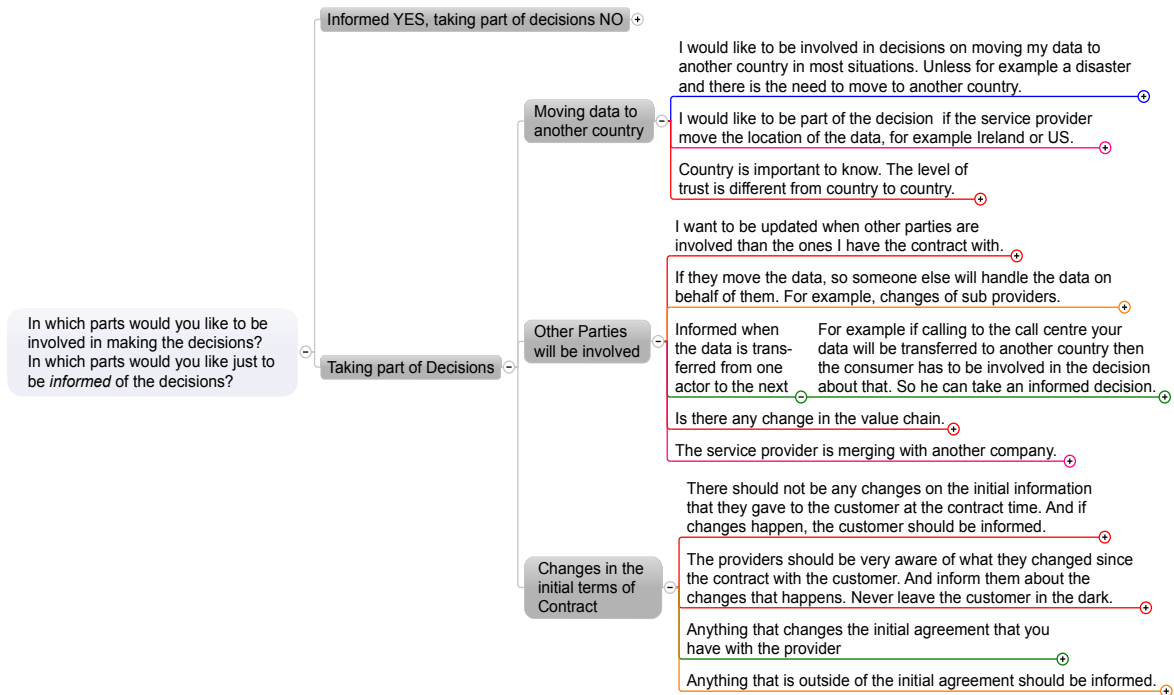


Figure 4: Involvement on making Decisions.

to be involved in the decision about that. So he can take an informed decision.” On changes in the initial terms of Contract, one respondent said: “the providers should be very aware of what they changed since the contract with the customer [was signed], and inform them about the changes that happen. Never leave the customer in the dark.”

When asked on what they would want to know about how the provider corrects data security problems, it was again surprising to learn that the participants have not thought much on what they could expect from the providers if some security issue happens. Most of the respondents needed further elaboration of the question before they would start saying something. Then, the participants stated that they would like to know what is planned before

something happens; when something happens they want to know how the providers are handling the situation, why the problem happened, and when will the services be back online. Interesting was also the fact that the participants wanted to know how the providers are improving their services after something happens, based on lessons learned. These responses are collated in the taxonomy shown in Figure 3.

5 DISCUSSION

After analyzing all the collected information we compiled a list of requirements elicited in the interviews, as shown in Table 2. The main “topics”

Table 2: List of Requirements from Transparency interviews.

List of Elicited Requirements	
What is possible to do with the data	<p>The provider should show clear statements of what is possible to do with the data</p> <p>The provider should allow the cloud customer to choose what is possible to do with his/data data</p> <p>The provider should have a page that they could tell the cloud customer about security mechanisms, e.g., firewalls, backup etc.</p> <p>The provider should have some kind of standard certification level of description or standard language that they have to make the situation easier to the buyer to evaluate which security level do we need, what is required from us and what is the provider offering.</p> <p>The provider should have a document explaining what are the procedures to leave the service and take the data out of their servers.</p> <p>The provider should have a document in which they describe the ownership of the data.</p>
Conformance to Data Agreement	<p>The provider should make available the technical documentation on how data is handled, how it is stored, and the procedures.</p> <p>There should be documentation of procedures in different levels of abstraction, for example for technical staff or for cloud subjects</p> <p>The provider should show that they follow the data handling agreement to the type of data that is in question.</p> <p>The provider should provide geographical information of where the data is stored.</p>
Data Handling	<p>The provider should provide functional, technical and security wise information about how they handle the data.</p> <p>The provider should provide very good information of how the data is stored and who has access to it.</p>
Value chain	<p>In case of using services from other parties, the provider should inform cloud customers on what are the responsibilities of the parts involved in the agreement.</p> <p>In case of using services from other parties, the provider should inform about the existence of sub providers, where they are located and whether they meet legal requirements of the country of the cloud customer.</p>
Multi-Tenant Services	<p>The provider should inform the cloud customers on cases of multi-tenant services.</p> <p>In case of multi-tenant services, the provider should inform how the customers are separated from each other.</p> <p>In case of multi-tenant services, the provider should inform how they assure that data from one customer will not be accessed by another customer.</p>
Protection of the data	<p>The provider should inform the cloud customer on how to protect the information or how the information is protected not much in detail for the end-user, but only for enterprises.</p> <p>The provider should have a document describing the mechanisms that secure data not only for data loss but also for data privacy vulnerabilities.</p>
Decisions	<p>The cloud providers should get the consent of the cloud customer before moving the data to another country, in cases where new parties will be involved in the value chain and on changes on the initial terms of contract.</p>
Correction of the data	<p>The cloud provider should have a document stating what are the procedures and mechanisms planned for cases of security breaches on customers' data.</p> <p>In case of security breaches, the cloud provider should inform the cloud customers on what happened, why did it happen, what are the procedures they are taking to correct the problem and when will services be normalized.</p>

mentioned by the respondents were related to what is possible to do with the data, conformance to data agreements, data handling, value chain, multi-tenant situations, protection of the data, decisions and corrections of the data.

Pauley (2010) designed a scorecard reproduced in Table 1 to cover the assessment areas frequently raised in the research, and to begin to establish high-level criteria for assessing provider transparency. When comparing our list of elicited requirements to Pauley's scorecard (Table 2), we can see some slight differences in the criteria that Pauley described as information that should be provided by the cloud providers and the information that the customers are looking for (Table 2). In the criteria about the business factors, the customers did not mention being concerned about the number of years in business, nor about membership of CSA, CloudAudit, OCCI, or other cloud standards groups, or if the providers are profitable or public. There is a possibility that the respondents did not mention these criteria because (a) companies in Norway are usually stable, and (b) membership of a group or association does not in itself guarantee good performance or compliance, even if the group or association promotes a certain standard.

On the security and privacy aspects, the customers mentioned all the criteria, but they did not mention directly the standards/certifying bodies, such as ISO/IEC 27000, COBIT and NIST, but they mentioned that it would be nice to know if the provider was certified somehow, based on some criteria. The customers also did not mention the need to know about "external" audits. One of the reasons for not mentioning security standards and certification bodies may be that companies that we have investigated are predominantly private companies in Norway, where there are not strong requirements from the certification bodies yet.

One important aspect not very much explored in Pauley's scorecard is that customers would like providers to be transparent about what is possible to do with the data. In addition, customers were quite concerned about transparency on exit procedures ("way out") and ownership of the data. The concern over data ownership is interesting seen in the light of Hon et al. (2012), who found no evidence of cloud contracts leading to loss of Intellectual Property Rights.

Another aspect further mentioned by the customers is on the decisions made on "ongoing" services, where the customers would like that: "The cloud providers should get the consent of the cloud customer before moving the data to another country,

in cases where new parties will be involved in the value chain and on changes on the initial terms of contract."

Physical location and legal jurisdiction, as well as specific information on the value chain was a very important aspect to be transparent about for the cloud customers, and it was not explicitly mentioned in Pauley's scorecard.

The interviewees did not show a desire for the kind of detailed information Durkee (2010) deems necessary (the inner workings of their cloud architecture as part of developing a closer relationship with the customer), and as also pointed out by Durkee, some respondents were also aware that the costs of such clarity may be prohibitive, and we might add that this level of disclosure seems highly unlikely for ordinary customers of commodity cloud services.

Many of the transparency mechanisms that customers expressed a desire for are actually being developed by the A4Cloud project (Jaatun et al., 2014). For end-users, the Data Track tool (Fischer-Hübner et al., 2014) enhances transparency by tracking which personal data has been released to whom. Furthermore, a central theme of A4Cloud is the development of the Accountability PrimeLife Policy Language (A-PPL), which allows end users to specify a privacy policy that also covers accountability requirements, including transparency (Azraoui et al., 2014). A4Cloud is developing an A-PPL Engine which will serve as a Policy Decision Point for the associated policies at each cloud provider. Other tools developed by A4Cloud include the Cloud Offerings Advisory Tool (COAT), which allow cloud customers to select an appropriate cloud provider based on relevant accountability requirements, including transparency (Alnemr et al., 2014). This will eventually allow transparency requirements to be built into standard cloud service level agreements (SLAs), where transparency is just one of several security attributes (Jaatun et al., 2012).

6 CONCLUSIONS

Cloud computing has been receiving a great deal of attention, not only in the academic field, but also amongst the users and providers of IT services, regulators and government agencies. The results from our study focus on an important aspect of accountability of the cloud services to customers: transparency.

The customers made explicit all the information that they would like the providers to be transparent about. Much of this information can be easily

provided at a provider's website. Our contention is that being transparent can be a business advantage, and that cloud customers who are concerned with, e.g., privacy of the data they put into the cloud, will choose providers who can demonstrate transparency over providers who cannot.

Our study increases the body of knowledge on the criteria needed for more accountable and transparent cloud services, and confirms the results from previous studies on these criteria. The list of requirements in Table 2 complements, in part, the existing criteria.

An area for future research is to further evaluate how cloud providers currently make the information required by cloud customers available. In addition, what are the effects of having transparent services in terms of costs and benefits to cloud customers and providers. Besides, we plan to increase the number of participants responding to our interview guide and adding strength to the evidence provided in this paper. Another aspect we would like to investigate, is if the results will be different for users of the different types of services (e.g., SaaS vs IaaS).

ACKNOWLEDGEMENTS

This paper is based on joint research in the EU FP7 A4CLOUD project, grant agreement no: 317550.

REFERENCES

- AbuKhoua, E., Mohamed, N., and Al-Jaroodi, J., "e-health cloud: Opportunities and challenges," *Future Internet*, vol. 4, no. 3, pp. 621–645, 2012.
- Ahuja, S. P., Mani, S. and Zambrano, J., "A Survey of the State of Cloud Computing in Healthcare," *Network and Communication Technologies*, vol. 1, no. 2, p. 12, 2012.
- Alnemr, R., Pearson, S., Leenes, R., and Mhungu, R., "COAT: Cloud Offerings Advisory Tool". *Proc. of the 2014 IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014)* 95-100, 2014.
- Azraoui, M., Elkhyaoui, K., Önen, M., Bernsmed, K., Sendor, J., and Santana de Oliveira, A., "A-PPL: An accountability policy language", in *DPM, 9th International Workshop on Data Privacy Management*, 10 September 2014.
- Bernsmed, K., Tountopoulos, V., Brigden, P., Rübsamen, T., Felici, M., Wainwright, N., Santana De Oliveira, A., Sendor, J., Sellami, M., and Royer, J.-C., "Consolidated use case report", A4Cloud Deliverable D23.2, October 2014 <http://www.a4cloud.eu/sites/default/files/D23.2%20Consolidated%20use%20case%20report.pdf>.
- Cruzes, D. S. and Dybå, T., *Recommended Steps for Thematic Synthesis in Software Engineering*. ESEM 2011: 275-284, 2011.
- Durkee, D., Why cloud computing will never be free. *Commun. ACM* 53(5): 62-69, 2010.
- Felici, M., Koulouris, T. and Pearson, S., "Accountability for Data Governance in Cloud Ecosystems", *Proc. of the 2013 IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013)*, 2013.
- Fischer-Hübner, S., Angulo, J., and Pulls, T., "How can Cloud Users be Supported in Deciding on, Tracking and Controlling How their Data are Used?", *Privacy and Identity Management for Emerging Services and Technologies, IFIP Advances in Information and Communication Technology* Vol. 421, 2014, pp 77-92.
- Gavrilov, G. and Trajkovic V., "Security and privacy issues and requirements for healthcare cloud computing," in *Proceedings of ICT Innovations*, 2012.
- Hon, W.K., Millard, C. and Walden, I., "Negotiating Cloud Contracts - Looking at Clouds from Both Sides Now" (May 9, 2012). 16 *STAN. TECH. L. REV.* 81 (2012); Queen Mary School of Law Legal Studies Research Paper No. 117/2012.
- Jaatun, M.G., Bernsmed, K., and Undheim, A.: "Security SLAs – an idea whose time has come?", *Proc. CD-ARES, Prague, LNCS Volume 7465*, pp 123-130, 2012.
- Jaatun, M.G., Pearson, S., Gittler, F., and Leenes, R., "Towards Strong Accountability for Cloud Service Providers", *Proc. of the 2014 IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014)*, 2014.
- Khorshed, M. T., Ali, A.S. and Wasimi, S.A.: A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. *Future Gener. Comput. Syst.* 28(6), 833–851 (2012).
- Kuo, A. M.-H., "Opportunities and challenges of cloud computing to improve health care services," *J. Med. Internet Res.*, vol. 13, no. 3, p. e67, 2011.
- Onwubiko, C., (2010) "Security Issues to Cloud Computing", in *Cloud Computing: Principles, Systems & Applications*, (Eds) Nick Antonopoulos and Lee Gillam, Springer-Verlag, August, 2010.
- Paquette S., Jaegar, P. T. and Wilson, S. C. Identifying the security risks associated with governmental use of cloud computing, *Journal of Government Information Quarterly* 27, pages 245-253, April, 2010.
- Pauley, W.A., "Cloud Provider Transparency: An Empirical Evaluation," *IEEE Security & Privacy* (8)6, pp. 32– 39, 2010.
- Rodrigues, J. J., Torre, I. de la, Fernandez, G., and Lopez-Coronado, M., "Analysis of the security and privacy requirements of cloud-based electronic health records systems," *J. Med. Internet Res.*, vol. 15, no. 8, p. e186, 2013.
- Yang, H. and Tate, M., "A Descriptive Literature Review and Classification of Cloud Computing Research," *Communications of the Association for Information Systems: Vol. 31, Article 2*, 2012.

OCCI and TTCN-3

Towards a Standardized Cloud Quality Assessment Framework

Yongzheng Liang

bwcon GmbH, Breitscheidstrasse 4, 70174 Stuttgart, Germany
liang@bwcon.de

Keywords: Cloud Quality Assessment, Standardized Testing, TTCN-3, Cloud Standards, OCCI, Software Defined Network, Network Functions Virtualization.

Abstract: Impacting basically all types of IT infrastructures The Cloud is one of the most important evolving IT paradigms. A standard-based Cloud quality and compliance assessment framework will be therefore of utmost importance. Bringing together the Open Cloud Computing Interface OCCI and the ETSI standardized test specification language TTCN-3 and related test methodologies this paper is going to demonstrate initial steps towards such a framework. Taking into account the diversity of Cloud infrastructures, of service providers, and related architectural, harmonization and standardization effort this approach is mainly motivated by studying Cloud-related effort of the NIST Cloud Computing Program and the ETSI Cloud Standards Coordination (CSC). Reflecting the “Cloudiness” of the Software Defined Network (SDN) and ETSI Network Functions Virtualization (NFV) this paper is considering these initiatives as necessary elements of the scope of every future standardized Cloud quality assessment framework as well.

1 INTRODUCTION

Impacting basically all types of IT infrastructures “The Cloud” is one of the most important evolving IT paradigms. A standard-based Cloud quality and compliance assessment framework will be therefore of utmost importance. Bringing together the Open Cloud Computing Interface OCCI and the ETSI standardized test specification language TTCN-3 and related test methodologies this paper is going to demonstrate initial steps towards such a framework. Taking into account the diversity of Cloud infrastructures, of service providers, and related architectural, harmonization and standardization effort our approach is motivated by studying Cloud-related effort of the NIST Cloud Computing Program, NIST CC, the ETSI Cloud Standards Coordination (CSC). Reflecting the “Cloudiness” of the Software Defined Network (SDN) and ETSI Network Functions Virtualization (NFV) this paper is considering these initiatives as necessary elements of the scope of every future standardized Cloud quality assessment framework.

The rest of the paper is organized as follows: Chapter 2 is introducing pertinent work of NIST CC

and ETSI CSC— here the role of the OCCI standard becomes already visible. The methodological look at NIST/ETSI will follow the triple “use cases – standards – testing” and corresponding mappings.

Chapter 3 describes how, following the virtualization paradigm, the “Software Defined Network”, SDN, and ETSI NFV have met the Cloud. It will be noticed that the NFV use case “IMS as a Service” (IMSaaS) has in its original 3GPP and ETSI context an elaborated TTCN-3 framework.

Chapter 4 introduces the OGF OCCI standard. Chapter 5 describes some OCCI related effort of relevance in the given context.

Chapter 6 introduces TTCN-3, the “Testing and Test Control Notation Version 3” the test specification language standardized by ETSI. Chapter 7 describes relevant TTCN-3 effort.

Chapter 8 describes “TTCN-3 on top of OCCI” for both a subset of the ETSI Interoperability test cases and for BonFIRE – a large European Multi-Cloud project.

Chapter 9 resumes the paper and gives an outlook on future work.

2 TOWARD A STANDARDIZED CLOUD QUALITY ASSESSMENT FRAMEWORK

Influenced by and possibly influencing the evolution of Cloud ecosystems potential Cloud adopters have developed related use cases of different abstraction level above the basic technologies in question. At the same time and in a similar interdependency relation in numerous bodies Cloud standards have evolved and are still evolving. In such a situation mapping use cases to compatible or even “integrated” standards is one of the natural important steps to happen next. Eventually, addressing different test types such as conformance, performance etc. test cases will be specified. Being a simplified one, this process is nevertheless a typical and necessary element in the evolution towards a quality assessment framework.

Following this process and given the sheer weight of the US Government as a Cloud adopter and the important role of ETSI concerning high-quality standards and formal testing methodologies we are going to use the NIST Cloud Computing Program and the ETSI Cloud Standard Coordination effort in order to argue for a TTCN-3- and OCCI-oriented, standardized Cloud quality assessment framework.

2.1 NIST CC Program

The NIST (National Institute of Standards and Technology) designed its Cloud Computing Program, CC, “to support accelerated US government adoption, as well as leverage the strengths and resources of government, industry, academia, and standards organization stakeholders to support cloud computing technology innovation” (NIST, 2014). The cited document “US Government Cloud Computing Technology Roadmap” comprising the Volume I “High-Priority Requirements to Further USG Agency Cloud Computing Adoption” and Volume II “Useful Information for Cloud Adopters” summarizes the results of now the finalized Phase I and defines and relates ten “high-level requirements” to the different NIST CC working groups for Phase II.

Key documents of Phase I are concerning Cloud taxonomy and vocabulary, reference architecture, standards and security; for references see (NIST, 2014).

The NIST projects and working groups apply a use case methodology to define business and technical operational scenarios and requirements. The NIST-chaired public Cloud Computing Business Use Case

Working Group (CCBUCWG) has produced use cases at the functional mission level. Those “business use case are decomposed into a list of high-level requirements, then into successively more detailed requirements, until they can ultimately be mapped to technical requirements that are required to identify and executed” as “technical use cases”. Dealt with by the group “Standards Acceleration to Jumpstart the Adoption of Cloud Computing” (SAJACC) the latter use cases are “designed to facilitate the qualitative testing of standards through the use of third-party APIs implemented in adherence to candidate specifications and emerging standards”. SAJACC use cases represent single activities, such as the “deletion of data, and the actions needed to successfully execute that activity (receive the request, respond to the request, execute the request, etc.)”.

Without any ambition towards formalization in terms of possible map-ability and automated processing, for the description of use cases two types of templates have been developed.

A particular set of standards in relation to a use cases was termed “compatible standards” – no specific exercise was undertaken to consider the “integration” of those specific standards in question – e.g. CDMI and OCCI; see also below (Edmonds, 2011) However, concerning the “current state of conformity assessment in Cloud Computing”, (NIST, 2014), section 6.2.4 states: In some cases, such as the CDMI, OCCI, OVF, and CIMI standards... industry-sponsored testing events and “plug-fests” are being advertised and conducted with participation from a variety of vendors and open source projects and community-based developers. In other cases, either the standards are not yet mature enough to permit such testing, or the participants have not yet exposed the conformity assessment processes to public view. – In this spirit NIST representatives gave presentations at the “First Cloud Interoperability Week” (Sill, 2013); see also (Liang, 2013a). Finally, in order to cope with questions like “is the proposed quality assessment framework not overkill?” - it should be mentioned that the NIST is considering Cloud ecosystems as eventually big, complex and potentially endangered by “catastrophes” comparable to the famous Internet or global power grid breakdowns. Accordingly – with participation of the OGF Research Group on Grid Reliability and Robustness - NIST has started the “Complex Information Measurement Project - Koala” (NIST, 2015).

It should be noticed that so far NIST doesn’t deal with SDN or NFV issues, see below.

2.2 ETSI CSC

Being part of the European Commission's Cloud related strategy the so-called key action "Cutting through the jungle of standards" was assigned by DG Connect to the specifically created ETSI working group "Cloud Standards Coordination", CSC. The latter in its mission's final step 3 created three "Specification identification gap analysis" working groups: SLAs – Security & Privacy – and – Interoperability, Data port, Reversibility. Launched in December 2012, the CSC provided a final report (ETSI, 2013). This report stated that "the Cloud Standards landscape is complex but not chaotic and by no means a 'jungle'".

In this report ETSI CSC introduces vocabulary and taxonomies applicable to Cloud Actors and their Roles within Use Cases. The analysis of Use Cases comprises the following dimensions: "Phases and Activities", "Perspectives" (SLAs, Interoperability, Security), generic domains (e.g. "Applications in the Cloud", "Cloud Bursting" etc.), and "Phases and Activities". This schema is then used in a mapping of use cases to standards.

Gaps related to SLAs, security and privacy are dealt with in the final report. Interoperability is specifically covered by the Technical Specification "CLOUD; Test Descriptions for Cloud Interoperability" (ETSI, 2013b). The standards dealt with herein are OCCI, see below, and CDMI, CAMP, OVF and CIMI. In Chapter 8 below we are going to demonstrate some initial work related to the OCCI-related test cases.

It should be mentioned that also ETSI CSC expresses a positive view concerning OCCI (together with CDMI and OVF): "OCCI as the universal and extensible interface description for the provisioning of virtualised computing resources."

ETSI CSC has called for a 2nd Phase of work to be started in early 2015 – and in close cooperation with NIST CC.

Without any further explanation the ETSI CSC final report provides a list of the ETSI NFV specifications; see next chapter.

3 ETSI NFV, SDN AND THE CLOUD

Instrumental as a key concept and as enabler of many aspects of computing, storage and networking "Virtualization" lies at the ground of both the Cloud and concepts or initiatives such as the "Software

Defined Network", SDN (ONF, 2011) and ETSI's "Network Function Virtualization", NFV (ETSI, 2012).

SDN has evolved as a potential solution to both the growing management complexity of the overly successful Internet and, in turn, the growing "ossification" of the latter. Aiming at more flexibility and dynamicity of network services through programmability of network hardware boxes such as routers, switches, firewalls etc. the OpenFlow™ protocol and API is a key element in the context. Launched in 2011 by Deutsche Telekom, Facebook, Google, Microsoft, Verizon, and Yahoo!, the Open Networking Foundation (ONF) is a non-profit organization with more than 140 members whose mission is to accelerate the adoption of open, standardized OpenFlow-based SDN.

Used as generic term "software defined networking" is also addressed by the "Network Functions Virtualization - Industry Specification Group", NFV(ISG). Initiated in 2012 within ETSI by seven telecom operators the group was joined by over 200 companies including network operators, telecoms equipment vendors. Opposed to SDN, NFV was primarily driven by concerns related to OPEX and CAPEX of typical telecom hardware appliances and service agility. NFV aims to use "advanced IT virtualization techniques" (aka Cloud plus Cloud enablers i.e. hypervisors etc.) in order to convert typical telecom appliances and service frameworks into "X as a Service" instances, the latter class being instantiated even into "IMS as a Service", IMSaaS.

SDN and NFV are highly complementary to and independent of each other.

In order to promote NFV through OpenFlow-based SDN in March 2014 ONF and ETSI agreed on a related strategic partnership.

The NFV(ISG) has produced since five specifications covering NFV use cases, requirements, the architectural framework, and terminology. The fifth specification defines a framework for coordination and promotion of public demonstrations of Proofs of Concept, PoC (ETSI, 2014). The PoC demonstrate key aspects of NFV use cases – specifically the explicitly Cloud-related "NFV Infrastructure as a Service" (NFVIaaS), the "Virtual Network Functions as a Service" (VNFaaS), the "Service Chain Forwarding Graphs" (VNF FG), the "Virtual Network Platform as a Service" (VNPaaS) and the mobility-oriented "Virtualization of the Mobile Core Network and IMS". The first results of the NFV PoC have been showcased.

While aiming at vendor and product neutrality the Cloud “core” of the PoC was the OpenDaylight Hydrogen release of OpenStack comprising inter alia the OpenStack Neutron component as OpenFlow oriented SDN controller.

Here, in the context of this paper, it should be noticed that this whole architecture is controlled by a (super-)set of the OpenStack RESTful APIs; see below the MCN project.

Finally, it should be mentioned that ETSI NFV doesn’t refer to ETSI CSC or the ETSI TC MTS, the Technical Committee Methods for Testing and Specification (ETSI, 2015); specifically, there is no hint given to the ample, standardized TTCN-3-oriented test framework (ETSI, 2015a).

4 OCCI

The Open Grid Forum’s (OGF) ‘Open Cloud Computing Interface’ (OCCI) is a well-defined, RESTful Cloud management protocol and interface, which can be applied to and extended from its initial target IaaS to functional and non-functional aspects also of PaaS and SaaS – even in Multi-Cloud ecosystems.

The definition of OCCI comprises a “Core” and a meta-model aspect according to the following figure, see (OCCI, 2011b).

The “Core” describes the foundation of the OCCI type system – “what types of resources can be out there”. This is orthogonal and complementary to the wire”.

The meta-model aspect represents the descriptive part allowing for extensibility, hierarchies, dynamic runtime modifications of resource instances and tagging via Mixins, and introspection via the mandatory discovery interface (Edmonds, 2012).

Members of the OCCI specification group developed a related conformance platform in Python (OGF, 2012b and OGF, 2012a). This work was not continued after 2012; it is/was not directly targeting whole OCCI-controlled Cloud systems but the conformance of (language) specific OCCI implementations.

The OCCI Working Group of the OGF is actively pursuing the further development of the OCCI standard; a completed specification is available e.g. for JSON rendering; a “Monitoring” specification and a related “Notification” specification are almost ready, and there is work for a “Platform” (PaaS) specification; see (OGF, 2014).

At the same time the WG is present at many related Cloud events such as the Cloud Interoperability

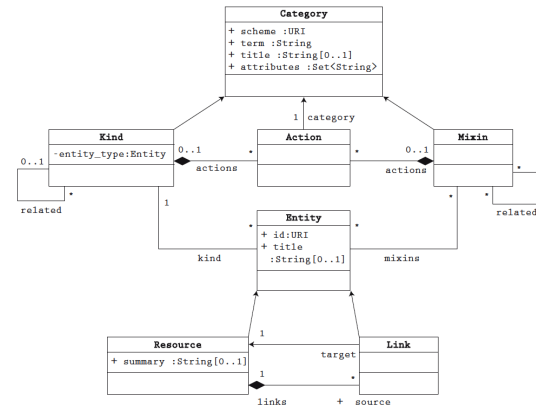


Figure 1: The OCCI “Core” Model.

Week mentioned above. Basically all WG members are also present in NIST CC or EGI (EGI, 2015) and MCN; see below.

5 OCCI-RELATED EFFORT

In order to further argue for the “robustness” of the OCCI case, in the following we are going to shortly mention effort covering technical and “market” aspects of OCCI applicability.

5.1 OCCI Technical Versatility

In (Edmonds, 2011) a standards conformant “integration scenario” of OCCI, CDMI and OVF is presented.

The “First Open Cloud Broker” developed in the CompatibleOne project and initiative is an early example for the extensibility of OCCI beyond IaaS (CompatibleOne, 2015).

The EU project MCN - Mobile Cloud – Networking, 2012-2015, “is motivated primarily by an ongoing transformation that drives the convergence between the Mobile Communications and Cloud Computing industry enabled by the Internet” (MCN, 2014). MCN’s two scenarios are “Exploiting Cloud Computing for Mobile Network Operations” and “The End-To-End Mobile Cloud”. While not fully concurrent with ETSI’s NFV PoC architectural principles MSC is about to realize a comparable SDN/NFV framework wherein the Cloud component will be represented by OpenStack too. In contrast to ETSI’s PoC non-standard set of related RESTful interfaces MCN is targeting OCCI. Referring to Core meta-model mechanisms, (MCN, 2013) section “2.4.1 OCCI Extensions” and “2.4.2 OpenStack Extensions”, the project has defined necessary extensions to both OCCI and OpenStack.

Finally, among the set of MCN's XaaS to be provided we are specifically mentioning MaaS, Monitoring as a Service (see also below the BonFIRE project) and IMSaaS, IMS as a Service.

The OCCI work in MCN is well aligned with the OCCI WG.

5.2 OCCI in Large Infrastructures

"The European Grid Infrastructure (EGI) is building a federated, standards-based IaaS Cloud platform, building on its decade-long experience in delivering a reliable, federated Grid infrastructure for scientific computing and e-Research across Europe and worldwide." "Federations are enabled by a set of core services such as seamless authentication and authorization of users, gathering of accounting information, information discovery, monitoring and VM management across multiple cloud domains; see (EGI, 2015)

In the given context it is of relevance that EGI Engage, the next large project of the initiative, is targeting well defined OCCI extensions in order to increase functions and performance of its pan-European Cloud federation. This work is closely aligned with the OCCI WG.

Our tests below are using the so-called rOCCI, an OCCI implementation in ruby. The rOCCI is part of the EGI effort.

6 TTCN-3

TTCN-3, the "Testing and Test Control Notation Version 3" is a successful Test Specification Language standardized by ETSI. Initially targeted at protocol conformance testing e.g. for IPv6, or SIP, the coverage of TTCN-3 was extended to new technical domains such as the Web, embedded and real-time systems, and new sectors such as Health, Automotive and "Intelligent Transport Systems" (ITS). Related organizations are e.g. 3GPP, OMA and AUTOSAR. The ETSI TTCN-3 standards have also been adopted by International Telecommunication Union (ITU-T) in the Z.160 series. The main characteristics of TTCN-3 are: Multi-Separation of Concerns by dividing a test system into an abstract but executable Test Specification Layer ("ATS" in Figure 2), and Concrete Codec and System-Adaptation Layers; see again Figure 2. From an effort point of view codec and adapter represent a major piece of (initial) work, paving the way towards a potential large testing framework at ATS level. This separation between concrete and abstract

layer is also allowing for a high degree of reusability. Targeting testing by design TTCN-3 provides an elaborated mechanism for the construction of Templates the latter to be used as test oracles; see e.g. (Schieferdecker, 2012). A related powerful Template matching mechanism then serves to validate output from the "System under Test" (SUT) on the level of the ATS; compare this e.g. with the language dependencies in (OGF, 2012a). - Related global Verdicts are computed, possibly composed from local Verdicts.

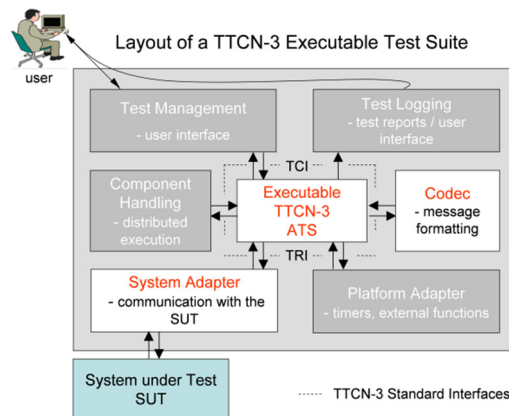


Figure 2: Layout of a TTCN-3 Executable Test Suite.

7 TTCN-3 RELATED EFFORT

In following, the first section is shortly describing effort related to TTCN-3 language developments. Section two is showing TTCN-3 as an element of ETSI's effort towards model-based testing.

7.1 TTCN-3 Development

TTCN-3 related effort is devoted to both the development of the language as such (via well-defined formal procedures within the ETSI); an example of relevance in context is "MTS The Testing and Test Control Notation version 3; Part 11: Using JSON with TTCN-3" - and other aspects. Such work may be carried out e.g. in cooperation with tool providers - to improve the efficiency of the coding/decoding process in a Web service environment would be an example. For a recent overview see (Stepien, 2014).

7.2 TTCN-3 in the ETSI TC MTS

TTCN-3 is not "just another standalone test specification language" but is part of an overall

effort within ETSI to further the development of methodologies in the spirit of “model-based testing” (ETSI, 2015).

Initially targeting communicating systems the ETSI MTS is addressing the formalization and mechanization/automation of a stack of processes and specifications ranging from requirements solicitation and “notation” over test and test purpose to test case specification.

Herein TTCN-3 is placed at the bottom layer.

Looking at the table format of the NIST technical and the ETSI CSC use cases the corresponding TC MTS historical effort is TPLan, ETSI ES 202 553. At present the TC MTS is pursuing with the TDL, Test Description Language, a more rigorous approach: integrating and unifying test description and test purpose specification layer above TTCN-3 TDL raises the abstraction layer of the latter and allows at the same time for down-mapping from the requirements layer; see (Makedonski, 2014).

8 TTCN-3 AND OCCI

“TTCN-3 on top of OCCI” was, to our knowledge, presented for the first time at the “Cloud Interoperability Week Workshop”, (Liang, 2013a) and at the UCAAT 2013 (Liang, 2013b). This work was related to the initial version of ETSI “Test Descriptions for Cloud Interoperability” (ETSI, 2013b).

We improved and extended this effort in the following way:

- We wrote new versions of the Codec and the System Adapter allowing specifically for a complete treatment of all coding and systems requirements of the OCCI tests of (ETSI, 2013b); see Figure 2 and Figure 3 again for the positioning these components.
- Using the current version of the ETSI document, so far we carried out all the OCCI Core and Infrastructure tests against a rOCCI-based EGI Cloud test infrastructure (EGI, 2015).
- We run initial tests of the BonFIRE Multi-Cloud project “Elasticity as a Service” (for “BonFIRE and OCCI” see below), (BonFIRE, 2014).

8.1 TTCN-3 and OCCI Mapping

The Figure 3 below shows the functional components and potential mappings of a TTCN-3 test system and those of an OCCI controlled Cloud system:

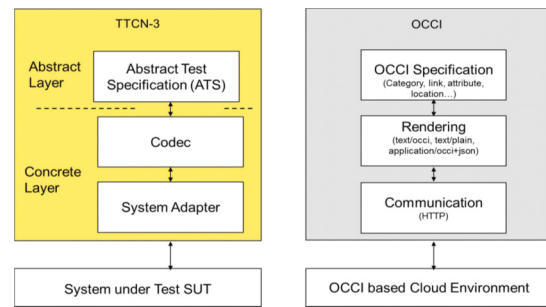


Figure 3: Mapping TTCN-3 - OCCI.

Elements formatted according to the OCCI specification can be expressed in terms of a TTCN-3 Abstract Test Specification. The rendering of the different MIME types will be accomplished by the Codec. The OCCI transport via HTTP will be provided by the System Adapter.

For example, the OCCI “Category” can be abstracted into the following TTCN-3 Data type:

```

Category {
    charstring category,
    CategoryValue category_value
}
type set CategoryValue {
    charstring term,
    charstring scheme,
    charstring class,
    charstring title optional,
    charstring rel optional,
    charstring location optional,
    charstring attributes optional,
    charstring actions optional
}
type set of Category CategoryList;
type record Category {
    charstring category,
    CategoryValue category_value
}

type set CategoryValue {
    charstring term,
    charstring scheme,
    charstring class,
    charstring title optional,
    charstring rel optional,
    charstring location optional,
    charstring attributes optional,
    charstring actions optional
}
type set of Category CategoryList;
    
```

In order to carry out the ETSI test case “TD/OCCI/INDRA/CREATE/004: Create an OCCI Compute Resource” one has to create the following TTCN-3 request template:

```

template OCCIReq
Req_TD_OCCI_INFRA_CREATE_004 := {
  url_req := {
    scheme := "http://",
    authority :=
      "rocci.herokuapp.com",
    path := "/compute/"
  },
  category_list := {
    {
      category := "Category",
      category_value := {
        term := "compute",
        scheme :=
          "http://schemas.occi.org/occi/infrastructure#",
        class := "kind"
      },
    },
    {
      category := "Category",
      category_value := {
        term := "small",
        scheme :=
          "http://my.occi.service/occi/infrastructure/resource_tpl#",
        class := "mixin"
      },
    },
    {
      category := "Category",
      category_value := {
        term := "my_os",
        scheme :=
          "http://my.occi.service/occi/infrastructure/os_tpl#",
        class := "mixin"
      },
    },
  },
  link_list := omit,
  x_occi_attribute_list := omit
}

```

This template represents the test oracle, i.e. the expected response of the SUT, for this conformance test.

The related HTTP verbs GET, POST, PUT and Delete and the OCCI rendering have to be parameterized as follows:

```

/* select HTTP verb */
modulepar boolean Create := true;
modulepar boolean Read := false;
modulepar boolean Update := false;
modulepar boolean Delete := false;

/* select OCCI Rendering */
modulepar charstring ContentType :=
"text/occi";
modulepar charstring AcceptValue :=
"text/occi";

```

The annotated Figure 4 shows the corresponding result of the test:

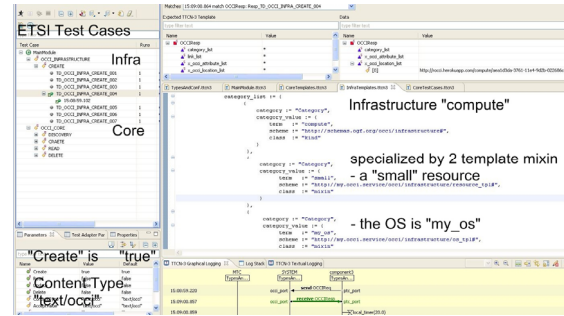


Figure 4: Creating an Infrastructure OCCI Compute resource modified by two mixins.

The tool window (TTworkbench, 2015) is showing:

- the list of all the implemented ETSI tests - the currently executed is highlighted (left upper corner)
- the action “create” and the related content type “text/occi”
- a “compute” “kind” modified by the two “mixins” (large window, middle right; see Figure 1 again for terminology); (the small window, upper corner right, is showing that the compute resource was created on a server of the PaaS provider HEROKU used by EGI for testing purposes).
- the OCCI Request/Response message exchange between the System_under_Test and the Test System (graphical window right bottom; the Verdict “pass” message is just not visible);).

8.2 TTCN-3 and “BonFIRE OCCI”

BonFIRE a recent EU project has realized and is providing a multi-site testbed on top of seven Cloud infrastructures operated by seven project partners. BonFIRE IaaS offers heterogeneous compute, storage and network resources, (BonFIRE, 2014).

In the given context, the main features of the BonFIRE (BF) architecture are the following:

- BF implements an “almost” OCCI-based resource manager on top of the participating IaaS testbed sites (no Categories etc., no MIXINS).
- The rendering uses the private type “application/vnd.bonfire+xml”
- BF provides a monitoring capability at both the VM and physical level. Under user control events generated by (Zabbix) monitoring agents are transported via AMQP to an “Aggregator”. From a functional point of view, the BF monitoring fits well the “Focused Technical (security) Requirements” of (NIST, 2014) Part II, “Visibility/Control for Consumers”.

- BF provides an experimental EaaS – Elasticity as a Service - across the test bed sides.

Formally, according to the BF data model, the BF user carries out “Experiments”. In a full OCCI setting “Experiments” would be defined as a Category above the participating infrastructures. Except for the description part and the fixed allocation of monitoring agents to user created VMs the monitoring architecture is close to the proposal presently discussed within the OGF OCCI WG.

The annotated Figure 5 shows:

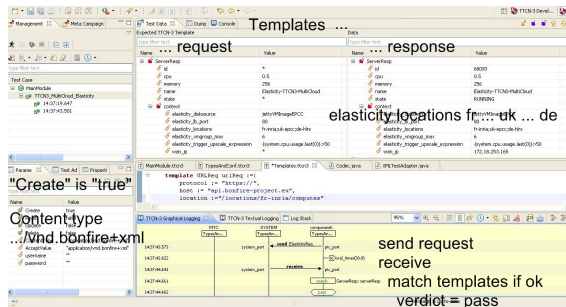


Figure 5: Creating a BonFire elasticity group.

- the creation of a elasticity group distributed over several BonFire geographical sites in France, the UK and Germany - in response to the request template (upper part right)
- the related action is (naturally) “create”
- (left below) the rendering’s private type “application/vnd.bonfire+xml”
- the verdict “pass” message (graphical window part).

Not considering the only “almost” OCCI compliance of the project BonFire is a clear and working example for the potential of OCCI beyond its initial specification.

9 SUMMARY AND OUTLOOK

Using Cloud related work of NIST and ETSI we have presented standardized testing of standard-based Cloud infrastructures as a necessary element of a Cloud quality assessment framework. We have shown that OCCI is well positioned to play a pivotal role within that context.

Assuming a key role of SDN/NFV in future Cloud provisioning we have also pointed to work using OCCI in SDN/NFV settings of Cloud infrastructures.

Then we have introduced the ETSI effort towards model-based testing – comprising TTCN-3 at the lowest layer.

In summary we propose – as strategically vision behind our effort - to adopt the Cloud world as the next big application field of the well-established ETSI TTCN-3-related testing methodologies.

Finally, as a proof-of-concept we demonstrated “standardized” TTCN-3 test cases against OCCI controlled Cloud test beds.

In order to gather and solicit support for our vision future work will include true interoperability tests in the spirit of ETSI CSC and further test types such as performance tests. If SDN/NFV Cloud infrastructures such as in the OCCI-oriented MCN become available tests exploiting advanced features both of TTCN-3 and OCCI are foreseen.

ACKNOWLEDGEMENTS

We would like to thank – Boris Parák of CESNET for support in using the rOCCI/EGI infrastructure – the former colleagues of BonFire for provision of the BonFire testbed – Prof. Ina Schieferdecker for guidance in the TTCN-3 world – Testing Technologies for the friendly provision of their TTCN-3 tool TTworkbench. The paper is partly funded by the EU project CloudSocket H2020-644690.

REFERENCES

- BonFire, 2014. www.bonfire-project.eu/. Accessed January 06, 2015.
- CompatibleOne, 2015. CompatibleOne The Open Source Cloud broker. <http://www.compatibleone.org/>. Accessed January 06, 2015.
- Edmonds, A., Metsch, T., Luster, E., 2011. An Open, Interoperable Cloud. <http://www.infoq.com/articles/open-interoperable-cloud>. Accessed January 06, 2015.
- Edmonds, A. et al. Towards an Open Cloud Standards. IEEE Internet Computing. July/August 2012.
- EGI, 2015. EGI European Grid Infrastructure. <https://www.egi.eu/infrastructure/cloud/>.
- ETSI, 2012. <http://www.etsi.org/technologies-clusters/technologies/nfv>. Accessed January 06, 2015.
- ETSI, 2013a. Cloud Standards Coordination Final Report November 2013 VERSION 1.0. November 2013.
- ETSI, 2013b. TS 103 142 V2.0.2 (2013-09) CLOUD; Test Descriptions for Cloud Interoperability.
- ETSI, 2014. NFV Proofs of Concept. <http://www.etsi.org/technologies-clusters/technologies/nfv/nfv-poc>. Accessed January 06, 2015.
- ETSI, 2015. TC Methods for Testing and Specification <http://www.etsi.org/images/files/ETSI>

- TechnologyLeaflets/MethodsforTestingand Specification.pdf. Accessed January 06, 2015.
- ETSI, 2015a. <http://www.etsi.org/technologies-clusters/technologies/testing/ims-testing>. Accessed January 27, 2015.
- Liang, Y., 2013a. Harnessing TTCN-3 Test Framework for OCCI-based Cloud Ecosystems. In DMTF, ETSI, OASIS, OCEAN Project, OGF, OW2 and SNIA. First Cloud Interoperability Week Santa Clara, USA, September 16-18 and Madrid, Spain, September 18-20, 2013 (co-hosted with the EGI and SDC conferences) <http://www.cloudplugfest.org/events/past-plugfest-agendas/cloud-interoperability-week/workshop> Accessed December 14, 2014.
- Liang, Y., 2013b. Towards a TTCN-3 test framework for OCCI-based Cloud ecosystems. In UCAAT, 1st User Conference on Advanced Automated Testing. Paris 22-24 October 2013. http://ucaat.etsi.org/2013/program_conf.html Accessed December 14, 2014.
- Makedonski, P. et al., 2014. Bringing TDL to Users: A Hands-on Tutorial. <http://www.swe.informatik.uni-goettingen.de/sites/default/files/publications/TDL%20Tutorial.pdf>. Accessed January 06, 2015.
- MCN, 2014. Mobile Cloud Networking project. <http://www.mobile-cloud-networking.eu/site/> (accessed December 14, 2014).
- MCN 2013 FUTURE COMMUNICATION ARCHITECTURE FOR MOBILE CLOUD SERVICES. FP7-ICT-2011-8 Project No: 318109. D3.1 Infrastructure Management Foundations – Specifications & Design for Mobile Cloud framework. 08 November 2013.
- NIST, 2014. Special Publication 500-293. Version 2. US Government Cloud Computing Technology Roadmap. Volume I. High-Priority Requirements to Further USG Agency Cloud Computing Adoption. Volume II. Useful Information for Cloud Adopters <http://dx.doi.org/10.6028/NIST.SP.500-293>. October 2014.
- NIST, 2015. Koala. In “Measurement Science for Complex Information Systems”. http://www.nist.gov/itl/antd/emergent_behavior.cfm. Accessed January 06, 2015.
- NIST, 2014b. Special Publication 500-299. Draft. NIST Cloud Computing Security Reference Architecture.
- OCCI, 2011a. <http://occi-wg.org/about/>. Accessed January 06, 2015.
- OCCI, 2011b. Core Specification: <http://ogf.org/documents/GFD.183.pdf>.
- OCCI, 2011c. Infrastructure: <http://ogf.org/documents/GFD.184.pdf>.
- OCCI, 2011d. HTTP Rendering: <http://ogf.org/documents/GFD.185.pdf>.
- OGF, 2012a. Grokking OCCI Syntax: OCCI ANTLR Grammar. <http://occi-wg.org/2012/02/29/occi-antlr-grammar/>. Accessed January 06 2015.
- OGF, 2012b. Do you Speak OCCI? <http://occi-wg.org/2012/03/05/do-you-speak-occi/>.
- OGF, 2014. OGF42 Updates from the Group. http://occi-wg.org/2014/09/15/updates_from_ogf42/. Accessed January 06 2015.
- ONF, 2011. <https://www.opennetworking.org/>. Accessed January 06, 2015.
- Schieferdecker, I. Oracles in TTCN-3 and UTP. In CREST Workshop. 2012, May 22nd, London.
- Stepien, B. Peyton, L. 2014 “Innovation and Evolution in Integrated Web Application Testing with TTCN-3”, International Journal on Software Tools for Technology Transfer. June 2014, Volume 16, Issue 3, pp 269-283.
- Sill, A., 2013. SAJACC: The NIST Cloud Use Case Test Definition Project. In - same as (Liang, 2013a).
- TTCN-3, 2013. <http://www.ttcn-3.org/>. Accessed January 06, 2015.
- TTworkbench, 2015. <http://www.testingtech.com/products/ttworkbench.php>.

Using Cloud-Aware Provenance to Reproduce Scientific Workflow Execution on Cloud

Khawar Hasham, Kamran Munir and Richard McClatchey

Centre for Complex Computing Systems (CCCS), Faculty of Environment and Technology (FET),

University of the West of England (UWE), Bristol, U.K.

{khawar.ahmad, kamran.munir, richard.mcclatchey}@cern.ch

Keywords: Cloud Computing, Scientific Workflows, Provenance, Reproducibility, Repeatability.

Abstract: Provenance has been thought of a mechanism to verify a workflow and to provide workflow reproducibility. This provenance of scientific workflows has been effectively carried out in Grid based scientific workflow systems. However, recent adoption of Cloud-based scientific workflows present an opportunity to investigate the suitability of existing approaches or propose new approaches to collect provenance information from the Cloud and to utilize it for workflow reproducibility on the Cloud infrastructure. This paper presents a novel approach that can assist in mitigating this challenge. This approach can collect Cloud infrastructure information along with workflow provenance and can establish a mapping between them to provide a Cloud-aware provenance. The reproducibility of the workflow execution is performed by: (a) capturing the Cloud infrastructure information (virtual machine configuration) along with the workflow provenance, (b) re-provisioning the similar resources on the Cloud and re-executing the workflow on them and (c) by comparing the outputs of workflows. The evaluation of the prototype suggests that the proposed approach is feasible and can be investigated further. Since there is no reference model for workflow reproducibility on Cloud exists in the literature, this paper also attempts to present a model that is used in the proposed design to achieve workflow reproducibility in the Cloud environment.

1 INTRODUCTION

The scientific community is processing and analysing huge amounts of data being generated in modern scientific experiments that include projects such as DNA analysis (Foster et al., 2008), the Large Hadron Collider (LHC) (<http://lhc.cern.ch>), and projects such as neuGRID (Mehmood et al., 2009) and its follow-on neuGRIDforUsers (Munir et al., 2013, 2014). In particular the neuGRID community is utilising scientific workflows to orchestrate the complex analysis of neuro-images to diagnose Alzheimer disease. A large pool of compute and data resources are required to process this data, which has been available through the Grid (Foster et al., 1999) and is now also being offered by the Cloud-based infrastructures.

Cloud computing (Mell and Grance, 2011) has emerged as a new computing and storage paradigm, which is dynamically scalable and usually works on a pay-as-you-go cost model. It aims to share resources to store data and to host services transparently among users at a massive scale (Mei et al., 2008). Its ability to provide an on-demand computing infrastructure

enables distributed processing of scientific workflows (Deelman et al., 2008) with increased complexity and data requirements. Recent work (Juve and Deelman 2010) is now experimenting with Cloud infrastructures to assess the feasibility of executing workflows on the Cloud.

An important consideration during this data processing is to gather provenance (Simmhan et al., 2005) information that can provide detailed information about both the input and the processed output data, and the processes involved in a workflow execution. This information can be used to debug the execution of a workflow, to aid in error tracking and reproducibility. This vital information can enable scientists to verify the outputs and iterate on the scientific method, to evaluate the process and results of other experiments and to share their own experiments with other scientists (Azarnoosh et al., 2013). The execution of scientific workflows in the Cloud brings to the fore the need to collect provenance information that is necessary to ensure the reproducibility of these experiments on the Cloud infrastructure

A research study (Belhajjame et al., 2012)

conducted to evaluate the reproducibility of scientific workflows has shown that around 80% of the workflows cannot be reproduced, and 12% of them are due to the lack of information about the execution environment. This information affects a workflow on two levels. It can affect a workflow's overall execution performance and also job failure rate. For instance, a data-intensive job can perform better with 2GB of RAM because it can accommodate more data in RAM, which is a faster medium than hard disk. However, the job's performance will degrade if a resource of 1GB RAM is allocated to this job as less data can be placed in RAM. Moreover, it is also possible that jobs will remain in waiting queues or fail during execution if their required hardware dependencies are not met. This becomes a more challenging issue in the context of Cloud in which resources can be created or destroyed at runtime.

The dynamic and geographically distributed nature of Cloud computing makes the capturing and processing of provenance information a major research challenge (Vouk 2008, Zhao et al., 2011). Since the Cloud presents a transparent access to dynamic execution resources, the workflow parameters including execution resource configuration should also be known to a scientist (Shamdasani et al., 2012) i.e. what execution environment was used for a job in order to reproduce a workflow execution on the Cloud. Due to these reasons, there is a need to capture information about the Cloud infrastructure along with workflow provenance, to aid in the reproducibility of workflow experiments. There has been a lot of research related to provenance in the Grid (Foster et al., 2002, Stevens et al., 2003) and a few initiatives (Oliveira et al., 2010, Ko et al., 2011) for the Cloud. However, they lack the information that can be utilised for re-provisioning of resources on the Cloud, thus they cannot create the similar execution environment(s) for workflow reproducibility. In this paper, the terms "Cloud infrastructure" and "virtualization layer" are used interchangeably.

This paper presents a theoretical description of an approach that can augment workflow provenance with infrastructure level information of the Cloud and use it to provision similar execution environment(s) and repeat a given workflow. Important points discussed in this paper are as follows: section 2 presents some related work in provenance related systems. Section 3 presents a reproducibility model designed after collecting guidelines used and discussed in literature. Section 4 presents an overview of the proposed approach. Section 5

presents an evaluation of the developed prototype. And finally section 6 presents some conclusions and directions for future work.

2 RELATED WORK

Significant research (Foster et al., 2002, Scheidegger et al., 2008) has been carried out in workflow provenance for Grid-based workflow management systems. Chimera (Foster et al., 2002) is designed to manage the data-intensive analysis for high-energy physics (GriPhyN) (GriPhyN 2014) and astronomy (SDSS) (SDSS 2014) communities. It captures process information, which includes the runtime parameters, input data and the produced data. It stores this provenance information in its schema, which is based on a relational database. Although the schema allows storing the physical location of a machine, it does not support the hardware configuration and software environment in which a job was executed. Vistrails (Scheidegger et al., 2008) provides support for scientific data exploration and visualization. It not only captures the execution log of a workflow but also the changes a user makes to refine his workflow. However, it does not support the Cloud virtualization layer information. Similar is the case with Pegasus/Wings (Kim et al. 2008) that supports evolution of a workflow. However, this paper is focusing on the workflow execution provenance on the Cloud, rather than the provenance of a workflow itself (e.g. design changes).

There have been a few research studies (Oliveira et al., 2010, Ko et al., 2011) performed to capture provenance in the Cloud. However, they lack the support for workflow reproducibility. Some of the work in Cloud towards provenance is directed to the file system (Zhang et al., 2011, Shyang et al 2012) or hypervisor level (Macko et al., 2011). However such work is not relatable to our approach because this paper focuses on virtualized layer information of the Cloud for workflow execution. Moreover, the collected provenance data provides information about the file access but it does not provide information about the resource configuration. The PRECIP (Azarnoosh et al., 2013) project provides an API to provision and execute workflows. However, it does not provide provenance information of a workflow.

There have been a few recent projects (Chirigati et al., 2013, Janin et al., 2014) and research studies (Perez et al., 2014a) on collecting provenance and using it to reproduce an experiment. A semantic-based approach (Perez et al., 2014b) has been

proposed to improve reproducibility of workflows in the Cloud. This approach uses ontologies to extract information about the computational environment from the annotations provided by a user. This information is then used to recreate (install or configure) that environment to reproduce a workflow execution. On the contrary, our approach is not relying on annotations rather it directly interacts with the Cloud middleware at runtime to acquire resource configuration information and then establishes mapping between workflow jobs and Cloud resources. The ReproZip software (Chirigati et al., 2013) uses system call traces to provide provenance information for job reproducibility and portability. It can capture and organize files/libraries used by a job. The collected information along with all the used system files are zipped together for portability and reproducibility purposes. Since this approach is useful at individual job level, this does not work for an entire workflow, which is the focus of this paper. Moreover, this approach does not consider the hardware configuration of the underlined execution machine. Similarly, a Linux-based tool, CARE (Janin et al., 2014), is designed to reproduce a job execution. It builds an archive that contains selected executable/binaries and files accessed by a given job during an observation run.

3 WORKFLOW REPRODUCIBILITY MODEL ON CLOUD

As per our understanding of the literature, there is not a standard reproducibility model proposed so far for scientific workflows, especially in Cloud environment. However, there are some guidelines or policies, which have been highlighted in literature to reproduce experiments. There is one good effort (Sandve et al., 2013) in this regard, but it mainly talks about reproducible papers and it does not consider execution environment of workflows. In this section, we have gathered basic points to present an initial workflow reproducibility model in Cloud that can provide guidelines for future work in this regard. These points are discussed as follows.

- **Share Code and Data**

The need for data and code sharing in computational science has been widely discussed (Stodden 2010). In computational science conservation, in particular for scientific workflow executions, it is emphasized that the data, code, and the workflow description

should be available in order to reproduce an experiment.

- **Execution Infrastructure details**

The execution infrastructure provided by the Grid or Cloud to execute a workflow is composed of a set of computational and storage resources (e.g. execution nodes, storage devices, networking). The physical approach, where actual computational hardware are made available for long time period to scientists, often conserves the computational environment including supercomputers, clusters, or Grids (Perez et al., 2014b). As a result, scientists are able to reproduce their experiments on the same hardware environment. However, this luxury is not available in the Cloud environment in which resources are virtualized and provisioned dynamically on-demand. A little focus is given to the underlying infrastructure, especially Cloud, in computational conservation in literature. This challenge has been tackled in this paper by collecting this information at runtime from the Cloud infrastructure. From resource provisioning point of view, parameters such as RAM, vCPU and Hard Disk are important in selecting appropriate resource especially on the Cloud and should be made part of the collected provenance. All these factors contribute to the job's execution performance as well as to its failure rate. For instance, a job will fail if it is scheduled to a resource with insufficient available RAM.

- **Software Environment**

Apart from knowing the hardware infrastructure, it is also essential to provide information about the software environment. A software environment determines the operating system and the libraries used to execute a job. Without the access to required libraries information, a job execution will fail. For example, a job, relying on MATLAB library, will fail in case the required library is missing. One possible approach (Howe et al., 2012) to conserve software environment is thought to conserve VM that is used to execute a job and then reuse the same VM while re-executing the same job. One may argue that it would be easier to keep and share VM images with the research community through a common repository, however the high storage demand of VM images remains a challenging problem (Zhao et al., 2014). In the prototype presented in this paper, the OS image used to provision a VM is conserved and thought to present all the software dependencies required for a job execution in a workflow. Therefore, the proposed solution should also retrieve the image information to build a virtual machine on which the workflow job was executed.

• Workflow Versioning

Capturing only a provenance trace is not sufficient to allow a computation to be repeated – a situation known as workflow decay (Roure et al., 2011). The reason is that the provenance systems can store information on how the data was generated, however they do not store copies of the key actors in the computation i.e. workflow, services, data. This paper (Sandve et al. 2013) suggests to archive the exact versions of all programs and enable version control on all scripts used in an experiment. This is not supported in the presented prototype, but it will be incorporated in next iterations.

• Provenance Comparison

The provenance traces of two executed workflows should be compared to determine workflow reproducibility. The main idea is to evaluate the reproducibility of an entire execution of a given workflow, including the logical chaining of activities and the data. To provide the strict reproducibility functionality, a system must guarantee that the data are still accessible and that the corresponding activities are accessible (Lifschitz et al. 2011). Since the focus of this paper is on workflow reproducibility on the Cloud infrastructure, the execution infrastructure should also be part of the comparison. Therefore the provenance comparison should be made at different levels; workflow structure, execution infrastructure, and workflow input and output. A brief description of this comparison is given below.

- Workflow structure should be compared to determine that both workflows are similar. Because it is possible that two workflows are having similar number of jobs but with different job execution order.
- Execution infrastructure (software environment, resource configuration) used on the Cloud for a workflow execution should also be compared.
- Comparison of input and output should be made to evaluate workflow reproducibility. There could be a scenario that a user repeated a workflow but with different inputs, thus producing different outputs. It is also possible that changes in job or software library result into different workflow output. There are a few approaches (Missier et al. 2013), which perform workflow provenance comparison to determine differences in reproduced workflows. The proposed approach in this paper incorporates the workflow output

comparison to determine the reproducibility of a workflow.

• Pricing Model

This point can be important for experiments in which cost is also a main factor. The resource provisioned on the Cloud has associated cost, which is based on the resource type and the amount of time it has been used for. This information can assist in reproducing an experiment with the same cost as was incurred in earlier execution. This point is not incorporated in the proposed design at the moment.

4 CLOUD-AWARE PROVENANCE APPROACH

An abstract view of the proposed architecture is presented in this section. This architecture is designed after evaluating the existing literature and keeping in mind the objectives of this research study. The proposed architecture is inspired by the mechanism used in a paper (Groth et al., 2009) for executing workflows on the Cloud. Figure 1 illustrates the proposed architecture that is used to capture the Cloud infrastructure information and to interlink it with the workflow provenance collected from a workflow management system such as Pegasus. This augmented or extended provenance information comprising of workflow provenance and the Cloud infrastructure information is named as Cloud-aware provenance. The components of this architecture are briefly explained below.

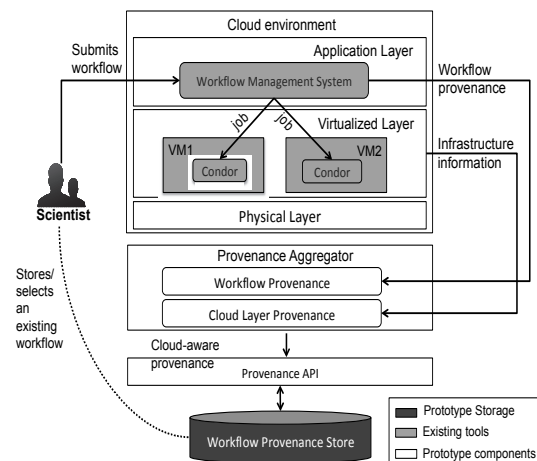


Figure 1: An abstract architecture of the proposed approach.

- **Workflow Provenance:** This component is responsible for receiving provenance captured at

the application level by the workflow management system (Pegasus). Since workflow management systems may vary, a plugin-based approach is used for this component. Common interfaces are designed to develop plugins for different workflow management systems. The plugin also translates the workflow provenance according to the representation that is used to interlink the workflow provenance along with the information coming from the Cloud infrastructure.

- **Cloud Layer Provenance:** This component is responsible for capturing information collected from different layers of the Cloud. To achieve re-provisioning of resources on Cloud, this component focuses on the virtualization layer and retrieves information related to the Cloud infrastructure i.e. virtual machine configuration. This component is discussed in detail in section 4.1.
- **Provenance Aggregator:** This is the main component tasked to collect and interlink the provenance coming from different layers as shown in Figure 1. It establishes interlinking connections between the workflow provenance and the Cloud infrastructure information. The provenance information is then represented in a single format that could be stored in the provenance store through the interfaces exposed by the Provenance API.
- **Provenance API:** This acts as a thin layer to expose the provenance storage capabilities to other components. Through its exposed interfaces, outside entities such as the Provenance Aggregator would interact with it to store the workflow provenance information. This approach gives flexibility to implement authentication or authorization in accessing the provenance store.
- **Workflow Provenance Store:** This data store is designed to store workflows and their associated provenance. This also keeps mapping between workflow jobs and the virtual compute resources in the Cloud infrastructure. This also keeps record of the workflow and its related configuration files being used to submit a user analysis on the Cloud. This information is later retrieved to reproduce the execution. However, it does not support workflow evolution in its current design.

4.1 Job to Cloud Resource Mapping

The CloudLayerProvenance component is designed in a way that interacts with the Cloud infrastructure as an outside client to obtain the resource configuration information. As mentioned earlier, this

information is later used for reprovisioning the resources to provide a similar execution infrastructure to repeat a workflow execution. Once a workflow is executed, Pegasus collects the provenance and stores it in its own internal database. Pegasus also stores the IP address of the virtual machine (VM) where the job is executed. However, it lacks other VM specifications such as RAM, CPUs, hard disk etc. The CloudLayerProvenance component retrieves all the jobs of a workflow and their associated VM IP addresses from the Pegasus database. It then collects a list of virtual machines owned by a respective user from the Cloud middleware. Using the IP address, it establishes a mapping between the job and the resource configuration of the virtual machine used to execute the job. This information i.e. Cloud-aware provenance is then stored in the Provenance Store. The flowchart of this mechanism is presented in Figure 2.

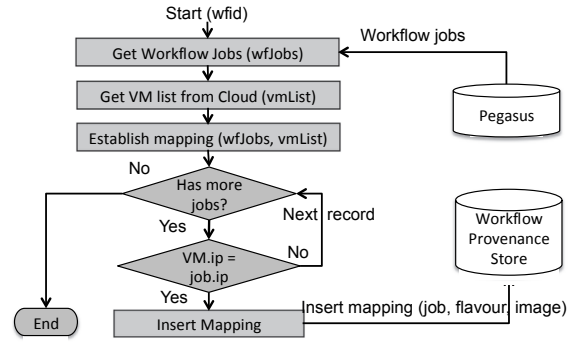


Figure 2: flowchart of job to Cloud resource mapping performed by ProvenanceAggregator component.

In this flowchart, the variable *wfJobs* – representing a list of jobs of a given workflow – is retrieved from the Pegasus database. The variable *vmList* – represents a list of virtual machines in the Cloud infrastructure – is collected from the Cloud. A mapping between jobs and VMs is established by matching the IP addresses (see in Figure 2). Resource configuration parameters such as flavour and image are obtained once the mapping is established. *flavour* defines resource configuration such as RAM, Hard disk and CPUs, and *image* defines the operating system image used in that particular resource. By combining these two parameters together, one can provision a resource on the Cloud infrastructure. After retrieving these parameters and jobs, the mapping information is then stored in the Provenance Store (see in Figure. 2). This mapping information provides two important data (a) hardware configuration (b)

software configuration using VM name. As discussed in section 3, these two parameters are important in recreating a similar execution environment.

4.2 Workflow Reproducibility using Cloud-Aware Provenance

In section 4.1, the job to Cloud resource mapping using provenance information has been discussed. This mapping is stored in the database for workflow reproducibility purposes. In order to reproduce a workflow execution, researcher first needs to provide the *wfID* (workflow ID), which is assigned to every workflow in Pegasus, to the proposed framework to re-execute the workflow using the Cloud-aware provenance. It retrieves the given workflow from the Provenance Store database (step 2(a) in Figure 3) along with the Cloud resource mapping stored against this workflow (step 2(b) in Figure 3). Using this mapping information, it retrieves the resource flavour and image configurations, and provisions the resources (step 3 in Figure 3) on Cloud. Once resources are provisioned, it submits the workflow (step 4).

At this stage, a new workflow ID is assigned to this newly submitted workflow. This new *wfID* is passed over to the ProvenanceAggregator component to monitor (step 5) the execution of the workflow and start collecting its Cloud-aware provenance information (see step 6 in Figure 3) This is important to recollect the provenance of the repeated workflow, as this will enable us to verify the provisioned resources by comparing their resource configurations with the old resource configuration.

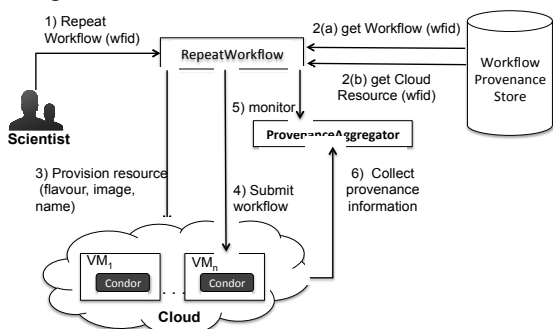


Figure 3: The sequence of activities to illustrate workflow repeatability in the proposed system.

4.3 Workflow Output Comparison

Another aspect of workflow repeatability is to verify that it has produced the same output that was

produced in its earlier execution (as discussed in section 3). In order to evaluate workflow repeatability, an algorithm has been proposed that compares the outputs produced by two given workflows. It uses the MD5 hashing algorithm (Stalling 2010) on the outputs and compares the hash value to verify the produced outputs. The two main reasons of using a hash function to verify the produced outputs are; a) simple to implement and b) the hash value changes with a single bit change in the file. If the hash values of two given files are same, this means that the given files contain same content.

The proposed algorithm (as shown in Figure 4) operates over the two given workflows identified by *srcWfID* and *destWfID*, and compares their outputs. It first retrieves the list of jobs and their produced output files from the Provenance Store for each given workflow. It then iterates over the files and compares the source file, belonging to *srcWfID*, with the destination file, belonging to *destWfID*. Since the files are stored on the Cloud, the algorithm retrieves the files from the Cloud (see lines 11 and 12). Cloud storage services such as OpenStack Swift, Amazon Object Store use the concept of a bucket or a container to store a file. This is why *src_container* and *dest_container* along with *src_filename* and *dest_filename* are given in the *GetCloudFile* function to identify a specific file in the Cloud. The algorithm then compares the hash value of both files and increments *ComparisonCounter*. If all the files in both workflows are the same, *ComparisonCounter* should be equal to *FileCounter*, which counts the number of files produced by a workflow. Thus, it confirms that the workflows are repeated successfully. Otherwise, the algorithm returns false if both these counters are not equal.

Algorithm 3 Compare Workflow Outputs Algorithm

Require: *srcWfID* : Source Workflow ID.

destWfID : Destination Workflow ID

```

1: procedure COMPAREWORKFLOWOUTPUTS(srcWfID, destWfID)
2:   srcWorkflowJobs ← GETWORKFLOWJOBS(srcWfID)
3:   destWorkflowJobs ← GETWORKFLOWJOBS(destWfID)
4:   FileCounter ← 0
5:   ComparisonCounter ← 0
6:   for all jobfiles ∈ srcWorkflowJobs do
7:     src_container ← jobfiles.container_name
8:     src_filename ← jobfiles.file_name
9:     dest_container ← destWorkflowJobs[jobfiles.jobname]
10:    dest_filename ← destWorkflowJobs[jobname].file_name
11:    src_cloud_file ← GETCLOUDFILE(src_container src_filename)
12:    dest_cloud_file ← GETCLOUDFILE(dest_container dest_filename)
13:    FileCounter ← FileCounter + 1
14:    if src_cloud_file.hash = dest_cloud_file.hash then
15:      ComparisonCounter ← ComparisonCounter + 1
16:   if FileCounter = ComparisonCounter then
17:     return True
18:   return False

```

Figure 4: Pseudocode to compare outputs produced by two given workflows.

5 RESULTS AND DISCUSSION

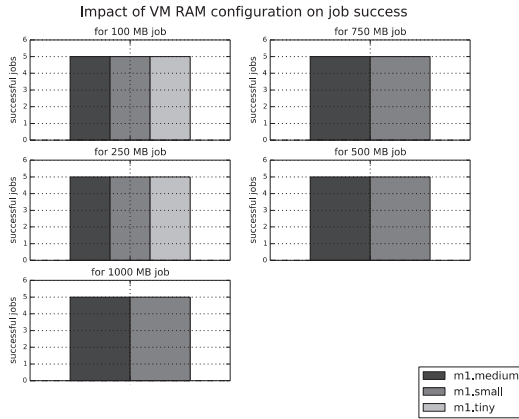


Figure 5: Cloud resource's RAM configuration impact on job success.

To demonstrate the effect of Cloud resource configuration such as RAM on job failure rate, a basic memory-consuming job is written in Java language. The job attempts to construct an alphabet string of given size (in MB), which is provided at runtime. To execute this experiment, three resource configurations, (a) m1.tiny, (b) m1.small and (c) m1.medium, each with 512 MB, 2048 MB and 4096 MB RAM respectively were used. Each job is executed at least 5 times with a given memory requirement on each resource configuration. The result in Figure 5 shows that jobs fail if required RAM (hardware) requirement is not fulfilled. All jobs with RAM requirement less than 500 MB executed successfully on all resource configurations. However, the jobs start to fail on Cloud resources with m1.tiny configuration as soon as the job's memory requirement approaches 500 MB because the jobs could not find enough available memory on the given resource. This result confirms the presented argument (discussed in section 1 and also in section 3) regarding the need for collecting Cloud resource configuration and its impact on job failure. Since a workflow is composed of many jobs, which are executed in a given order, a single job failure can result in a workflow execution failure. Therefore, collecting Cloud-aware provenance is essential for reproducing a scientific workflow execution on the Cloud.

To evaluate the presented mapping algorithm, which collects the Cloud infrastructure information and interlinks it with the workflow provenance, a Python based prototype has been developed using Apache Libcloud (Apache Libcloud – <http://libcloud.apache.org>), a library to interact with

the Cloud middleware. The presented evaluation of the prototype is very basic currently. However, as this work progresses further a full evaluation will be conducted. To evaluate this prototype, a 20 core Cloud infrastructure is acquired from the Open Science Data Cloud (OSDC) (<https://www.opensciencedatacloud.org/>). This Cloud infrastructure uses the OpenStack middleware (openstack.org) to provide infrastructure-as-a-Service capability. A small Condor cluster of three virtual machines is also configured. In this cluster, one machine is a master node, which is used to submit workflows, and the remaining two are compute nodes. These compute nodes are used to execute workflow jobs. Using the Pegasus APIs, a basic *wordcount* workflow application composed of four jobs is written. This workflow has both control and data dependencies (Ramakrishnan and Plale, 2010) among its jobs, which is a common characteristic in scientific workflows. The first job (*Split* job) takes a text file and splits it into two files of almost equal length. Later, two jobs (*Analysis* jobs), each take one file as input, and then calculate the number of words in the given file. The fourth job (*merge* job) takes the outputs of earlier analysis jobs and calculates the final result i.e. total number of words in both files.

This workflow is submitted using Pegasus. The wfID assigned to this workflow is 114. The collected Cloud resource information is stored in database. Table I. shows the provenance mapping records in the Provenance Store for this workflow. The collected information includes the *flavour* and *image* (*image name* and *Image id*) configuration parameters. The *Image id* uniquely identifies an OS image hosted on the Cloud and this image contains all the software or libraries used during the job execution. As an image contains all the required libraries of a job, this prototype does not extract the installed libraries information from the virtual machine at the moment for workflow reproducibility purpose. However, this can be done in future iterations to enable the proposed approach to reconfigure a resource at runtime on the Cloud.

The reproducibility of the workflow using the proposed approach (discussed in section 4.2) has also been tested. The prototype is requested to repeat the workflow with wfID 114.

Upon receiving the request, it first collects the resource configurations, captured from earlier execution, from the database and provisions the resources on the Cloud infrastructure. The name of re-provisioned resource(s) for the repeated workflow has a postfix '-rep.novalocal' e.g. *mynova-*

Table 1: Cloud infrastructure mapped to the jobs of workflow with ID 114.

wfID	Host IP	nodename	Flavour Id	minRAM (MB)	minHD (GB)	vCPU	Image name	Image id
114	172.16.1.49	osdc-vm3.novalocal	2	2048	20	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169
114	172.16.1.98	mynode.novalocal	2	2048	20	1	wf_peg_repeat	102960c-557c-4253-8277-2df5ffe3c169

Table 2: Cloud infrastructure information of repeated workflow (wfIDs: 117 and 122) after repeating workflow 114.

wfID	Host IP	nodename	Flavour Id	minRAM (MB)	minHD (GB)	vCPU	Image name	Image id
117	172.16.1.183	osdc-vm3-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169
117	172.16.1.187	mynode-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169
122	172.16.1.114	osdc-vm3-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169
122	172.16.1.112	mynode-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169

Table 3: Comparing outputs produced by workflows 114 (original workflow) and 117 (repeated workflow).

Job	WF ID	Container Name	File Name	MD5 Hash
Split	114	wfoutput123011	wordlist1	0d934584cbc124eed93c4464ab178a5d
	117	wfoutput125819	wordlist1	0d934584cbc124eed93c4464ab178a5d
	114	wfoutput123011	wordlist2	1bc6ffead85bd62b5a7a1be1dc672006
	117	wfoutput125819	wordlist2	1bc6ffead85bd62b5a7a1be1dc672006
Analysis 1	114	wfoutput123011	analysis1	494f24e426dba5cc1ce9a132d50ccbd
	117	wfoutput125819	analysis1	494f24e426dba5cc1ce9a132d50ccbd
Analysis 2	114	wfoutput123011	analysis2	127e8dbd6beffdd2e9dfed79d46e1ebc
	117	wfoutput125819	analysis2	127e8dbd6beffdd2e9dfed79d46e1ebc
Merge	114	wfoutput123011	merge_output	d0bd408843b90e36eb8126b397c6efed
	117	wfoutput125819	merge_output	d0bd408843b90e36eb8126b397c6efed

rep.novalocal as shown in Table 2. It was named *mynova.novalocal* in original workflow execution as shown in Table 1. From Table 2, one can assess that similar resources have been re-provisioned using the proposed approach to reproduce the workflow execution because the RAM, Hard disk, vCPUs and image configurations are similar to the resources used for workflow with wfID 114 (as shown in Table 1). This preliminary evaluation confirms that the similar resources on the Cloud can be re-provisioned with the Cloud-aware provenance information collected using the proposed approach (discussed previously in section 4). Table 2 shows two repeated workflow instances of original workflow 114.

The other aspect to evaluate the workflow reproducibility (as discussed in section 3) is to compare the outputs produced by both workflows. This has been achieved using the algorithm presented in Figure 4 (discussed in section 4.3). Four jobs in both the given workflows i.e. 114 and 117 produce the same number of output files (see Table 3). The *Split* job produces two output files i.e. *wordlist1* and *wordlist2*. Two analysis jobs, *Analysis1* and *Analysis2*, consume the *wordlist1* and *wordlist2* files, and produce the *analysis1* and *analysis2* files respectively. The merge job consumes the *analysis1* and *analysis2* files and produces the *merge_output* file. The hash values of these files are shown in the MD5 Hash column of

the Table 3, here both given workflows are compared with each other. For instance, the hash value of *wordlist1* produced by the *Split* job of workflow 117 is compared with the hash value of *wordlist1* produced by the *Split* job of workflow 114. If both the hash values are same, the algorithm returns true. This process is repeated for all the files produced by both workflows. The algorithm confirms the verification of workflow outputs if the corresponding files in both workflows have the same hash values. Otherwise, the verification process fails.

6 CONCLUSION AND FUTURE DIRECTION

In this paper, the motivation and the issues related to workflow reproducibility due to workflow execution on the Cloud infrastructure have been identified. The dynamic nature of the Cloud makes provenance capturing of workflow(s) and their underlying execution environment(s) and their reproducibility a difficult challenge. A workflow reproducibility model (discussed in section 3) has been presented after analysing the literature and workflow execution scenario on the Cloud infrastructure. A proposed architecture has been presented that can augment the existing workflow provenance with the information of the Cloud infrastructure. Combining these two can assist in re-provisioning the similar execution environment to reproduce a workflow execution. The Cloud infrastructure information collection mechanism has been presented in this paper in section 4.1. This mechanism iterates over the workflow jobs and establishes mappings with the resource information available on the Cloud. This job to Cloud resource mapping can then be used to reproduce a workflow execution. The process of reproducing a workflow execution with the proposed approach has been discussed in section 4.2. In this paper, the workflow reproducibility is verified by comparing the outputs produced by the workflows. An algorithm has been discussed in section 4.3 (see Figure 4) that compares the outputs produced by two given workflows. A python-based prototype was developed for evaluating the proposed approach. The results show that the proposed approach can capture the Cloud-aware provenance information (as discussed in section 4) by collecting the information related to Cloud infrastructure (virtual machines) used during a workflow execution. It can then provision a similar execution infrastructure i.e. same resource configura-

tion on the Cloud using the Cloud-aware provenance information to reproduce a workflow execution. In future, the proposed approach will be extended and a detailed evaluation of the proposed approach will be conducted. Different performance matrices such as the impact of the proposed approach on workflow execution time, impact of different resource configuration on workflow execution performance, and total resource provision time will also be measured. In this paper, only workflow outputs have been used to compare two workflows' provenance traces. In future, the comparison algorithm will also incorporate workflow structure and execution infrastructure (as discussed in section 3) to verify workflow reproducibility. The proposed approach has not addressed the issue of securing the stored Cloud-aware provenance. In future, the presented architecture will be extended by adding a security layer on top of the collected Cloud-aware provenance.

ACKNOWLEDGEMENTS

This research work has been funded by a European Union FP-7 project, N4U – neuGrid4Users. This project aims to assist the neuro-scientific community in analysing brain scans using workflows and distributed infrastructure (Grid and Cloud) to identify biomarkers that can help in diagnosing the Alzheimer disease. Besides this, the support provided by OSDC by offering a free Cloud infrastructure of 20 cores is highly appreciated. Such public offerings can really benefit research and researchers who are short of such resources.

REFERENCES

- (2014). GriPhyN: <http://www.phys.utb.edu/griphyn/> [Last visited 30-12-2014].
- (2014). SDSS: <http://www.sdss.org> [Last visited 30-12-2014].
- Azarnoosh, S., Rynge, M., Juve, G., Deelman, E., Niec, M., Malawski, M., and da Silva, R. (2013). Introducing precip: An api for managing repeatable experiments in the cloud. In 5th IEEE Conference on Cloud Computing Technology and Science (CloudCom), volume 2, pages 19–26.
- Belhajjame, K., Roos, M., Garcia-Cuesta, E., Klyne, G., Zhao, J., De Roure, D., Goble, C., Gomez-Perez, J. M., Hettne, K., and Garrido, A. (2012). Why workflows break - understanding and combating decay in taverna workflows. In Proceedings of the 2012 IEEE 8th International Conference on E-Science (e-

- Science'12), pages 1–9, USA. IEEE Computer Society.
- Stodden, V. C. (2010). Reproducible research: Addressing the need for data and code sharing in computational science. *Computing in Science & Engineering*, 12.
- Chirigati, F., Shasha, D., and Freire, J. (2013). Reprozip: Using provenance to support computational reproducibility. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, TaPP '13*, pages 1–4, Berkeley, USA. USENIX Association.
- Oliveira, D., Ogasawara, E., Baíao, F., and Mattoso, M. (2010). Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 378–385.
- Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.-H., Vahi, K., and Livny, M. (2004). Pegasus: Mapping scientific workflows onto the grid. In Dikaiakos, M., editor, *Grid Computing*, volume 3165 of *Lecture Notes in Computer Science*, pages 11–20. Springer Berlin Heidelberg.
- Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2008). Workflows and e-science: An overview of workflow system features and capabilities.
- Foster, I. and Kesselman, C., editors (1999). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., USA.
- Foster, I., Vöckler, J., Wilde, M., and Zhao, Y. (2002). Chimera: a virtual data system for representing, querying, and automating data derivation. In *Scientific and Statistical Database Management, Proceedings. 14th International Conference on*, pages 37–46.
- Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10.
- Groth, P., Deelman, E., Juve, G., Mehta, G., and Berriman, B. (2009). Pipeline-centric provenance model. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, WORKS '09*, pages 4:1–4:8, USA. ACM.
- Howe, B. (2012). Virtual appliances, cloud computing, and reproducible research. *Computing in Science Engineering*, 14(4):36–41.
- Janin, Y., Vincent, C., and Duraffort, R. (2014). Care, the comprehensive archiver for reproducible execution. In *Proceedings of the 1st ACM SIG-PLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering, TRUST '14*, pages 1:1–1:7, USA. ACM.
- Juve, G. and Deelman, E. (2010). Scientific workflows and clouds. *Crossroads*, 16(3):14–18.
- Kim, J., Deelman, E., Gil, Y., Mehta, G., and Ratnakar, V. (2008). Provenance trails in the wings-pegasus system. *Concurr. Comput. : Pract. Exper.*, 20(5):587–597.
- Ko, R., Lee, B., and Pearson, S. (2011). Towards achieving accountability, auditability and trust in cloud computing. In *Advances in Computing and Communications*, volume 193 of *Communications in Computer and Information Science*, pages 432–444. Springer Berlin Heidelberg.
- Lifschitz, S., Gomes, L., and Rehen, S. K. (2011). Dealing with reusability and reproducibility for scientific workflows. In *Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on*, pages 625–632. IEEE, 38, 69.
- Macko, P., Chiarini, M., and Seltzer, M. (2011). Collecting provenance via the xen hypervisor. 3rd USENIX Workshop on the Theory and Practice of Provenance (TAPP).
- Mehmood, Y., Habib, I., Bloodsworth, P., Anjum, A., Lansdale, T., and McClatchey, R. (2009). A middleware agnostic infrastructure for neuro- imaging analysis. In *Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE International Symposium on*, pages 1–4.
- Mei, L., Chan, W. K., and Tse, T. H. (2008). A tale of clouds: Paradigm comparisons and some thoughts on research issues. In *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, APSCC '08*, pages 464–469, USA. IEEE Computer Society.
- Mell, P. M. and Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States.
- Missier, P., Woodman, S., Hiden, H., and Watson, P. (2013). Provenance and data differencing for workflow reproducibility analysis. *Concurrency and Computation: Practice and Experience*.
- Munir, K., Kiani, S. L., Hasham, K., McClatchey, R., Branson, A., and Sham-dasani, J. (2013). An integrated e-science analysis base for computation neuroscience experiments and analysis. *Procedia - Social and Behavioral Sciences*, 73(0):85 – 92. *Proceedings of the 2nd International Conference on Integrated Information (IC-ININFO 2012)*, Budapest, Hungary, August 30 – September 3, 2012.
- Munir, K., Liaquat Kiani, S., Hasham, K., McClatchey, R., Branson, A., and Shamdasani, J. (2014). Provision of an integrated data analysis platform for computational neuroscience experiments. *Journal of Systems and Information Technology*, 16(3):150–169.
- Ramakrishnan, L. and Plale, B. (2010). A multi-dimensional classification model for scientific workflow characteristics. In *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science, Wands '10*, pages 4:1–4:12, USA. ACM.
- Roure, D. D., Manuel, J., Hettne, K., Belhajjame, K., Palma, R., Klyne, G., Missier, P., Ruiz, J. E., and Goble, C. (2011). Towards the preservation of scientific workflows. In *Procs. of the 8th International Conference on Preservation of Digital Objects (iPRES 2011)*. ACM.
- Sandve, G. K., Nekrutenko, A., Taylor, J., and Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLoS Comput Biol*, 9(10):e1003285.
- Santana-Perez, I., Ferreira da Silva, R., Rynge, M., Deelman, E., Pérez-Hernández, M., and Corcho, O.

- (2014a). A semantic-based approach to attain reproducibility of computational environments in scientific workflows: A case study. In *Parallel Processing Workshops*, volume 8805 of *Lecture Notes in Computer Science*, pages 452–463. Springer International Publishing.
- Santana-Perez, I., Ferreira da Silva, R., Rynge, M., Deelman, E., Perez-Hernandez, M. S., and Corcho, O. (2014b). Leveraging semantics to improve reproducibility in scientific workflows. In *The reproducibility at XSEDE workshop*.
- Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J., and Silva, C. (2008). Tackling the provenance challenge one layer at a time. *Concurr. Comput. : Pract. Exper.*, 20(5):473–483.
- Shamdasani, J., Branson, A., and McClatchey, R. (2012). Towards semantic provenance in cristal. In *Third International Workshop on the role of Semantic Web in Provenance Management (SWPM 2012)*.
- Simmhan, Y. L., Plale, B., and Gannon, D. (2005). A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36.
- SMS, C., CE, P., D, O., MLM, C., and M., M. (2011). Capturing distributed provenance metadata from cloud-based scientific workflows. *Information and Data Management*, 2:43–50.
- Stallings, W. (2010). *Cryptography and Network Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition.
- Stevens, R. D., Robinson, A. J., and Goble, C. A. (2003). myGrid: personalised bioinformatics on the information grid. *Bioinformatics*, 19:i302–i304.
- Tan, Y. S., Ko, R. K., Jagadpramana, P., Suen, C. H., Kirchberg, M., Lim, T. H., Lee, B. S., Singla, A., Mermoud, K., Keller, D., and Duc, H. (2012). Tracking of data leaving the cloud. 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 0:137–144.
- Tannenbaum, T., Wright, D., Miller, K., and Livny, M. (2002). Beowulf cluster computing with linux. chapter Condor: A Distributed Job Scheduler, pages 307–350. MIT Press, Cambridge, MA, USA.
- Vouk, M. (2008). Cloud computing #x2014; issues, research and implementations. In *Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on*, pages 31–40.
- Zhang, O. Q., Kirchberg, M., Ko, R. K., and Lee, B. S. (2011). How to track your data: The case for cloud computing provenance. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 446–453. IEEE.
- Zhao, X., Zhang, Y., Wu, Y., Chen, K., Jiang, J., and Li, K. (2014). Liquid: A scalable deduplication file system for virtual machine images. *Parallel and Distributed Systems, IEEE Transactions on*, 25(5):1257–1266.
- Zhao, Y., Fei, X., Raicu, I., and Lu, S. (2011). Opportunities and challenges in running scientific workflows on the cloud. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, pages 455–462.

Addressing Issues of Cloud Resilience, Security and Performance through Simple Detection of Co-locating Sibling Virtual Machine Instances

John O’Loughlin and Lee Gillam

*Department of Computing, University of Surrey, Guildford, GU2 7XH, Surrey, U.K.
{john.oloughlin, l.gillam}@surrey.ac.uk*

Keywords: Virtualisation, Xen, Cloud Computing, Co-location, Security, Performance.

Abstract: Most current Infrastructure Clouds are built on shared tenancy architectures, with resources shared amongst large numbers of customers. However, multi tenancy can lead to performance issues (so-called “noisy neighbours”) and also brings potential for serious security breaches such as hypervisor breakouts. Consequently, there has been a focus in the literature on identifying co-locating instances that are being affected by noisy neighbours or suggesting that such instances are vulnerable to attack. However, there is limited evidence of any such attacks in the wild. More beneficially, knowing that there is co-location amongst your own Virtual Machine instances (siblings) can help to avoid being your own worst enemy: avoiding your instances acting as your own noisy neighbours, building resilience through ensuring host-based redundancy, and/or reducing exposure to a single compromised host. In this paper, we propose and demonstrate a test to detect co-locating sibling instances on Xen-based Clouds, as could help address such needs, and evaluate its efficacy on Amazon’s EC2.

1 INTRODUCTION

Infrastructure Clouds offer compute resources for rent on-demand, typically on a per hour basis (Armbrust et al, 2009). One of the most popular offerings is the virtual server, which is the mainstay of providers of Infrastructure Clouds such as Amazon, Google and Microsoft. Infrastructure Clouds use virtualisation technologies such as Xen and KVM to offer physical servers as (often, multiple) virtual servers. Customers can rapidly acquire virtual servers, use them for as long as required, then release them back to the provider when no longer needed, with the equivalent resource then available for use by other customers.

At any given time, a physical server in a Public Cloud could be running virtual servers, also referred to as instances, for a number of different users (customers). From the user’s perspective, shared tenancy raises various concerns, of which security and performance are key. For security, one particular concern is hypervisor breakouts, where hypervisor security can be compromised and a resulting privilege escalation can be used to obtain data from other customers’ instances. For performance, one such concern is noisy neighbours, where

performance degradation occurs in one instance due to the (legitimate and not necessarily malicious) resource consuming actions of another.

In such cases, the concern tends to focus on the security or performance impact from *other* users. Consequently, research has tended to be focused on identifying vulnerable instances, or hiding from potential attackers. However, identifying co-locating instances may be of even more use for the majority of users with respect to their *own* instances. We will refer to instances started by the same user, irrespective of when, as *sibling instances* in the remainder of this paper.

Sibling instances that are co-located on the same host may be undesirable for the following reasons:

1. They may degrade the performance of each other when running compute bound workloads.
2. They are all vulnerable to failure, or degradation, of the underlying host.
3. They are all vulnerable to other noisy neighbours.
4. There is a greater exposure to a security compromise on a single host.

Determining co-location is difficult, and to date, no simple methods have been proposed that would reliably allow for such detection. This paper aims to

address this problem by exploring one kind of trace left by virtualization on Xen based Clouds - domain ids (*domids*). The rest of this paper is structured as follows: In section 2 we review relevant related work to offer background to the problem; in section 3, we discuss the Xen hypervisor and the generation of domain ids, and in section 4 we discuss the results of *domids* collected from a small sample (100) of virtual servers in the Amazon Cloud, and use this as a basis for tests for co-location in section 5. In section 6 we use *domids* collected from further samples to demonstrate likely recycling of resources. Finally, in section 7 we present our conclusions, and future directions of this work.

2 RELATED WORK

The ability for one instance to degrade the performance of other co-located instances is well known, and is referred to as noisy neighbours. Intel identifies the primary cause of the problem as the sharing of resources, such as the L2 cache, which cannot be partitioned (Intel, 2014); that is, there is no mechanism to limit how much of the resource an instance may consume. Consequently, it is possible for instances to use such resources disproportionately, to the detriment others.

The standard metric for compute performance is execution time. Identifying if a running task is likely to suffer from poor performance i.e. need increased execution time, is difficult. On their production clusters, Google detects likely poor performance by repeatedly measuring a task's cycles per instruction (CPI), i.e. the number of cycles required to execute an instruction, and comparing with the known CPI distribution (Zhang et al, 2013). If more outliers (defined as more than 2 standard deviations from the mean) are detected than expected, then performance of the task is likely to be poor. The protagonist, i.e. the noisy neighbour, is identified by correlating other instances' CPU usages with the increase in CPI outliers for the victim.

On a Public Cloud, information about when an instance is scheduled for CPU time by the hypervisor is only available to the provider, and is not subsequently made available to customers. As such, it is not possible to precisely state when an instance is running or not. A coarser approach would be to attempt to correlate instance performance using compute benchmarks. Such an approach would likely require a minimum number of co-located instances on a given host in order to be successful, and so this already requires co-location to be knowable, and there is the potential to miss a small degree of co-location per host.

The problem of extracting information between co-locating virtual machines has been investigated by a number of authors. In (Zhang et al, 2012) the sharing of an L2 cache between VMs was shown to be vulnerability when it was demonstrated that one VM may extract cryptographic keys from another VM on the same host. Such an attack is known as an access driven side channel attack. Particularly noteworthy, is the fact that the attack was demonstrated on an SMP system. In this case the challenge of core migrations i.e. the scheduling of VMs onto different cores during its lifetime, as would be encountered in a Cloud environment, needs to be overcome. However, the demonstration was on a standalone Xen system rather than on a Public Cloud.

The vulnerability of a shared cache relies, in part, on exploiting hypervisor scheduling. Methods to increase the difficulty of successfully using such attacks are under development (Lui, Ren and Bai, 2014), and indeed, are already being integrated into Xen. Whilst such work mitigates fine grained attacks, denial of service attacks, which seek to obtain a large share of the L2 cache, are considered viable.

This has led to work on targeted attacks in the Cloud, whereby an attacker seeks to co-locate with a specific target. This requires methods for determining co-location with the target before the attack can be launched. In (Ristenpart, Tromer, Shacham and Savage, 2010) a number of network based probes have been proposed, for example ping trip time and common IP address of *dom0*. In order to test the veracity of these methods they also use access timings of shared drives. No details are provided of the type of drive being used (local or network) or how the disk is being shared.

However, as the authors state, a provider can easily obfuscate network based probes and this already appears to be the case. From our experiments we can confirm this. Whilst access times to shared drives may potentially be used for detecting co-locating siblings, there are a number of issues not discussed that need require further investigation. Perhaps most importantly, is the widely reported variation in disk read/write timings on EC2 (Armbrust et al, 2009), which clearly needs to be accounted for in any test that proposes to use them.

In (Bates et al, 2013) watermarking of network flows is proposed and demonstrated on a variety of stand-alone virtual systems. However, as the authors state, there are a number of defences against watermarking in place in Public Clouds, and in particular on EC2, which prevented them from successfully using the tests.

In (Zhang, Juels, Oprea and Reiter, 2011) a cache avoidance strategy is used so that instances

can co-ordinate their use (or avoidance) of the L2 cache and measure resulting cache use. This, then, is a basis for detecting co-locating siblings. The method is applicable to Xen-based Clouds but requires modification of how the guest OS kernel manages memory, and has a performance overhead when cache use is turned off. Such an approach is technically challenging, as it involves kernel changes, and this is likely beyond the capabilities of most Cloud users.

In summary, neither simple network probes nor network flows watermarking co-location tests work on EC2 due to measures in place, whilst cache avoidance technically challenging. There is a need for simple methods then.

3 THE XEN HYPERVISOR

The Xen system (Xen, no date) is a widely deployed hypervisor in Infrastructure Cloud systems, and is in use at Amazon, Rackspace, IBM and GoGrid, amongst others. The Xen system consists of the Xen *hypervisor* together with a *privileged* VM called domain 0 or *dom0*. Xen is a bare-metal hypervisor, started by the BIOS, which in turn starts *dom0*. The *dom0* is a *privileged* VM and can directly access hardware such as network cards and local disk storage. *Dom0* provides a management interface for the Xen system, from which system administrators can launch and manage the life cycle of VMs. These VMs are unprivileged domains and are referred to as *domUs*.

The Xen hypervisor is responsible for scheduling VM CPU time, managing memory, and handling interrupts. On an x86 CPU, *dom0* privilege escalation is provided by running *dom0* in ring 1, whilst the Xen hypervisor runs in ring 0 (and the unprivileged VMs, *domUs*, run in ring 3). *DomUs* gain access to hardware devices such as disks and network cards via calls to *dom0*.

Each domain is given two identifiers, a *domid* and a UUID. The UUID is a unique identifier amongst a deployment of multiple Xen systems; that is, it uniquely identifies a domain amongst the set of all domains across the Xen systems. For example on EC2, the UUID assigned to a new instance will (in theory) be unique to that instance, at least within the Region it was launched in.

In addition, a newly launched domain is assigned a domain identifier, referred to as the *domid*. This uniquely identifies domains on the physical server only. On EC2, instances on the same physical server will have different *domids*. However, these may well clash with *domids* for instances on other hosts. The *domid* is a 16 bit integer and allocation is

monotonically increasing - Xen assigns the next available *domid*. This means that instances that are started one after the other will obtain consecutive *domids*. On EC2, therefore, we would expect co-locating instances, started at the same time, to have consecutive *domids* - or, with other requests also being satisfied, being quite close to each other.

Xen *domids* have a rather interesting property, and one which will be crucial to us later: an instance can increase its own *domid* simply by rebooting. An instance's new *domid* will be its old *domid* plus the number of instances that have started on the same host since it was last rebooted, plus the number of reboots that have occurred. *Domids* do not, however, seem to survive an underlying host reboot, and in this case the next available *domid* is reset to 1.

A user does not have administrative access to Xen on EC2 (or indeed any Public Cloud). However, we can determine an instance's *domid* via Xenstore. Xenstore (Xenstore, 2014) is a data area exported from *dom0* to *domUs*, the interface of which is a pseudo file system which can be mounted on */proc/xen* within a guest. This is analogous to the */proc* and */sys* pseudo file systems in Linux which provide an interface for user space processes to the Linux kernel. Under a standard Xen system, a domain can extract information such as the *domids* of all the running domains and the CPU weightings assigned to them. As one would expect, on EC2 the data exported to the instances via Xenstore is restricted, and does not allow a domain to obtain any information other than about itself. However, it is particularly useful, for our purposes that a domain can obtain its own *domid*.

In the next section we present the results of *domids* collected on EC2 via Xenstore from some 120 instances.

4 COLLECTING DOMIDS

We can initially collect *domids* from instances launched on EC2, and examine the extent to which these hint at co-location. Using an Ubuntu precise 12.04 AMI, we can readily launch 20 *m1.small* instances as a single request in the Region US-East-1, in AZ *us-east-1b*. Each instance gets *xenstore-utils* installed, and has the exported Xen store file system mounted on */proc/xen*. In this setup, it is then possible to obtain an instance's *domid*, *uuid* and *cpuid*.

In Table 1, below, we list 20 *domids* obtained from just such a setup (on 07/10/2014), which are readily organised into three sequences of consecutive *domids*. For all instances, the CPU model was an E5-2651.

Table 1: Consecutive Domids.

Seq	Domids
1	563, 564, 565, 566, 567 and 568
2	723, 724, 725, 726, 727, 728 and 729
3	752, 753, 754, 755, 756, 757 and 758

The simplest explanation for these consecutive *domids* is that the 20 instances are allocated to just three hosts. It may also be possible that these sequences are obtained simply by chance across a large number of hosts that are churning VMs at similar rates, and we discuss this possibility in section 5.

The AZ *us-east-1b* appears homogeneous (just one CPU model) for the account we were using. To simplify concerns further, we instead examine domids in *us-east-1a* as this provides heterogeneous hosts. This helps to improve clarity over co-location since instances with consecutive domids on *different CPU models* are clearly not co-located, and so here the consecutive *domids* are more likely to indicate co-locating instances – unless, of course, *cpuid* and *domid* values are spoofed.

We ran 5 requests, with 20 instances per request, on Amazon’s spot market for *us-east-1b*. Of the 100 instances started, 3 were reclaimed and so we have results for just 97 instances. As before, we determine the *domid*, *uuid* and *cpuid*. After this information was obtained, the instances were released. Each request was made at a different time over a 2 day period, from 07/10/2014 to 08/10/2014. In Table 2, below, we list only the sequences with consecutive domids found in each request, together with the instance CPU models – one of E5645, E5507 or E5-2650.

Table 2: Domids from Multiple Time-Separated Requests.

Request, Date & Time	Consecutive Domids and CPU Model
1 07/10/2014 17:05	242,243,244 – E5645 469,470 – E5645 1499,1500 – E5645 1671, 1672 – E5645 + E5-2650 2627, 2628, 2629 – E5-2650
2 07/10/2014 17:58	None
3 07/10/2014 21:57	250, 251, 252, 253, 254, 255, 256 – E5645 732, 733 – E5507 1501, 1502 – E5645 2630, 2631, 2632 – E5-2650
4 08/10/2014 10:25	263, 264, 265, 266 – E5645 501,502 – E5645 1505, 1506 – E5645 2637, 2638, 2639, 2640 – E5-2650
5 08/10/2014 21:50	None

3 out of 5 of the requests evidence consecutive *domids* with E5645 CPUs, and all three contain at least 2 such sequences. The most common pattern is of two consecutive domids, and the longest sequence is 7. We note consecutive domids in request 1 of 1671 and 1672, with different CPU models – E5645 and E5-2650 respectively – which clearly cannot be co-located (unless, again, the *cpuid* is spoofed). In request 1, it would appear that 10 of 20 instances are not host separated, in request 3 this is 14, and in request 4 it is 12.

5 CO-LOCATION TEST

Based on the discussion and results in section 4 we can state that for any pair of instances the following initial conditions must be satisfied if the instances are more likely to be co-located:

1. Same CPU model
2. Values of *domids* are sufficiently close to each other

For the second condition, we do not require that the *domids* be consecutive but should be sufficiently close to each other. In order to understand why consider the following: two sibling instances are scheduled onto the same host, but in between them being launched an existing instance is rebooted. In this case then, they will not have consecutive domids but the *domids* will differ by (at least) 2. We discuss how close is ‘sufficiently close’ later in this section.

Whilst the two conditions listed above are necessary for co-location, they are not sufficient. It is entirely possible that the instances have been allocated to hosts whose next available domids were within the *domid* distance simply by chance. Indeed, this becomes more likely if the hardware platform and configuration is identical, and if the churn rate of VMs is the same. In fact, we have already seen an example in batch 1 of instances with *domids* of 1671 and 1672 that had different CPU models.

For the second condition, closeness of the domids depends in large part on how many instances a host has been configured to support. If a host supports *k* instances, then any instances started within a short period of time on the host would likely have their domids within *k* of each other. We cannot state this for certain, since it’s possible that within that period (1) a number of instances on the host were rebooted (2) a number of instances were terminated, and a number more were started.

We also cannot state the value of *k* for a host with certainty, since it depends on its CPU model, the CPU configuration, how many sockets the host has, and the degree of over commitment. As an example, we have previously shown (blind ref, no

date) that m1.small instances on EC2 may be backed by 6 different CPU models, including the AMD 2218 and the Intel Xeon E5-2651. The former is a dual core CPU, so a host with dual socket can have at most 4 cores. The latter, however, has 10 cores per socket and dual socket would have 20 cores. Further, if hyper threading is enabled (as is common practice on EC2), the core count rises to 40. Finally, the configured ratio of vCPUs to physical cores determines k . As EC2 does not advertise socket count, and only specifies vCPU to cores for some instance type, as a rule of thumb we will take ‘close’ to be 2 times the core count of a CPU, and times again by 2 if the CPU supports hyper threading.

In table 3 below we list the 6 models we have identified to date as backing m1.small instances together with a domids closeness range based on the above reasoning:

Table 3: CPU model and Domid Range.

CPU Model	Domid Range (m1.small only)
AMD 2218	4
Intel Xeon E5430	8
Intel Xeon E5507	8
Intel Xeon E5645	24
Intel Xeon E5-2650	32
Intel Xeon E5-2651	40

We are naturally led to the question of the likelihood that non-co-locating instances have *domids* near to each other. This question is similar to the well known ‘birthday’ and ‘almost birthday’ problems. The birthday problem can be stated like this: How many people do we need in a room in order for there to be a 0.5 chance that at least 2 people will share the same birthday? In this case the answer is 23. As we are interested in near *domids* our problem is more akin the ‘almost birthday problem’: In a room of 23 people how likely is it to have at least one pair of consecutive birthdays? An analytic solution to this is presented in (Dasgupta, 2004), with the answer 0.89.

Monte Carlo methods can be used to tackle the birthday problems stated above. We can assume that a birthday is equally likely to fall on any day in the year. We then generate random samples, of size 23, drawn from the uniform distribution. For each sample we record a success if there is the matching (or consecutive, depending upon the problem of interest) birthday. The number of successes divided by the number of trials is then the estimate of the probability.

We note that the assumption that birthdays are uniformly distributed is not entirely accurate and that seasonal variations do exist. However, the

uniform distribution does provide a good approximation.

Can we apply such methods to estimate the probabilities of instances having consecutive, or near, *domids* by chance – and not because they are necessarily co-locating? An immediate requirement is a reasonable approximation for the distribution of *domids* across hosts. In theory, a *domid* is in the range $[1, 65536]$, however we have so far only observed *domids* within a restricted range. Further, the *domid* distribution is likely CPU dependent to some degree. CPUs with more cores, such as the E5-2651, will likely increment *domids* at a different rate to the E5645, as they can run more instances.

We could assume that the range of *domids* for hosts with the same CPU model is equally likely to be between the observed minimum and maximum. Applying this to the E5645, that would be between 252 and 20708. Using a Monte Carlo simulation, we find that 20 non co-located instances, placed on randomly selected hosts with E5645 CPUs, will have at least one pair of consecutive *domids* with a probability of 0.009. That is, approximately 1 in 100 batches of 20 instances would have at least one pair with consecutive *domids*.

However, it is not obvious that we can model the problem in a manner similar to the birthday problems. Consider for example, a power failure in one portion of a data centre resulting in a large number of E5645 hosts being rebooted. In this case then, we initially have a large number of E5645 hosts with small *domids*. Instances allocated to these hosts would have a far greater chance of consecutive, or near, *domids* than our estimate would imply. Whereas birth dates do not tend to change in such a way.

Indeed, it is not clear that the *domid* range should be well approximated by any statistical distribution. Further, the VM allocation mechanisms in use, which are not advertised, may well produce *domid* ranges whereby near *domids* are more likely, and perhaps considerably so, than our assumptions would allow for. As such, developing a model to accurately represent *domid* distribution across hosts is beyond the scope of this paper, so we do not rely on purely statistical arguments and instead look for further evidence for co-location, which we describe now.

We have already seen that when an instance is rebooted it acquires a new *domid*. This will be the number of new instances started on the host plus the number of instance reboots. This observation allows us to add an additional condition:

Suppose, then, that we have two instances both on hosts with the same CPU model. If they have identical *domids* they are not on the same host. Suppose that the instances’ *domids* are different and

within a host's *domid* range (from Table 3). We denote the lower *domid* by m , and refer to the instance with this *domid* by A . We refer to the higher *domid* by n and the instance with this *domid* by B . Upon rebooting A , its new *domid* must, simply, be greater than the *domid* of B .

We now state this as a third necessary condition for co-location:

3. A and B are instances with *domids* (m, n) respectively, where $m < n$. If A and B are co-locating, then upon rebooting B , its new *domid*, p , must satisfy $p > n$.

Of course, we still do not have a sufficient condition – instances may satisfy the above by chance. However, a user is free to reboot their instances as often as the like. So we can strengthen the condition as follows:

- a. Reboot the instance A , which has *domid* p , k times. When rebooted, the instance with *domid* n will obtain a new *domid*, q , that must satisfy $q > p + k$.

Whilst again this may be satisfied by chance, further repetition should lead to greater confidence as co-locating instances will satisfy these conditions.

To test this, we used 2 pairs of instances, the first pair with *domids* (7635, 7638) respectively, and the second pair (9536, 9538). As the first pair of instances were on E5-2650 hosts (condition 1), and have close *domids* (condition 2) they are good candidates for co-location. However, upon rebooting instance with *domid* 7635, its new *domid* was 7636, and so cannot be co-locating with the instance with *domid* 7638 (due to condition 3). For the second pair, again both with CPU model of E5-2650 (condition 1) when rebooting the instance with *domid* 9536, its new *domid* was 9539, and so greater than 9538 (condition 3). We rebooted this instance a further 5 times and after the last reboot its *domid* was 9544. We then rebooted the instance with *domid* 9538, after which its *domid* was 9545 (condition 3a). This more strongly suggests co-location, and we note again that a user is of course free to set the *domid* distance to any value they like by rebooting (we set to 6), and to repeat as many times as they wish.

To now, we only considered instances started within a short space of time of each other. A user may have long running instances, and want to know if newly started instances are co-located with any long running instance. In this case, a long running instance's *domid* is likely not representative of the current *domids* available from the host due to requests and reboots in the intervening period. In this case, rebooting the long running instance will update its *domid*, and bring the *domid* into range of

new instances, allowing for further confirmatory tests to be run.

We now state our test for co-location as follows: Two instances, A and B , chosen because they have *domids*, m and n , such that $m < n$ are likely co-locating if they satisfy the following:

1. Same CPU model
2. Values of *domids* are in range (by Table 3). That is, $n - m \leq k$ where k is the CPU *domid* range in Table 3.
3. Upon rebooting instance A , its new *domid* satisfies $p > n$.

If 3 is satisfied, then we strengthen the condition as follows:

- 3a. Upon rebooting instance A a further k times, a reboot of B results in a new *domid*, q satisfying $q > p + k$.

We reiterate that (3a) can be carried out as many times as the user wishes, for any value of k .

6 RECYCLED RESOURCES

In addition to some degree of co-location, we also observe that instances started from later requests appear to be scheduled onto the same hosts as earlier ones. This observation is also based on *domids*, as we explain now.

In request 1 we obtain instances with *domids* 1499 and 1500, and both have E5645 CPUs. In request 3 we obtain instances with *domids* of 1501 and 1502, and in request 4 we have 1505 and 1506 – again all E5645. One explanation is that these instances were scheduled onto just one host. As another example, we have the *domids* 2627, 2628 and 2629 in request 1, followed by 2630, 2631 and 2632 in request 3 and then followed by 2637, 2638, 2639 and 2640 in request 4. All of the instances were running on a host with a E5-2650 CPU, so could again have been scheduled onto just one same host.

In a follow up experiment, we launched 100 instances and found 4 consecutive *domids*. We terminated these instances, and 5 minutes later started another 100 instances (5 of which were reclaimed). The *domids* in the two sets ranged between 759 and 7292. Comparing *domids* in the first set to the second, we found a remarkable 51 *domids* in the first set with consecutive *domids* in the second set, 27 *domids* in the first set with a 'plus 2' in the second, 7 at 'plus three' and 1 at 'plus 4'. The likelihood of our second set of instances being on a completely different set of hosts to the first, but having *domids* so close to the first set would appear to be small.

Running 3 further requests, again of size 100, we find the same behavior of later instances appearing to be scheduled on to previously used hosts. This is also not just a feature of either on-demand or spot instances, as we observe this for both. Indeed, when running a batch of spot instances after a batch of on-demand, we again observe such behavior, suggesting that requests are being satisfied from the same resource pool.

It is unclear whether this might be a temporal or spatial issue. In the former, it may simply be the case that whilst there is a large amount of available resource, instances started shortly after earlier ones are scheduled back on to previously obtained hosts. In the latter, it may be that a user is restricted to a subset of the available resources. We know that EC2 is vast in scale, with 28 AZs, most of which comprise at least 2 data centres - with the largest AZ having 6 - and each data centre houses between 50,000 to 80,000 physical servers (Vanian, 2014). For each user, an AZ identifier, such as us-east-1a, relates to some pool of resources out of which requests are served. It is possible that AZ identifiers may map to a data centre in an AZ, or indeed to some rather smaller subset thereof.

Recycling of resources has the clear potential to impact on a user's ability to separate co-locating instances. In this case, a user may be interested in the number of attempts needed, and so the cost, to ensure separation. Perhaps more intriguingly, if a user is restricted to a subset of resources then launching a targeted attack against them on EC2 would be much harder - you would only be able to target users that you share the same resource partition with. With sufficient data, it may be possible to answer these questions, and also estimate the size of resource pool available for use. From this, one might also estimate a likely number of people with whom the resource pool is shared, and could use this number to suggest the risk of security and performance issues arising.

Finally, given the well established problem of performance variation due to the heterogeneous (Osterman et al, 2010, Iosup, Nezih and Dick, 2011) nature of Public Clouds, there has been interest in so-called 'instance seeking' or 'deploy and ditch' strategies (Farley et al, 2012, Zhuang, Liu, Ou and Arberer, 2013). The assumption behind these strategies is that a poorly performing instance can be released and a new, better performing one, found. However, as the performance of an instance is determined by the hosts it is running on, such strategies are rather less likely to produce performance gains in the face of resource recycling.

7 CONCLUSIONS

Identifying when sibling instances are co-locating is beneficial to users in a number of situations:

1. Co-located instances may degrade the performance of each other when running compute bound workloads.
2. Co-located instances are all vulnerable to failure, or degradation, of the underlying host.
3. Co-located instances are all vulnerable to other noisy neighbours.
4. Co-located instances imply is a greater exposure to a security compromise on a single host.

Determining co-location is challenging, particularly so on Public Clouds. The simple approach we have presented in this paper is based on information provided from Xen, which is currently the dominant hypervisor technology used in Public Infrastructure Clouds. Xenstore provides an interface for domains to obtain information such as *domids* and *uuids*. However, as would be expected, on EC2 the interface is restricted so a domain can only obtain information about itself. But the *domid* is still very useful for our purposes. On a standard Xen system, *domids* are assigned consecutively when starting domains and are not recycled - except when the range itself cycles. Instances are assigned the next available (new) *domid* when rebooted. *Domids* also do not survive host reboots, which resets the next available *domid* to 1.

These characteristics of *domids* allow for the formulation of the simple test for co-locating sibling instances as described, based on the same CPU model and close *domids* (per Table 3 for the various CPU models we have observed backing m1.small instance types). It is still, as we have elaborated, possible that such instances have close *domids* simply by chance, and indeed we have seen such examples. Simulation methods could be employed to determine the likelihood of this, but assumptions regarding the distribution of *domids* are required, the validity of which is difficult to establish. Whilst nearness hints at co-location, further evidence is required.

Further evidence is provided by the observation that one instance can restrict the possible range of values for another instance's *domid* - simply via rebooting itself and so increasing the next available *domid* value. The second instance, upon a reboot, can then in turn restrict possible *domid* values for the first instances. This process can be repeated as often as a user chooses, and at the *domid* distance the user chooses (the reboot value), and therefore each time this is done the probability that this happens by

chance decreases. Further, this is not limited to instances started close to each other in time, but can be used when any pair of instances is suspected of co-locating.

We should be clear that whilst passing the tests described in section 5 decreases the likelihood that the instances are not co-locating, increasingly so when repeated, we cannot say for certain that the instances are co-locating. From a pragmatic point of view, a user must balance the risk of having co-located instances with the cost of (determining and then ensuring) separation.

Determining such costs may be difficult as there appears to be a degree of recycling of resources, as described in section 6. This also has an immediate and significant consequence for the probability of success in carrying out a targeted side channel attack on a Public Cloud. Indeed, from our work here, we find the chance of intentionally co-locating with sibling instances to be fairly small. Co-locating with any intended target would therefore be more unlikely still, if it is indeed possible at all. We also note the impacts for so-called ‘performance seekers’, whereby a user releases back underperforming instances in the hope of acquiring better performing new instances. A user may simply be paying to obtain resources they have already had.

In summary, our test is simple to implement and works on Linux, Windows and FreeBSD Operating Systems, with the appropriate Xenstore client tool. Future work is largely aimed at further exploration and confirmation of the ideas discussed in this paper. In particular, we would like to be able to identify behaviours of instances that can be detected by others as would confirm co-location, without incurring the effort involved with rewriting (for Linux) kernel memory management features to spot avoidance of shared cache use, and further ensuring that any such observation are not due to chance.

REFERENCES

- Armbrust, M. et al. (2009) “Above the clouds: a Berkeley view of cloud computing”. Technical Report EECS-2008-28, EECS Department, University of California, Berkeley.
- Intel, (2014) [Online]. Available at: www.intel.com/content/dam/www/public/use/en/documents/white-papers/intel-saa-performance-white-paper.pdf. [Accessed on 02/01/2015]
- Zhang, X. et al. (2013) CPI²: CPU performance isolation for shared compute clusters, Proc of EuroSys 2013, pp 379-391.
- Zhang, Y. et al. (2012) Cross-VM Side Channels and their use to Extract Private Keys, Proc of the 2012 ACM Conference on Computer and communications Security, pp305-316.
- Ristenpart, T. Tromer, E. Shacham, H. Savage, S. (2010) Hey you get off my Cloud, Proc of the 16th ACM Conference on Computer and communications Security, pp199-212.
- Bates, A. et al (2013) On Detecting Co-resident Cloud Instances using Network Flow Watermarking Techniques, International Journal of Information Security, Vol 13, Issue 2, pp 171-189.
- Lui, F. Ren, L. Bai, H. (2014) Mitigating Cross-VM Side Channel Attacks on Multiple Tenants Cloud Platform, Journal of Computers, Vol 9, No 4, pp1005-1013.
- Zhang, Y. Juels, A. Oprea, A. Reiter, M.K. (2011) Home Alone: Co residency detection in the cloud via side channel analysis, Proc 2011 IEEE Symposium on Security and Privacy, pp313-328.
- Xen, (no date) [Online]. Available at: www.xenproject.org [Accessed: 08/02/2015].
- Xenstore, (2014) [Online]. Available at: <http://wiki.xen.org/wiki/XenStoreReference> [Accessed: 08/02/2015].
- Blind Ref, no date:
- Dasgupta, A. (2004) The Matching, Birthday and Strong Birthday Problem: A Contemporary Review, Journal of Statistical Planning and Inference 130, pp377-389, 2004.
- Vanian, J., 2014. [Online]. Available at: <https://gigaom.com/2014/11/12/amazon-details-how-it-does-networking-in-its-data-centers/> [Accessed: 08/02/2015].
- Osterman, S., et al. (2010) A performance analysis of EC2 cloud computing services for scientific computing, Cloud Computing, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, vol 34, pp115-131.
- Iosup, A. Nezih, Y. and Dick, E. (2011) On the performance variability of production cloud services. In Cluster, Cloud and Grid Computing (CCGrid), 2011.
- Farley, B. et al. (2012) “More for your money: exploiting performance heterogeneity in Public Clouds”, in Proc. of the Third ACM Symposium on Cloud Computing, article no. 20.
- Zhuang, H. Liu, X. Ou, Z. Arberer, A. (2013) “Impact of Instance Seeking Strategies on Resource Allocation in Cloud Data Centres”, in Proc. Of the IEEE Sixth International Conference on Cloud Computing, pp27-34.

SHORT PAPERS

P-TOSCA Portability of SOA Applications

Marjan Gusev, Magdalena Kostoska, Sasko Ristov and Aleksandar Donevski

University Ss. Cyril and Methodius, Faculty of Computer Sciences and Engineering, 1000 Skopje, Macedonia
{marjan.gusev, magdalena.kostoska, sashko.ristov}@finki.ukim.mk, aleksandar.donevski@outlook.com

Keywords: Application Portability, SOA, Cloud Computing.

Abstract: Even more frequently, the customers express their increasing need to change the cloud provider and/or the operating cloud environment in order to avoid vendor lock-in. We analyze portability as the transferability of an application from on-premise onto a cloud (migration) and among different clouds (porting). The contribution of this paper is twofold: 1) demonstration of the P-TOSCA model for automated migration and porting of SOA applications onto a cloud and/or switch between cloud providers, and 2) evaluation of a significant time reduction in migration and porting.

1 INTRODUCTION

The Service Oriented Architecture (SOA) is commonly used architecture in the past few years, especially for enterprise applications. It became popular due to the benefits and flexibilities of integrating loosely-coupled and reusable services (Erl, 2004)

Cloud offers a scalable and elastic environment for hosting SOA applications as computing utilities (Buyya et al., 2009). It saves not only companies' OPEX and CAPEX, but also management and administration costs (Rana, 2014). As more customers adopt and use cloud technologies, they encounter turbulence along the increasing number of SOA applications hosted on clouds. A variety of cloud service providers (CSPs) with different service level agreements (SLAs) are now available on the market.

Customers appreciate being able to switch among CSPs and environments, so as to choose the most suitable one to their needs. The ability of a software to run on different cloud platforms in general defines the cloud portability. It presents a hot topic research challenge, although CSPs do not think to offer mechanisms to enable migration of applications onto clouds or to enable a possibility to transfer them between clouds. Mostly, CSPs are stuck to the philosophy that the best solution on the market will become a standard. However, we analyze a possibility to define a specification and build an engine that will enable portability and migration. Our research also includes *migration* as a process of transferring an application onto a cloud. Actually, we aim to present a sophisticated procedure to realize the transfer process

to make the application run in the new cloud environment, which is realized manually by cloud experts.

Cloud application portability is a general ability to move applications between CSPs, no matter which cloud environment they are using as infrastructure. Cloud application portability is not considered as a data or service transfer between clouds, but as a transfer of a whole set of functionally organized services and data. Our goal is *automated cloud application portability* by defining an automated procedure that will realize migration or porting, without or with minimal user intervention. *Porting* is a complex process that usually needs a lot of support by cloud vendors, and especially by CSPs. Our goal is to use standard interfaces to cloud management, which will realize porting as a process. These cloud management features used by most of the existing operating systems (OS) and cloud environments integrate essential cloud management functions, such as invocation of virtual machines (VMs), instantiation, customization and management.

The approach for automated porting of a SOA application between different CSPs (or cloud environments) is based on the P-TOSCA model and practical implementation (Kostoska et al., 2014b). This model enables automated porting of Platform-as-a-Service (PaaS) hosted applications from one cloud environment to another as an extension of TOSCA 1.0 standard (OASIS, 2014).

Cloud portability requires the CSPs to enable cloud interoperability (Toosi et al., 2014). It means that a CSP must be able to replicate the application environment and enable application deployment.

2 RELATED WORK

As more CSPs and cloud environment vendors become available, users at some point would like to transfer their data and applications from one CSP to another, but there is no standard to do it seamlessly. A nice overview about cloud portability approaches and opportunities is given by Petcu and Vasilakos (2014). Gonidis et al. (2013) have classified three types of cloud portability solutions: 1) adoption of existing or emerging standards (like TOSCA, CDMI, OCCI, OCF), 2) usage of intermediary levels (like jClouds or mOSAIC) and 3) adoption of semantics and model-based solutions. Several approaches have been analyzed in (Ortiz Jr, 2011). IEEE P2301 is just one initiative to design a roadmap for application portability, management and interoperability interfaces, as well as for file formats and operating conventions.

Open Virtualization Format (OVF) (DMTF, 2010) establishes a transport mechanism for moving VMs from one hosted platform to another. It is an approach for definition of an open standard for packaging and distributing virtual appliances or more generally software to be run in multiple VMs. Their approach is based on using different hypervisors.

The OCCI, OVF and similar approaches work on IaaS level. When analyzed on PaaS level, the most promising approach is defined by TOSCA (OASIS, 2014) as a portable and manageable specification of services and applications deployed on any CSP. Our research with TOSCA specification was initially directed to test the feasibility of a TOSCA model deployment. Recently, analyzing this process we have identified critical points where original TOSCA specification requires further refinement and extended the specification with additional cloud-specific elements to enable automated porting (Kostoska et al., 2014b).

No commercial solution supports processing of TOSCA specification at this moment. (Binz et al., 2013) present the OpenTOSCA environment for imperative Cloud Service Archive (CSAR) processing. Unlike this proposal, P-TOSCA offers declarative processing and implementation for multiple cloud environments. Other initiatives include creation of visual environments for TOSCA specifications like Winery (Kopp et al., 2013) and Vino4TOSCA (Breitenbücher et al., 2012). Some approaches were concerned with creation of TOSCA specification for existing projects (Li et al., 2013; Kostoska et al., 2014a).

Petcu and Vasilakos (2014) also give an overview of tools and services that support a certain degree of portability, including Aoleus (cloud management software, written in Ruby), CompatibleOne (cloud broker, defining a language for management of cloud

services), CloudFoundry (works on top of VMware-based IaaS), ConPaaS (federation support), Docker (deployment engine), mOSAIC (API that allows deployment and configuration management), Nimbus (a virtual site layer for dynamic provisioning of distributed resources), etc.

Katsratos et al. (2014) present a proof of concept for the portability problem on OpenStack cloud. They use Opscode Chef as a configuration management tool that describes and manages system configuration using a Ruby based domain-specific language. It automates the cloud management tasks that are obtained by translating a TOSCA-based application specification into Chef environment. Similar approaches are used by the existing EU funded research projects, such as SeaClouds, Remics, Cloud4SOA, Optimis, Contrail, Artist, PaaSage, MODAClouds, or even RighScale or CloudFoundry. Instead of building a TOSCA engine, these approaches translate a TOSCA-based application specification into a specification that can use cloud management tools, such as CAMP, Brooklyn, Chef, Puppet etc. However, our approach uses a practical implementation of P-TOSCA engine and direct application porting between clouds.

The concept of SOA services is based on unified communication and collaboration to produce the desired result (OASIS, 2014). It offers many benefits due to scalability and adaptability. The main SOA characteristics are Discoverable and Dynamically Bound, Self-Contained and Modular, Interoperability, Loose Coupling, Location Transparency, Composability, and Self-Healing (Valipour et al., 2009).

SOA applications consist of independent service elements orchestrated to communicate and exchange information in order to achieve the desired functionality. The services can be composed as applications (assembly of services and components bound by application logic), service federations (collections of services bound in large service domain) and service orchestration (execution of one business process by multiple successful service invocation) (Valipour et al., 2009).

Building and deploying a distributed SOA depends upon successful orchestration to enable services to be orchestrated in unified and defined process, successful deployment to enable proper configuration of security, reliability, scalability and successful management (Papazoglou and Heuvel, 2007).

3 P-TOSCA CONCEPTS

A P-TOSCA specification of an application contains XML description of the application topology (types,

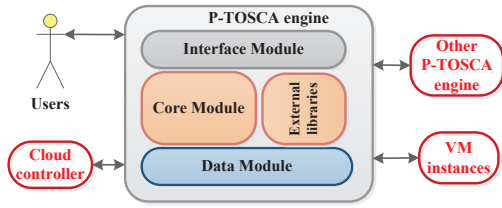


Figure 1: Context diagram of P-TOSCA engine.

templates and artifacts). This specification along with the required artifacts is packed in a CSAR. Artifacts represent files needed for application deployment or configuration like scripts, war files, zip files, libraries etc. The P-TOSCA approach is used to: *Migrate* an application onto a cloud; or *Port* an application from one CSP to another.

In our earlier work (Kostoska et al., 2014b), we have identified several TOSCA weaknesses and ambiguities and suggested extensions to enable a fully automated application life-cycle management. Most of these identified problems arise because the main goal of TOSCA is to be cloud, technology and hardware agnostic, while the real implementation needs proper definitions of several hardware based specific parameters, such as: a) Specifying the external namespace of `ServerProperties`; b) Specifying the initial number of instances child element to the `ServerProperties` element; c) Extending the `ServerProperties` element of `Node Template` with `ServerIPAddress` element; d) Specifying the external namespace of `ScriptArtifact Properties`; e) Extending the `Properties` element of `Node Template` with `ServerSecurityProperties` element; and f) Introducing XML defined plan.

All these extensions still keep the P-TOSCA cloud and technology agnostic. It just defines all those specific elements required for real implementation.

Details on the practical P-TOSCA engine implementation are also described in (Kostoska et al., 2014b). The software is hosted on a separate VM by the CSP and its architecture is presented in Fig. 1.

The interface module contains two parts, one to establish communication to the users and the other to collaborate with other P-TOSCA engine implementations. The core module is responsible for managing the CSAR archives. It executes the artifact plans and communicates with the cloud controller to manage, invoke and revoke various instances. All relevant data are stored and managed by the corresponding database module. Software is developed in Java programming language using Linux specifics.

The first use case, which presents the *cloud migration* (to migrate an application onto a cloud), is performed by a direct user interaction with a web application provided by the platform. The second use

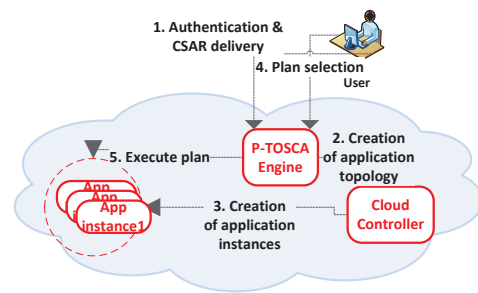


Figure 2: P-TOSCA based migration.

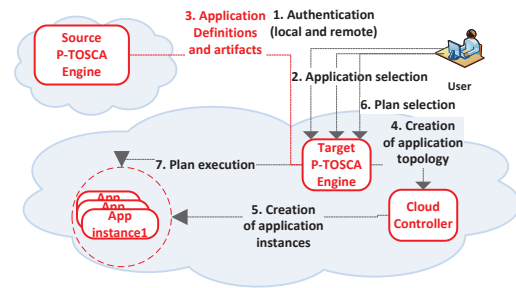


Figure 3: P-TOSCA porting.

case, which presents the *cloud application portability* (porting of an application from one CSP to another), is performed by a direct user interaction with a web application and communication between platforms using web services.

Both use cases may require only two clicks by the customer, one to select and upload the CSAR archive (or to select the appropriate application to be ported together with the P-TOSCA engine), and another for plan selection. We assume that the customer had already prepared the CSAR archive and the execution plan. As the plan selection may be a part of a script file, the complete transfer can be done automatically.

Fig. 2 presents a conceptual diagram of a sequence of activities to realize the cloud migration using the P-TOSCA platform. This use case includes the following actions: 1) A user authenticates at the P-TOSCA engine platform and uploads the CSAR archive; 2) The platform processes the archive and requires creation of instances according to the topology of the application; 3) The cloud controller creates the instances; 4) The user selects execution plans; 5) The platform executes the plans on the created instances.

The conceptual diagram describing the sequence of activities that realize interaction between the user and the P-TOSCA engine for cloud application portability is shown in Fig. 3. Porting an application from one CSP to another using P-TOSCA includes the following actions: 1) A user authenticates to the P-TOSCA engine, selects the remote P-TOSCA engine and authenticates to the remote platform; 2) The user

selects an application to be ported from the remote platform; 3) Application's definitions and artifacts are obtained from the remote platform via web services; 4) The platform requires creation of instances according to the specified application topology; 5) The cloud controller creates the instances; 6) The user selects execution plans; 7) The platform executes the plans on the created instances.

4 P-TOSCA DEMONSTRATION

A modified version of the eBay SOA Shopper project (Hansen, 2007) is used as a proof-of-concept of transferring a SOA application between clouds using the P-TOSCA model. The developed application uses the eBay SOAP API to retrieve offers from eBay by given search terms. It offers different interfaces (web service and application for web browser). It is a typical transaction-based application realized with the SOA approach that consumes services from one provider and offers services to other parties. This application is selected as SOA demo since it represents both a consumer and a service provider in same type (i.e. covers the two important aspects of SOA).

One Java EE container hosts the application. It consists of two main modules: *Interface module* that contains the services, which are offered as web interface, REST and SOAP services; *Core and consumer module*, which consumes services from eBay using the eBay SOAP API and converts the data in the required format. When an application user accesses an interface service (whether using SOAP, XML message or HTTP parameter), then the appropriate software module is invoked. The control then continues with the SOA Finder API (which represents a wrapper) with goal to invoke eBay services. Finally the result is returned to the user via corresponding interface. The application is hosted on one VM instance.

The application topology describes the application deployment architecture. The orchestration of application required elements is needed for: a) Enabling a platform independency; b) Easier installation; c) Cloud deployment. Orchestration, in this context, describes the way the services are invoked and managed.

TOSCA specifies the application topology nodes by node types divided in three main categories: *Base type*, which defines the basic components required by the application, such as OS, web server and web application; *Specific types* used to define the specific components required by the application topology, such as Linux OS, GlassFish web server and Java EE Web Application. The deployment and configuration of the application also requires usage of Maven

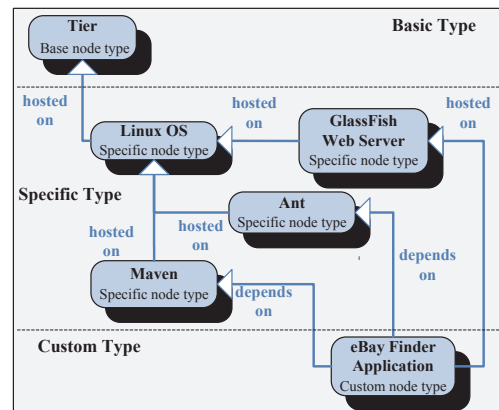


Figure 4: eBay Finder SOA application's topology template.

```
<NodeTemplate id="Tier"
  name="Instance for eBay Finder App"
  type="Tier">
  <Properties>
    <ServerProperties>
      <NumCpus>2</NumCpus>
      <Memory>2048</Memory>
      <Disk>10</Disk>
      <InitialNumInstances>
        1
      </InitialNumInstances>
    <ServerSecurityProperties>
      <ServerSecurityProperty>
        <protocol>TCP</protocol>
        <port>80</port>
      </ServerSecurityProperty>
      <ServerSecurityProperty>
        <protocol>TCP</protocol>
        <port>443</port>
      </ServerSecurityProperty>
    </ServerSecurityProperties>
  </Properties>
  ...
</NodeTemplate>
```

Listing 1: Tier node template definition.

and Ant tools. For that reason these tools are defined as specific types; *Custom types* that define the components developed specifically for the eBay Finder Application.

The application topology template of the eBay Finder SOA application is presented in Fig. 4. Each node of the topology is specified by a node template, and the relationships between the nodes are specified using relationship template, depicted by blue color arrows. Same as nodes, the relationship types are initially set in the specification and each relationship template specifies the type of relationship. All elements are defined by XML. P-TOSCA model uses external namespaces for different custom defined elements, but they will be intentionally omitted from the further listings for clearer representation.

Tier node template definition is presented in Listing 1. In this context we use P-TOSCA extended specification of the *ServerProperties* element with the following: Initial number of instances to be

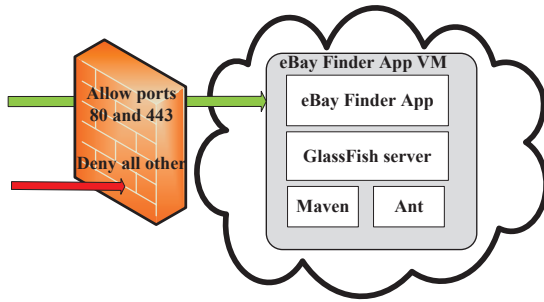


Figure 5: Communication layout of the eBay Finder.

```
<NodeTemplate id="GlassFishWebServer"
  name="GlassFish Web Server"
  type="GlassFishWebServer">
  <Properties>
    <httpport>80</httpport>
    <httpsport>443</httpsport>
    <username>admin</username>
    <password>adminadmin</password>
  </Properties>
</NodeTemplate>
```

Listing 2: *GlassFishWebServer* node template definition.

1; Server security properties with TCP protocol and port 80 (for HTTP); Server security properties with TCP protocol and port 443 (for HTTPS). The *ServerProperties* element was extended by definition of the *ServerSecurityProperty* element with corresponding port identification for HTTP and HTTPS protocols. It specifies details of the communication layout presented in Fig. 5.

Listing 2 shows definition of the GlassFish server node template. According to the P-TOSCA extended specification we define the *Properties* element with Port 80 (for HTTP) and Port 443 (for HTTPS). The user credentials are also defined within the *Properties* element.

Listing 3 shows the definition of the artifact template for configuration of the GlassFish server. The *ScriptArtifactProperties* element within the *ArtifactTemplate* was extended with the *InputParameters* element used to define the input ports and user credentials. Note that these parameters correspond to the already defined properties of the GlassFish Web Server node template.

Unlike the standard TOSCA specification (that suggest usage of BPMN and BPEL languages), P-TOSCA uses XML defined plans. XML definition is used instead of use of BPMN and BPEL languages due to the ambiguity of these languages and the lack of BPMN processing engine at the moment when this project was developed (Kostoska et al., 2014b). Listing 4 shows the definition of the XML *Plan* element for application deployment.

Each *Plan* element contains the node templates and their operations for execution of the required ac-

```
<ArtifactTemplate id="glassfish-configsh" type="ScriptArtifact">
  <Properties>
    <ScriptArtifactProperties>
      <ScriptLanguage>sh</ScriptLanguage>
      <PrimaryScript>
        scripts/GlassFishWebServer /configure.sh
      </PrimaryScript>
      <InputParameters>
        <InputParameter
          nodeTemplateId="GlassFishWebServer"
          property="httpport"/>
        <InputParameter
          nodeTemplateId="GlassFishWebServer"
          property="httpsport"/>
        <InputParameter
          nodeTemplateId="GlassFishWebServer"
          property="username"/>
        <InputParameter
          nodeTemplateId="GlassFishWebServer"
          property="password"/>
      </InputParameters>
    </ScriptArtifactProperties>
  </Properties>
  <ArtifactReferences>
    <ArtifactReference
      reference="scripts/GlassFishWebServer">
      <Include pattern="configure.sh"/>
    </ArtifactReference>
  </ArtifactReferences>
</ArtifactTemplate>
```

Listing 3: GlassFish Artifact template.

```
<Plan id="InstallApplication"
  <NodeTemplateOperations>
    <NodeTemplateOperation ref="GlassFishWebServer">
      <Operation name="install"/>
      <Operation name="configure"/>
    </NodeTemplateOperation>
    <NodeTemplateOperation ref="Maven">
      <Operation name="install"/>
    </NodeTemplateOperation>
    <NodeTemplateOperation ref="Ant">
      <Operation name="install"/>
    </NodeTemplateOperation>
    <NodeTemplateOperation ref="eBayFinderApp">
      <Operation name="install"/>
      <Operation name="configure"/>
    </NodeTemplateOperation>
  </NodeTemplateOperations>
</Plan>
```

Listing 4: XML *Plan* element for eBay Finder deployment.

tion. The operations specified in the plan are executed sequentially in the order of description, unless some operation defines precondition (that should be executed before the operation).

The first activity of the P-TOSCA portability sequence is the preparatory step, where the user authenticates and prepares the CSAR archive.

All definitions and the required artifacts are packed in the CSAR archive as a zip file. The CSAR archive for the eBay Finder SOA application contains: 1) Java EE 7 installation (which represents a zip file); 2) The application deployment artifact (war file); 3) Scripting artifacts for GlassFish; 4) Configuration, installation and deployment scripts. The final archive has a substantial size (over 90MB) due to the size of Java EE 7 installation file.

The next step includes copying to the P-TOSCA engine and starting a procedure defined by the corresponding migration or porting scenario. All these

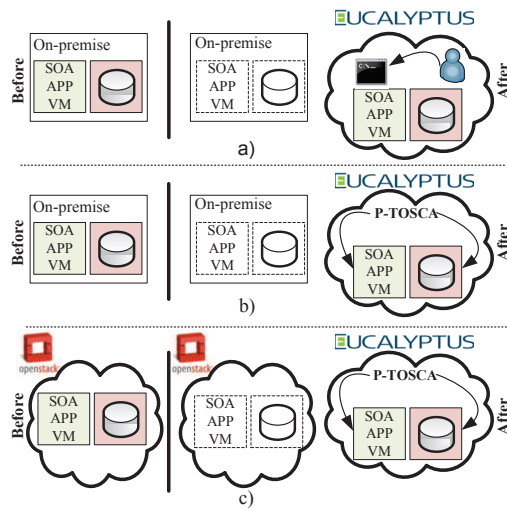


Figure 6: Three scenarios.

steps are performed in a sequence. The user interaction is required only to specify the execution plans.

5 TESTING METHODOLOGY

The test goal was to evaluate the functionality and deployment performance of the P-TOSCA portability model, and to provide a proof-of-concept of automated cloud application portability. A demonstration will be successful if the application migration and porting can be realized by using a P-TOSCA engine following the definitions of the P-TOSCA model. Performance evaluation will show if transferring the application using this approach is realized by a click of a button and consumes less time. Further on, we discuss details on the testing environment, test cases, and test data.

For testing purposes we used two cloud environments: OpenStack and Eucalyptus, isolated in separated VLANs. The OpenStack cloud was installed on one server with Ubuntu server 12.04 LTS. The server hosts all the OpenStack elements: node, cluster and tiers. The Eucalyptus cloud was installed on three physical servers with CentOS 6.5 in such a way that each server uses one Eucalyptus element.

Both frameworks were set with a P-TOSCA engine, which can communicate with the user and the cloud controller. Communication with other P-TOSCA engines is realized by web services.

Fig. 6 presents three different scenarios, where Eucalyptus is presented as the target cloud. *Manual migration onto a cloud (MM)* is a test scenario to evaluate the performance of application migration on the cloud, where the user manually creates an instance,

deploys the application and sets authentication and security rules (Fig. 6 a); *P-TOSCA based migration onto a cloud (PTM)* is a test scenario to evaluate the performance of activities to upload the defined CSAR and to deploy the application using the web interface of the P-TOSCA engine (Fig. 6 b); *P-TOSCA based porting the application from one cloud to another (PTP)* is a test scenario to evaluate the performance to transfer a PaaS hosted application from one cloud environment to other using web services (Fig. 6 c).

The target cloud to migrate or transfer the application is either OpenStack or Eucalyptus cloud. Since both cloud frameworks are used for the three test scenarios, there are a total of 6 use cases.

Functionality testing is realized by testing the application after its deployment on the target cloud for its proper functioning. For this purpose we have selected several characteristic input parameters and specified the expected output. The functionality test actually realizes matching of the expected output and the real obtained output for the same input. Performance testing is defined by evaluation of the total time needed for deployment of the use case.

In the manual migration use case, the time measurement starts with user authentication and the activity of manual preparation, including collection of necessary files and packing into a zip archive, followed by the manual copying of the archive, manual extracting, and manual starting of the scripts, VM instantiation, execution plans and ends when the final installation and deployment activity is finished.

In the P-TOSCA based migration and porting, all activities are performed automatically by the corresponding scripts. The time is measured from the moment when the user authenticates, followed by starting the initial script and finishes with realization of the execution plan. In this use case we have defined user interaction for selection of an appropriate execution plan, although it can also be automated by a corresponding script.

6 EVALUATION

In addition to the original TOSCA specification, our P-TOSCA model gives details on implementation specifics, such as initial number of instances, used protocols and ports for communication, and relevant data about CPUs, memory, disks etc. This information is required for implementation and deploying purposes. XML specified plans and exiting P-TOSCA engine cannot support the full coverage of architecture configurations and deployments as those by the BPEL engine, but this is planned for next P-TOSCA

Table 1: Average time for use case scenarios.

Use case	id	sec	id	sec
MM	$T_d(O)$	1918	$T_d(E)$	1914
PTM	$T_m(O)$	1247	$T_m(E)$	1205
PTP	$T_p(O)$	1093	$T_p(E)$	1022

software release. Currently all sequential deployment and installation activities can be successfully applied by P-TOSCA.

Recently we have presented demo cases by using the P-TOSCA approach: a demo on a small SOA application (Ristov et al., 2014) and a demo of porting the N -tier applications (Gusev et al., 2014). Here we extend the approach on a more general SOA application and give comprehensive details on P-TOSCA application, along with functional and performance evaluation.

Proof of concept is demonstrated for all 6 use cases although OpenStack is not specifically designed neither for interoperability nor portability (Toosi et al., 2014). The overall application porting from Eucalyptus to OpenStack and vice versa are correspondingly demonstrated in the publicly available videos: <http://youtu.be/92KaHt0CyxE> and <http://youtu.be/NRnqrPqO41k>. Some performance related results are presented next to explore the influence of the cloud infrastructure.

Table 1 presents the average of 10 successful test case executions, calculated for manual, P-TOSCA based migration and for P-TOSCA based porting, where OpenStack and Eucalyptus are the target clouds, correspondingly. We observe similar results for both cloud platforms, that is, P-TOSCA based cloud migration is faster than the manual. P-TOSCA based porting is even faster, since the transfer is realized directly between the clouds. Let $c \in \{O, E\}$ identifies the cloud environment, where O stands for OpenStack and E for Eucalyptus. Denote by T_d the time for default manual migration (MM), $T_m(c)$ for P-TOSCA based migration (PTM), and $T_p(c)$ the time for P-TOSCA based porting (PTP).

The target cloud infrastructure is important for the overall deployment process. The obtained results show that migration and porting using P-TOSCA model is faster on Eucalyptus for 3.5% in case of migration and 6.95% in case of porting on Eucalyptus as target cloud instead of on OpenStack. This is due to the different cloud infrastructure used for hosting the cloud environment. The OpenStack cloud environment uses one server as infrastructure to host the node controller, cluster and tier, while the Eucalyptus uses three different physical servers as infrastructure. These results actually show that the better the infrastructure is, the faster the deployment.

The time needed for initial migration using P-TOSCA is slightly greater than the time for porting between the clouds because during the initial migration we access the platform from outer network. The reason is in the setting of the cloud environment and the duration of the copying process, which is due to the network latencies and throughput.

The experiment environment uses user interaction with WAN (Wide Area Network) protocol, while both clouds (OpenStack and Eucalyptus) are connected with 1Gb LAN (Local Area Network). So, the data transfer time for archive upload requires more time when using WAN, while the data transfer between clouds was faster due to the faster LAN protocol.

Although the size of the archive is greater than 90MB, most of the time differences between the manual and P-TOSCA migration are due to the time required by the user to access the visual interface of the cloud and to manually execute the actions.

Another time-consuming activity is introduction of the new cloud environment. In the conducted experiments we do not analyze the time needed for learning the environments (i.e. we assume them as known environment). In real life, the time needed for manual migration may be significantly longer if the users meet the new cloud environment for the first time. It takes time to get acquainted to the interface and the options offered by the CSP.

During this process the user does not have control over the instances (as in manual migration), since the platform uses specific security policies and authentication, but at the end of the process the user specific requirements are set.

Other SOA characteristic is modularity. If the SOA application consists of several loosely-coupled modules, each module can be defined as a separate node in the application topology and can be deployed on a dedicated instance, while at the same time the relationships between the modules can be described using the relationship template.

Our approach described on a PaaS level can be evaluated by the approach (Petcu and Vasilakos, 2014) with a high portability degree, since, no code has to be rewritten and recompiled, no restructuring of data and applications are required and no services are re-configured. All activities are realized straight forward and automatically, starting from archiving, transferring of archives, deployment and installation.

7 CONCLUSION

So far, TOSCA can be used with the BPEL engine, or extended to P-TOSCA and using the P-TOSCA en-

gine. All other published research results concern development of various tools that support the process, while the on-going projects translate the TOSCA definitions in specification used by specific cloud management tools. One of the benefits of the P-TOSCA platform is that the user does not have to learn and use the native interfaces of CSPs, making the management of hosting a SOA application an easy task.

A proof-of-concept of automated cloud application portability was demonstrated in this paper using P-TOSCA portability in case of migration or porting of a transaction-based SOA application.

The demonstration of migration and porting on OpenStack and Eucalyptus cloud environments can be used also for other cloud environments, since P-TOSCA uses a generalized approach for specification and modeling of an application, and provides a scripting mechanism for automated sequence of activities. The main benefit is the possibility to easily switch CSPs and port the application between clouds. It seems that this functionality is unattractive to CSPs, since they prefer vendor lock-in and would prefer not to give the customer an easy way out of their cloud. This looks similar to the ongoing fight for mobile phone devices by mobile providers. However, as the time goes by, the customers would prefer portability and easy way in and easy go out options from future CSPs. It is not a question of should CSPs do it, but when to do it.

REFERENCES

- Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., and Wagner, S. (2013). OpenTOSCA – a runtime for TOSCA-based cloud applications. *11th Int. Conf. on Service-Oriented Computing, LNCS* vol. 8274, pages 692–695. Springer.
- Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., and Schumm, D. (2012). VINO4TOSCA: A visual notation for application topologies based on TOSCA. *OTM 2012, Part I, LNCS* vol. 7565, pages 416–424. Springer-Verlag.
- Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616.
- Distributed Management Task Force (2010). Open virtualization format specification version 1.1.0.
- Erl, T. (2004). *Service-oriented architecture: Concepts, Technology, and Design*. Prentice Hall.
- Gonidis, F., Simons, A. J., Paraskakis, I., and Kourtesis, D. (2013). Cloud application portability: an initial view. *6th Balkan Conf. in Informatics*, pages 275–282. ACM.
- Gusev, M., Kostoska, M., and Ristov, S. (2014). Cloud P-TOSCA porting of N-tier applications. *22nd Int. TELFOR Forum, IEEE Conf. Publications*, pages 935–938.
- Hansen, M. D. (2007). *SOA Using Java Web Services*. Pearson Education, Inc, Upper Saddle River, NJ.
- Katsaros, G., Menzel, M., Lenk, A., Revelant, J. R., Skipp, R., and Eberhardt, J. (2014). Cloud application portability with TOSCA, Chef and Openstack. *Cloud Engineering (IC2E), 2014 IEEE Int. Conf.*, pages 295–302.
- Kopp, O., Binz, T., Breitenbücher, U., and Leymann, F. (2013). Winery – modeling tool for TOSCA-based cloud applications. *11th Int. Conf. on Service-Oriented Computing, LNCS* vol. 8274, pages 700–704. Springer.
- Kostoska, M., Chorbev, I., and Gusev, M. (2014a). Creating portable TOSCA archive for iKnow university management system. *Federated Conf. Computer Science and Information Systems (FedCSIS), IEEE Conf. Publications*, pages 767–774.
- Kostoska, M., Gusev, M., and Ristov, S. (2014b). P-TOSCA portability model for PaaS hosted applications. Tech. Report LiT:22/2014, University Ss Cyril and Methodius, Computer Science and Engineering.
- Li, F., Vogler, M., Claessens, M., and Dustdar, S. (2013). Towards automated IoT application deployment by a cloud-based approach. *Service-Oriented Computing and Applications (SOCA), 6th IEEE Int. Conf.*, pages 61–68.
- OASIS (2014). Online files.
- Ortiz Jr, S. (2011). The problem with cloud-computing standardization. *IEEE Computer*, 44(7):13–16.
- Papazoglou, M. and Van Den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415.
- Petcu, D. and Vasilakos, A. V. (2014). Portability in clouds: approaches and research opportunities. *Scalable Computing: Practice and Experience*, 15(3):251 – 270.
- Rana, O. (2014). The costs of cloud migration. *Cloud Computing, IEEE*, 1(1):62–65.
- Ristov, S., Kostoska, M., and Gusev, M. (2014). P-TOSCA portability demo case. *2014 IEEE 3rd Int. Conf. on Cloud Networking (CLOUDNET)*, pages 269–271.
- Toosi, A. N., Calheiros, R. N., and Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Comput. Surv.*, 47(1):7:1–7:47.
- Valipour, M., Amirzafari, B., Maleki, K., and Daneshpour, N. (2009). A brief survey of software architecture concepts and service oriented architecture. *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE Int. Conf.*, pages 34–38.

A Cloud-based Data Analysis Framework for Object Recognition

Rezvan Pakdel and John Herbert

Department of Computer Science, University College Cork, Cork, Ireland
{rp3, j.herbert}@cs.ucc.ie

Keywords: Cloud-based Big Data Analytics, Big Data, OpenCV, Machine Learning, Real-time Object Recognition.

Abstract: This paper presents a Cloud-based framework using parallel data processing to identify and recognize an object from an image. Images contain a massive amount of information. Features such as shape, corner, color, and edge can be extracted from images. These features can be used to recognize an object. In a Cloud-based data analytics framework, feature detection algorithms can be done in parallel to get the result faster in comparison to a single machine. This study provides a Cloud-based architecture as a solution for large-scale datasets to decrease processing time and save hardware costs. The evaluation results indicate that the proposed approach can robustly identify and recognize objects in images.

1 INTRODUCTION

Object recognition is one of the fundamental challenges in computer vision. Object recognition by computers has been active for more than two decades (Torralba et al., 2010) and includes image processing algorithms which extract features from an image to detect an object. Detection and recognition depend on the quality of the amount of images, noise, and occlusion. Nowadays, the number of collections of images is increasing quickly, especially in critical areas such as medicine, health, astronomy. Processing these images therefore has a key role in science.

In this paper, features such as edge, corner, shape, and color are used. Basically, finding an unknown object is easy, however recognizing it, is difficult. For being able to recognize what an object is, image features are helpful however one feature is not enough to recognize an object. A combination of features is needed to have better object recognition accuracy (Hetzl et al., 2001). The features can be used to make a unique object signature. In this work, eleven classification models with a machine learning model are used in parallel to recognize an object from an image. Each classification model has one or more image features.

High performance processing of a huge amount of data across multiple machines in parallel requires much resources and a reliable infrastructure. Cloud Computing can be used to solve this issue by leveraging distributed data, computing resources and services. Cloud Computing has several advantages.

First, the multi-core architecture decreases hardware cost and increases computing power and storage capacity. It also is the widespread adoption of Services Computing and Web applications. It is the exponentially growing data size (Foster et al., 2008).

2 STATE OF THE ART AND RELATED WORK

Finding and identifying objects in an image is an important task in computer vision. Humans can recognize objects with irrespective view, size/scale, and rotation or translation. Extracting some features from an image, then applying some machine learning classifier to the extracted features, is a method that is used in Image Processing (Rosten et al., 2010). Various approaches have been used to detect objects accurately in images, such as geometry-based, appearance-based and feature-based approaches (Yang, 2009). Feature extraction is the main element in most object recognition methods.

The image features can be divided into two groups, Global and Local. Local features calculate features over the results of subdivision of the image based on the image segmentation or edge detection. On the other hand, global features calculate features over the entire image or just regular sub-area of an image (Choras, 2007). So, the global features describe the visual content of the entire image. Global features like shape and texture, are attractive because they produce very compact representations of images, where

each image corresponds to a point in a high dimensional feature space. Also, standard classification algorithm can be used for global features(Lisin et al., 2005).

There are several algorithms and methods proposed for extracting features from images. Harris corner detection is a method to detect and match point features like corners or edges(Schmid and Mohr, 1997). Canny edge detection, developed by John F. Canny in 1986(OpenCV,), uses a multi-stage algorithm to detect a wide range of edges in images. Region and contour detectors are also methods for object recognition. Detectors using image contours or region boundaries, should be less likely to be disrupted by cluttered backgrounds near object boundaries. Region detectors are used for category recognition(Andrew and Brady, 2004) but are not practical for a large number of images representing different categories. The performance for object class recognition approaches is often reported for entire methods (Berg et al., 2005; Fergus et al., 2003). Recognizing an object can be done by extracting these features from an image. Research shows a combination of methods can be useful to recognize objects in an image(S.Arivazhagan1, 2010). Feature detectors can use machine learning algorithms. For instance a corner detector can create a model and then apply it directly to the image(Rosten et al., 2010). Lots of different machine learning algorithms are used for image classification. After considering various machine learning algorithms including Bayesian Nets, Decision Trees, Genetic Algorithms, Nearest Neighbors and Neural Nets, J48 decision tree is used for this work. Decision trees are popular because they are easy to understand. Rules can also be extracted from decision trees easily.

In this work, the OpenStack Cloud Computing platform is used. The framework contains eleven models, each of which is assigned to a worker role in the Cloud environment. The models are created based on labeled objects in images. When new image data is sent to the Cloud, each worker role creates a signature for each object in order to recognize it. There will be 11 results for an object. The evaluation component of our architecture, processes the results and provides the most accurate result.

3 CLOUD-BASED DATA ANALYSIS ARCHITECTURE

In this work, a Cloud-based data analytics framework is proposed. It will use classification models for object recognition utilizing machine learning methods. Utilizing the Cloud infrastructure will provide a better

performance of smart-phones, laptops and computers even with limited computational resource.

Analyzing large volumes of heterogeneous data can be done by data analysis methods such as machine learning, computational mathematics, and artificial intelligence. A Cloud-based architecture is chosen for this work, due to its scalability, efficiency, and manageability. As Cloud Computing is designed for the distributed systems with fault tolerance, it uses a pools of resources to deploy a virtual machine. (Han et al., 2010) presents the average cost of parallel image pattern recognition tasks in Cloud, supercomputers and clusters and shows that the cost for running the task in the Cloud is cheaper. Virtual machine instance can be simply moved or scaled up or down to make the best use of the hardware without compromising performance. It significantly improves the cost efficiency under the limitation of computing power of smart-phones, computers and etc. One of the other advantages of a Cloud-based data analysis framework is that with a queue-based architecture an asynchronous scheme is applied where worker roles are asynchronously coupled. This means that scaling or adding/removing instances does not affect other worker roles. Furthermore, (L.S.Kmiecik, 2013) pointed out that the size of the classifier model is independent of the size of the training data whereby even though the training dataset is very huge the model does not need to be very big. Huge amounts of training data can be stored in the Cloud as backup for future analysis.

The Cloud framework consists of four types of queues:

- **Data Queue.** It is used to communicate between the Client and the Controller Node. This queue is considered as a general queue. When a Client uploads data into the Blob the URL of the link is added into the Data Queue.
- **Task Queue.** There is separate Task Queue for each worker roles. The Controller reads data from the Data Queue and assigns it into different Task Queues.
- **Model Queue.** When the result of each worker role is prepared and evaluated, it will be sent to this queue as the evaluation result .
- **Response Queue.** The best result, identified object label and its class will be in this queue for users.

The Cloud framework also consists of four types of worker roles. Here are the definitions of these worker roles.

- **Controller Node.** This controls and manages the incoming data and adding the new messages to the

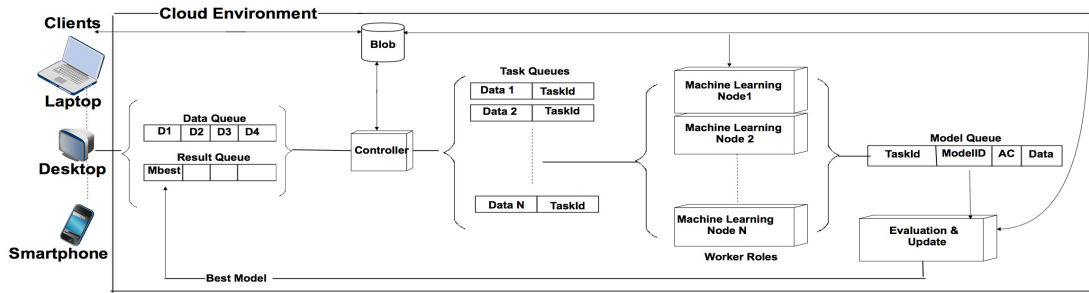


Figure 1: Structure of The Object Recognition System.

Task Queue are the main role of this queue. It also stores the incoming data to the Blob storage.

- **Machine Learning Nodes.** Each of the Machine Learning Nodes reads a message from its own Task Queue and starts producing a model based on a machine learning classifier with different features. It generates a data record from the extracted features and uses its own classifier model to identify the data's label and class. It then submits the data record, its label, and the accuracy to the Model Queue.
- **Evaluation Node.** This worker role will check if all models produced results. If yes, it will find the best model. The best label satisfying the threshold will be submitted to the result queue.
- **Response Node.** This worker role will show the result to the user.

In this framework, RabbitMQ, which is a queue service provider, is used as a messaging system between worker roles. A dedicated virtual machine is assigned for the RabbitMQ service. Also, one virtual machine is assigned for each Machine Learning Node. Each of Controller, Evaluation, and Response worker roles is assigned a virtual machine. They have different responsibilities in order to finish a task.

The process of Cloud-based data analysis is illustrated in Figure 1. Once the client uploads new data, the URL of the data is collected by the Controller node and assigned as different tasks to machine learning worker roles. Each of the worker roles is designed to handle a machine-learning task. Each machine learning classifier generates and evaluates a model. Based on all evaluation results of all models in the Evaluation node the best model is identified. The best identified label and class for the recognized object will be presented to the user by the Response node.

In this system, a comprehensive Cloud-based data analysis framework is developed by combining big data analytics and Cloud Computing technologies. This provides analysis as a service from data deliv-

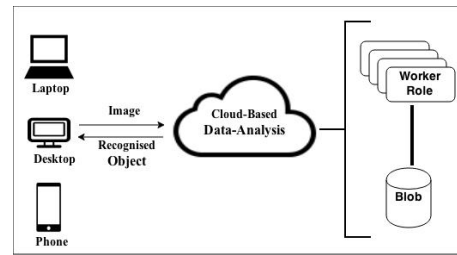


Figure 2: Overview of The Object Recognition System.

ery and analysis to storage, in order to optimize data analysis. An overview of the Cloud-based data analysis framework is presented in Figure 2.

4 CASE STUDY OF OBJECT RECOGNITION

In the case study of object recognition, a Cloud-based architecture is designed and used. It is deployed in the OpenStack Cloud environment. We use nine physical machines to implement our private Cloud environment, four of them to be used for compute nodes (Nova), one for the controller node, one for the block storage (Cinder), one for object storage (Swift), one for dashboard and accessing the Cloud environment (horizon), one for neutron. The process of control flow is shown as follows:

- **Training Model:**
 - 1) Dataset is collected and manually labeled.
 - 2) The dataset is processed, by passing it to the eleven machine learning.
 - 3) Eleven models will be created, each by a worker role.
 - 4) Eleven models are stored.
- **Testing Model:**
 - 1) User sends data to the Cloud-based Data Analysis framework.
 - 2) Data is loaded into the framework.
 - 3) Data will be passed to each worker role.
 - 4) Worker roles will process the data by using the existing models.
 - 5) The results containing labels

of the data and accuracy, will be prepared. 6) Results will be compared and the best model is chosen. 7) The labeled data will be presented to the user.

There is a set of features for each worker role and classifier model. Color, Shape, and some basic features such as Edge, Corner. There are also two shape signature methods proposed by the authors of this work, Ordered and Sorted Signature. Here is an explanation of each model.

1. M1: Basic features (Edge and corner)
2. M2: Color (Three color schemes, RGB, YCbCr, and HSV, and also the combination of all of them)
3. M3: The Ordered-Signature.
4. M4: The Sorted-Signature.
5. M5: Basic feature and Color.
6. M6: Ordered-Signature and Basic features.
7. M7: Sorted-Signature and Basic features.
8. M8: Ordered-Signature and Color features.
9. M9: Sorted-Signature and color features.
10. M10: Color, Ordered-Signature, and Basic features.
11. M11: Color, Sorted-Signature, and Basic features.

Each of these eleven models has a different set of features. Each worker role assigns to a model has a set of image processing algorithms to extract the needed features and build data for model evaluation. The data is evaluated by a stored model of each worker role and a label and the accuracy of its evaluation will be presented as an output. When all eleven worker roles finish their jobs, another worker role will check all results. It will find the best model by checking the accuracy and will then send the best result back to the user.

4.1 Data Collection

The dataset used in this paper consists of 219 leaf images. The dataset is divided into 3 classes, type3 (Pittosporum Tobira), type14 (Betula Pendula), and type21 (Cercis Siliquastrum). Each image is 255 by 255 pixels and in JPEG format. A total of 120 images are used as the Training set (T) and the remaining 99 images as the Testing set (S). Figure 3 shows the dataset leaf types.



Figure 3: Dataset: type3 (Pittosporum Tobira), type14 (Betula Pendula), and type21 (Cercis Siliquastrum).

4.2 Feature Extraction

Image feature extraction is at the heart of this framework. In this work, Four methods are used to recognize an object in an image.

4.2.1 Edge Detection

Edge is an important feature and digital image processing, edge detection is an important subject (Nadernejad et al., 2008). The boundaries between regions in an image are defined as edges (Nadernejad et al., 2008). There are several algorithms to perform edge detection. The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. Figure 4 shows the Canny edge detection for one type of the leaf.



Figure 4: Left Side: Original Image. Right Side: Canny Edge Operator Applied.

4.2.2 Corner Detection

The corner is defined as a location in the image where the local autocorrelation function has a distinct peak. There are various methods to detect corners in computer vision. Harris corner detection is used to extract information in this work (Malik et al., 2011). Harris Corner Detection is based on the autocorrelation of image intensity values or image gradient values. The corner features extracted by using Harris corner detector method are analyzed different values of sigma, threshold and radius (K. Velmurugan and Baboo, 2011). Figure 5 shows the Harris Corner Detection for one type of the leaf.



Figure 5: Left Side: Original Image. Right Side Harris Corner Detector Applied.

4.2.3 Shape Detection

Shape representation and description techniques can be generally classified into two classes: contour-based and region-based methods (Shotton, 2005). Several methods that have been developed by past researchers for the shape detection such as using generalized Hough transform (Duda and Hart, 1972),

and template matching(Korman et al., 2013). The contour-based method is used as the shape detector in this work. Contour shape techniques only exploit shape boundary information. Contour-based approaches are more popular than region-based approaches. This is because human beings are thought to discriminate shapes mainly by their contour features. Another reason is because, in many of the shape applications, the shape contour is of interest. While the shape interior content is not important (Zhang and Lu, 2004). The shape detection signature algorithm has been designed and developed by the authors of this work.

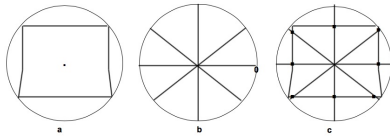


Figure 6: (a) Contour of Object, (b) Clock-wise Lines (c) Intersection of Lines and Contour.

In figure 6, (a) contour detection is used to detect the outer boundary of an object. Then based on the contour, the center of the object and radius is calculated. In the mask image that is shown in (b), the lines are drawn based on an angle step. This is the angle between successive radius lines drawn from the center to the boundary. For example, In Figure 6, an angle step of 45 degree produces eight lines from the center of the circle to the boundary. The overlap of (a) and (b) will produce (c). From (c), the intersection of object outer boundary and lines can be extracted.

The distance between the center of the object and the intersections are absolute distance. Absolute distance does not work well, as it depends on scale. However, if the absolute distance is divided by the radius of the circle, it gives us a scale invariant number between zero to one. By using this technique, it does not matter how small or big the object is and the numbers produced by this technique will be the same for any size of the same object. If the same object is rotated, the center of the object and radius are possibly different. So, our solution is to divide the absolute distance by the longest radius distance of each object. It gives better results for the same object with different sizes and rotations.

Starting from the longest radius line and continuing clockwise, the list of radius lengths for an object produces a signature, called the Ordered-Signature. Sorting this list from high to low produces another signature, the Sorted-Signature. Our results shows better shape recognition accuracy for Sorted-Signature. These two techniques are size and rotation invariant.

4.2.4 Color Detection

There are different color spaces in images such as RGB, HSV, YCbCr, to distinguish an object in an image(Patil et al., 2011). The first step in this work was to resize the image to 256 by 256 pixels. Second, calculating the average of each space like R,G and B for all pixels. This process will be done for all spaces and finally nine numbers will be produced. The next step is to get a histogram to calculate their statistical moments (mean, std and skewness). After applying this process we will get 27 features of each image($3 \times (\text{rgb} + \text{YCrCb} + \text{HSV component}) \times 3$ features(mean, std, skewness value)). Figure 7 shows the three color spaces.

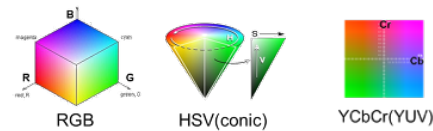


Figure 7: Color Model.

4.3 Machine Learning Classifier

In this work, machine learning is used for classification. The general idea of any feature-based method is to first find a set of discriminative features that can help distinguish between objects in an image, then run a machine learning model based on those features over a training set. Finally apply the model to classify a new object in an image. The machine learning classifier was trained to produce the classification model. The Weka machine-learning package(Witten and Frank, 2005) was used in this study to develop the machine learning mechanism for the object detection and recognition. There are various classifier algorithms in Weka for classification such as Naive-Bayes, Neural Network, NB-three, Decision Tree known as J48. J48 is used because it is an open source Java implementation of the C4.5 algorithm in the Weka data mining tool.

5 EXPERIMENT AND RESULT

The framework requires a machine learning model classifier for each model. There are 11 worker roles each of which needs a model classifier in order to recognize an object in an image. The model classifier should be trained properly. The training process is done with 120 images for all three classes of leaf type. Once the training process is done, 11 trained model classifiers will be assigned to 11 Machine Learning Nodes. In the test process, our dataset contains 99

images for all classes of leaf type. We start the test process by sending 99 requests to the framework. We discuss the model accuracy as well as the worker roles performance below.

Figure 8 reports the accuracy of distinguishing objects based on their features for each model. The result shows that the retrieval accuracy is increased in the models that contain the Sorted-Signature feature. Three models, Basic (Corner and Edge), Sorted-Signature and color features and Sorted-Signature and Basic features have good performance. There are 89 images are correctly recognized by these features. Figure 8 shows that the color model we use, does not provide good results. Analyzing the result shows that either the color feature or the method we used needs to be replaced or improved. Figure 9 also

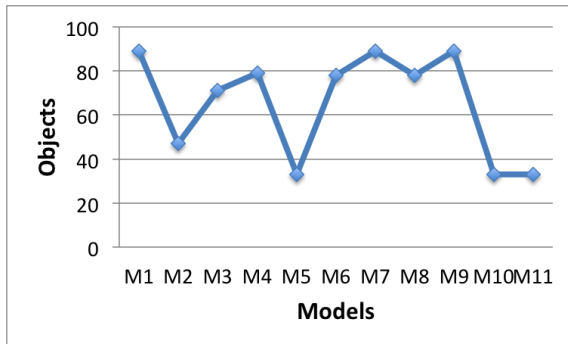


Figure 8: Model Validation: M1-Basic features M2-Color M3-Ordered-Signature M4-Sorted-Signature M5-Basic feature and Color M6-Ordered-Signature and Basic features M7-Sorted-Signature and Basic features M8-Ordered-Signature and Color features M9-Sorted-Signature and color features M10-Color, Ordered-Signature, and Basic features M11- Color, Sorted-Signature, and Basic features.

shows the validity of object recognition for three different leaf classes. The accuracy of the framework depends on the training datasets and the type of the test images. Object recognition process flow consists of *Data-Created*, *Controller-Received*, *Waiting-To-Process*, *Processed*, and *Evaluated* statuses. First, a user sends data to our framework (*Data-Created*). The Controller node receives the data (*Controller-Received*) and puts the data into Task Queues for Machine Learning nodes to process (*Waiting-To-Process*). After processing, the result will be sent to the Model Queue (*Processed*). The evaluation node reads all the results, evaluates them, and finds the best result for the requested data (*Evaluated*). The final result will be sent to the user in the final step. As figure 10 shows, the waiting time for processing data is different for each model. This step is where the data is in the queue to be processed and waiting for a computing node to take and process it. The waiting time

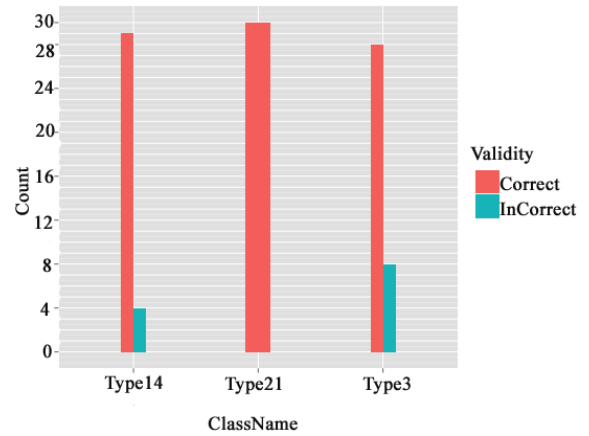


Figure 9: Class Validation

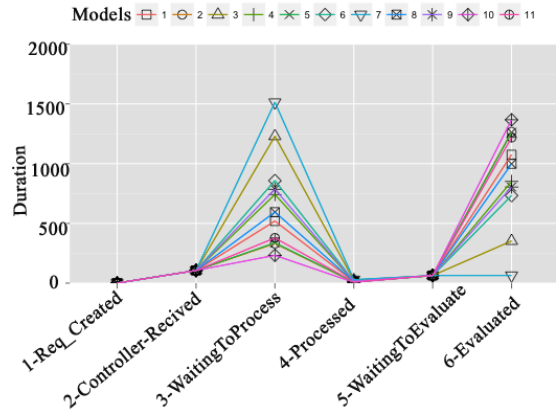


Figure 10: Detailed Framework Performance Configuration A.

is higher when the actual processing time for a model is higher. It helps us to detect the models that has higher processing time. As this is a dynamic Cloud environment, then a controller can increase the number of compute nodes for the detected model when it's required. The model with Sorted-Signature feature has the highest processing time and the model with basic features has the best. The Cloud environment provides us the opportunity to control the number of worker roles and compute nodes in order to utilize the current nodes and add more nodes in order to improve performance. This would not happen in a local environment. Figure 11 shows our second experiment with the same data but with the different configuration. In this experiment, we twice the number of worker roles but not the number of compute nodes (VMs). As it is obvious in the Figure 11, with the new configuration we could achieve a better performance as the processing time for the worst model became one third less in comparison to the first experiment. It also improve the overall performance as the evaluation is 33% faster. An intelligent controller

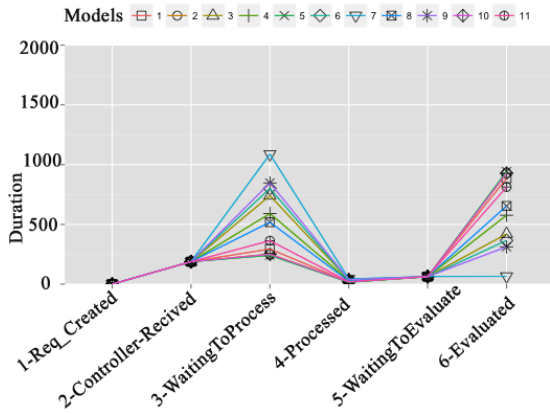


Figure 11: Detailed Framework Performance Configuration B.

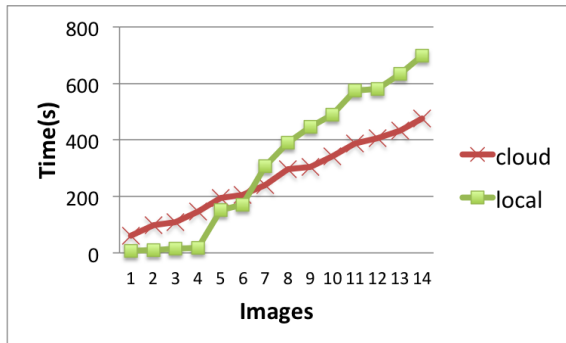


Figure 12: Processing Time In Local and Cloud Environment.

can help to make our framework providing a better performance that is in our future work.

5.1 Processing Time Cloud Vs. Local

Comparing our framework in our private Cloud environment with a local machine provides us an overview of the differences between them. We could not run our framework for more than 14 images in a local machine as the processing time was significantly high. Figure 12 shows the difference between the local and Cloud performance for 14 images. The local machine for small number of data is better than a Cloud environment. However, when we have Big Data, the local machine is not helpful. As the figure shows, the local machine is faster for less than 7 images, but as the number of images are increased, the performance is worst than our Cloud framework. The processing time is as important as the accuracy. Figure 13 shows the average processing time for each model. The average processing time for the models with Basic feature and with Color feature is about 15 seconds. The models with more than one feature has a higher processing

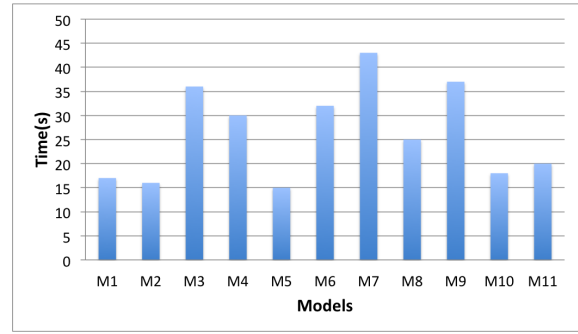


Figure 13: Average Processing Time.

time like the model with Basic and Color features that has an average of 35 seconds. The most significant processing time is related to our Ordered-Signature and Sorted-Signature algorithms that is very helpful in the accuracy.

6 FUTURE WORK

In our future work, there will be a controller node deciding to add or remove nodes to the framework, send the request to an appropriate node, and resend a request to another node when a response for a specific request does not show up before a threshold waiting time. So in order to improve the performance of Cloud-based data analytics, new mechanisms is needed to be exploited to dynamically allocate system resources for different machine learning nodes based on the performance of each machine learning classifier. There also will be a reusable mechanism for the output of the models with one feature for reusing in the models with multiple features in order to reduce the processing time. It is planned to further evaluate this work with datasets that are bigger in size, and variety.

7 CONCLUSION

In this paper, we proposed a robust approach of object recognition using hierarchical classification by combining feature detection and machine learning algorithms. With integration of the Cloud infrastructure, the system provides superior scalability and availability for data analysis and model management. The data analysis and model evaluation are conducted remotely in the Cloud. The experimental results show that feature detection algorithms can be done in parallel in the Cloud to get the result in a fast efficient way.

REFERENCES

- Andrew and Brady, M. (2004). An Affine Invariant Saliency Region Detector. In *European Conference on Computer Vision*, pages 228–241.
- Berg, A. C., Berg, T. L., and Malik, J. (2005). Shape matching and object recognition using low distortion correspondence. In *In CVPR*, pages 26–33.
- Choras, R. S. (2007). Image feature extraction techniques and their applications for cbir and biometrics systems. *International Journal of Biology and Biomedical Engineering*.
- Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15.
- Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *In CVPR*, pages 264–271.
- Ferzli, R. and Khalife, I. (2011). Mobile cloud computing educational tool for image/video processing algorithms. In *2011 Digital Signal Processing and Signal Processing Education Meeting, DSP/SPE 2011*, pages 529–533. Affiliation: Microsoft Corp., Unified Communications Group, Redmond, WA, United States; Affiliation: Group of Inf. and Comm. Sys., Scientific Park, Universitat de Valencia, Spain; Correspondence Address: Ferzli, R.; Microsoft Corp., Unified Communications Group, Redmond, WA, United States; email: rferzli@ieee.org.
- Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared. *2008 Grid Computing Environments Workshop*, pages 1–10.
- Han, L., Saengngam, T., and van Hemert, J. (2010). Accelerating data-intensive applications: a cloud computing approach image pattern recognition tasks. In *The Fourth International Conference on Advanced Engineering Computing and Applications in Sciences*.
- Hetzl, G., Leibe, B., Levi, P., and Schiele, B. (2001). 3d object recognition from range images using local feature histograms. In *Proceedings of CVPR 2001*, pages 394–399.
- Korman, S., Reichman, D., Tsur, G., and Avidan, S. (2013). Fast-match: Fast affine template matching. In *CVPR'13*, pages 2331–2338.
- K.Velmurugan and Baboo, L. D. S. (2011). Article: Image retrieval using harris corners and histogram of oriented gradients. *International Journal of Computer Applications*, 24(7):6–10. Full text available.
- Lisin, D. A., Mattar, M. A., Blaschko, M. B., Benfield, M. C., and Learned-miller, E. G. (2005). Combining local and global image features for object class recognition. In *In Proceedings of the IEEE CVPR Workshop on Learning in Computer Vision and Pattern Recognition*, pages 47–55.
- L.S.Kmiecik (2013). Cloudcentered,smartphonebasedlong-termhumanactivity recognition solution. *IEEE Transactions on Image Processing*.
- Malik, J., Dahiya, R., and Sainarayanan, G. (2011). Article: Harris operator corner detection using sliding window method. *International Journal of Computer Applications*, 22(1):28–37. Full text available.
- Nadernejad, E., Sharifzadeh, S., and Hassanpour, H. (2008). Edge detection techniques: Evaluations and comparison. *Applied Mathematical Sciences*, 2(31):1507–1520.
- OpenCV. *OpenCV*.
- Patil, N. K., Yadahalli, R. M., and Pujari, J. (2011). Article: Comparison between hsv and ycbcr color model color-texture based classification of the food grains. *International Journal of Computer Applications*, 34(4):51–57. Full text available.
- Rosten, E., Porter, R., and Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):105–119.
- S.Arivazhagan1, R.Newlin Shebiah1, S. N. L. (Oct 2010). Fruit recognition using color and texture features bibtex. *Journal of Emerging Trends in Computing and Information Sciences*.
- Schmid, C. and Mohr, R. (1997). Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):530–535.
- Shotton, J. (2005). Contour-based learning for object detection. In *In Proc. ICCV*, pages 503–510.
- Torralba, A., Murphy, K. P., and Freeman, W. T. (2010). Using the forest to see the trees: Exploiting context for visual object detection and localization. *Commun. ACM*, 53(3):107–114.
- Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Yang, M.-H. (2009). Object recognition. In LIU, L. and ZSU, M., editors, *Encyclopedia of Database Systems*, pages 1936–1939. Springer US.
- Zhang, D. and Lu, G. (2004). Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1 – 19.

Factors Affecting Cloud Adoption and Their Interrelations

Radhika Garg and Burkhard Stiller

*Communication Systems Group CSG, University of Zürich UZH, Binzmühlstrasse 14, CH-8050, Zürich, Switzerland
{garg, stiller}@ifi.uzh.ch*

Keywords: Cloud Computing, Cloud-based Services, Cloud Adoption, Technical Factors, Economical Factors, Organizational Factors.

Abstract: Cloud Computing has emerged as a paradigm that relies on sharing resources over the network and, therefore, potentially has cost advantages in terms of lower variable and capital cost. However, the adoption of cloud-based technology for a given IT (Information Technology) setting is a complex decision as it is influenced by multiple interdependent factors. To successfully adopt cloud-based services and evaluate their consequential impact, relevant factors, which denote the performance of such services, have to be identified. This paper, therefore, analyzes and identifies relevant technical, economical, and organizational factors. This is performed as exploratory research consisting of performing (a) a literature review and (b) multiple case-studies with 17 organizations, who have adopted or plan to adopt cloud-based services. Also, as these factors are not mutually exclusive, this paper discusses interrelations of these factors and its complexity.

1 MOTIVATION AND INTRODUCTION

Ever since the advent of Cloud Computing (CC) numerous definitions have been proposed. The definition provided by NIST (Badger, 2012) states, “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” In addition to the explicit listing of major technical characteristics of CC, this definition also hints at an economical and organizational impact of CC.

CC is based on computing technologies such as of virtualization, Service-oriented Architecture, Web 2.0, Web 3.0, and Distributed Computing. The major benefits being pay-as-you-go model, on-demand scalability, business agility, increase in economies of scale. Depending on the provisioning location, CC has four deployment models (1) Private Cloud, (2) Public Cloud, (3) Hybrid Cloud, and (4) Community Model. Initially, CC delivered three fundamental service models: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). But, today it is extended to XaaS (Anything-as-Service), which can include anything

such as Network-as-a-Service, Database-as-a-Service, or Communication-as-a-Service.

As numerous cloud-based alternative solutions are available, in order to successfully adopt one in an organization it is important to evaluate value and impact of incorporating the cloud into business for fulfilling IT (Information Technology) requirements. Currently, many organizations tend to fail to retrieve the best return from the cloud-based solution. This is due to the lack of complete understanding of factors (both by cloud providers and customers) that impact organizations, which adopt cloud-based services to fulfill their IT requirements. Factors that affect CC depend on (a) requirements of the cloud-customer, (a) type of service model, and (c) deployment model. Therefore, in order to formalize the impact of cloud-based services, and to take a decision whether to adopt cloud or not, identifying factors from technical, economic, and organizational perspective is necessary (Garg, 2014b).

The identification of factors in this paper here is done based on exploratory research. In exploratory research, conclusions are based on the review of available literature/data, or qualitative approaches such as discussions, focus group, or case-studies. Therefore, for identifying major factors from the technical, economical, and organizational perspective, this paper uses (a) review of available literature and (b) case-studies with 17 organizations,

who have adopted cloud-based services, or plan to do so. Once relevant factors are identified, the method to identify interrelations between these factors is discussed. This leads to assisting organizations in successfully adopting cloud-based services and predicting impact with a possibility of preparing counter-measures in advance in case a failure occurs in future.

The remainder of this paper is structured as follows. Section 2 discusses related work done in the field of the identification of relevant factors from all perspectives influencing the adoption of clouds in an organization. It also highlights existing gaps in this field and how this paper bridges them. Section 3 determines the research methodology followed in terms of research questions addressed, the design, and the study. Section 4 summarizes and analyzes key findings and explains how interrelations between identified factors can be identified. Section 5 summarizes and concludes the paper.

2 RELATED WORK

According to a recent report of 2014 by the International Data Corporation (IDC), spending on public IT cloud services would increase to a compound annual growth rate of 22.8 percent over the next five years, hence making it a \$127 billion value (IDC, 2014). In order to completely utilize benefits of CC, industry and research have tried to understand and solve challenges affecting the cloud adoption, such as that of security and privacy. However, these efforts have been concentrated mainly toward addressing technical issues, such as multi-tenancy, scalability, monitoring of cloud-architecture, or performance (Tang, 2014), (Kaur, 2013), (Kuyoro, 2011). There are some efforts toward optimizing cost or Return-of-Investment (ROI) of adopting cloud-based services (Chaisiri, 2012), (Misra, 2011). In addition, there are studies to understand and calculate how cloud-based services conserve capital and reduce ongoing cost. Comparing Total Cost of Ownership (TCO) of cloud-based service and on premise solution leads to an assessment of total costs involved in deploying these two models (Walterbusch, 2013), (Martens, 2012). However, efforts in both of these directions follow a narrow approach and do not identify and analyze the impact of adopting cloud-based services from all perspectives (Garg, 2014a).

There was effort invested in the direction of addressing how the decision of adopting cloud-based services can be taken (Geczy, 2012), (Hoesseini,

2011), (Saripalli, 2011). They do identify that this decision is influenced by multiple factors that can be interrelated. However, neither do these approaches list factors that should be considered to take such a decision nor do they identify that these factors belong to all technical, economical, and organizational fields. Also, the identification of interrelations is only restricted to the analysis of their relative importance and it does not include their interdependence in terms of their performance requirements and evaluation.

Table 1: Comparison of Related Work with Respect to Main Characteristics of Current Work.

Features	Methods for Decision of Adoption of Cloud	Methods for Optimizing Technical Factors	Methods for Optimizing Cost
Technical Analysis	×	✓	×
Economical Analysis	×	×	✓
Organizational Analysis	×	×	×
Inter-relations Between Factors	✓ (partially)	(only between technical factors)	×

As shown in Table 1, gap still exists in research efforts in terms of identifying factors, which influences the decision of cloud adoption. The comparison of related work to the work done in this paper is based on four key features; “✓” describing the presence and “×” denoting the lack of that feature. This paper, therefore, fills this gap by (a) identifying factors from all perspectives-technical, economical, and organizational, and (b) identifying interrelations between these factors.

3 RESEARCH METHODOLOGY

Given the lack of empirical data for these factors that should be considered, while evaluating impact of cloud-based services or decision to adopt cloud, this paper follows an exploratory method. This is a qualitative approach, to understand information in depth and analyze diverse and complex data. In order to identify relevant factors, two methods were used. First is that of a case study, wherein semi-structured interviews were conducted with organizations. Second is that of analyzing available literature, both from industrial and academic surveys.

3.1 Case Study Design

Case studies are useful for collecting data, where little or no information exists. It helps to understand a “case” from holistic and real-world perspective (Yin, 2013). This paper here lists and analyzes data collected from case studies conducted with 17 organizations, who have adopted or plan to adopt cloud-based services for fulfilling their IT requirements. These interviews were conducted between June 2013 and October 2014, and their duration varied between 45 and 60 minutes. These interviews were either conducted on landline phone or as face-to-face meetings.

3.1.1 Selection of Participants for Case Studies

The selection of organizations interviewed was based on random and convenience sampling. Random sampling is considered as a fair way of selecting a sample from a given population since every member is given equal opportunities of being selected (Gravetter, 2010). This was combined with

convenience sampling, due to the availability and proximity of participants. Convenience sampling helps to collect information in more depth as participants are in proximity (Gravetter, 2010).

Bias, which can often result from convenience sampling, was avoided with two countermeasures: (a) Participants were selected with varied geographical scope and domain of expertise. This helped in collecting data, which can be representative of the complete population. (b) Questions were based on interviewees’ experience of adoption of cloud-based services (varied as per their domain of expertise) and with general benefits or challenges associated with the adoption of cloud-based services, therefore, making generalizations possible. Details of organizations are listed in Table 2. Interviewees from these organizations were senior decision-makers with experience of assessing various cloud alternatives. Participation was voluntary and their identity is kept anonymous, while reporting and analyzing the data collected. This was mainly due to the confidentiality and sensitivity of data and opinions shared by the decision makers of various organizations.

Table 2: Details of Organizations involved in Case Studies.

Org	Domain of Expertise	Organization's Size ^a	Geographic Scope Served
C1	ICT Provider	60000	Europe, USA, Singapore
C2	Health Insurance	450	Switzerland
C3	Communications	20000	Switzerland
C4	IT Infrastructure Provider	5000	Europe, USA, Australia, China
C5	Financial Services	2600	Worldwide
C6	Property and Life Insurance	4000	Switzerland
C7	Professional Services	180000	Worldwide
C8	Networking Solutions	67000	Worldwide
C9	ICT Association	-	Switzerland
C10	Financial Services	140000	Worldwide
C11	Banking Services	255000	Worldwide
C12	Technology and Consulting	431000	Worldwide
C13	Technology and Consulting	305000	Worldwide
C14	IT services	318000	Worldwide
C15	IT Infrastructure Provider	107000	Worldwide
C16	Life Insurance	3000	Switzerland
C17	Digital Media Solutions	12000	Worldwide

^aNumber of employees as per October 2014.

3.1.2 Research Questions

These case studies were conducted as semi-structured interviews. Owing to the semi-structured format of the interview, the interviewer was able to adapt the interview based on individual circumstances. All topics discussed (major ones listed below) within this interview supported two research questions that served as trigger point for discussion.

- What are the factors (technical, economical, and organizational) that should be considered while making a decision to adopt cloud-based services for fulfilling IT requirements?
 - Key reasons for adopting a cloud-based solution.
 - Factors that decide the eligibility of candidate to be migrated to cloud-based solution
 - Limiting factors and risks for selecting a cloud-based service.
 - Factors that decided which deployment model will be selected.
- Are these factors interdependent? If yes, then how?
 - Impact of migration to cloud-based service on organization.
 - Evaluation of success or failure of adoption

3.2 Literature Study

The literature review is used as the second method to collect data in terms of factors affecting the adoption of clouds in an organization. This covered reviewing various technical and economic papers, white papers, and surveys provided by industry and academic research. Even though these efforts do not list factors from all relevant perspectives, they collectively give a valuable insight into challenges and benefits of the cloud adoption decision.

The topics covered in literature available can be broadly categorized into the following categories:

- Security and privacy issues related to the adoption of cloud computing
- Technical issues in migrating and integrating cloud-based services with existing systems to fulfill IT requirements
- Generalized benefits and challenges of adopting cloud-based services
- Structural changes in ROI and TCO models, including cost benefits

4 KEY FINDINGS AND ANALYSIS

The data analysis in this paper was based on the targeted result of identifying factors and their interrelations. In order to avoid any misinterpretations all case studies were fully transcribed. The data (both of case studies and literature review) was aggregated, converged, and aligned in a database, thereby helping in identifying multiple occurrences of factors and cross case-study synthesis. This enabled the identification of regularities and differences across and within various data sources and provided for plausible explanations on importance of a particular factor. Also, due to the presence of multiple data sources, result credibility was ensured. The qualitative data was categorized in three categories (technical, economic, and organizational). Factors, found in the exploratory research, can have a different priority or relevance for different organizations. This depends on overall requirements and expectations from the cloud-based service. Thus, key findings of this exploratory research in terms of technical, economic, and organizational factors and their interdependencies are derived as follows.

4.1 Technical Factors

CC has major benefits in terms of its technical

Table 3: Relevant Technical Factor.

Scalability
Availability
Elastic Resourcing
Network Quality
• Bandwidth
• Connectivity
Interoperability
Speed/Latency
Quality of Service
Portability
Compliance and Standards
Usability
• Application Launch Time
• Graphics Agility
• Simplicity
Data Loss
Reliability
• Elasticity
• Disaster Recovery
Privacy
Compatibility with Existing Systems
Software Assurance
Customization
Integration
Management and Maintenance of Identity Platform
Management of Authentication Platform
Security Configuration and Maintenance
• Confidentiality
• Integrity
• Availability
• Auditability
• Multi-tenant Trust
Functionality
Triability
Delay in Migration and Data Transfer
Vendor Lock-in
Process Redesign
Accessibility
Standards for API
Backup
• Data
• Application
Workload Management
• Classification
• Capacity Planning
• Performance Management
• Configuration Management
• Mission Criticality
Multi-tenancy

characteristics. Table 3 lists the key factors, which must be evaluated before adopting a cloud-based service. An IDC report reported 75% of the respondents mentioned security as one of most important factor to be evaluated, while adopting cloud-based services (Sultan, 2011). This is of utmost importance for the public cloud. Another important factor is that of sensitivity of data. As the provider has full access to the data, responsibility of data theft, loss, and adherence to legal and regulative guidelines for storage of data has to be carefully evaluated. This factor has higher priority in cases, when public or hybrid deployment models are selected (as compared to private deployment model). For Small and Medium Enterprises (SME), it is important to take measures to increase the network quality in terms of bandwidth and connectivity (Yeboah-Boateng, 2014). Network quality is important, because in many cloud architectures (*e.g.*, Amazon Elastic Block Store (EBS) architecture) the data storage layer is abstracted in the compute layer of the application. These compute and data storage nodes are connected via a network. If the network is not of good quality, the application can fail to respond (Joyent, 2014).

As pointed out by every organization, which participated in these case studies, vendor lock-in is an obstacle for a successful adoption of cloud-based services. It also has high negative impact in terms of cost and interoperability in case when the service provider has to be switched. It highlights the need of common standards for APIs across cloud-service providers, so that interoperability is possible. Public cloud tends to get significant advantage over private cloud for all organizations, which participated in the case study, because of its capability to handle unexpected hike in workloads. Therefore, a flexible infrastructure capacity and a provisioning time determine a critical factor for the adoption of a cloud. Organizations participating in this case study also mentioned usability and functionality as deciding factors. Not only the technological know how is important for a successful adoption of cloud based services, but also the ease-of use is crucial for these organization

4.2 Economic Factors

As found in these case studies (specifically pointed out by SMEs) and within the literature review, cloud-based services reduce upfront costs and operational complexities of converting small businesses into larger ones (Chaisiri, 2012), (Misra, 2011). These costs are shifted to data centers, which

benefit from economics of scale and scalability. CC follows the Operating Expenditure (OPEX) model and offer elasticity in terms of scaling resources as per demand. This transfers the risk of over- or under-provisioning to the service provider. However, customers should evaluate, if scaling-up of resources (*e.g.*, increasing the power of server) or scaling-out of resources (*e.g.*, increasing the number of servers) is more appropriate for their specific use-case. This is specifically required as clouds operate at the large scale (Hasan, 2012). For example, in some cases, where the number of customers pre-decides the number of software licenses, increasing them later for an unexpected increase in demand will be very expensive or even impossible. Table 4 lists the key factors from the economic perspective.

All organizations, irrespective of its size and geographical scope, considered the reduction of their carbon footprint as one of the major goals. This leads to evaluation of alternatives of cloud-based services so that a best trade-off is achieved between performance, Quality-of-Service (QoS), and energy consumption of storing, processing, and transportation (Mouftah, 2012). Service Level Agreements (SLA) indicate the description of an agreed upon service, service level parameters, guarantees, actions, and penalties in case of failure or violations (Wu, 2012). SLAs help the organization to monitor the performance and billing of the service provider. If any of the guaranteed metric is not fulfilled, the provider incurs penalties.

Table 4: Relevant Economical Factors.

Cost
<ul style="list-style-type: none"> • License • Maintenance • Back-up • Energy • Hardware • Migration • Future Requirements • Performance • Data Loss • Switching Providers • Integration
Operating Cost (OPEX)
Marginal Cost and Profit
Energy Use and Carbon Emission (Carbon Foot Print)
Contracts and Service Level Agreements (SLA)
Billing and Metering of Resource Usage
Traceability and Audibility
<ul style="list-style-type: none"> • Data • Application
Return-of-Investment (ROI)
Total Cost of Ownership (TCO)
Migration Time

However, the major consequence of this is on the business continuity of the customer. Consequent economical losses due to failure of any cloud-based services can be very high. Another important factor is the calculation of the Return-of-Investment (ROI). For calculating ROI of a traditional IT infrastructure, the initial cost of project, the investment made, and the cost savings done owing to the new investment has to be identified (Chang, 2012). However, for calculating the ROI associated with cloud-based services, increase in profit, reductions in cost, license cost, and any implicit cloud costs have to be identified also. Based on the calculated ROI, the suitability of an adoption of a specific cloud-based service can be recognized (Misra, 2011).

4.3 Organizational Factors

As determined from both, literature review and case studies, currently organizational factors are the least evaluated factors for the decision on a cloud adoption. This is because understanding the significance and extent of impact of an adoption of cloud-based services in an organization on the management and operation of IT infrastructure is a challenge. However, the evaluation of a cloud-based service from this perspective is equally important. To name a few required changes in an organization due to CC are a change in the accounting model, the security model, compliance requirements, and the project management (Hoesseini, 2011). Table 5 lists other key factors from the organizational perspective, which must be evaluated before adopting a cloud-based service.

CC has a distinct disadvantage in terms of loss of control of both data and resources. This leads to issues of privacy and security. Also, as cloud-based services lack transparency in terms of location where data is stored and performance levels of the application as compared to the terms in the SLA, it raises problems related to legal and regulative issues (Battey, 2012). Legal risks also include the liability of the service provider to protect the data from security threats and privacy breaches. Security threats include the deletion of data, multi-level risks, physical attacks, and isolation failure (Kuyoro, 2011). On one hand, CC has advantage in terms of improved process efficiency and increased employee productivity by better internal collaboration. On other hand, CC can have a major disadvantage in the overall efficiency and productivity of the organization, if employees are unable to adapt themselves to changes brought by CC. A successful

adoption of cloud-based services is dependent on how easily can the new technology be learnt by employees of the organization (Saini, 2012). As identified by organizations, which participated in case studies, CC has the capability of transforming business, as the employees need to concentrate only on the innovation of application, the cloud-service provider handles everything else. However, to achieve this, technical support from service-providers and the competence of employees of the organization are two crucial factors.

Table 5: Relevant Organizational Factors.

Size of Organization
Degree of Centralization
Managerial Structure
Competence of Employees
Control
Transparency
Business Flexibility and Agility
User and Technical Support from the Provider
Legal and Regulative Compliance
Skills and Expertise of the Cloud Providers

4.4 Relation between Factors

The factors outlined above can have numerous and complex interrelations based on use case-specific requirements of the organization. Therefore, understanding and identifying these interrelations is equally important to completely evaluate any cloud-based service. Figure 1 illustrates an example of how factors can be interrelated. Scalability leads to cost savings, which can be used in acquiring end user systems and training. To manage failures, such as that of an uptime time failure or a data loss, many cloud-service providers recommend customers to maintain multiple levels or redundancy. This, however, leads to higher capital and operational cost for establishing and maintaining such a system. The lack of standards causes major difficulties, when a decision is made to move applications or data between clouds. These problems include (a) security levels, (b) handling data movement and encryption of data, and (c) setting up of network with the same configuration as that of the source cloud. These issues consequently have an economic impact in terms of cost and ROI. Security and privacy issues associated with cloud computing include multiple issues as that of (a) regulatory compliance in terms of liability of data, location of data storage, (b) proper means of data segregation so that availability, reliability, and confidentiality of data is ensured (Kaur, 2013), and (c) a cloud-based solution that has been able to replicate data across multiple sites to ensure proper recovery in case of any disaster

(Kuyoro, 2011). These issues affect the cost, performance, and control an organization has on a cloud-based service. The costs of cloud-based services are lower, mainly owing to the sharing of resources. This, however, means less guarantees of performance. Therefore, it is important to have strict SLAs, where each relevant performance metric is identified with an expected level of performance.

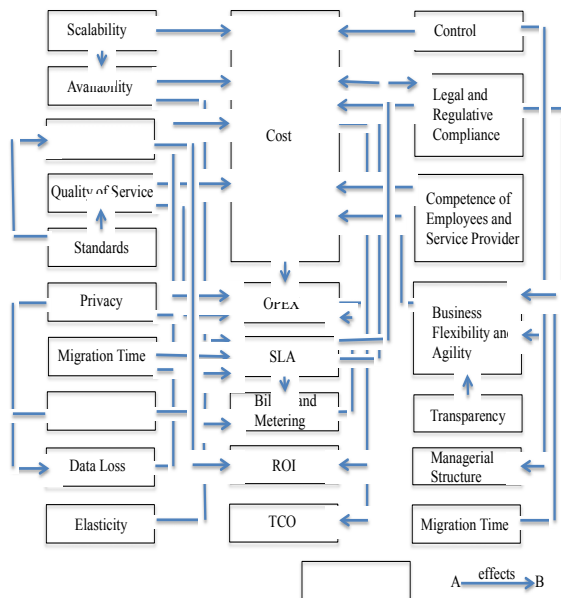


Figure 1: Interrelations between Factors.

As cloud-based services allow scalability and variable levels of resource usage, billing and metering as per the usage is essential. Cloud-based services contribute primarily to business agility and flexibility, but it also restricts an organization in terms of control it has on its own data and applications. Cases of data loss or the need to move data between cloud providers can lead to huge losses. A successful adoption of a cloud-based service is also dependent on the adaptability of the organization in terms of its managerial roles, structure, and competence. Identification of interrelations leads to a systematic evaluation for all tradeoffs and risks involved in considering specific clouds (Garg, 2014b).

5 SUMMARY AND CONCLUSIONS

This paper has bridged the existing gap between identifying relevant technical, economic, and organizational factors and their interrelations. To

achieve this, exploratory research was used in terms of a literature review and 17 case studies with organizations that have either adopted or plan to adopt cloud-based service in the future. In turn, the work showed that interrelations exist between factors of multiple domains and how these relations can be identified.

In conclusion, these factors and their interrelations have a clear influence on (a) the decision of the adoption of cloud-based services and (b) on the impact analysis of a cloud-based service. These lists of factors developed classify available cloud-based services. This classification can be done on the basis of a capability of cloud service providers to fulfill the expected level of performance for each of these factors, thereby aiding organizations to select the best alternative as per IT requirements and business objectives. Furthermore, organizations can ensure that all relevant and critical factors are specified in the SLA with a guaranteed level of expected performance. Lastly, it has also been identified that areas of standardization, interoperability, security, and privacy need to evolve (e.g., ease in encryption of data while in transit between cloud service provider). This is because of their wide impact in terms of technical, economic, and business value.

ACKNOWLEDGEMENTS

This work was partly funded by FLAMINGO, the Network of Excellence Project ICT-318488, supported by the European Commission under its Seventh Framework Program.

REFERENCES

- Badger, L., Grance, T., Patt-Corner, R, Voas, J (2012). "Cloud Computing Synopsis and Recommendations". *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology*, MD, USA.
- Bathey, J., Coyner, D., Krass, P., Mangelsdorf, J. (2012). "The Connected Consumer - Challenging Traditional Financial Roles". Sapardanis. C., (Editor). *CSC World- Summer*.
- Chaisiri, S., Lee, B. S., Niyato, D. (2012). "Optimization of Resource Provisioning Cost in Cloud Computing". *IEEE Transactions on Services Computing*. Vol. 5, No. 2, pp. 164-177.
- Chang, V., Wills, G., Walters, R. J., Currie, W. (2012). "Towards a Structured Cloud ROI: The University of Southampton Cost". *Sustainable ICTs and*

- Management Systems for Green Computing*, IGI Global, pp. 179-200.
- Garg, R., Stiller, B. (2014a). "Design and Evaluation of an Impact Analysis Methodology for the Adoption of Cloud-based Services (IAMCIS)". *10th International Conference on Network and Service Management (CNSM 2015)*, Rio de Janeiro, Brazil, pp. 260–263.
- Garg, R., Stiller, B. (2014b). "Trade-off-based Adoption Methodology for Cloud-based Infrastructures and Services". *8th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2014): "Monitoring and Securing Virtualized Networks and Services"*, Brno, Czech Republic, pp 1–13.
- Gravetter, F., Forzano, B., L. (2010). "Research Methods-For Behavioral Sciences". 4th Edition, *Warsworth Cengage Learning*.
- Geczy, P., Izumi, N., Hasid, K. (2012). "Cloudsourcing: Managing Cloud Adoption". *Global Journal of Business Research*. Vol. 6, No. 2, pp. 57-70.
- Hasan, M. Z., Magana, E., Clemm, A., Tucker, L., Gudreddi, S. L. D. (2012). "Integrated and Autonomic Cloud Resource Scaling". *IEEE Network Operations and Management Symposium (IM 2012)*, Hawaii, USA, pp. 1327-1334.
- Hoesseini, K.A., Greenwood, D., Smith, W. J., Sommerville, I. (2011). "The Cloud Adoption Toolkit: Supporting Cloud Adoption Decisions in the Enterprise". *Software: Practice and Experience, Special Issue: Software and Architectures and Application Development Environments for Cloud Computing*. Vol. 42, No. 4, pp. 447-465.
- IDC (2014). "IDC Forecasts Public IT Cloud Services Spending Will Reach \$127 Billion in 2018 as the Market Enters a Critical Innovation Stage". <http://www.idc.com/getdoc.jsp?containerId=prUS25219014>. Last accessed in February 2015.
- Joyent. (2014). "Joyent vs. Amazon Web Services". <https://www.joyent.com/products/public-cloud/aws-comparison>. Last accessed in November 2014.
- Kaur, K., Vashisht, S. (2013). "Data Separation Issues in Cloud Computing". *International Journal for Advance Research in Engineering and Technology*, Vol. 1, No. X, pp. 26-29.
- Kuyoro S. O., Ibikunle F. & Awodele O. (2011). "Cloud Computing Security Issues and Challenges". *International Journal of Computer Networks*. Vol. 3, No. 5, pp. 247-255.
- Martens, B., Walterbusch, M. Teuteberg, F. (2012). "Costing of Cloud Computing Services: A Total Cost of Ownership Approach". *45th IEEE Hawaii International Conference on System Science (HICSS 2012)*. Hawaii, USA, pp. 1563-1572.
- Misra, C. S., Mondal, A. (2011). "Identification of a Company's Suitability for the Adoption of Cloud Computing and Modeling its Corresponding Return of Investment". *Mathematical and Computer Modeling*. Vol. 35, pp. 504-521.
- Mouftah, H. T., & Kantarci, B. (2012). "Energy-efficient Cloud Computing—A Green Migration of Traditional IT". *Handbook of Green Communications*, Academic Press, pp. 295-329.
- Saini, I., Khanna, A., Kumar. V. (2012). "ERP Systems: Problems and Solution with Special Reference to Small & Medium Enterprises". *International Journal of Research in IT & Management*. Vol. 2, No. 2, pp. 715-725.
- Saripalli, P., Pingali, G. (2011). "MADMAC: Multiple Attribute Decision Methodology for Adoption of Clouds". *4th IEEE International Conference on Cloud Computing (CLOUD 2011)*, Washington DC, USA, pp. 316-323.
- Sultan, N.A. (2011). "Reaching For the "Cloud": How SMEs can manage". *International Journal of Information Management*. Vol. 31, pp. 272–278.
- Tang, B., Sandhu, R., Li, Q. (2014). "Multi-tenancy Authorization Models for Collaborative Cloud Services". *Concurrency and Computation: Practice and Experience*, John Wiley & Sons, Ltd.
- Walterbusch, M., Martens, B., Teuteberg, F. (2013). "Evaluating Cloud Computing Services From a Total Cost of Ownership Perspective". *Management Research Review*. Vol. 36, No.6, pp. 63-638.
- Wu, L., & Buyya, R. (2012). "Service Level Agreement (SLA) in Utility Computing Systems". *Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications*. IGI Global, pp. 286-310.
- Yeboah-Boateng, E.O., Essandoh, K.A. (2014), "Factors Influencing the Adoption of Cloud Computing by Small and Medium Enterprises in Developing Economies", *International Journal of Emerging Science and Engineering*. Vol. 2, No. 4, pp. 13-20.
- Yin, R. K. (2013). "Case Study Research: Design and Methods". Sage Publications.

A Comparative Study of Current Open-source Infrastructure as a Service Frameworks

Theo Lynn, Graham Hunt, David Corcoran, John Morrison and Philip Healy

Irish Centre for Cloud Computing, Dublin 9, Ireland

{theo.lynn, graham.hunt, david.corcoran}@dcu.ie, {j.morrison, p.healy}@cs.ucc.ie

Keywords: Cloud Computing, Open Source, IaaS, Openstack, Cloudstack, Opennebula, Eucalyptus.

Abstract: With the growth of cloud computing in recent years, several commercial and open source IaaS frameworks have emerged. The development of open source IaaS solutions offers a free and flexible alternative to commercial cloud services. The main contribution of this paper is to provide a qualitative comparative of current open-source IaaS frameworks. Existing research papers examining open source IaaS frameworks have focused on comparing OpenStack with a small number of alternatives. However, current research fails to adequately compare all major open source frameworks in a single study and notably lacks the inclusion of CloudStack. Our research paper provides the first overview of the five main open source cloud IaaS frameworks – OpenStack, CloudStack, OpenNebula, Eucalyptus and Nimbus. As such, this review provides researchers and potential users with an up to date and comprehensive overview of the features of each solution and allows for an easy comparison between the open source solutions.

1 INTRODUCTION

Cloud Computing technologies have seen significant adoption in recent years by enterprises, researchers and individuals. It is forecast that the Cloud Computing industry will reach a market size of \$241 billion by 2020 (Reid and Kilster, 2011). Cloud Computing has become an important tool in delivering Infrastructure as a Service (IaaS) for users that require a high level of flexibility and management of their software stack.

The last eight years has seen the rapid proliferation of vendors in the IaaS market, which includes major enterprise players such as Amazon (Amazon, n.a.), Microsoft (Microsoft, n.a.), Google (Google, n.a.), and IBM (Google, n.a.). While these services may offer flexibility to the user, commentators cite cost, lack of portability and lack of interoperability as drawbacks to enterprise IaaS (Mahjoub et al., 2011; Zhang et al., 2013). With each vendor promoting its own infrastructure and incompatible standards, users can suffer from vendor lock-in (Mahjoub et al., 2011; Zhang et al., 2013; Wen et al., 2012).

In some environments it may be preferable for an organization to develop their own cloud, based on an open source framework. Open source frameworks offer an alternative to enterprise clouds by offering

the freedom to modify the source code and build a cloud that is pluggable and open to extensions while reducing costs and avoiding vendor lock-in (Zhang et al., 2013; Wen et al. 2012; Bist et al., 2013). The development of open source solutions is of particular importance for the further proliferation of private and hybrid clouds (Sefraoui, 2012).

The decision to select the most appropriate open source solution can be a challenge for organizations given each framework's specific characteristics. While previous studies (Zhang et al., 2013; Dukarić, R. and Jurić, 2013; Endo et al., 2010; Yaday, 2013) have compared these frameworks, most have focused only on OpenStack and OpenNebula, neglecting CloudStack or have focused on comparing frameworks with different hypervisors (Ristov et al., 2013; Cordeiro, 2010).

The purpose of this paper is to provide a qualitative review of the top five open source IaaS frameworks and present a comparative analysis to aid in framework selection decisions. The frameworks included in this paper have been chosen based on literature reviews and perceived industry acceptance. The open source frameworks detailed in this paper are OpenStack, CloudStack, OpenNebula, Eucalyptus and Nimbus.

The paper is structured as follows. In Section II, we present a background to the study providing an

overview of the benefits of Cloud Computing adoption and discuss the different service models. Section III provides a qualitative review concerning the five chosen frameworks and examines their components. This section also provides a short overview of closed vs. open source IaaS solutions.

Drawing on the review presented in Section III, Section IV provides a comparative analysis of the identified frameworks based on selected criteria. Finally, Section V will discuss our conclusions and recommendations for future developments.

2 BACKGROUND

The National Institute of Standards and Technology (NIST) defines Cloud Computing as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” (Mell and Grance, 2011). Today it is the most widely cited and accepted definition. This definition can be expanded to include five additional characteristics of cloud computing. These characteristics have also become widely accepted and cited; they are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The NIST definition of cloud computing outlines three service models: software as a service (SaaS); platform as a service (PaaS); and infrastructure as a service (IaaS). In addition, it outlines four deployment models: private cloud; community cloud; public cloud; and hybrid cloud.

2.1 Service Models

Cloud Computing is typically classified according to a variety of service models. In recent years, service models such as Storage as a Service (Fielder et al., 2012) and Business Process as a Service (Lynn et al., 2014) have emerged. However, the most common differentiation of service models remains as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), all of which have been given a definition by NIST.

Software as a Service is when the supplier provides working applications running on their own cloud infrastructure, typically these applications would be accessed through a web browser or some other type of web interface or application (Mell and Grance, 2011).

Platform as a Service is when the supplier provides the user with the tools required for deploying an app onto the cloud infrastructure (Mell and Grance, 2011).

Finally, Infrastructure as a Service, the focus of this paper, is when the supplier provides the consumer with processing, memory, storage, networks, and other computing resources (Mell and Grance, 2011).

3 OPEN SOURCE IaaS FRAMEWORKS

Enterprise Solutions (or closed frameworks) typically supply servers and storage to their customers using their own infrastructure through a public cloud development (Zhang et al., 2013). Open source IaaS frameworks take advantage of open source code which can be modified by users to create a single functional package that can be applied to a network of servers and storage to produce IaaS (Zhang et al., 2013). While open source solutions are typically used to develop private clouds they are also suitable for hybrid and public cloud development models (Salih and Zang, 2012; Ristov et al., 2013; Wen et al., 2012). Recent acquisitions such as Citrix’s purchase of CloudStack (Keeps, 2014) suggest that we are likely to see further development of public clouds based on open source frameworks, leading to improved interoperability between different clouds.

The main player offering enterprise solutions is Amazon Web Services, with growing competition from competitors including Microsoft, Google, and IBM (Gartner, 2014). The remainder of this paper will discuss open source frameworks.

Five major projects, OpenStack, Apache CloudStack, OpenNebula, Eucalyptus, and Nimbus dominate the market for Open Source IaaS. A survey conducted in July 2014, by Linux.com, of 550 Open Source cloud computing experts found that OpenStack, CloudStack, OpenNebula, and Eucalyptus are the most popular Open Source IaaS options with all others receiving less than 2% of the result (Williams, 2014). OpenStack was favoured with 63 per cent of the vote, followed by CloudStack, OpenNebula, and Eucalyptus, respectively. OpenStack was also found to be the most popular overall Open Source Cloud Computing project. One of the reasons suggested for OpenStack’s popularity is that the strength of the OpenStack community and the prestige and size of the companies who back OpenStack has given it a

significant advantage over its alternatives (Voras et al., 2013). Another reason suggested is that with the maturation of the market there has been a focus on the bigger projects available in search of a de facto standard (Voras et al., 2013).

An alternative view is that no one Open Source IaaS framework will dominate the market, as each framework is suited to different end use scenarios or applications. For example it is proposed that OpenNebula is more suited to security-sensitive and smart-city scenarios while OpenStack is more suited for time-constrained scenarios (Kostantos, 2013).

At this stage in the market's maturation, the likelihood of new entrants is slim (Voras et al., 2013). Considering the lead the five mentioned projects have over their alternatives, we have decided to focus our analysis on these projects for the purpose of this paper. In the remainder of this section we provide a short discussion of the qualitative features of the different open source IaaS frameworks.

3.1 OpenStack

Announced in 2010, OpenStack is an entirely open sourced IaaS project for private and public clouds. Initially developed by Rackspace Hosting and NASA, it has since developed into a global collaboration of developers and cloud computing technologists. With over 4500 members, 850 companies, and support from major tech industry companies, OpenStack has a very powerful foundation. The OpenStack Open Source Cloud Mission is "to produce the ubiquitous Open Source Cloud Computing platform that will meet the needs of public and private clouds regardless of size, by being simple to implement and massively scalable" (Openstack, n.a.).

OpenStack has three core official software programs, Compute (Nova), Object Storage (Swift),

and Image Service (Glance).

Nova is the computing engine behind OpenStack. It is the cloud computing fabric controller and the central element of the IaaS system (Openstack, n.a.). It manages all the compute resources of the OpenStack cloud. It includes an API (nova-api) server that accepts requests for creating virtual machines (VM), and their local or remote disks (Baset, 2012), and associated metadata in a VM. Other processes within Nova include:

- nova-schedule – takes a VM request and determines where it should be placed.
- nova-compute – creates and terminates VM instances through a hypervisors' API. It includes support for Xen, XenServer/XCP, KVM, VMware vSphere, LXC, QEMU, UML, Hyer-V, etc.

Swift offers cloud storage software enabling the storage and access of objects across different nodes using a simple API. It is designed to be extremely scalable in size and capacity.

Glance provides image services for OpenStack. It provides a catalogue and repository of VM images, also allowing them to be used as templates when deploying new VM instances (Kumar et al., 2014).

Other official OpenStack programs include Cinder, Neutron, Horizon, Keystone, Ceilometer, Heat, and Trove.

Cinder is the block storage component of OpenStack. It provides persistent block level storage devices for use with OpenStack compute instances.

Neutron is OpenStack Networking; it provides networking capabilities to OpenStack and manages networks and IP addresses (Kumar et al., 2014).

Horizon is the dashboard behind OpenStack. It provides a graphical interface for users and administrators.

Keystone provides identity services to OpenStack. It is a means of access to the OpenStack services available to different users.

Ceilometer provides billing services to individual users of the cloud. It also keeps an account of the users' usage of different components of OpenStack.

Heat uses a template based orchestration mechanism to launch multiple composite cloud applications. The templates are in the form of text files that are treated like code and define what resources are necessary for the code.

Trove is a database as a service for OpenStack. It provides functionality for both relational and non-relational database engines.

The latest OpenStack release, Icehouse, was released in April 2014. This release saw

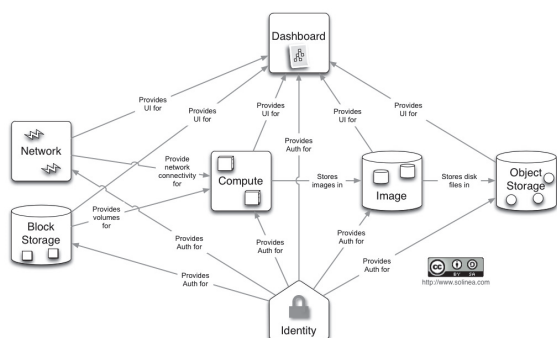


Figure 1: Conceptual OpenStack Architecture, Source: Openstack, n.a..

approximately 350 new features and 2,902 bug fixes added. Features added included support for rolling upgrades, the new feature discoverability that improves workflow, mandatory testing for external drivers, automated scaling of additional resources across the platform.

3.2 CloudStack

CloudStack was developed by Cloud.com and in 2010 released the majority of the project's source code. In 2011 CloudStack were acquired by Citrix who subsequently released the remainder of CloudStack's code. In April 2012 Citrix donated CloudStack to the Apache Software Foundation (ASF). It was accepted into the Apache Incubator and then graduated from the Apache Incubator in March 2013 to become a Top Level Project of ASF (Cloudstack, n.a.) CloudStack doesn't have quite the same foundation of users as OpenStack but a number of large companies do use Apache CloudStack including Apple, British Telecom, Dell, Nokia, and Fujitsu.

CloudStack consists of the management server and the resources to be managed. The management server is the CloudStack software that manages the cloud resources and allocates resources in the cloud deployment. It provides a web interface for administrators and end users, as well as API interfaces for CloudStack API and also the Amazon EC2 interface. The management server also manages assignment of guest VMs, assignment of IP addresses, allocates storage during VM instantiation, and manages snapshots, disk images, and ISO images. It is a single point of configuration for the IaaS cloud and during deployment the user informs it of the resources to be managed.

The CloudStack deployment is organised with hosts contained within clusters, clusters contained within pods, pods contained within zones, and zone contained within regions. Regions are the largest available units in the CloudStack deployment.

Regions essentially consist of a number of available zones, with zones roughly being the equivalent of a data enter. Regions can be used to provide fault tolerance and achieve higher availability and scalability. Zones, as mentioned typically correspond to a single data centre, but a data centre can also contain multiple zones. Pods often represent a single rack. Clusters are a way to group hosts; hosts in a cluster will all have identical hardware and run the same hypervisor. Hosts are single computers and provide the resources that run guest VMs. Figure 2 provides an outline of the

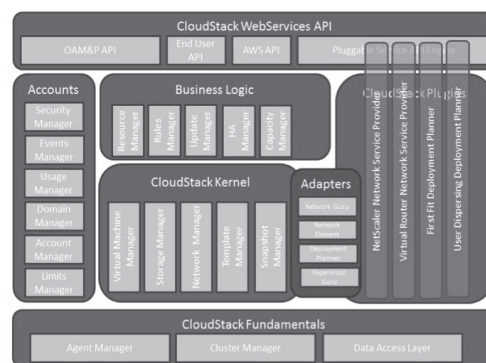


Figure 2: CloudStack Architecture, Source: Cloudstack.

conceptual architecture of a simple CloudStack deployment, note that does not include regions since these would not be necessary for a simple smaller scale deployment (Cloudstack).

The latest version of Apache CloudStack is 4.4.0 and was released in August 2014 and had a number of updates including updating to the latest version of Java.

3.3 OpenNebula

OpenNebula was initially released in 2008 and now operates as an open source project. While it is most commonly used for private cloud, it also supports public and hybrid clouds. The OpenNebula mission is “to become the simplest cloud enabling platform for the enterprise” and their purpose is “to bring simplicity to the private and hybrid enterprise cloud”. OpenNebula has a large user base with telecommunication companies (e.g. Telefonica), system integrators (e.g. Logica), supercomputing centers (e.g. SARA), etc. (Kostantos et al., 2013). Additionally, in March 2010 the main authors of OpenNebula founded C12 Labs to provide a value-added professional service, which many enterprises require for internal adoption, and to enhance OpenNebulas long-term sustainability (Yadav, 2013).

The OpenNebula architecture is a classical cluster with a front end and a set of cluster nodes to run the VMs. At least one physical network is needed for connecting all cluster nodes with the front end (OpenNebula, n.a.). This architecture is flexible and modular, making it easier to integrate multiple storages, network infrastructures and hypervisor technologies than using OpenStack (Wen, 2012).

The OpenNebula software consists of three layers: tools, drivers, and core. An example of an

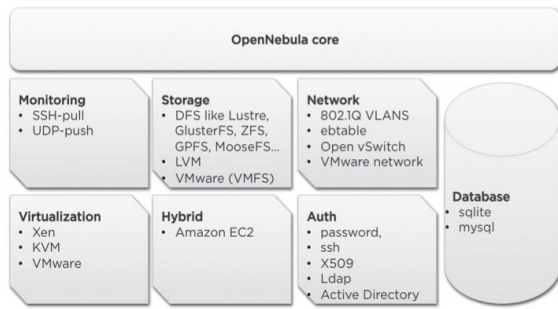


Figure 3: OpenNebula Architecture, Source (OpenNebula, n.a.).

architecture reference model for IaaS clouds is shown in Figure 3.

The **Tools** layer provides interfaces to communicate with users and allows users to manage VMs through the interfaces, which include command line interface (CLI) and libvirt API (Wen, 2012). This layer also contains a scheduler that manages the functionality of the core layer (Wen, 2012).

The **Drivers** layer contains components that communicate directly with the underlying operating system and capture the underlying infrastructure as an abstract service (Wen, 2012).

The **Core** layer (Figure 3) contains the components used to perform user requests and control resources. In this layer, disk images for VMs are stored using datastores (referred to as Image Repositories in previous releases). The monitoring subsystem gathers information on the hosts and the VMs, such as basic performance indicators and capacity consumption, and it also contains the authentication system, which comes with a built-in user/password authentication driver and the choice from SSH Authentication, X509 Authentication, and LDAP Authentication also.

The latest OpenNebula update is version 4.8.0, Lemon Slice, released on the 12th of August 2014. The release saw improvements to the hybrid model, changes to virtual networks that can now include any combination of ranges to accommodate any address distribution, and several other improvements throughout every other OpenNebula component.

3.4 Eucalyptus

Eucalyptus was developed by the University of California-Santa Barbara as an open source Infrastructure as a service and was released in 2008. It allows for the installation of private and hybrid clouds. Eucalyptus target is hybrid installations with

its compatibility with Amazon Web Service AWS. Eucalyptus' scalability is limited in comparison to some of the alternative open source IaaS solutions and, as such, has lost some of the popularity it had in the earlier days of its release (Sefraoui et al., 2012). The introduction of Scalable Object Storage has helped improve the scalability of Eucalyptus.

Eucalyptus has six components (one being an optional component) (Ristov et al., 2013): Cloud Controller (CLC), Scalable Object Storage (SOS), Cluster Controller (CC), Storage Controller (SC), VMware Broker components and Node Controller (NC). These components are divided into three layers: Cloud Layer, Cluster Layer and Cloud Controller.

Cloud Controller is contained in the Cloud Layer and is the entry-point for administrators, developers, project managers, and end users. It queries the other components in Eucalyptus for information and makes requests to the Cluster Controllers. It is responsible for exposing and managing the underlying virtualized resources and offers a web-based administrative interface.

Scalable Object Storage is also contained in the Cloud Layer; this is the Eucalyptus equivalent to Amazon Web Services Simple Storage Service (S3). SOS allows for the implementation of scale-out storage that implements the S3 interface. Eucalyptus also provides a basic storage solution known as Walrus, suitable for smaller cloud deployments.

Cluster Controller is contained in the Cluster Layer. It communicates with the SC and NC and manages the execution of VM instances on specific NCs. The CC must have network connectivity to the machines running the CLC and the machines running the NC.

Storage Controller is also contained in the Cluster Layer. It communicates with the CC and the NC and manages the VM volumes and snapshots from volumes in the cloud. This component functions similarly to the Amazon Elastic Block Store.

The **VMware Broker** is the final component in the Cluster Layer and is the only optional component. It mediates interactions between the CC and VMware and can connect directly to ESX/ESXi hosts or to vCenter Server.

Node Controller is the sole component in the Node Layer. The Node Controller is written in C, hosts the VM instances and manages the virtual network endpoint (Eucalyptus, n.a.).

The latest release of Eucalyptus is version 4.0.1, a maintenance update for version 4.0.0 that was released on the 30th of May 2014. Some of the

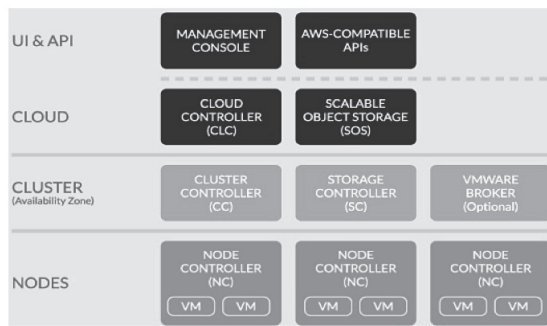


Figure 4: Eucalyptus Architecture, Source: Eucalyptus, n.a..

changes in this release include the possibility to have multiple user-facing services, the possibility to use different object storage backends, and more improvements throughout most of the Eucalyptus software.

3.5 Nimbus

Nimbus is an open source IaaS project that was initially released in 2009. Nimbus' mission is "to evolve the infrastructure with emphasis on the needs of science". Nimbus is built with three goals in mind:

- Enable resource owners to provide their resources as an infrastructure cloud.
- Enable cloud users to access infrastructure cloud resources more easily.
- Enable scientists and developers to extend and experiment with both sets of capabilities.

The main infrastructure components are the Workspace Service site manager, a web services resource framework (WSRF) based remote protocol implementation, an EC2 based remote protocol implementation of their SOAP and Query APIs, Cumulus, RM API, the cloud client, the reference client, the Workspace Pilot, the workspace-control agent, and the metadata server.

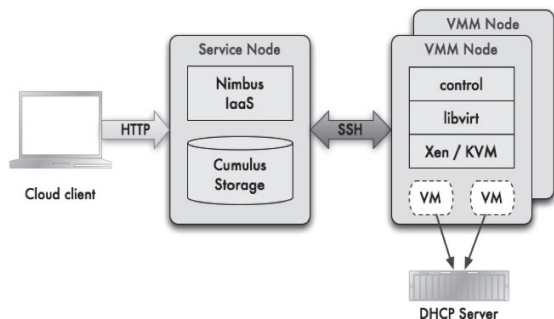


Figure 5: Nimbus Architecture, Source: Nimbus, n.a..

The **Workspace Service** is a site VM manager that different remote protocol frontends can invoke. The **metadata server** responds to HTTP queries from VMs. The **Cloud Client** is the component that enables users to launch instances quickly. The **workspace-control** is a program that can start, stop and pause VMs, implement VM image reconstruction and management, connect the VMs to the network, and deliver contextualization information.

4 COMPARISON OF OPEN SOURCE SOLUTIONS

While the IaaS frameworks discussed in Section III are designed to allow users to manage their own virtual infrastructures, they have different features that should be considered when selecting a framework. This will differ depending on the expected use case and organization. These qualitative features are summarized in Table 1. The set of criteria outlined in Table 1 are derived from a review of the supporting framework documentation and related literature. Based on this review 19 criteria emerged as the main discussed points when comparing open source IaaS frameworks. These criteria are outlined below:

- **Philosophy:** Concerns the main use cases for the frameworks deployment (Wen et al., 2012).
- **Suitability:** Different frameworks will better service the needs of different users. As a result, large commercial organizations are likely to adopt a different framework than research institutes.
- **Architecture:** The architecture of the platform provides details on how the framework was built and how it operates (Mahjoub et al., 2011)
- **API Support:** Cloud APIs are application programming interfaces that are used to build applications (Wen et al., 2012; Endo et al., 2010; Semolinski and Thain, 2010).
- **Amazon Support:** Given that AWS is the predominant enterprise cloud framework, compatibility with such public clouds is an important factor for many users (Wen et al., 2012).
- **Cloud Implementation:** The cloud deployment models (Mell & Grance, 2011) that are suitable for the IaaS framework.
- **Hypervisor:** A hypervisor also known as a VM Manager (VMM) is used to describe a technique to allow multiple guests to run concurrently on a

Table 1: Comparison of Open Source IaaS Solutions.

Capability/features	OpenStack	CloudStack	Eucalyptus	OpenNebula	Nimbus
Established Origin	2010 Rackspace, NASA, Dell, Citrix, Cisco, Canonical etc.	2010 Cloud.com	2008 Santa Barbara university, Eucalyptus System Company	2008 European Union	2009 University of Chicago
Philosophy	Offers Cloud Computing services		Mimic Amazon EC2	Private, highly customizable cloud	Cloud tailored to scientific researchers
Suitability	Enterprises, service providers and researchers	Enterprises, service providers and researchers	Large commercial enterprises, Research institutions	Large commercial companies and public institutions	Research institutions
Architecture	Integration of OpenStack object and OpenStack compute	Hierarchical with four main components: - Management Server, - Availability Zone, - Pod, - Computer Nodes.	Hierarchically grouped from CLC via the CC to the NC ; - Hierarchical - Five components - Minimum two servers	Three modules contain all components; - Centralized - Three components - Minimum two servers	Three modules contain all components; - Centralized - Three components - Minimum two servers
API Support	Native API, Amazon EC2 API, CloudFiles REST API.	Amazon EC2 API, S3	Amazon EC2 API,	Native API in Ruby and JAVA. XML-RPC API for interfaces creation. OGF OCC1 & Amazon EC2 APIs.	EC2 APIs, S3 APIs, JAVA client APIs.
Amazon Support	EC2, S3	EC2, S3	EC2, S3, EBS, IAM, AMI	EC2, EBS, AMI	EC2, S3
Cloud Implementation (deployment)	Public Hybrid Private	Public Hybrid Private	Private Hybrid	Private Hybrid	Private Community
Hypervisor	KVM, Xen, VMware ESX, ESXi, Hyper-v, LXC, QEMU, UML, PowerVM, Bare metal	VMware, Oracle VM, KVM, XEN	KVM, Xen, VMware	KVM, Xen, VMware ESX, ESXi	Python, Bash, Ebttables, Libvirt, KVM, Xen
Programming Language	Python	Java	Java, C, Python	Java, Ruby and C++	Java, Python
Community	+++++	++++	+++	+++	++
Release Frequency	<4 months	4 months	>4 months	>6 months	<4 months
Ease of use	+++++	+++++	++	+++	+++
Supported OS	Linux, Windows, Requires x86 Server	Depending on the Hypervisor and hardware - Mac OS X, Asianux, CentOS, Debian, DOS, Fedora, FreeBSD, Novell Netware, Oracle Enterprise Linux, Ubuntu, Red Hat Enterprise Linux, Sun Solaris, SUSE Linux Enterprise, Windows.	Linux (Ubuntu, Fedora, CentOS, OpenSUSE et Debian)	CentOS, Debian, Fedora, RHEL open-SUSE, SLES, and Ubuntu.	Most Linux distributions
Storage	Object and block storage supported. Volumes are persistent (data retained until the volume is deleted, independently from the VM). File storage is supported through Swift (organizing the files in containers).	Supports for iSCSI, NFS, SMB/CIFS; support for OpenStack Swift and Amazon S3	Support for iSCSI, EBS, Amazon S3. Hardware support for industry-standard Storage Hardware.	Hardware support for Fibre Channel, iSCSI, NAS shared storage, SCSI / SAS / SATA. Non-shared and shared file systems (NFS, LVM with CoW, VMFS, etc.).	
Networking	VLAN NO VLAN Public IP's Private IP's SDN IDS Load- balance Firewalls VPN; OpenStack Compute	VLAN, Public IP,	VLAN NO VLAN Public IP's Private IP's; DHCP server on the cluster controller	VLAN NO VLAN Public IP's Private IP's Ebttables OVSwith; Manual configuration	DHCP server installed on nodes
User Interface	Web interface (i.e. Dashboard) and Command line interface to deploy VMs and a console to manage the VMs.	Web interface and Command Line Interface (CLI)	euca2ools (CLI)	Web interface and Command Line interface (CLI)	Web-Services, specifically: Nimbus Web
Security	API includes protection against DoS attacks or faulty clients. The project concept is introduced by Nova, allowing administrators to manage other user accounts and the project resources. Keystone used for identity management.	CloudStack Security Groups	The Cloud Controller generates a public/private key code pairing for user authentication	Authentication by passwords, secure shell and RSA key code pairings Lightweight Directory Access Protocol ; Authentication framework based on passwords, SSH RSA key-pairs or LDAP. Various administration roles. Multi-tenancy for public clouds.	Public Key Infrastructure
Error Robustness	Replication	Replication	Separate clusters reduce likelihood of correlated errors; Cluster controller's separation	Permanent database to store information about hosts, networks and virtual machines; Database backend (registers virtual machine information)	Regular check and backup of worker nodes; Periodic verification of cloud nodes
Load Balancing	The Cloud Controller	TCP Load Balancer	The Cloud Controller	Nginx	Le Context Broker
Licensing	ApacheLicence Version2	ApacheLicence Version2	BSD-Licence	ApacheLicence Version2	ApacheLicence Version2
Document Support	+++++	+++++	+++	+++	++

host machine (Mahjoub et al., 2011; Wen et al., 2012; Von Laszewski et al., 2010).

- **Programming Language:** Concerns the scripting language in which the framework is written.
- **Community:** A community may consist of individuals, teams or organizations that are users, developers or service providers. While all may not contribute to the development, the size of the community is a good indication of future developments of the framework (Wen et al., 2012).
- **Release Frequency:** The strength of the community will influence the regularity of updates to the open source frameworks. Typically these range from between 4-6 month release cycles.
- **Ease of Use:** The complexity of setting up the system and managing VMs. This can be impacted by the supporting documentation available (Wen et al., 2012). Easy of management is crucial for successful implementing Infrastructure as a service (Voras et al., 2013).
- **Supported OS:** Concerns the operating systems that are implemented on the machines.
- **Storage:** Concerns the support for storage technologies such as network-attached storage, direct-attached storage and backup technologies (Voras et al., 2013). How data is stored on the virtualized pools is critical to achieving flexibility, scaling and ease of use (Wen et al., 2012; Voras et al., 2013).
- **Network:** Cloud frameworks must give attention to virtual networks to ensure optimal performance (Voras et al., 2013). Networks can be considered a means of using services deployed on the VMs and managing the cloud environment (Voras et al., 2013; Von Laszewski et al., 2013). This includes support for VLAN, and firewalls.
- **Interface:** Refers to how the user will access and manage their resources, typically through a command line interface (CLI) or browser interface (Mahjoub et al., 2011; Wen et al., 2012).
- **Security:** Given the perceived security treat of cloud computing, the protection of data is imperative to any IaaS framework Lynn et al., 2014; Mahjoub et al., 2011).
- **Error Robustness:** Concerns how the system will tolerate component failure and continue to operate effectively if there is a component failure (Mahjoub et al., 2011).

- **Load Balancing:** In computer networking terminology, load balancing denotes how a workload is distributed across multiple machines to optimize resource utilization (Mahjoub et al., 2011).
- **Licensing:** Refers to the restrictions of the user on distributing and modifying the software with or without the payment or royalties (Wen et al., 2013). The most common license of open source IaaS frameworks is Apache License Version 2.0 (Openstack, n.a.; Cloudstack, n.a., Von Laszewski et al., 2012).
- **Documentation Support:** The level of documentation available to users will impact on the ease of use for setup and management. One of the challenges is keeping documentation updated in line with frequent new releases

5 CONCLUSIONS

In this paper, we present an up to date qualitative review of the main open source infrastructure as a service frameworks. Building on this, we deliver a means of comparison between these frameworks based on a set of criteria developed to evaluate open source frameworks. The paper adds to the literature by providing a comparison of five open source solutions based on the latest releases. We believe that the evaluation will to help potential adopters understand the functions of these frameworks and aid in the early stage adoption decision-making process. This work aims to illustrate the difference offerings for each of the IaaS frameworks and that user requirements may be significantly different, impacting on their framework selection.

In future research, the criteria outlined in this paper will be used to expand this study to compare open and closed IaaS frameworks. We also look to develop performance criteria that can be combined with the existing paper to develop a comprehensive quantitative and qualitative comparison tool.

ACKNOWLEDGEMENTS

The research work described in this paper was supported by the Irish Centre for Cloud Computing and Commerce, an Irish national Technology Centre funded by Enterprise Ireland and the Irish Industrial Development Authority.

REFERENCES

- Amazon Web Services Homepage. <http://aws.amazon.com/>.
- Baset, S. (2012). Open Source Cloud Technologies. *Proceedings of the Symposium on Cloud Computing – SoCC'12*, 28-27.
- Bist, M., Wariya, M., & Agarwal, A. (2013, February). Comparing delta, open stack and Xen Cloud Platforms: A survey on open source IaaS. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, 96-100.
- CloudStack Homepage. <http://cloudstack.apache.org/>.
- Cordeiro, T. D., Damalio, D. B., Pereira, N. C. V. N., Endo, P. T., de Almeida Palhares, A. V., Gonçalves, G. E., Mångs, J.-E. (2010). Open Source Cloud Computing Platforms. *2010 Ninth International Conference on Grid and Cloud Computing*, 366-371.
- Dukarić, R., & Jurić, M. B. (2013). A Taxonomy and Survey of Infrastructure-as-a-Service Systems. *Lecture Notes on Information Theory*, 1(1), 29-33.
- Endo, P. T., Gonçalves, G. E., Kelner, J., & Sadok, D. (2010, May). A survey on open-source cloud computing solutions. In *Brazilian Symposium on Computer Networks and Distributed Systems*, 3-16.
- Eucalyptus Homepage. <https://www.eucalyptus.com/>.
- Fielder, A., Alleweldt, F., Kara, S., Brown, I., Weber, V., & McSpedden-Brown, N. (2012) Cloud Computing Study. For the European Parliament's Committee on Internal Market and Consumer Protection. IP/A/IMCO/ST/2011-18.
- Gartner. (2014). *Magic Quadrant for Cloud Infrastructure as a Service*. Available: <http://www.gartner.com/technology/reprints.do?id=1-1UKQQA6&ct=140528&st=sb>. Last accessed 30th Sept 2014. Last accessed 30th Sept 2014.
- Google Cloud Platform Homepage. <https://cloud.google.com/compute/>.
- IBM IaaS Homepage. <http://www.ibm.com/cloud-computing/ie/en/iaas.html>.
- Keeps, B. (2014). *Cloudstack execs take the money and run or were they pushed*. Available: <http://www.forbes.com/sites/benkepes/2014/09/04/on-the-three-year-anniversary-cloudstack-execs-take-the-money-and-run-or-were-they-pushed/>. Last accessed 30th Sept 2014.
- Kostantos, K., Kapsalis, A., Kyriazis, D., Themistocleous, M., & Rupino de Cunha, P. (2013). Open-Source IaaS Fit For Purpose: A Comparison Between OpenNebula and OpenStack, *International Journal of Electronic Business Management*, 11(3), 191-201.
- Kostantos, K., Kapsalis, A., Kyriazis, D., Themistocleous, M., & Rupino, P. (2013). OPEN-source iaas fit for purpose: a comparison between opennebula and openstack, *11(3)*, 192-202.
- Kumar, R., Gupta, N., Charu, S., Jain, K., & Jangie, S. (2014) Open Source Solution for Cloud Computing Platform Using OpenStack. *International Journal of Computer Science and Mobile Computing*, 3(5), . 89-98.
- Lynn, T., O' Carroll, N., Mooney, J., Helfert, M., Corcoran, D., Hunt, G., Van Der Werff, L., & Morrison, J. (2014) Towards a Framework for Defining and Categorising Business Process-As-A-Service (BPaaS), *Proceedings of the 21st International Product Development Management Conference*.
- Mahjoub, M., Mdhaftar, A., Halima, R. Ben, & Jmaiel, M. (2011). A Comparative Study of the Current Cloud Computing Technologies and Offers. *2011 First International Symposium on Network Cloud Computing and Applications*, 131-134.
- Mell, P., & Grance, T. (2011) The NIST definition of cloud computing. Recommendations of the National Institute of Standards and Technology, Washington, D.C. (NIST Special Publication 800-145).
- Microsoft Azure Homepage. <https://azure.microsoft.com>.
- Nimbus Homepage. <http://www.nimbusproject.org/>.
- OpenNebula Homepage. <http://www.opennebula.org/>.
- OpenStack Homepage. <http://www.openstack.org/>.
- Reid, S., Kilster, H., "Sizing the Cloud", Forrester Research Report, 2011.
- Ristov, S., Gusev, M., & Donevski, A. (2013). OpenStack cloud security vulnerabilities from inside and outside. In *CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDS, and Virtualization*, 101-107.
- Salih, N. K., & Zang, T. (2012). Survey and comparison for Open and closed sources in cloud computing. *arXiv preprint arXiv:1207.5480*.
- Seffraoui, O., Aissaoui, M., & Eleuldi, M. (2012). Comparison of multiple IaaS Cloud platform solutions. *7th WSEAS International Conference on Computer Engineering and Applications, (Milan-CEA 13)*. 978-1000.
- Sempolinski, P., & Thain, D. (2010). A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus. *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 417-426.
- Vaquero, L. M., Roderio-Merino, L., Caceres, J., & Lindner, M. (2009) A break in the clouds: towards a cloud definition, *ACM SIGCOMM Computer Communication Review*, 39(1), 50-55.
- Von Laszewski, G., Diaz, J., Wang, F., & Fox, G. (2012, June). Comparison of multiple cloud frameworks. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, 734-741.
- Voras, I., Orlic, M., & Mihaljevic, B. (2013). The Effects of Maturation of the Cloud Computing Market on Open Source Cloud Computing Infrastructure Projects, *Proceedings of the ITI 2013 35th International Conference on Information Technology Interfaces*. 101-106.
- Wen, X., Gu, G., Li, Q., Gao, Y., & Zhang, X. (2012). Comparison of open-source cloud management platforms: OpenStack and OpenNebula. *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, (Fskd)*, 2457-2461.
- Williams, A. (2014). *The Top Open Source Cloud Projects of 2014* Available: <http://www.linux.com/news/enterprise/cloud-computing/784573-the-top-open>

- source-cloud-projects-of-2014. Last accessed 30th Sept 2014.
- Yadav, S. (2013). Comparative Study on Open Source Software for Cloud Computing Platform: Eucalyptus, Openstack and Opennebula, *International Journal of Engineering and Science*, 3(10), 51-54.
- Zhang, Z., Wu, C., & Cheung, D. W. (2013). A survey on cloud interoperability: taxonomies, standards, and practice. *ACM SIGMETRICS Performance Evaluation Review*, 40(4), 13-22.

CSP Formulation for Scheduling Independent Jobs in Cloud Computing

M'hamed Mataoui¹, Faouzi Sebbak², Kada Beghdad Bey² and Farid Benhammadi²

¹*IS & DB Laboratory, Ecole Militaire Polytechnique, Algiers, Algeria*

²*AI Laboratory, Ecole Militaire Polytechnique, Algiers, Algeria*

mataoui_mhamed@umbb.dz, faouzi.sebbak@gmail.com, {bey_kadda, benhammadif}@yahoo.fr

Keywords: Scheduling Algorithms, Cloud Computing, Job Scheduling, Resource Allocation, CSP Formulation.

Abstract: This paper investigates the use of Constraint Satisfaction Problem formulation to schedule independent jobs in heterogeneous cloud environment. Our formulation provides a basis for computing an optimal Makespan using job and machine reordering heuristics based on Min-min algorithm result. The combination of these heuristics with the weighted constraints allows improving the efficiency of the tree search algorithm to schedule jobs with considerable space search reduction. The proposed CSP model is validated through simulation experiments against clusters of 10 virtual machines. The results demonstrate that our model is able to efficiently allocate resources for jobs with significant performance gains between 18% - 40% compared to the Min-Min heuristic results to optimize the Makespan.

1 INTRODUCTION

Nowadays heterogeneous cloud computing is expanding its services to data-intensive computing on cloud platforms because each job (application) of users runs on a separate virtual machine. In these platforms, the jobs are independent and different from one another and it needs an optimal maximal completion time (Makespan). Hence the Scheduling process in cloud computing systems is useful for several different user needs. Static or dynamic heuristics are proposed for cloud to find an optimal solution to the scheduling. Static heuristics define a schedule at compiled time based on the knowledge of the processors availability and tasks to be executed. Dynamic heuristics, on the other hand, are applied when the tasks arrival time is not beforehand known and therefore the system needs to schedule tasks as they arrive (Barbosa and Moreira, 2009). The scheduling strategy defines the instants when the scheduling algorithm is called to produce a schedule based on forecasting resources performances and independent tasks to be executed. The aim of task scheduler in cloud computing environment is to determine a proper assignment of resources to the tasks of jobs to complete all the jobs received from clients. Large numbers of jobs scheduling heuristics are available for maximizing profit via resources allocation in cloud computing systems (Kuribayashi, 2011; Abirami and Ramanathan, 2012; Gouda et al., 2013; Irugurala and

Chatrapati, 2013).

The challenge that needs to be addressed is how efficiently schedule jobs in cloud computing based on the job completion time's optimization to increase resource utilization. In this paper, we consider the problem of resource allocation in heterogeneous cloud environment. The proposed solution in this work is based on the computing power parameter for resources allocation in cloud environment. Clients in our case are jobs decomposed into various tasks where each task should be assigned to one of the resources, which is best suited for its execution to maximize the profit. The proposed approach uses constraint satisfaction problem formulation to schedule independent jobs in heterogeneous cloud environment. Our formulation provides a basis for computing an optimal Makespan using job and machine reordering heuristics based on Min-min algorithm result. The combination of these heuristics with the weighted constraints allows improving the efficiency of the tree search algorithm to schedule jobs with considerable space search reduction. The proposed CSP model is validated through simulation experiments. Our evaluation shows that the proposed approach can improve efficiency and effectiveness of heterogeneous cloud computing systems with significant performance gains between 18% - 40% compared to the Min-Min heuristic results to optimize the Makespan.

This paper is organized as follows. Related works

about resources allocation strategies in cloud computing environment are introduced in Section 2. Section 3 presents the makespan optimization problem definition. Section 4 describes the proposed solution in resource allocation in heterogeneous cloud computing environments, assuming both task arrive simultaneously and machine available time updated. The simulation results are presented in Section 5 and Section 6 concludes this paper.

2 RELATED WORK

There has been a large amount of work focusing on static scheduling approaches on cloud computing platforms and are currently prevalent in clouds. These approaches use static heuristics which are suitable for known prior time execution of jobs. Yuan et al. (Yuan et al., 2011) propose an intelligent scheduler which can handle heterogeneous resources, and be able to allocate resources according to user needs. The proposed intelligent scheduler shows an improved scheduling algorithm for making efficient resources allocation in cloud. Zhang et al. (Xie et al., 2012) uses the a dynamic constraint programming to solve the problem of virtual cloud resource allocation model. This approach takes into account both users' QoS requirements and the cost of virtual cloud resources. The simulation results show that the proposed approach can efficiently allocate and manage the virtual resources of the cloud platform, and are in agreements with those of (Zhang et al., 2013). Goudarzi and Pedram (Goudarzi and Pedram, 2011), address the Service Level Agreements (SLA)-based resource allocation problem for cloud and a distributed solution for this problem is proposed. The response time of the request based on the different allocation of resources for different servers and the cluster is modeled and used in the profit optimization problem.

In (Santos et al., 2002), the authors propose a mathematical formulation for the resource allocation problem in clusters. The authors describe a method to find the best resource assignment in a cluster in the case that the application has certain resource requirements. Experimental results proved that the proposed method was able to realize best load balancing and reasonable resources utilization. In (Li et al., 2010), an adaptive resource allocation algorithm for the cloud system with preemptable tasks is considered. The proposed algorithms adjust the resource allocation adaptively based on the updated of the actual task executions. A. Kundu et al (Kundu et al., 2010) proposed the memory utilization method in cloud computing environment based on transparency. The

proposed mechanism enables users to access memories depending on the predefined criteria. The resource allocation is made based on the selection criteria which will improve the efficiency of the cloud environment. The memory manager is responsible for allocating memory resources to the clients. The authors introduced a cloud service based memory utilization which is an effective mechanism for allocating memories in cloud computing environment. A scheduling algorithm named as Linear Scheduling for Tasks and Resources (LSTR) is designed in (Abirami and Ramanathan, 2012). This algorithm performs tasks and resources scheduling respectively. The combination of Nimbus and Cumulus services are imported to a server node to establish the IaaS cloud environment. The virtualization technique used with the scheduling algorithm will yield higher resource utilization, and improve the performance of the cloud resources.

Chen et al. (Chen and Tseng, 2012) introduced an Improved Load Balanced algorithm on the ground of Min-Min algorithm to reduce the Makespan and increase resource utilization. Another optimal joint multiple resource allocation method based on the above resource allocation model is presented in (Kuribayashi, 2011). The Author develops a resource allocation model for cloud computing environments, assuming both processing ability and bandwidth are allocated simultaneously to each service request and rented out on an hourly basis. Gouda et al. (Gouda et al., 2013) proposed a new approach that allocates resource with minimum wastage and provides maximum profit. This approach used priority algorithm which decides the allocation sequence for different jobs requested among the different users after considering the priority based on some optimum threshold decided by the cloud owner. An innovative admission control and scheduling algorithms for efficient resource allocation to maximize profit by minimizing cost and improving customer level is introduced in (Irugurala and Chatrapati, 2013). The authors showed that the algorithms work well in a number of scenarios and give the maximum profit among all proposed algorithms by varying all types of QoS parameters.

Silva et al. (Silva et al., 2008) presented a heuristic for optimizing the number of machines that should be allocated for processing an analytical task so that maximum speedup can be achieved within a limited budget. The traffic of web applications is dynamic and random; hence predicting the optimal number of machines for the completion of the client applications in real time and within budget is not a trivial task. Gomathi and Karthikeyan (Krishnasamy and Gomathi, 2013) proposed Hybrid Particle Swarm Op-

timization (HPSO) based scheduling heuristic to balance the load across the entire system while trying to minimize the makespan as well as to utilize the resources in an efficient way in cloud environment. In addition, the results are in agreement with those of (Guo et al., 2012).

In (Katyal and Mishra, 2014), authors have discussed a selective algorithm for resources allocation in cloud environment to end-users on-demand basis. The proposed algorithm is based on min-min and max-min conventional task scheduling algorithms. The selective algorithm uses certain heuristics to select between the two algorithms so that overall makespan of tasks on the machines is minimized. In (Han et al., 2013), the authors presented a QoS guided task scheduling model based on Sufferage-Min-Min heuristic algorithm. An efficiency improvement has been obtained by dividing the tasks and resources into two groups of high QoS level.

3 RESOURCES ALLOCATION STRATEGY

To meet the increasing computational requirements of scientist needs, cloud computing environments are promising platforms which ensure the resources allocation with high quality of service. Therefore the essential challenge of cloud computing scheduler is to provide an optimal scheduling of the jobs based on Makespan optimization to allocate jobs on suitable resources.

The scheduling problem of finding the optimal Makespan is a known NP-complete problem. The scheduling problem that we consider can be stated as follows. Let $J = \{j_1, j_2, \dots, j_n\}$ denote the set of jobs which are independent and let $M = \{m_1, m_2, \dots, m_n\}$ be the set of machines in the cloud computing environment. We assume that each machine can estimate how much time is required to perform each job. In (Minarolli and Freisleben, 2011), Expected Time to Compute (*ETC*) is an $m \times n$ matrix, used to estimate the expected execution time of the job J_j on the machine m_i . In *ETC* matrix, n is the number of jobs and m is the number of machines. One column of the *ETC* matrix contains the estimated execution time for a given job on each machine. Similarly one row of the *ETC* matrix contains the estimated execution time of a given machine for each job. Hence, for a given job J_j and a given machine m_i , ETC_{ij} is the estimated execution time of job J_j on machine m_i . For this problem, we assume take the hypothesis that the computing capacity of each resource and the running time of each job are known.

The Makespan is equal to maximum completion time of all jobs and can be estimated using the following equation (Eq.1):

$$makespan = \max_{i \in \{1, \dots, m\}} \left\{ \sum_{j \in J_{m_i}} ETC_{ij} \right\} \quad (1)$$

where J_{m_i} is the set of the jobs mapped on the machine m_i .

4 OUR STATIC SCHEDULING AS CSP FORMULATION

In this study we start with a presentation to the practical part of our Constraint Satisfaction problem modelling for independent tasks scheduling to improve the Min-Min algorithm result. Thereafter we show that this problem can be described using this formalism using the Min-Min developed job and machine ordering heuristics. These heuristics aim to minimize the total completion time (Makespan).

4.1 SCSP Problem Formulation

The Constraint Satisfaction Problem model is widely used to represent and solve various AI related problems such as Scheduling or Optimization. A SCSP (Scheduling CSP) is defined by a set of jobs, a set of allowed estimated execution time of machines (the domain) is associated to each job and a Global Completion Time constraint (GCT). Solving a SCSP means finding an assignment for each job on one machine that satisfies a GCT constraint.

Based on the Min-min scheduling results, we present a formal model for minimizing the completion time obtained by this algorithm. Using this model, we formulate the static scheduling problem for independent job scheduling in heterogeneous environment as a constraint satisfaction problem (CSP). Our formulation provides a basis for computing an optimal completion time based on several CSP search strategies to refine the Makespan obtained by Min-min algorithm.

The SCSP consists of:

- N jobs J_1, J_2, \dots, J_n , and M machines M_1, M_2, \dots, M_m .
- $D = \{D_1, \dots, D_n\}$ is a set of n domains where each $D_n = \{ETC_{n1}, ETC_{n2}, \dots, ETC_{nm}\}$ is associated with J_n .
- $GCT_m = \sum_j ETC_{jm} < (1 - \alpha) \times C_{max}$ for all $m \in \{1, \dots, M\}$. GCT is the global completion time constraint on the machine m . The parameter $\alpha \in [0, 1]$ represents the improvement of the C_{max} obtained by the Min-Min algorithm. The search

Table 1: An ETC matrix example.

	Jobs						Completion Time
Machines	j_1	j_2	j_3	j_4	j_5	j_6	
m_1	129	109	42	218	113	168	779
m_2	89	73	33	178	83	106	562
m_3	164	141	45	305	148	221	1024

Table 2: Execution results of Min-Min algorithm.

	Jobs						Total
Machines	j_1	j_2	j_3	j_4	j_5	j_6	
m_1	0	0	0	218	113	0	331
m_2	0	73	33	0	0	106	212
m_3	164	0	0	0	0	0	164

space can be reduced by applying this parameter as mentioned in experimental section. If $\alpha = 0$ we use the C_{max} of the Min-Min algorithm to avoid the systematic search assignment which explores systematically the whole search space. So using this GCT constraint, we minimize the maximum completion time for all machines. Note that the C_{max} value is modified in the search process according to the obtained maximal completion time.

As example, consider a simple SCSP of 3 machines m_1 , m_2 and m_3 and 6 jobs J_1, J_2, \dots, J_6 . A scenario of ETC (durations) is given in Table 1. First, Min-Min algorithm determines that the minimum completion time for J_3 will be achieved on m_2 , and makes this assignment. After the first assignment, Min-Min algorithm continues to schedule the jobs J_2, J_5, J_1, J_6 and J_4 as well on m_2, m_1, m_3, m_2, m_1 machines respectively. Consequently, this algorithm finds that the maximum completion time is 331 seconds on m_1 as reported on Table 2. The scheduling, like the following can be expressed as SCSP:

- (06) Jobs J_1, J_2, J_3, J_4, J_5 and J_6 as variables
- (06) Domains $D_1 = \{89, 129, 164\}$, $D_2 = \{73, 109, 141\}$, \dots , and $D_6 = \{106, 168, 221\}$.
- $GCT < (1 - \alpha) \times 331$

Most algorithms for solving CSP_s search systematically through the possible assignments of values to variables. These algorithms seek any solution or all solutions of a CSP. Or they try to prove that no solution exists. In the present work, we have adapted the Forward Checking (FC) algorithm to find all SCSP solutions because the original algorithm aimed simply at finding a feasible solution. So we use FC algorithm with an incremental and modified maximal completion time process. However, the order in which jobs are considered for allocation on machines (instantiation) has a dramatic effect on the time taken to solve our SCSP, relatively to the order in which each job's ETC_s are considered. There are general principles

which are commonly used in selecting the jobs and their ETC values on the machines ordering. The job and machine ordering may be either a static or dynamic ordering according to the current state of the search. In our approach, we use both job and machine ordering heuristics. The job ordering uses the inverse job order obtained by the Min-Min algorithm. The machine ordering heuristic is based on the global completion time of each machine under hypothesis that all jobs are affected to the same machine.

The SCSP formalism allows defining the space of a combinatorial search as a tree. To cut branches in the search tree based on our adapted FC algorithm, we add job and machine ordering heuristics with the incremental maximal completion time as follow:

- The job ordering heuristic uses the order $J_4 \ll J_6 \ll J_1 \ll J_5 \ll J_2 \ll J_3$ which is the inverse order obtained by Min-Min algorithm. (see Fig.1)
- The value (machine) ordering heuristic uses the completion time reported in Table 1. So we obtained $m_2 \ll m_1 \ll m_3$ for J_4 . (see Fig.1)
- Modification of the maximal completion time C_{max} in the GCT constraint in the search tree process.

Applying our search algorithm, more work per node but, presumably, the extra effort will be compensated by the reduction on the number of visited nodes. Fig. 2 shows an example of search space reduction obtained by our FC algorithm. As can be seen the use of the C_{max} modification in the search process offers more reduction of the visited nodes to skip the instantiations with no possible C_{max} improvement. For instance, the new obtained $C_{max} = 260$ allows to cut practically the whole branches in the rest of the search tree. So for this example, our algorithm visited only 117 nodes instead of 1093. For example, our algorithm FC has detected that the partial assignment with $GCT_{m_3} = 305$ is inconsistent with the global constraint ($GCT_{m_3} > C_{max} = 260$), and the algorithm will

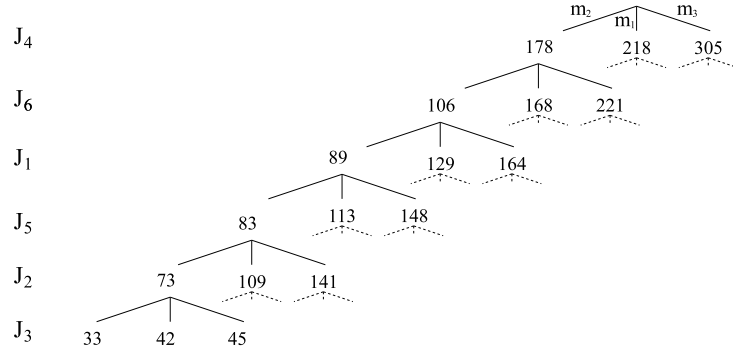
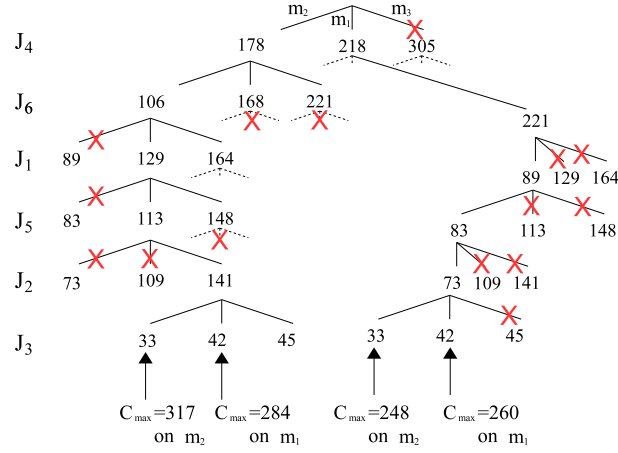
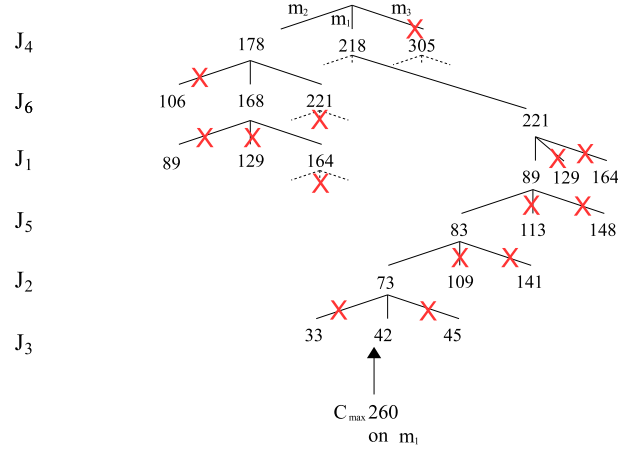


Figure 1: Search tree for the instance: 6 jobs and 3 machines (data from Table 1).

Figure 2: The search tree generated by our algorithm (with $\alpha = 0$).Figure 3: The search of tree generated by our algorithm ($\alpha = 0.2$).

therefore backtrack immediately.

In order to make our search algorithm more efficient, another preprocessing treatment can be added that initially introduces the α parameter value for the global completion time constraint. Hence, if we fix $\alpha = 0.2$, the space of a combinatorial search tree will

be reduced to only 66 visited nodes (Fig. 3).

5 SIMULATION RESULTS

Simulation evaluations of our formalism have been

Table 3: The resulting Makespans compared to Min-Min algorithms for 20 jobs ($\alpha = 0$).

Machines	Time	Total nodes	Explored nodes	C_{max}		Makespan Improvement
				Min-Min Algorithm	Proposed approach	
2	2.00	2097151	33794	1074	973	9%
3	18.22	5.23E+09	434382	744	692	7%
4	58.73	1.47E+12	1718228	684	562	18%
5	80.73	1.19E+14	2747020	571	480	16%

performed. The SCSP resolution uses the inverse of the obtained total order from the Min-Min algorithm to optimize the maximum completion time. The details of the simulation setting are presented in the following. In the literature, many heuristic-based techniques have been proposed for independent job scheduling in heterogeneous environment. The Min-Min heuristic algorithm is the most efficient and used one.

The proposed scheduling algorithm has been applied on simulated data, with 3 different types of ETC matrices up to 3 heterogeneous machines, and up to 20 randomly generated heterogeneous jobs used in (Ibarra and Kim, 1977). These different types of ETC matrices are generated based on the following properties (Inomata et al., 2011):

- **Job Heterogeneity** – represents the amount of variance among the jobs execution times for a given machine. The job heterogeneity is defined as: **J(l)**: Job low and **J(h)**: Job high.
- **Machine Heterogeneity** – represents the variation among the execution times for a given job across all machines. The machine heterogeneity is defined as: **M(l)**: Host low and **M(h)**: Host high.
- **Consistency** – an ETC matrix is said to be consistent (c) whenever a machine m executes all jobs faster than another machine and the inconsistency (i) if the machine m may be faster than another machine for some jobs and slower for others. Partially-consistent (s) matrices are inconsistent matrices that include a consistent sub-matrix of a predefined size and are a combination of consistent and inconsistent matrices (Minarolli and Freisleben, 2011). Instances are labeled as **J(x)M(y)C(z)** as follows: x indicates the job heterogeneity, y represents the machine heterogeneity and z shows the type of consistency.

Table 3 shows the results of the maximum completion times compared to the Min-Min algorithm for scheduling 20 jobs (**J(l)M(l)c(c)**) based on the C_{max} of this algorithm ($\alpha = 0$). The results are based on the computation of job completion times across explored nodes. As can be seen the search space can be reduced

by applying our heuristics where the total explored nodes are widely lower compared to the total nodes of the search space. Moreover, we obtain minimal completion times with the improvements 9%, 7%, 18% and 16% for 2, 3, 4 and 5 machines respectively.

Table 4 reports the speed-ups for the same instance with different values of α parameter. The efficiency observed is very good where the computation time's decrease for all instances compared to the above results from 1 to 5 machines. Also, it is observed that for 6, 7, 8, 9 and 10 machine instances, the execution times are considerably reduced. For example, for the last instance (20 jobs on 10 machines) the execution time 2748 seconds and is reduced to 3.79 seconds with $\alpha = 40.45$. It is interesting to note that the execution time is reduced by 99.85% for 10 machines and we note that the resulting Makespan of Min-Min heuristic is constant starting from 571 because the Min-Min algorithm does not maximize the use of resources. Overall, our results demonstrate that where there is a consistency for low jobs, and having large number of machines, we obtain an efficiency superior to 95% with $\alpha \geq 30\%$.

Finally, Table 5 presents the results of the completion times compared to Min-Min heuristic using maximal values of the parameter α . It is interesting to note that our heuristic outperforms the Min-Min algorithm in all cases (job heterogeneity, machine heterogeneity and consistency) by obtaining the minimal Makespan. Clearly, with the use of an adequate α value, our algorithm performs well in all cases and reduces the tree space search and the execution time to schedule these instances from many hours to a few minutes. A remark has to be made on the computation time irregularity observed for the inconsistent cases for high jobs and low machines. So a major drawback of α values determined by our simulations is that the execution times for the same scheduling problem can be very different from an execution to another for different types of ETC matrix up to 10 heterogeneous machines, and up to 20 randomly generated heterogeneous jobs of the same instance category. An important point to notice is that our approach cannot be considered as very effective for large scheduling problems. To be efficient we remedied the poor performance of

Table 4: The computation time reduction using different values of α parameter for 20 jobs.

Machines	Time (second)	Total nodes	Explored nodes	$\alpha(\%)$	C_{max}	
					Min-Min Algorithm	Proposed approach
2	01.53	2097151	24990	10	1074	973
3	11.50	5.23E+09	266832	8	744	692
4	39.75	1.47E+12	1167804	18	684	562
5	49.73	1.19E+14	1683660	15	571	480
6	51.92	4.39E+15	1945518	23.5	571	434
7	89.72	9.31E+16	3690876	30.29	571	398
8	27.67	1.32E+18	1215384	33.97	571	377
9	10.09	1.37E+19	469152	38.01	571	353
10	03.79	1.11E+20	185420	40.45	571	340

Table 5: Makespan results for all heterogeneity and consistency cases.

Instance	Time (s)	$\alpha(\%)$	C_{max}	
			Min-Min Algorithm	Proposed approach
J(l)M(l)C(c)	03.79	40	571	340
J(l)M(h)C(c)	35.1	32	17271	11627
J(h)M(l)C(c)	1029.4	13	10640	8167
J(h)M(h)C(c)	36.3	37	685303	427236
J(l)M(l)C(i)	1290.4	10	442	397
J(l)M(h)C(i)	62.7	35	22947	14734
J(h)M(l)C(i)	1304.2	38	11495	7109
J(h)M(h)C(i)	1265.9	38	1068180	654517
J(l)M(l)C(s)	7.3	18	341	278
J(l)M(h)C(s)	6.37	16	20025	14734
J(h)M(l)C(s)	3.86	22	10286	6992
J(h)M(h)C(s)	8.82	22	969294	654517

our FC search algorithm by avoiding the recursive tree traversal based on a parallel computation for global completion time constraint. This parallelization uses decomposition methods which distribute the search tree at a particular depth level (Habbas et al., 2005).

6 CONCLUSION

In this work, a static scheduling problem in cloud environment based on a combination of a CSP formulation and Min-Min job ordering heuristic. To improve the Min-Min algorithm result a refinement process uses the incremental maximal completion time as weighted global constraint.

We used various ETC matrixes to investigate efficiency of our approach based on different degrees of job and machine heterogeneities and consistencies. The results indicated that our CSP solver provides to reach an optimal completion time in very short time for small instances compared to Min-Min algorithm. However, our approach cannot considered as very ef-

fective for large instances.

For future work, there are still some aspects for further investigation in our CSP job scheduling algorithm especially for parallel CSP solver using decomposition strategy of the search tree in cloud environment and prediction model for the job completion time distribution that is applicable to making decisions in scheduling.

REFERENCES

- Abirami, S. and Ramanathan, S. (2012). Linear scheduling strategy for resource allocation in cloud environment. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 2(1):9–17.
- Barbosa, J. and Moreira, B. (2009). Dynamic job scheduling on heterogeneous clusters. In *Eighth International Symposium on Parallel and Distributed Computing, 2009. ISPDC'09.*, pages 3–10. IEEE.
- Chen, C.-Y. and Tseng, H.-Y. (2012). An exploration of the optimization of excutive scheduling in the cloud computing. In *Advanced Information Networking and*

- Applications Workshops (WAINA), 2012 26th International Conference on*, pages 1316–1319. IEEE.
- Gouda, K., Radhika, T., and Akshatha, M. (2013). Priority based resource allocation model for cloud computing. *International Journal of Science, Engineering and Technology Research (IJSETR)*, ISSN, 2(1):2278–7798.
- Goudarzi, H. and Pedram, M. (2011). Maximizing profit in cloud computing system via resource allocation. In *31st International Conference on Distributed Computing Systems Workshops (ICDCSW), 2011*, pages 1–6. IEEE.
- Guo, L., Zhao, S., Shen, S., and Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm. *Journal of Networks*, 7(3):547–553.
- Habbas, Z., Krajecki, M., and Singer, D. (2005). Decomposition techniques for parallel resolution of constraint satisfaction problems in shared memory: a comparative study. *International Journal of Computational Science and Engineering*, 1(2):192–206.
- Han, H., Deyui, Q., Zheng, W., and Bin, F. (2013). A qos guided task scheduling model in cloud computing environment. In *Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), 2013*, pages 72–76. IEEE.
- Ibarra, O. H. and Kim, C. E. (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM (JACM)*, 24(2):280–289.
- Inomata, A., Morikawa, T., Ikebe, M., Okamoto, Y., Noguchi, S., Fujikawa, K., Sunahara, H., and Rahman, M. (2011). Proposal and evaluation of a dynamic resource allocation method based on the load of vms on iaas. In *4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2011*, pages 1–6. IEEE.
- Irugurala, S. and Chatrapati, K. S. (2013). Various scheduling algorithms for resource allocation in cloud computing. *The International Journal Of Engineering And Science (IJES)*, 2(5):16–24.
- Katyal, M. and Mishra, A. (2014). Application of selective algorithm for effective resource provisioning in cloud computing environment. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 4(1):1–10.
- Krishnasamy, K. and Gomathi, B. (2013). Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing environment. *Journal of Theoretical & Applied Information Technology*, 55(1):33–38.
- Kundu, A., Banerjee, C., Guha, S. K., Mitra, A., Chakraborty, S., Pal, C., and Roy, R. (2010). Memory utilization in cloud computing using transparency. In *5th International Conference on Computer Sciences and Convergence Information Technology (IC-CIT), 2010*, pages 22–27. IEEE.
- Kuribayashi, S.-i. (2011). Optimal joint multiple resource allocation method for cloud computing environments. *International Journal of Research & Reviews in Computer Science*, 2(1).
- Li, J., Qiu, M., Niu, J.-W., Chen, Y., and Ming, Z. (2010). Adaptive resource allocation for preemptable jobs in cloud systems. In *10th International Conference on Intelligent Systems Design and Applications (ISDA), 2010*, pages 31–36. IEEE.
- Minarolli, D. and Freisleben, B. (2011). Utility-based resource allocation for virtual machines in cloud computing. In *IEEE Symposium on Computers and Communications (ISCC), 2011*, pages 410–417. IEEE.
- Santos, C., Zhu, X., and Crowder, H. (2002). A mathematical optimization approach for resource allocation in large scale data centers. *Technical Report HPL-2002-64, HP Labs, March 2002*.
- Silva, J. N., Veiga, L., and Ferreira, P. (2008). Heuristic for resources allocation on utility computing infrastructures. In *Proceedings of the 6th international workshop on Middleware for grid computing , MGC '08*, pages 1–6. ACM.
- Xie, W.-j., Tang, Z., Yang, L., and LI, R.-f. (2012). Research on the virtual machine placement algorithm in cloud computing based on stochastic programming. *Computer Engineering & Science*, 5(5):95–100.
- Yuan, J.-B., Lee, Y.-C., Wu, W., Young, H.-C., and Liang, K.-H. (2011). Building an intelligent provisioning engine for iaas cloud computing services. In *13th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2011*, pages 1–6. IEEE.
- Zhang, L., Zhuang, Y., and Zhu, W. (2013). Constraint programming based virtual cloud resources allocation model. *International Journal of Hybrid Information Technology*, 6(6):333–344.

Quality of Service Trade-offs between Central Data Centers and Nano Data Centers

Farzaneh Akhbar and Tolga Ovatman

*Department of Computer Engineering, Istanbul Technical University, 34469, Istanbul, Turkey
{akhbar, ovatman}@itu.edu.tr*

Keywords: Nano Data Center, Distributed Cloud Architectures, Quality of Service in Cloud Services.

Abstract: Nano data centers are one of the latest trends in cloud computing aiming towards distributing the computing power of massive data centers among the clients in order to overcome setup and maintenance costs. The distribution process is done over the already present computing elements in client houses such as tv receivers, wireless modems, etc. In this paper we investigate the feasibility of using nano data centers instead of conventional data centers containing accumulated computing power. We try to draw the lines that may affect the decision of nano data center approach considering important parameters in cloud computing such as memory capacity, diversity of user traffic and computing costs. We also investigate the thresholds for these parameters to find out the conditions that make more sense to set up nano data centers as the best replacement of Central Data Centers. We use a CloudSim based simulator, namely CloudAnalyst, for Data Center performance experiments in java. Our results show that 1 gigabyte memory capacity can be seen as a threshold for response time improvement of nano data centers. For nano data centers with more memory capacity there will not be any improvement in response times that leverages the performance cost. We also combine the results of response time and performance cost to provide a similar threshold.

1 INTRODUCTION

Cloud computing is continuously getting more mature over time as the challenges retaining the concept (Qi Zhang et al. 2010) keeps evolving. Challenges like energy efficient computing in cloud environments (Kliazovich et al., 2010) and optimal resource management (Adami et al., 2013) is heavily studied while new concepts like nano data centers (NaDa) are proposed as well through time (Laoutaris et al., 2008) NaDa concept is based on the idea of distributing the computing power of central data centers (CDC) among the customers of the computing service by using relatively less powerful computing devices at customer site. However, CDC should manage requests of different servers, NaDa could consent request of their local users, which are in edge of their networks, for example their home gateways or set-top-boxes.

The basis drive in the development of NaDa is the thriving towards pertaining QoS issues where continuous low latency (Ousterhout et al., 2011) (Zeng and Veeravalli, 2012) is an important parameter to improve. Even more importantly, inducing the cost (Papagianni et al., 2013) to setup

and maintain a large CDC may increase the cost of services (Srajan Kumar and Saxena, 2011).

In this paper we show that distributed data centers as a new version of data centers have advantages in contrast to current CDC in cloud based infrastructures. We use CloudAnalyst simulator (Wickremasinghe et al., 2010) to study the behavior of data centers in both central and distributed topologies. After that we present the tradeoff between data center properties: memory capacity, computing costs and latency under different configurations of data centers to study if they can be used in the decision process of migrating to a distributed NaDa approach. Finally, we take into consideration parameters like the number of user bases: cumulated areas of incoming user traffic and the ratio between CDC's memory capacity and a single node's capacity in the distributed nano network.

The rest of the paper is organized as follows: In Section 2 we present related work on NaDa. In Section 3 we present our simulation environment and in Section 4 we discuss the results obtained from our experiments. In the last section we conclude our study and present future work.

2 RELATED WORK

As cloud computing is a modern technology, recently a lot of studies on different aspects have been done. Since, in these studies data centers play an important role, always attaining big attention. Majority of articles that exist in literature consider only energy consumption of data centers in their studies. In this study we try to find some thresholds to adjust different characteristics of our nano data centers as a replacement for current central ones.

For example, Ning Liu et al suggested an optimization model for energy consumption (Ning Liu et al., 2013). They used greedy algorithm for allocating tasks to different open server and maintained the response time and energy consumption and compared results with the results of random task scheduling in Internet. Their results show greedy task scheduling gives less energy consumption and at the same time less response time. Another research proposed genetic algorithm based approach, namely GABA for virtual machine online reconfiguration in large-scale cloud computing data centers with aim of energy efficiency. In the study by Lin Yuan et al. GABA algorithm is suggested to conserve consumption energy by decreasing the number of physical machine that should be turn on when tasks get arrived in cloud based infrastructures (Haibo Mi et al., 2010).

Moreno and Xu suggested Nano data center again for energy conservation in a way that data centers be located at the edge of the network, like home gateways or set-top-boxes, and cooperate in a peer-to-peer manner (Moreno and Xu, 2011). Valancius et al. applied NaDa in video on demand (VoD) services in cloud computing environment and verified energy utilization in traditional current centric data centers and the new version of data centers, NaDa (Valancius et al., 2009). In this study NaDa utilized ISP-controlled home gateways to provide computing and storage services and adopts a managed peer-to-peer model to form a distributed data center infrastructures. By developing energy consumption pattern with using a large set of empirical VoD access data in traditional and in NaDa data centers they demonstrated, even under the most pessimistic scenarios, NaDa saves at least 20% to 30% of the energy compared to traditional data centers. In the study, it is claimed such kind of energy savings is result of cooling costs avoidance, or reduction of network energy consumptions.

3 SIMULATION ENVIRONMENT

In contrast with traditional data centers that provide services for a large variety of consumers, NaDa supply just local consumers. Since cloud computing developed with the aim to as needed service, so equipment in cloud based infrastructure, should have enough facilities to resolve the requests they take and this may cause data centers and virtual machines over provision. In all, Disadvantages of traditional data centers include majorly three factors (Valancius et al., 2009): 1) over-provisioning, 2) height cost of heat dissipation and 3) increased distance to end-users. In this paper we show how NaDa could overcome these three factors in best. Actually our aim is to find a threshold could guarantee privilege of NaDa in comparison of CDC, while NaDa get maximum proficiency.

We simulate the performance of traditional DCs and Nano ones. We use Cloud Analyst simulator in Java with Intel Core i7-3537U-2.0GHz. During the paper we show response time and performance costs in both traditional and NaDa, and compare them to prove that NaDa works more better than the current CDC and reach the saturation points in which NaDa give their best QoS. We show results as performance cost and response time in charts.

3.1 Cloud Analyst Simulator

Cloud Analyst simulator (CA) have written in java. CA built on Cloudsim, which is a toolkit for modeling and simulation of cloud computing environment and evaluation of resource provisioning algorithms and studying the data center's response time patterns (Buyya et al., 2009) In CA whole worlds considered as 6 different regions. These regions could hosts data centers and user bases. For studying the traditional data centers as CDC, we define a data center in central region and distribute users in all around regions but for NaDa we define one data center for every user. The topology of CDCs in our simulator, model the configuration of CDCs in real world. We put a datacenter in central region of simulator for investigate CDCs performance, because current central data centers receive tasks from lots of consumer and different machines all over their environment, so by placing a CDC in central region we try to force that data center to get task from all user bases in all regions around to act like real central data centers. For modeling NaDas, we try to put users in shortest distance, by placing them in the same region as a NaDa data center is in. NaDas could communicate

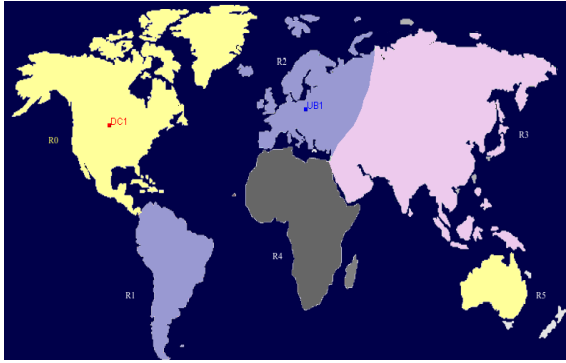


Figure 1: Cloud Analyst Regions.

peer to peer to get their required data instead they send their requests via internet to the servers and waste lots of time and energy during their transfer.

With this simulator we could change different data center's configurations and user bases properties. Figure 1 shows regions distribution in Cloud Analyst simulator. UB1 near R2 shows we have one user in region 2 and DC1 close to R0 shows there is one data center in region 0.

We show the result of our investigation for different configuration of data centers and user bases. By these results we can determine which configuration make NaDa work better. Results are in forms of the response time in millisecond and performance cost. In Table1 there are the characteristics of our CDC. These characteristics are default in our cloud analyst simulator. In fact these values are the average amount of specification we need in our data centers totally (Qi Zhang et al., 2010). Below amount are the average values which guarantee satisfying quality of services in a normal size data center with small task of video demand or such kind of tasks (Pepelnjak, 2014).

Table 1: Properties of CDC.

Central Data Center	
Band Width(Mb/s)	1000000
Memory Capacity(Mb)	204800
Processor Speed(MHz)	10000

4 SIMULATION RESULTS AND EVALUATION

For next step at first we consider bandwidth fluctuation. As shown in Figure 2, we consider bandwidth between 1Mbps and 25Mbps. As we expected, by increasing the bandwidth amount, response time decreases. We can see behavior of the

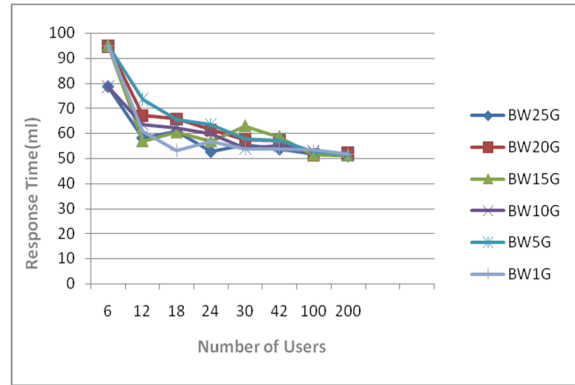


Figure 2: NaDa bandwidth effect on response times.

lines almost are linearly and the same for all sizes of bandwidth. Also we can see when the number of user bases exceeds the 100, response time going to stay constant near the 50 ml second.

Figure 3 demonstrates pattern of response time when we change the amounts of processor speed. The amounts interval is between 2 and 6 GHz. Although response time for different amounts of processor speed at first act differently but as the number of user bases increase it goes to be constant again near 50 ml second. So based on our purpose, network structure or the number of user bases; we could select the bandwidth size. Then we start investigate the memory changes effects on response time behavior. We can see from Figure 4, after we increase NaDa's memory storage capacity more than 1GB, response time show the constant behavior, with 50mls value. It means, with this threshold we will have no concern about response time fluctuations and guarantee the average response time for consumer whose their data centers has this amount of memory capacity in their local data center. In other mean with help of these results we can design local Data center that provides lower response time.

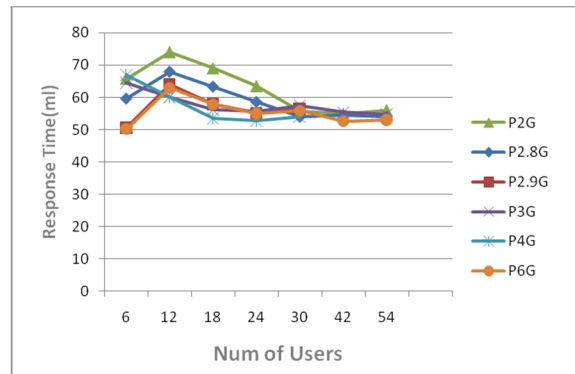


Figure 3: NaDa processor speed effect on response times.

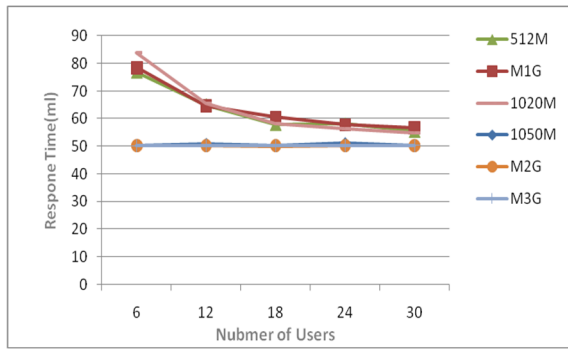


Figure 4: NaDa memory storage effect on response times.

Up to this point we examine bandwidth, processor speed and memory capacity that are the most important properties in data centers. For demonstrating how the behavior of distributed NaDa and central ones are different, we collect the maximum response time value in NaDa and CDC for all three properties that we have examined for different user bases in Table 2. In this way we can see the difference between NaDa and CDC response times. As we can see there is a significant difference between response time values of NaDa and CDC.

Between the characteristic we have checked, memory capacity has some kind of exception; cause after a point response time become a horizontally line for all different number of user bases. These all lead us to consider the memory capacity proportion of traditional data center on our NaDa, till we explore more precisely point of memory storage value which NaDa gain their best performance and could be substitute with traditional data centers in best way. Maybe proportion comparison could help more, because the central datacenter which will be replace with NaDa could have different memory storage amount in different places, depends on network structures or some other parameters. But this time we pay attention to the performance cost values, because we knew how response time fluctuations based on Figure 4.

Figure 5 presents performance cost pattern of NaDa memory capacity on CDC memory capacity.

Table 2: Comparison of CDC and NaDa Response Times.

# of Users	CDC Response Time	NaDa Bandwidth	NaDa Mem.Cap.	NaDa Proc. Speed
		Worst Case Response Time		
6	309.89	94.86	84	65.8
18	308.36	65.5	58.38	69.01
30	308.68	57.72	54.96	55.7
100	307.95	51.52	50.42	52.49
500	309.01	50.98	50.12	50.23

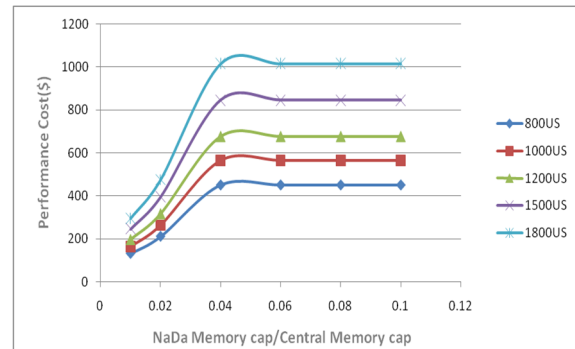


Figure 5: The effect of memory capacity ratio on performance cost with respect to small number of users.

There may be other interpretations for this chart; for example for different number of user bases after almost 0.05 point, we see horizontal constant response time. In deed for one CDC substitution we could consider the NaDa with memory capacity that make this proportion, till NaDa get the best quality and service. Another amassing thing in Figure 5, is lines are close to each other after number of user bases increase from 20, and this shows, as the numbers of consumer of NaDa increase, response time going to be close to each other and it could give us more opportunity to choose the user base number to ascribe local data center as a NaDa. In Figure 5 we consider 640 GB for our central for being sure that the 0.05 points for memory proportion is a correct point.

Data size is another important factor that affect data center behavior drastically. Hence we choose data size for our next investigation. we start to verify the effect of transferring data in different size between user bases and data centers in different CDC and NaDa, so we consider three different CDC with different data sizes and also three different NaDa with the same data size values.

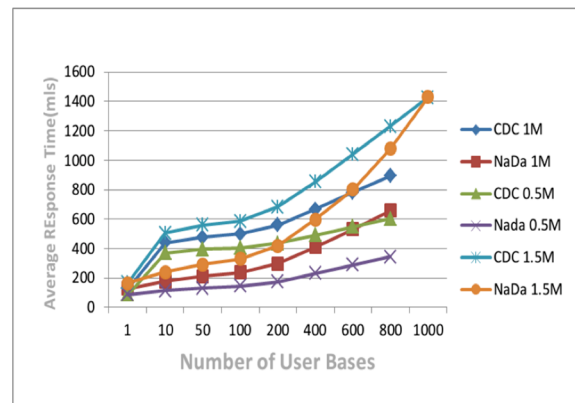


Figure 6: NaDa and CDC Response Times for Different Data Sizes.

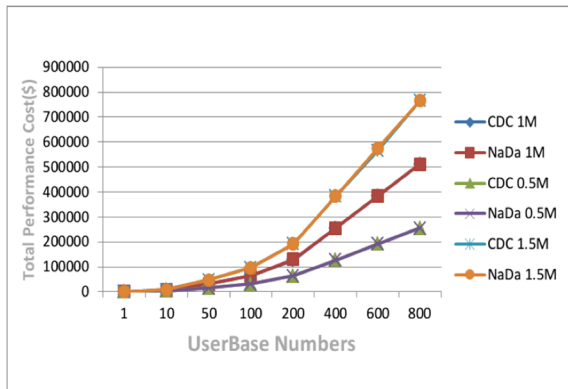


Figure 7: NaDa and CDC Performance Cost for Different Data Sizes.

Figure 6 shows average response time of CDC and NaDa. As we can see in chart, for all three, 0.5, 1 and 1.5 MB data sized packets, response time has less amount for NaDa in comparison with CDC.

NaDa response time has smaller amount than CDC, but we except performance cost be higher for NaDa. So we start check it and surprisingly we realize performance cost of all CDC and NaDa have the same amount for different data packet sizes. Figure 7 demonstrates the results. In all for data packet size properties again we see our NaDa have better proficiency than the CDC. Less response time with equal performance cost really satisfying to substitute NaDa with current traditional central ones.

Finally for having the better perspective of how our research could help construct the NaDa for replacing with CDC, we put the results of performance cost and average response time in one chart together till we could have better comparison. Because user bases number and memory capacity are both of most important properties, at Figure 8 and Figure 9 we have results of them respectively.

As we can see for 1.5MB data packet sizes, response time line and cost line intersect with each other at one point which belong to a user base, so based on our aim, if less response time is important for us or less performance cost, we could replace a CDC with nano one for apparent number of user bases we reach in our charts till we get best proficiency.

In Figure 9, we repeat showing the result of performance cost and response time in one diagram this time vs. the proportion of memory capacity of nano data center on memory capacity of CDC for 50 user bases. This chart fluctuation is less, because as you can see response time always has constant amount near 50 milliseconds.

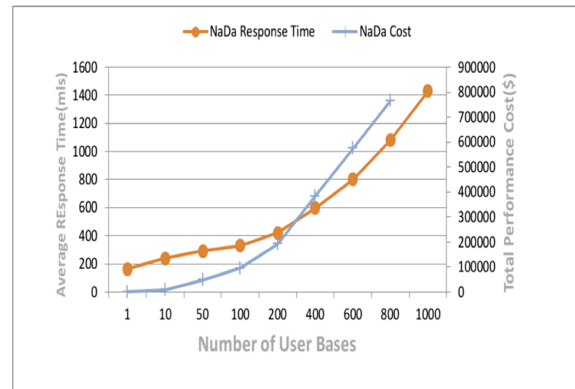


Figure 8: Average Response Performance Cost vs. Number of User Bases.

If we need to have less response time in contrast with performance cost we need to give the amount of memory to NaDa in a way they have more than 0.02 ratio of the previous CDC's memory capacity, because response time line is under the performance cost line after 0.02 ratio. In our studies we examine distributed data centers as nano data center to find the properties that make nano data center as a good replacement for central data centers. We find response time and performance cost for different properties amounts like memory capacity, bandwidth, processor speed and user bases. For example our research shows for more than 1 Gigabyte memory capacity response time will not change. This approach could help people who concern with data centers performance to construct needed data center in a way they could reach maximum quality of services in different cloud architectures.

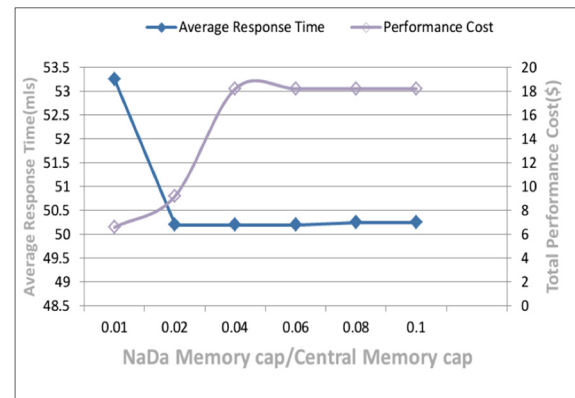


Figure 9: Average Response Time and Performance Cost vs. Memory Capacity Ratio.

5 CONCLUSIONS AND FUTURE WORK

We reach threshold points for different properties of NaDas, include: memory capacity, bandwidth and processor speed. We show how NaDa could have it's maximum quality of services in these points. We also show Our NaDas performance while giving services to different number of user bases. In all of our simulations neighbor NaDas could ask services from each other in peer to peer form. Trying other ways of communication between neighbor data centers could be considered as a next level of performance investigation.

In addition, our studies can be extended by using real cloud based architectures for experiments. The ways of how it could help the industry for more financial profit and improvement could be another charming spark to use this approach. Web application providers could adopt their products based on our new thresholds for NaDa for get better QoS values and in follow reach more profit. This work shows that maybe we should investigate cloud structure more precisely and researchers should look at our work as a spark for more and deeper investigation.

Our studies show that still there are gap in cloud computing structures and shows we could prepare data centers in a way they be more proportional. Our threshold can be used almost in all of the application served over Internet. ISP Provider or who other adjust the data centers characteristics could consider our work to reach the better performance and QoS. The thresholds in this study give hints for adjusting the properties of the NaDa to improve their services by having the minimum response time of task delivery, or less performance cost.

REFERENCES

- Adami, D., Martini, B., Gharbaoui, M., Castoldi, P., Antichi, G., Giordano, S., 2013. *Effective resource control strategies using OpenFlow in cloud data center*. IM, page 568-574. IEEE.
- Buyya, R., Ranjan, R. and Calheiros, R.N., 2009. *Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities*. Proceedings of the 7th High Performance Computing and Simulation Conference HPCS2009, IEEE Computer Society.
- Haibo Mi, Huaimin Wang, Gang Yin, Yangfan Zhou, Dianxi Shi, Lin Yuan, 2010. *Online Self-reconfiguration with Performance Guarantee for Energy-efficient Large-scale Cloud Computing Data Centers*. IEEE SCC, page 514-521.
- Kliazovich, D., Bouvry, P., Audzevich, Y., Khan, S.U., 2010. *GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Center*. GLOBECOM, page 1-5. IEEE.
- Laoutaris, N., Rodriguez, P., Massoulie, L., 2008. *ECHOS: Edge Capacity Hosting Overlays of Nano Data Centers*. Computer Communication Review 38(1):51-54.
- Moreno, I.S., Jie Xu, 2011. *Customer-Aware Resource Overallocation to Improve Energy Efficiency in Real-Time Cloud Computing Data Centers*. SOCA, page 1-8. IEEE.
- Ning Liu, Ziqian Dong, Rojas-Cessa, R., 2013. *Task Scheduling and Server Provisioning for Energy-Efficient Cloud-Computing Data Centers*. ICDCS Workshops, page 226-231. IEEE.
- Ousterhout, J., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G., Rosenblum, M., Rumble, S.M., Stratmann, E., Stutsman R., 2011. *The Case For RAMClouds*. Commun. ACM 54(7):121-130.
- Papagianni, C., Leivadeas, A., Papavassiliou, S., Maglaris, V., Cervello-Pastor, C., Monje, A., 2013. *On the Optimal Allocation of Virtual Resources in Cloud Computing Networks*. IEEE Trans. Computers 62(6):1060-1071.
- Pepelnjak, I., 2014. *Data Center Design Case Studies*. In Space Publication. First edition.
- Qi Zhang, Lu Cheng, Boutaba, R., 2010. *Cloud computing: state-of-the-art and research challenges*. Journal of Internet Services and Applications In Journal of Internet Services and Applications. Vol. 1, No. 1. pp. 7-18.
- Sravan Kumar, R., Saxena, A. R., 2011. *Data Integrity Proofs in Cloud Storage*. COMSNETS, page 1-4. IEEE.
- Valancius, V., Laoutaris, N., Massoulié, L., Diot, C., Rodriguez, P., 2009. *Greening the Internet with Nano Data Centers*. CoNEXT. page 37-48. ACM.
- Wickremasinghe, B., Calheiros, R.N., Buyya, R., 2010. *CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications*. AINA, page 446-452. IEEE Computer.
- Zeng, Z., Veeravalli, B., 2012. *Do More Replicas of Object Data Improve the Performance of Cloud Data Centers*. UCC, page 39-46. IEEE.

Cloud Readiness Assessment of Legacy Application

Flavio Corradini, Francesco De Angelis, Andrea Polini and Samuele Sabbatini

School of Computer Science, University of Camerino, Via del Bastione 1, Camerino, Italy
{name.surname}@unicam.it

Keywords: Cloud Assessment, Cloud Migration, Software Evaluation, Reengineering, Cloud Application Portability, Cloud Governance.

Abstract: Applications and services hosted in the cloud are increasing continuously. Cloud technology offers important perspectives (performance, high availability, elasticity) and it enables new business models. Unfortunately, this new paradigm faces unprecedented requirements not addressed in legacy application (multi-tenancy, scalability, etc.). This leads to complex re-engineering phases in order to migrate existing software into a cloud environment. Before starting a migration, it is important to analyze the cloud compliance of the application, what to expect after the migration and the effort required to fulfill these expectations. This paper assesses a way to extract an index that describes the feasibility of the re-engineering. We test the metric with a real application that needs to be migrated to a private cloud.

1 INTRODUCTION

Migrate legacy application to the cloud is one of the biggest challenges that cloud paradigm has brought (Buyya et al., 2010). Although the concept of utility computing was introduced about fifty years ago (Parkhill, 1966), it began to be a commercial need only in the early 2000s. The fact that this new paradigm is driven by commercial aspects and not from a real scientific study has led to the creation of different definitions (Vaquero et al., 2008) depending on the commercial context. NIST (Mell and Grance, 2011) provides the most used definition of cloud. The cloud is totally revolutionary in software development as foundries have been in the hardware industry (Armbrust et al., 2009). This model is completely detached from the past, but it results in some problems. One of the main being the ability to migrate legacy application developed with previous methodologies into a new environment and making them cloud compliant. This challenge is due to the fact that legacy application have been implemented with previous methods without taking into considerations concepts unknown until the advent of cloud (i.e. elasticity and scalability) (Menychtas et al., 2013). To migrate a legacy application in a cloud environment, it is necessary to update the application to exploit these new capabilities. To do this, it is necessary to evaluate the application to migrate how and where the application is to be evolved. On the other hand, as mentioned previously, the cloud is not a unique concept, and so this as-

essment should also take into account the technology used in the cloud infrastructure. This article presents the definition of a metric to evaluate the compliance of an application respect to a cloud environment. This proposed research is related to the Open City Platform project (OCP project) founded by the Italian Ministry (Ministero dell'Istruzione, dell'Università e della Ricerca) in the Smart Cities and Communities and Social Innovation initiative (OCP, 2014). This project aims to migrate the applications used by some Public Administration in an private cloud infrastructure. In this context, the metric will be based mainly on technological concepts, ignoring the change in business models needed in a migration to a public cloud. The presented metric will be based on the specific request of this environment, and it will also be portable in order to be applied in other contexts. In Section II, the state of the art is analyzed. In Section III, the metric is proposed. Section IV will focus on the results provided by the metric. In Section V the application context is presented. The paper finishes with a section of Conclusions.

2 RELATED WORK

These last years, the issue of cloud migration was faced by researchers and industrials and a quite variety of solutions were presented.

Di Biase (Biase, 2013) proposes a questionnaire

to evaluate both organizational and application migration in order to identify which migration type can be applied. Hosseini et al. (Khajeh-Hosseini et al., 2012) propose a migration tool-kit that involves all decision making in order to evaluate the feasibility of the application. Related to our work the application assessment is a list of question divided in different areas. Vu et al. (Vu and Asal, 2012) proposes a methodology approach is presented in order to establish which step are needed in legacy application evaluation process.

ARTIST (Artist, 2014) and REMICS (Remics, 2014) are two projects very closed to the aim of the research herein. These projects are funded by the European Community, and they focus their aim on migration using Model Driven Engineering (OMG, 2014). Both projects aim to develop different tools of different part of the migration. REMICS ended in the 2013 and it focused the attention on the recovery, migration, validation and supervising processes of the migration itself. However this project did not cover challenges such as elasticity, multi-tenancy and other non-functional properties. ARTIST focuses on migrating legacy software written in Java and C. The project is still open and it tries to support the migration in every aspect. Strictly related to the purpose of this article, ARTIST presents a work (Alonso et al., 2013) strictly related to the purpose of this article, where the pre-migration phase of the project is proposed. The method used to elaborate the maturity of the software is a questionnaire that has to be answered by a person with a good knowledge of technical and businesses aspects.

The evaluation of legacy application is a business used also by big cloud infrastructure player. Company such as Ibm (IBM, 2014), Cisco (CISCO, 2014), VmWare (VmWare, 2014) and other (RedHat, 2014) (Rackspace, 2014) (Amazon, 2014) offer a self-assessment tool or whitepaper to evaluate the advantages to migrate the application in their cloud. However, the problem of these approaches is that they are based on closed proprietary tools that are not widely available; and they are often accompanied by expensive consultancy periods. The advantage of our proposed metric is to create an agile process in order to fill out complex questionnaires readily.

3 EVALUATION CRITERIA

This section will presents the metric to assess if a legacy application is cloud compliant. This metric analyzes a series of questions that are asked to the software engineer. These questions are used to analyze the current state of the application and the status that

should be achieved by migrating to the cloud. For the realization, the following categories were taken into account: (a) Workload, (b) Application Type, (b) Component, (c) Loose Coupling, (d) Distributed application, (e) Security, (f) Multi-Tenancy and (g) Database. Each category was then divided into several sub-categories in order to be able to identify the level of applications cloud compliant relating to specific category.

3.1 Workload

To migrate an application from in-house environment to cloud, it is necessary to take into account the workload. In Cloud computing patterns (Leymann et al., 2013) presents 5 different workloads: (a) Static, (b) Periodic, (c) Once a life, (d) Continuously grow and (e) Elastic. This paragraph goes in details of each type of load will be presented.

An application with *static workload* does not take any advantage to be migrated into cloud. This is due to the elasticity concept. Indeed, a static workload needs the same resources over the time, this means that having the automation in the allocation and deallocation of resources is almost useless. The migration of application with *periodic workload* into cloud will exploit the concept of resources elasticity. On the other hand, this workload is often too easy to be predicted, so it is possible to avoid the cloud by providing the necessary resources to meet the peak load and this would lead to a waste of resources during other periods. *Once a life workload* consists in a static load with rare peak of resources utilizations. This It results to be more advantageous than periodic load due to the fact that the single peak cannot be predicted so if in-house solution is used it would be probable to remain without sufficient resources. *Continuously grow workload* nearly represents the optimal case in which cloud migration will adds many advantages. In this type of load, the necessary resources grow with time and an automation of resource allocation would bring many benefits. In a static environment (in-house), this type of load would result in many problems as there would be either a state of over-sizing of the allocated resources or a lack of resources when the load has exceeded its capacity. This then leads to a waste of money when the available resources are greater than the actual demand, and it lacks of reliability and performance when the required resources are greater than those actually available. The cloud instead, thanks to its elasticity, allows the resources provided to be exactly those needed. For that reason *Elastic workload* is the optimal load for which the migration to the cloud is essentially required.

3.2 Architecture Type

The number of layer by which is divided the application is very important. The layers are a logical division that separate the various application processes and make them independent. Obviously, this type of classification is related to the server part. The applications that are taken into account are the classic: (a) 1-Layer: Monolithic, (b) 2-Layer, (c) 3-Layer (or more), (d) client-server.

1-Layer application has no subdivision, stateless and stateful components are related to each other and then it becomes difficult to carry out policies of scalability. In *2-Layer application*, it is possible to identify a data layer and a Presentation & Logic layer. This division helps the migration phase due to the fact that a division between stateful and stateless components has already been executed. Thus, the top layer can scale without any problem and not having to deal with the data redundancy. Applications with *3 or more layer* have a physical and logical subdivision already well defined. Each layer is able to be independent and satisfy a given operation. In a cloud environment, this subdivision allows a migration faster than the other cases giving the chance to each layer to scale. *Client-Server application* is another common architecture used in legacy application. This type of architecture has different problems when it is migrated into the cloud. In the cloud model the client part of the application has to be converted to work in a cloud infrastructure. However, if the migration affects only the server, this type of application could be seen as one of the previous specification.

3.3 Component

The analyses of components that would be migrated is an important aspect to be considered when a migration in cloud is performed. With the term *component* we consider a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard. Regards this category three different components are take into account herein: (a) Stateful with Strict Consistency, (b) Stateful with Eventual Consistency, (c) Stateless.

Stateful with Strict Consistency components being the most difficult to migrate to cloud. The consistency must be guaranteed in all its replicas, thus the system must be able to keep synchronized all copies. The system needs a lot of work to fulfil this problem and when the number of replicas grows exponentially the performance of the system highly decreases. *Stateful components* can be accessed without having read the

most updated data. It is not possible to use this model on critical components, which are the ones where information must be precise. That is why it is important to make sure data are as up-to-date as possible when read. In case of several instances, the updates are not executed synchronously but asynchronously, thus allows a better response. In a cloud view, this kind of component is able to scale much more easily than the previous one. *Stateless components* have no information that needs to be duplicated or updated in the other instances. This means it does not undergo efficiency loss as the number of replicas increases. In a cloud environment, the resources for components would be 100% exploitable, as they do not need any policy to be implemented and the updating of the data could be both asynchronous and synchronous.

3.4 Database

The database level is probably the most delicate to migrate. In order to achieve a good level of scalability the usage of NoSQL databases is recommended. However most legacy application uses Relational DataBase (RDB). Great might be the effort to migrate data from this databases to NoSQL, both in terms of reengineering the database and migrate the data. This might result in discouraging this migration. Therefore, it is important to consider that components are only stateful at this level and data must remain intact. So, the issues to be considered are remarkable. The following list shows different database types: (a) relational database with stored procedure (SP), (b) RDB without SP, (c) RDB divided by area and (d) NoSQL.

Relational databases with Stored Procedures are the worst cases of migration to cloud. Until the advent of the cloud, using stored procedures was recommended as they were able to speed up and optimize the process. This also allowed to not have large deployment of resources for components that were not DB. With cloud and the theoretical availability of infinite and elastic resources, it is better to demand processing and workload to components able to scale without problems. Consequently, the part of the application more difficult to scale will be weighed down by stored procedures that load static components. As mentioned before, stored procedures are not recommended in cloud environment, due to the fact that they load components difficult to scale. *Without stored procedures*, the database must perform only the necessary operations on the data, without excessively overloading the database. All applications that use *multiple relational databases without stored procedure* apply to another sub-category of Database

classification. The data layer of the application consists of multiple databases where each database has specific competences. In this case, the migration to cloud is better than in the previous cases since the size of the components different types of access are created, and the overhead of an area does not affect the performance of the other. *NoSQL databases* are the best for the migration to cloud. All non-relational database are included in this category. They are very useful because they can scale very easily and therefore the bottleneck that relational databases have is no longer present.

3.5 Loose Coupling

Loose coupling is another feature to be considered when an application is migrated to cloud environment is the component autonomy. A component is considered autonomous when its changes do not affect the other ones or vice-versa. With reference to the application elasticity in the cloud, components with a low level of autonomy make them difficult to divide, and then it becomes difficult to manage the scalability of a single component. Components with high autonomy can be scaled without affecting the other components with which it interacts. Starting from the 7 levels presented in (Krafzig et al., 2005), 5 levels of components autonomy were considered relevant during the migration: (a) Physical, (b) Format, (c) Time, (d) Reference, (e) Platform.

The highest level of coupling is the *Physical* one, in this configuration the migration in the cloud is very difficult to accomplish unless a review of the application is performed. A direct physical connection has a number of limitations. In this context, it is very complicated to undertake policies of scalability is since the increase of instances of a component implies major changes on how it interacts with the other components. *Format coupling* is intended for the components that are interfaced through a common format. The limitations are minor compared to a physical coupling carrying this specific type of applications on cloud has relevant advantages. However, this dependency leads to great limitations to the component. A component that does not have to be synchronized with other components falls into *Time coupling*. The level of autonomy in this case starts to become significant, and the advantages resulting from the migration to cloud become clear. At this level, a component can exchange data with another one even if it is not available or it works at different speeds. In a cloud environment the component is easily scalable, thus achieving the benefits of scalability with little difficulty. A component is autonomous at a refer-

ence level when it does not need to know the address of other components to interact with them. In this case, the autonomy of the component is very high so moving component in a cloud environment does not create problems. With a *platform autonomy*, the component does not have any binding to the others, it can be implemented in any technology, and this would not affect the behavior of components around him. It is the best solution for a migration to cloud, each component is in fact completely independent on the other and it can perform all the operations without interfere with the other components.

3.6 Distributed Application

The migration on cloud of a distributed application can be easier than migrating other applications. Indeed, this means that Loose Coupling and Components concepts have already been taken into account. There are 3 different classification in distributed application: (a) pipe based through message, (b) process based, (c) layer based.

In *pipe based through message* distributed application, the division is made through the data. The components expect a certain input and provide certain output. Often, pipes and filters are used check the data format to ensure that the "chain" between all components works. A *process based* distributed application is an application that focuses on decomposition based on business models. In this kind of applications, it is necessary to have an engine for the management of the processes, which manages each step of the application and ensures a proper work and the order of the components. A *layer based* distributed application decomposes the application into separate logical layers. Each layer is made of application components providing a certain function. Components are restricted to access components of the same layer or one layer below.

3.7 Multi-tenancy

Multi-tenancy is basis concept of cloud environment, that terms indicate the use of a single instance of the software by multiple tenants. The value defined to this category are: (a) multiple instance in separate hardware, (b) hardware in common with dedicated virtual machine for each tenant, (c) shared middle-ware with separated address space and multiple application instances and different db, (d) shared middle-ware with separated address space and multiple application instances and (e) shared middle-ware and one application instance.

In *multiple instance in separate hardware*, the

multi-tenancy is not yet implemented. Each client organization has a dedicated stack from hardware to application level. In *hardware in common with dedicated virtual machine for each tenant*, the various tenants are located in the same physical machine in which VM were created specifically for each tenant. This configuration is not yet possible to be considered as multi-tenancy due to the fact that a large part of the stack is completely dedicated. *Shared middle-ware with separated address space and multiple application instances and different db* manages the multi-tenancy for a large part of the stack. Middle-ware in this case is in common, and only the application part is still running at one instance per tenant. Although middle-ware is shared in this configuration, it is possible to see that the database uses different structures and tables depending on the tenant reference. *Shared middle-ware with separated address space and multiple application instances* is similar to the previous one, the only noticeable difference is in the management of data. In this case, the data of the various tenant reside on the same tables and there are no dedicated tables for each tenant. The rest remains unchanged, with middle-ware shared among multiple tenants and an application instance for each tenant. In *shared middle-ware and one application instance* the entire stack is managed through the multi-tenancy model. The tenant access to the same application instance and so the entire stack is shared. This is the best case for a migration to cloud because it makes the most out of this paradigm.

3.8 Security

Security is another key issue for cloud environment, even if in our context this problem is very mitigated. In our case, we are talking about private cloud where will run own applications. This differentiates our model from the classic scene where public cloud is taken into account. Despite the context fades this issue, it is always important to take security into consideration. In our case the security concerns basically two aspects: permissions and data protection. This is the only category that is divided in other categories due to the different meaning of security. The result of this category is the sum of the value of Authorization and Data Protection.

3.8.1 Authorization

The first aspect of security taken into account concerns the Permissions. Access management is very important in cloud because of the multi-tenancy, that, as explained above, must manage multiple tenants on the same instance. For this reason, the applications

were classified as: (a) application without login, (b) application with simple login, (c) application with login managed by roles

3.8.2 Data Protection

The data protection is the second aspect to be considered in security. The classification in this case concerns the type of data, considering whether they are sensitive or not, and whether they are encrypted or not. This classification is due to the fact that having a shared environment requires special attention to data protection and it is extremely important for the data not to be violated by unauthorized users.

4 MIGRATION ASSESMENT

The objective of the metric is to provide to the user the cloud compliance of an application. To define a clear output, two modes were defined: a numeric output represented by three values, and a graphical output. To perform the assessment, for each category and sub-category we assigned a specific value (Table 1). The values assigned in this article have been set according to our infrastructure. However it can be changed depending on the needs of the cloud service provider that delivers the platform and the questionnaire of the assessment.

In order to calculate the metric, the users fills out a questionnaire of technical questions in order to define the positioning of the application in the various criteria. In addition, the questionnaire presents some questions to determine which would be the goal of the migration. Indeed, to exploit the potential of the cloud, it is not necessary that the application reaches the maximum value in each category to exploit the potential of the cloud. The metric provides values for the current state (equation 3) and the final one (equation 4) for each category. These extracted values are then processed according to the equations 5 and 6. These values represent the percentage of actual (equation 5) and future (equation 6) application compliance respect to the cloud provider (equation 5). These two values are used to extract the percentage of fulfilment of the desired objective (equation 7). This is the most important value of the metric. This value can be used to estimate the effort of the migration in terms of time. This calculation can be made taking into account the costs of the implementation of the software until now, and relate them to the index presented in equation 7. Although this effort is not predictable in a precise way, the index can help in the estimation when it is calculated for more than one

Table 1: Weight of Categories and Sub-Categories.

WORKLOAD	10
- Unknown	0
- Static	2
- Periodic	6
- Once a life	7
- Continuously grow	9
- Elastic	10
LOOSE COUPLING	8
- Physical	0
- Format	2
- Time	4
- Reference	7
- Platform	10
NUMBER OF LAYER	7
- No (1-Layer)	0
- client-server	2
- 2-Layer	5
- 3+-Layer	10
DISTRIBUTED APPLICATION	5
- No	0
- Pipe Based through message	7
- Process Based	9
- Layer Based	10
DATABASE	9
- RDB with SP	0
- RDB without SP	4
- RDB divided by area	7
- NoSQL	10
COMPONENT	8
- Stateful with Strict Consistency	2
- Stateful with Eventual Consistency	6
- Stateless	10
MULTI-TENANCY	9
- Multiple instances in different separate hardware	0
- Hardware in common with VM for each tenant	2
- Shared middleware, separated address space, multiple application instances, different DB	6
- Shared middleware, separated address space, multiple application instances	8
- Shared middleware and one application instance	10
SECURITY	8
- AUTHORIZATION	5
- No Login	0
- Simple Login	2
- Login with specific role	5
- DATA PROTECTION	5
- Non encrypted sensitive data	0
- Non encrypted non sensitive data	2
- Encrypted sensitive data	4
- Encrypted non sensitive data	5

application, the underline implementation technology is comparable, and we know the effort made in other migrations.

$$n = \text{Number of Categories} \quad (1)$$

$$W_i = \text{Weight of the Category } i \quad (2)$$

$$V_{i,x} = \text{Current value of the Category } i \quad (3)$$

$$V_{i,y} = \text{Desidered value of the Category } i \quad (4)$$

$$V_1 = \frac{\sum_{i=1}^n (W_i \cdot V_{i,x})}{\sum_{i=1}^n (W_i \cdot V_{i,max})} \cdot 100 \quad (5)$$

$$V_2 = \frac{\sum_{i=1}^n (W_i \cdot V_{i,y})}{\sum_{i=1}^n (W_i \cdot V_{i,y})} \cdot 100 \quad (6)$$

$$V_3 = V_1 / V_2 \cdot 100 \quad (7)$$

A radar chart was chosen to represent graphically the results. The choice fell on this type of chart because it gives the possibility to have a rough assessment of the status of the application and to understand what are the gaps to be faced. The various categories are represented in each edge of the chart. By means of the questionnaire previously issued, each edge shows two values representing the score obtained by the application in a category. The first value is related to the current state of the application, while the second one refers to the value that should be achieved when migrating to the cloud. As mentioned above, each category has a different score, so before the chart is drawn, all the values are normalized according to the weight of the category. The difference lies in the effort that must be put in before the migration. Moreover, radar chart allows the increasing or decreasing of various edges without changing the meaning of it, that advantage permit to export the assessment in other environments only configuring the metric.

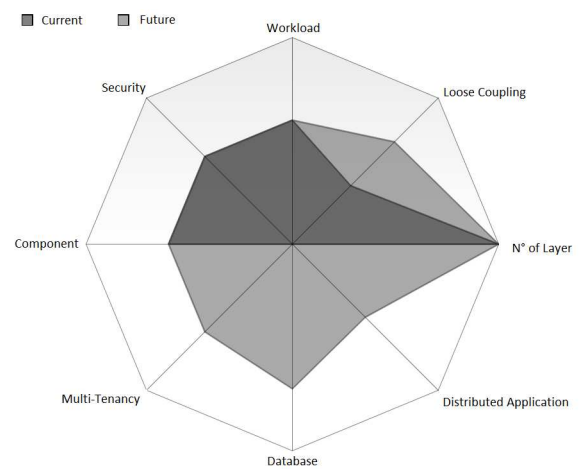


Figure 1: Example of radar results after the evaluation.

4.1 Metric Utilization Example

In this section we introduce an example of the metrics presented above using an application that handles the calculation of vehicle taxes.

4.1.1 Questionnaire

The questionnaire is composed of a set of questions with multiple answers, each answer is needed to extrapolate one of the value presented in the Table 1. The set of questions regarding "database" criteria is presented in Fig. 2 as example. The question regards both current status of the application (i.e. Q1-Q2-Q3) and future status (i.e. Q4). When the questionnaire is completed the system elaborates the answer to select proper value. In example, Fig. 2 shows how the first three questions allow us to extrapolate the current value of the application (i.e. RDB with SP", value =0) whereas the fourth one refers to the value the application would have after the migration (i.e. RDB divided by area, value=7).

DATABASE SECTION

1. Has your application a database?
 - ☒ Yes
 - ☐ No
2. Is the database a relational database or NoSQL database?
 - ☐ NoSQL
 - ☒ Relational
3. Do you use stored procedure?
 - ☒ Yes
 - ☐ No
4. Do you plan the reengineering of the database?
 - ☐ No
 - ☐ Yes, the stored procedure will be extracted from the database
 - ☒ Yes, the database will be divided in different areas
 - ☐ Yes, the database will be migrate to NoSQL environment

Figure 2: Database's questionnaire section.

4.1.2 Results

In this section we present the results of the application tested. The results of the questionnaire are presented in Table 2(the database values are already explained in the previous section). These values are then used to calculate the indexes presented earlier. The results of the elaboration are: $V_1 = 32.25 \%$, $V_2 = 56.125 \%$, $V_3 = 57,461 \%$. As said in the previous paragraph, V_3 is the important value. Indeed, this value can be used in particular application (such as homogeneous applications) to estimate the effort needed for the migration. Considering the effort used until now to achieve V_3 , it is possible to estimate the effort necessary to complete the migration. The figure shown previously (Fig.1) represents the results of this elaboration in a graphical manner. Every edge has a maximum value fixed to 100 ($W_{max} \cdot V_{max}$), the value of the current application (dark grey) is ($W_i \cdot V_{i,x}$) (in the

Table 2: Questionnaire's results of example application.

CATEGORY	CURRENT VALUE	FUTURE VALUE
Workload(10)	Periodic (6)	Periodic (6)
Loose Coupling(8)	Time (4)	Reference (7)
N Layer(7)	3+ -Layer (10)	3+ -Layer (10)
Distributed App.(5)	No (0)	Layer Based (10)
Database(9)	RDB with SP (0)	RDB divided by area (7)
Component(8)	Stateful with eventual consistency (6)	Stateful with eventual consistency (6)
Multi-Tenancy(9)	Multiple instances in different separate hardware (0)	Shared middleware, separated address space, multiple app. instances, diff. DB (6)
Security(8)	Simple Login + Encrypted sensitive data (7)	Simple Login + Encrypted sensitive data (7)

example $9 \cdot 0 = 0$) and the desired value (light grey) is ($W_i \cdot V_{i,y}$) (in the example $9 \cdot 7 = 63$).

5 FUTURE WORK AND CONCLUSIONS

The next step will test the assessment other application involved in OCP project in order to better validate the index. Moreover, the presented metric is used to evaluate software engineering aspects. However, the adoption of cloud computing involves all the aspects of the enterprise. The aim is to integrate this evaluation metric with other metrics that evaluate business and organizational aspects. The integration with other metrics imply the automation of the metric in order to have a migration methodology to automate the entire process (Fig. 3). The idea is to use Model Driven Engineering to extract information directly from the artifacts in order to have as accurate information. In this way the questionnaire to submit will undergo to some changes. The defined weights will also be automated in order to have the assessment made automatically from a data storage in which patterns and services of the cloud infrastructure are described. The crucial aspect of this assessment is the value presented in equation 7 that is used to estimate the effort needed to the migration. The proposed estimation of effort uses generic assumption. The idea is to study in depth how

it would be possible to improve the accuracy of the estimation effort by transforming it to person/month or other valuable metrics. Another output will generate documents describing the weakness of this evaluation. These documents will be used as an input by the modernization process of the application that will perform an evolution of the application even going to insert, where possible, specific patterns of the infrastructure to create optimal Platform Specific Model (PSM). The objective of these implementations is: (a) to have a precise and clear situation through automated processes, (b) to make the metric portable in other contexts.

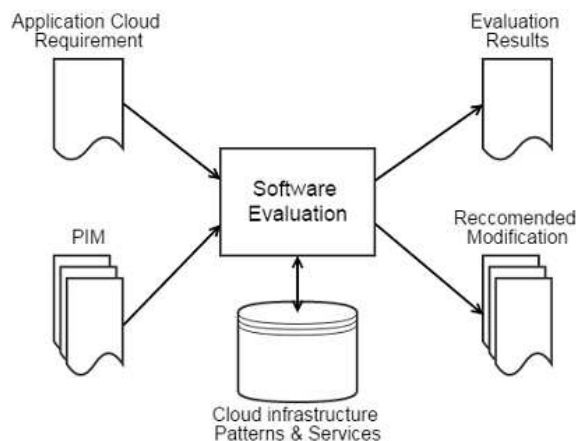


Figure 3: High level architecture diagram of a migration phase related with the metric.

Public institutions suffer the issue of legacy application migration to the cloud. They offers a wide range of heterogeneous services to citizen, company and other institution and they could take advantage from cloud computing handling automatically and dynamically resources. The metric proposed in this paper is intended to helps public institution that wants to create private cloud infrastructure with services installed in it. Considering the context of the OCP project, in which this research is performed, it was decided to consider only the technological part of the migration ignoring aspects of business. At the same time, a platform independent setting was given not to limit the metric only to this case but to apply it in other areas.

ACKNOWLEDGEMENTS

This work has been supported by the Open City Platform project and EUREKA project funded by Regione Marche, UNICAM and APRA SpA.

REFERENCES

- Alonso, J. et al. (2013). Cloud modernization assessment framework: Analyzing the impact of a potential migration to cloud. In *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, pages 64–73. IEEE.
- Amazon (2014). <http://media.amazonwebservices.com/cloudmigration-main.pdf>.
- Armbrust, M. et al. (2009). M.: Above the clouds: A berkeley view of cloud computing.
- Artist (2014). Artist project. <http://www.artist-project.eu/>.
- Biase, F. D. (2013). Legacy to cloud migration: Assessing the cloud readiness of legacy software systems. Master's thesis, University of Applied Sciences Western Switzerland.
- Buyya, R., Ranjan, R., and Calheiros, R. N. (2010). Inter-cloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer.
- CISCO (2014). <http://www.ciscowebsites.com/cloud>.
- IBM (2014). <http://www-01.ibm.com/software/rational/info/cloud-services/self-assessment.htmls>.
- Khajeh-Hosseini, A., Greenwood, D., Smith, J. W., and Sommerville, I. (2012). The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. *Software: Practice and Experience*, 42(4):447–465.
- Krafzig, D., Banke, K., and Slama, D. (2005). *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional.
- Leymann, C. F. F., Retter, R., Schupeck, W., and Arbitter, P. (2013). Cloud computing patterns.
- Mell, P. and Grance, T. (2011). The nist definition of cloud computing.
- Menychtas, A. et al. (2013). Artist methodology and framework: A novel approach for the migration of legacy software on the cloud. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*, pages 424–431. IEEE.
- OCP (2014). Open city platform project. <http://www.opencityplatform.eu/>.
- OMG (2014). Mda. <http://www.omg.org/mda/>.
- Parkhill, D. F. (1966). Challenge of the computer utility.
- Rackspace (2014). <http://www.rackspace.com/cloud/hybrid>.
- RedHat (2014). <https://engage.redhat.com/forms/cloud-readiness-assessment>.
- Remics (2014). Remics project. <http://www.remics.eu/>.
- Vaquero, L. M., Roderio-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55.
- VmWare (2014). <http://www.cloudflightcheck.com/>.
- Vu, Q. H. and Asal, R. (2012). Legacy application migration to the cloud: Practicability and methodology. In *Services (SERVICES), 2012 IEEE Eighth World Congress on*, pages 270–277. IEEE.

Development of an Anything Relationship Management Prototype for the Smart Factory

Jonathan P. Knoblauch, Rebecca Bulander and Thomas Greiner
Pforzheim University of Applied Sciences, Tiefenbronner Str. 65, 75175 Pforzheim, Germany
{jonathan.knoblauch, rebecca.bulander, thomas.greiner}@hs-pforzheim.de

Keywords: Anything Relationship Management, Customer Relationship Management, Industrial Internet, Smart Factory, Internet of Things, Services and Data.

Abstract: The Internet of Things, Services and Data (IoTSD) enters into more and more areas of the business, private and public sector. Typical areas are Smart Factory, Smart Home, Smart Grid, Connected Vehicles and Smart City. The area of Smart Factory (also called industrial internet) will be the most important one in the manufacturing sector. For several years There has been another development in information and communications technology (ICT) observable, called Anything Relationship Management (xRM), trying to systematically manage all stakeholders, physical objects and virtual entities of an enterprise through the use of powerful IT platforms. xRM can be used as a cloud management platform for smart industrial production units combined with stakeholder management. In this paper we use xRM to develop a top-down prototype in the Smart Factory environment. The main objective is to demonstrate how xRM could be used in the future Smart Factory. We therefore recreate the structure of an existing machine for mixing liquids as a service on an xRM cloud platform. Furthermore typical data exchange activities between machine and an xRM cloud platform as well as customers and production machine are simulated. The xRM prototype demonstrates why using an xRM cloud platform is helpful for flexible production environments.

1 INTRODUCTION

The Internet of Things, Services and Data enters into more and more areas of the business, private and public sector. Typical areas are Smart Factory Smart Home, Smart Grid, Connected Vehicles and Smart City or newer areas like Smart Farming. Cisco estimates that by 2020, 50 billion devices and objects will be connected to the internet (Evans, 2011). This progress will lead to a huge “Value at Stake” and bring new innovations which are not imaginable today. Cisco believes that the internet of Things and Services will create \$14.4 trillion in “Value at Stake” from 2013 to 2022 (Bradley, 2013, p. 1). The area of Smart Factory (also called industrial internet) will be the most important one in the manufacturing sector.

For several years there has been another development in information and communications technology observable, called Anything Relationship Management, trying to systematically manage all stakeholders, physical objects and virtual entities of an enterprise through the use of powerful IT platforms. xRM can be used as a management

platform for smart industrial production units combined with stakeholder management.

In this paper we use xRM to develop a top-down prototype in the Smart Factory environment. The main objective is to demonstrate how xRM could be used in the future Smart Factory. We therefore recreate the structure of an existing machine for mixing liquids as a service on an xRM cloud platform. Furthermore typical data exchange activities between machine and xRM cloud platform and customers and machine are simulated. The xRM prototype demonstrates why using an xRM cloud platform is helpful for flexible production environments. The benefit of using xRM is that a flexible cloud platform is provided with the ability to map and create almost any entity, relationship and process in business fields and to manage them systematically. Next to the field of Smart Factory there are other areas, where to use xRM; just to mention some examples: Business Partner Relationship Management, Employee Relationship Management but also Patient Relationship Management at a hospital or Student Relationship Management at a university.

2 FUNDAMENTALS

2.1 Definition of xRM

The term xRM has already been defined several times in a variety of ways. In a previous paper we did an extensive literature review about the term xRM (see Knoblauch and Bulander, 2014). In most definitions xRM is seen as the further stage of Customer Relationship Management (CRM) as well as the realization of the theoretical foundations of relationship management. In addition, xRM includes a technological component (IT system or platform) and a conceptual component (management concept and strategy). In newer definitions xRM is seen as an opportunity to manage objects in the IoTSD (Internet of Things, Services and Data, also called Internet of Everything in an intersectoral way). The following definition covers the main aspects of xRM: “Anything Relationship Management, as an advancement of CRM, is a consistent and holistic concept of Relationship Management between and in-between enterprises, people, physical things and virtual assets. It is based on one or more flexible, modular and scalable IT platforms, which can be focussed on different branches. xRM helps enterprises to capture, coordinate and analyse entities and their relationships as well as processes in the Internet of Everything” (Knoblauch and Bulander, 2014).

2.2 xRM Cloud Platforms

In this section a brief overview about xRM platforms is given. An xRM platform as an extensible foundation provides core functionalities which can be used by multiple modules; each module can interact with each other through defined interfaces (Tiwana et al., 2010). Most xRM platforms use Cloud Computing technologies based on the fundamentals of SOA. xRM platforms provide high-value functions that improve the relationship structures and also any related applications. Business logic and associated user interfaces are based on the defined model. Also the creation of business objects as well as their networking is supported at a conceptual level.

The architecture of an xRM platform can be divided into three layers. The management layer in xRM describes the conceptual approach to manage the n:n interaction, coordination and collaboration between all entities. The middleware layer interconnects people, enterprises, virtual assets and smart objects to create virtual organizations and

cross-company business processes. This layer has to be implemented as a highly efficient and dynamical platform with the capability of interoperability. The back-end layer integrates various systems in a homogeneous system landscape. Besides ERP and SCM systems this layer also has to integrate intelligent physical things like Cyber Physical Systems (CPS) or virtual things such as cloud computing services.

xRM platforms should be highly connected and integrated in multiple ways, also across business operations and domain boundaries. The provisioning of electively networked, cooperating, and human-interactive systems will be an essential component in the adoption of such solutions in the future.

In our research project we did also an extensive xRM market analysis over 26 software suppliers which offer CRM platforms with xRM functionalities. We want just mention some global players out of this analysis to underline the importance of this topic: Microsoft with Dynamics CRM 4.0, Salesforce with their Salesforce Platform or Selligent GmbH with CRM & Interactive-Marketing-Suite (see Knoblauch and Bulander, 2014).

2.3 Advantages through xRM

By using xRM platforms and appropriate management concepts a range of advantages for enterprises are given. In the following the most important ones are summarized.

xRM as platform-as-a-Service (PaaS) provides a cloud-based software development environment for xRM applications (Britsch et al., 2012, p. 86). Such a platform has a flexible and scalable infrastructure as well as the ability of interoperability. Thus, the use of well-defined communication models and communication protocols is necessary (Günthner and Hompel, 2010, p. 79). The software development environment includes components like a repository or has debug functions and the ability to install plug-ins.

Another advantage of xRM is the existence of a configurable framework. Such a framework provides an implementation of important application services like access management or administration functions and a first area of application (typically CRM).

One benefit of xRM is the possibility to build Point-and-Click apps and to customize them easily out of the box. This is one of the core principles that xRM brings along and therefore allows apps to be built fast and easily without the need for extensive implementation skills.

Furthermore, the data models of xRM platforms don't have a fixed schema but a flexible and extensible one. This means, that xRM platforms can hold any data model and can generate or extend the data model without much programming knowledge.

xRM enables the mapping of any kind of entity (stakeholder, virtual asset or physical object) in an application. This allows comprehensive business requirements to be fulfilled on one IT platform. The next level of xRM is integrating smart objects or shared virtual objects through the Internet Things, Services and Data.

Many xRM platforms follow the service orientation paradigm and are built on a service-oriented architecture (SOA). This allows the platform consumer to be served with service orientated capabilities like immediate availability and well-defined behavior of servicers or service composition.

With xRM company-wide and system-wide workflows can be established more easily, since one or more interoperable platforms or well-defined communication standards are in place. This leads to less workflow disruptions and a faster cycle time as well as a more consistent management of workflows and business processes.

Finally any graphical user interface (GUI) of an xRM application can be customized by the user. Depending on user preferences and access restrictions one and the same xRM application can have a completely different GUI.

2.4 xRM, IoTSD and Smart Factory

The Internet of Things, Services and Data is "a dynamic global network infrastructure with self-configuring capabilities based on standards and interoperable communication protocols where physical and virtual "things" [like services] have identities, physical attributes and virtual personalities and use intelligent interfaces and are seamlessly integrated into the information network" (Martinez, 2012, p. 3). In the Internet of Things, Services and Data xRM can be used to define clear relationship structures and to link real and virtual entities dynamically with the right context (Britsch et al., 2012, p. 87). Furthermore, xRM enables the stakeholders of an organization to be connected to enterprise services, virtual assets and physical things. In a nutshell, xRM brings people, things, services and data together on a business platform that allows the systematic management of all relevant business objects.

xRM must also be considered in the context of

the Smart Factory. According to acatech (2013) there are three overarching aspects for implementing the Smart Factory (in Germany called "Industry 4.0"): A horizontal integration through value networks, a holistic integration of engineering across the entire value chain and a vertical integration along networked manufacturing systems (acatech, 2013, pp. 20). By using xRM concepts and platforms it will be possible to build powerful solutions across the vertical integration by reconfiguring whole manufacturing systems over an xRM user interface with regards to business use cases. xRM platforms will help to make connections between multiple companies as well as stakeholders in the horizontal integration of interoperable, to share business context and to extend value networks.

3 REQUIREMENTS ENGINEERING

The definition of objectives are essential in the requirements engineering process of software development. We want to highlight the main objectives of the xRM prototype, demonstrate the added value, describe the software development items and explain restrictions in this chapter.

3.1 Achievable Objectives

Objective 1: The proposed xRM prototype should map and link customers, suppliers, employees, business partners and the industrial production units within the xRM application.

Objective 2a: The proposed xRM prototype should simulate an industrial production unit for mixing liquids as a service.

Objective 2b: The service of mixing liquids is explained as an example for an industrial production unit.

Objective 3: The xRM prototype sensors of the industrial production unit should receive fictive sensor values and save them within the entity.

Objective 4: A saved sales order can also be saved as an XML file that could be sent to an industrial production unit for further processing.

Objective 5: A business process of the sales order via the mixed liquid as a service should be demonstrated.

3.2 Advantages and Added Value of the Prototype

Different advantages and added values are shown through the implementation of the corresponding xRM prototype. The main advantage of the xRM prototype is to demonstrate how xRM can be used in the Smart Factory as a platform for relationship management and value network design. The xRM prototype refers to a use case of process engineering. It is not useful to think in objects (e. g. with RFID-Tags or barcodes) in process engineering, but rather to think in the category of sales orders and their items as well as production services. The xRM prototype elucidates why this is necessary and useful in the Smart Factory. The elements of the industrial production unit of the Smart Factory are called Cyber Physical Production Systems (CPPS), since they are CPS for production.

The tracing of ingredients, products, batches etc. can be carried out with the xRM prototype. Furthermore monitoring and maintenance of the production machine is enabled by receiving important key figure values like temperature, power consumption, number of revolutions or plant utilization in real-time. This in turn enables machines and their components to be checked remotely through specific xRM GUIs. Key figures can also be used to alert if values are out of range. Moreover an improved accounting and reporting by using actual material and production plant consumption is possible. A stronger relationship between customers and customer needs is given by thinking in services. The customer becomes the producer of his product with the xRM prototype. Customers can choose their production services and start their production process over the cloud.

3.3 Development Items

In this section we want to give an overview of the chosen development items for the xRM prototype. There are three basic development items that are described in the following section.

The objective of the first development item is to model an industrial production unit with all existing components on an xRM platform. The industrial production unit is used to mix two different liquids. Furthermore customers, suppliers, employees, business partners and ingredients are also modeled. Therefore, a suitable entity relationship model (ERM) is needed. After the specification of the ERM this can be used to build the logic on an xRM platform via Point-and-Click-Customization. This is

the primary development item of the xRM prototype.

In the second development item the objective is to transfer a sales order with corresponding order items into an XML file that can be sent to an industrial production unit. Besides information about the customer information about the product and the industrial production unit also has to be saved in the created XML file. The idea behind this development item is that information saved as XML can easily be merged in a data exchange format like PLCopen XML. We also want to backtrace the effort that is needed to implement such a function with this development item.

In the last development item we want to implement a simulation of the real-time data exchange of the industrial production unit and the xRM platform. Data out of the machine sensors is saved in the related xRM entity of the xRM platform. Furthermore, each new sensor value is saved within the entity. The current sensor value is always set to the main sensor value attribute. Older sensor values are saved in an XML file.

3.4 Restrictions of the Prototype

The xRM prototype has some restrictions that have to be mentioned. Even though the xRM prototype simulates an existing industrial production unit, there is no direct communication linkage for now. The xRM prototype follows the top-down approach since a working smart industrial production unit is unavailable. Additionally, the data sent to the xRM prototype is randomly generated data. Finally, the generated XML files of the sales order are not sent directly to the industrial production unit, they are saved in a storage location for further processing.

4 IMPLEMENTATION

4.1 xRM Software (SugarCRM)

The xRM platform SugarCRM was chosen to implement the xRM prototype. In its basics SugarCRM is a Customer Relationship Management application that was founded in 2004 as an open source project for Silicon Valley companies. Today there are different product editions of SugarCRM existing. These are a Community Edition (open source), that is licensed under the GNU General Public License and several fee-based software editions. The SugarCRM platform is written in the programming language PHP. SugarCRM has

evolved into an xRM platform over the last years, fulfilling xRM principles like the existence of a configurable framework, a plugin installation module and a Point-and-Click functionality. The reason why SugarCRM was chosen is listed below.

1. The platform in its community edition is open source and therefore free in use.
2. SugarCRM offers a big community and there have been lots of installations.
3. The platform can be installed as on-premise software or used as an on-demand service.
4. SugarCRM allows the software developer to easily access and modify the code as well as the database.
5. The platform offers good functions to build Point-and-Click applications.
6. SugarCRM has a consistently good usability.

The Sugar Community Edition 6.5.17 was the chosen version for developing the xRM prototype.

4.2 Data Model and Business Logic of the Prototype

A corresponding data model is needed first for implementing the structure of a smart machine for mixing liquids. This data model includes the following entities.

In figure 1, the data model of the xRM prototype is visualized. In order to reduce complexity not all attributes of the entities are shown.

Table 1: Entities of the xRM prototype – Part 1.

Entity	Description
Customer	A customer is an external stakeholder who wants to buy a certain amount of mixed liquid.
Sales Order	A customer places a sales order to buy mixed liquid.
Order Item	Each sales order has one or more order items that describe what the customer wants to have mixed, with which mixing ratio, how much and in which volume per filling.
Product/ Ingredient	A mixed liquid is made up out of at least two ingredients/products. Therefore, an order item always includes at least two ingredients.
Supplier	The ingredients are delivered by a supplier.
CPPS-Service	A CPPS-Service is a virtual entity. Thus, mixing liquids is defined as a service.
CPPS-Module	The physical modules of a CPPS-Service are named CPPS-Module. A CPPS-Module is a distinguishable part of a machine that is responsible for a specific task in the production process.

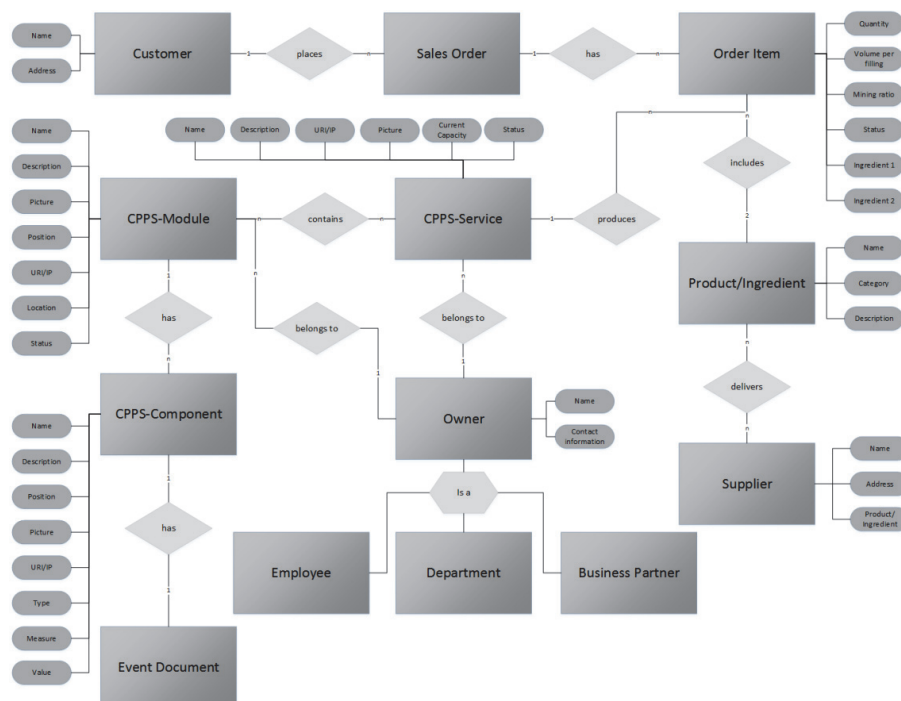


Figure 1: Data model of the xRM prototype.

Table 2: Entities of the xRM prototype – Part 2.

CPPS-Component	A CPPS-Module is built up out of smaller components called CPPS-Components. For example these elements can be seen as the sensors and the actuator of the module.
Event Document	In a CPPS-Component different events are triggered from inside or outside. These events get saved in an entity named Event Document which is linked to the corresponding entity.
Owner	Each CPPS-Service and CPPS-Module has an Owner who is responsible for maintenance, order processing etc. The Owner can be an employee of the organization, a department of the organization or a business partner.

An xRM platform enables an application to be built out of this data model with the Point-and-Click functionality. In SugarCRM this is carried out in the function “Module Builder”. This data model was transferred one-to-one on the xRM platform. A customer places a sales order with order items. Each order item has two ingredients (liquid for mixing) that are delivered by a supplier. In addition, each order item gets a mixing service allocated. This mixing service (CPPS-Service) is responsible for the production of the mixed liquid. A CPPS-Service is constructed out of CPPS-Modules that can be seen as parts of an industrial production unit. A CPPS-Module in turn has various sensors and actuators (CPPS-Components).

CPPS-Components also have an Event-Document that records activities. Furthermore CPPS-Components and CPPS-Services have an Owner who is responsible for production and

predictive maintenance.

The next step is to fill the entities with content. Here we focus on the content of the industrial production unit for liquid mixing. Figure 2 shows a liquid mixing service we have defined with five CPPS-Modules that have sensors and actuators.

4.3 Sales Orders via XML

In future customers will configure and produce their own products in the Smart Factory over the cloud. We simulated what such a business process could look like with the xRM prototype (see section 4.5). The customer selects which liquids he wants to have mixed, defines the mixing ratio and the boxing (bottles, barrels etc.) over the xRM cloud interface. Additionally, the customer also chooses a CPPS-Service which will produce his mixed liquid. After the order item is saved the production can be triggered by sending the data of the item to a CPPS-Service.

We implemented a function on the SugarCRM platform in the main menu of the sales order interface that allows the creation of an XML file that contains all relevant data of an order item. This can be done by defining own PHP-classes that are extensions of the class DOMELEMENT.

The XML file can be sent via HTTP-POST to an existing cloud server of the industrial production unit. Communication via SOAP web services is also feasible. In the following, there is a short example what content in the XML file might look like.

This generated example contains information about an order item that wants to have the liquids cola and soda mixed in five bottles with the mixing ratio 90/10.

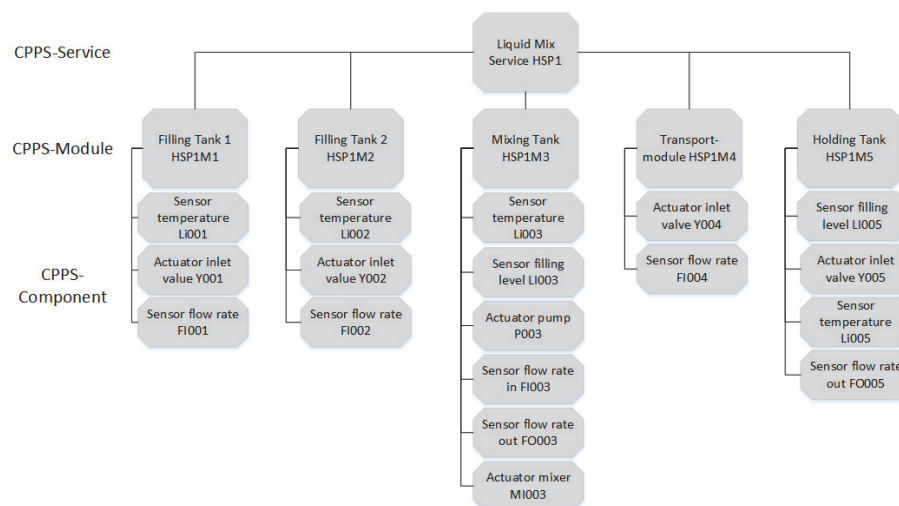


Figure 2: Elements of the liquid mixing service.

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <cppsorder machineid="378909e4-c90f-54bd-fc07-54203c4aa3ee"
3   machineaddress="http://192.168.101.54">
4   <orderitem itemid="5a493058-82ab-816a-ebb0-542bee73ce51">
5     <ratio>b90d10</ratio>
6     <ingredient1>Cola</ingredient1>
7     <ingredient2>Soda</ingredient2>
8     <quantity>5</quantity>
9     <unit>Bottle</unit>
10  </orderitem>
11 </cppsorder>

```

Figure 3: Example of an XML file for data exchange.

4.4 Real-time Data Exchange via Web Services

The exchange of data is necessary to trace the production process of the mixing liquid service. In SugarCRM data can be sent directly to the platform entities (SugarCRM calls them “Modules”) via web services. We implemented a PHP script for the xRM prototype that sends data to all of the CPPS-Components of a CPPS-Service. This PHP script simulates how a mixing liquid production process would actually send data to the SugarCRM platform. Thus, it enables employees to monitor the production process in real-time. Besides monitoring sensor values, finished process steps and status can also be visualized through the platform interface. The following figure illustrates the interface to monitor values of CPPS-Components.

Name	Service Modul	Sensor value
.Actuator inlet value Y001	Filling Tank 1 HSP1M1	58.000
.Actuator inlet value Y002	Filling Tank 2 HSP1M2	0.000
.Actuator pump P003	Mixing Tank HSP1M3	600.000
.Sensor filling level L1003	Mixing Tank HSP1M3	10.000
.Sensor flow rate FI001	Filling Tank 1 HSP1M1	125.000
.Sensor flow rate FI002	Filling Tank 2 HSP1M2	150.000
.Sensor flow rate in FI003	Mixing Tank HSP1M3	0.000
.Sensor flow rate out FI003	Mixing Tank HSP1M3	5.000
.Sensor temperatruue	Filling Tank 2 HSP1M2	88.000
.Sensor temperatruue Li001	Filling Tank 1 HSP1M1	54.000

Figure 4: Interface to monitor production process values.

To evaluate past data every new sensor value can also be saved in the entity Event Document that has a relationship to the corresponding CPPS-Component. This event document is linked to an XML file which saves sensor values and time stamps. Hence, it is possible to evaluate past data by analysis tools. Through this approach we want to emphasize the importance of saving data in the entity it belongs to and not to save unstructured data somewhere else.

4.5 Implemented Business Scenario

To illustrate the big picture of the implemented business scenario a corresponding business process is shown in the Business Process Management Notation (BPMN) in figure 5. Thereby, only the important process steps were depicted.

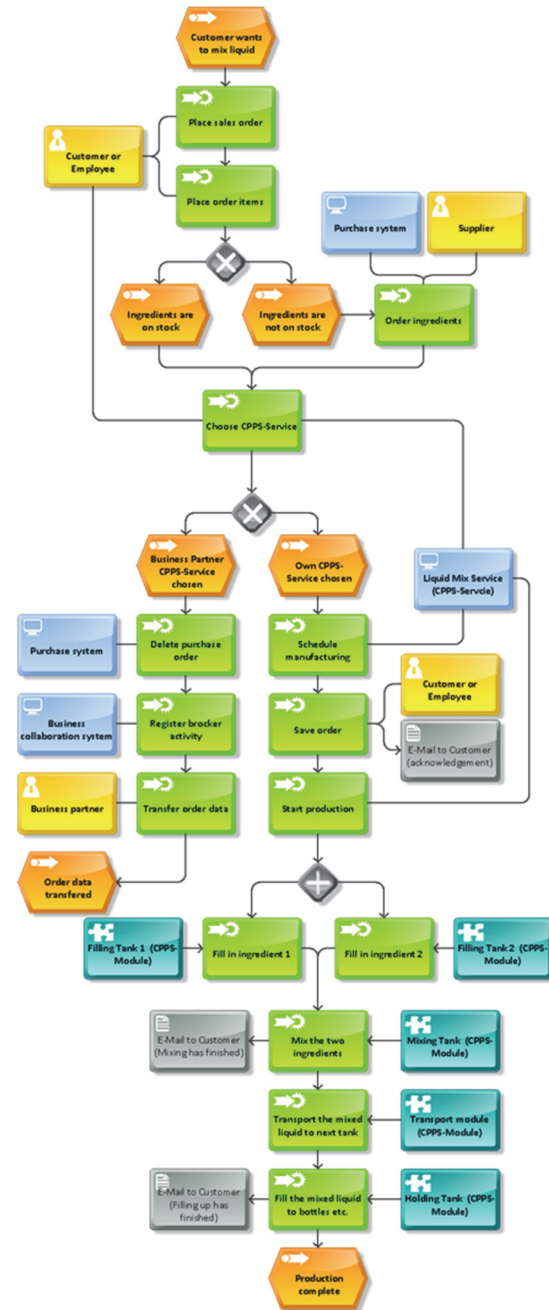


Figure 5: Business process of the xRM prototype.

The business process starts with the event

“Customer wants to mix liquid”. While placing the sales order and the order items it is verified if necessary ingredients are in stock. If not, they are ordered from a supplier through a purchase system. Afterwards a CPPS-Service for mixing liquids as a Service is chosen by the customer or by the employee (if customer doesn’t have the skills or permissions). Depending on the chosen CPPS-Service the business process on the one hand is transferred to a business partner (if its own organization cannot accept the sales order) and on the other hand further processed in its own organization. If the sales order is transferred to a business partner this is registered and will lead to a brokerage for the organization.

The further process steps in its own organization are the scheduling for the manufacturing, the final saving of sales order with an acknowledgment via email and the start of the manufacturing process when possible or desired. The two filling tanks are first filled with the chosen ingredients of the order item in the manufacturing process. The next step is to mix these ingredients in the mixing tank and to send notification to the customer when finished. In the third step the mix liquid gets pumped to the final holding tank. The last step is to fill the mixed liquid out of the holding tank in chosen volumes per filling (bottles etc.) of the order item and to send another email to the customer when completed. After the manufacturing process is finished the mixed liquid is prepared for shipping to the customer.

5 CONCLUSIONS

Organizations are confronted with a rapidly changing environment today in which relationship management is more important than ever. By using xRM concepts and xRM platforms an approach is given to handle the increasing complexity. In future, production services will also be able to automatically allocate their sales orders among their related industrial production units.

We predict that industrial production units will independently configure themselves according to the relations in the xRM. As an example the liquid mixing service we have shown could have a third filling tank added on the xRM platform. This would create a task in the Smart Factory that ends up by adding such a tank to the industrial production unit and connecting it to the other modules. Vice versa, adding a new tank cloud also automatically creates the relationship in xRM.

Regardless of the data flow direction, xRM

platforms and their relationship networks will become more and more important in the future.

ACKNOWLEDGEMENTS

The authors would like to thank the research program of Karl Steinbuch of the MFG Innovation Agency for ICT and Media for the financial support of the research project “Ma-x-RM – Management concept of Anything Relationship Management”.

REFERENCES

- acatech, 2013. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0*. URL http://www.forschungsunion.de/pdf/industrie_4_0_final_report.pdf, (accessed 10/10/2014).
- Bradley, J., Barbier, J., Handle, D., 2013. *Embracing the Internet of Everything To Capture Your Share of \$14.4 Trillion*. URL: http://www.cisco.com/web/about/ac79/docs/innov/IoE_Economy.pdf, (accessed 12/05/2014).
- Britsch, J., Schacht, S., Mädche, A., 2012. *Anything Relationship Management*. In: Business & Information Systems Engineering : BISE (4:2), pp. 85-87.
- Evans, D., 2011. *The Internet of Things. How the Next Evolution of the Internet Is Changing Everything*, Cisco. URL: https://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411F_INA_L.pdf (accessed: 09/03/2014).
- Günter, B., Helm, S., 2006. *Kundenwert. Grundlagen – Innovative Konzepte – Praktische Umsetzungen*. Wiesbaden: Gabler Verlag.
- Günthner, W., Hompel, M. (2010): *Internet der Dinge in der Intralogistik*. Berlin, Heidelberg: Springer-Verlag (VDI-Buch).
- Knoblauch, J. P.; Bulander, R., 2014. *Literature Review and an Analysis of the State of the Market of Anything Relationship Management (xRM) – xRM as an Extension of Customer Relationship Management*. In: Proceedings of 11th International Conference on E-Business and Telecommunications (ICE-B), INSTICC, Wien, Austria, 28–30 August, 2014, pp. 236–244.
- Martinez, C., 2012. *Objective ICT-2013.1.4 - A reliable, smart and secure Inter- net of Things for Smart Cities*. URL: http://www.oko-ist.cz/calls/ncp-infoday_12-06-19/Obj_1_4.pdf (accessed: 10/12/2014).
- Mertic, J., 2009. *The Definitive Guide to SugarCRM - Better Business Applications*. USA: Apress.
- Tiwana, A., Konsynski, B., Bush, A. A. 2010. *Platform evolution: coevolution of platform architecture, governance, and environmental dynamics*. Information Systems Research 21 (4), pp. 675–687.

Redefining the Cloud based on Beneficial Service Characteristics

A New Cloud Taxonomy Leads to Economically Reasonable Semi-cloudification

Bastian Kemmler and Dieter Kranzlmüller

*Leibniz Supercomputing Centre (LRZ), Bavarian Academy of Sciences and Humanities,
Boltzmannstr. 1, 85748 Garching n. Munich, Germany
kemmler@lrz.de*

Keywords: Cloud, Semi-cloud, Service, Cloud Service, Semi-cloud Service, Service Management.

Abstract: Cloud services promise benefits for customers and providers such as scalability, elasticity and reduced investment costs. Unfortunately, many of the promised benefits are not fulfilled by today's cloud offerings and not every service can be cloudified, e.g. if the service's intrinsic structure contains unavoidable time-consuming or manual tasks. A new cloud definition, based on a survey and comparison of existing cloud definitions, but derived from beneficial cloud characteristics, leads to a service-oriented understanding of clouds and provides an extension to the usual cloud service types. The characteristics of the given cloud definition uncover the so-called "MOUSETRAPS" of cloud services. The term "semi-cloudification" for the transformation of services towards a cloudified state presents a solid foundation for further discussions on the topic and enables the improvement of non-cloudifiable services by semi-cloudification. Even services which partly consist of unavoidable time-consuming or manual tasks qualify for semi-cloudification.

1 INTRODUCTION

Cloud computing influences the way today's IT related businesses work. The percentage of companies which utilize clouds continually increases (Wallraf and Pols, 2014, p. 15). According to Gartner *the use of cloud services is growing faster than the overall enterprise IT market. (...) Cloud computing is set to have a considerable impact on business in the future. (...) Three key factors (...) will significantly impact enterprise cloud use in the near to midterm future* (Rivera and Meulen, 2013):

1. Cloud services will be primarily used as a **solution for specific problems** with limited scope.
2. Cloud services will have an **increased business impact**, while the use of **cloud services moves up the service chain** from infrastructure towards business process services.
3. Cloud solutions will lead to a **more diverse solution portfolio**, widely varying in timelines, resource requirements, benefit profiles, business criticality and complexity.

Although the migration challenges for the cloud service customer are debated extensively (Khajeh-Hosseini et al., 2010a, Khajeh-Hosseini et al., 2011, Khajeh-Hosseini et al., 2010b, Kaisler and Money,

2011, Ward et al., 2010, Andrikopoulos et al., 2013, Paulus and Riemann, 2013), there is little discussion on the service provider challenges for transforming legacy services into cloud services.

Besides other aspects, the three key factors indicate four major requirements, which will be highlighted in this work:

1. Cloud taxonomies need to consider the promised benefits of the cloud.
2. Services besides the general well-known classification Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) need to be included in a more generalistic approach to the cloud. Especially, solutions with limited scope, IT-related services of the service chain towards business processes and highly complex services should be considered.
3. Cloud taxonomies should enable and improve discussions about the emerging service provider challenges, which accompany the transformation of a legacy service towards a cloud service.
4. Cloud services in the long run have the potential to substitute the vast majority of legacy IT services (Münzl et al., 2009, p. 22). Therefore, the cloud service definition needs to be based on a solid definition for IT services.

Based on this list, we extract the following:

- Which benefits are expected from a cloud service?
- Which of the existing cloud definitions support these benefits?

Unfortunately, the promised benefits of cloud services are often not fulfilled. A potential reason for the lack of fulfilled benefits could stem from a major design fault in existing cloud definitions. Therefore, we need to ask the question:

- Which cloud service characteristics need to be fulfilled for a service to become a cloud service, thus fulfilling the promised benefits?

In order to evaluate these questions, we also need to investigate the following:

- Which services are cloudifiable, which are not?
- Which cloud service characteristics can be fulfilled for non-cloudifiable services?

Following this introduction, Section 2 discusses cloud benefits, that are promised in relevant literature. Of these benefits several are still unfulfilled in existing cloud implementations. Section 3 compares existing cloud definitions and presents their inherent inability to fulfill the promised benefits, while Section 4 proposes a new cloud taxonomy. By focusing on the limits of cloudification Section 5 introduces the term semi-cloud as a midway point for cloudification and a possible solution for non-cloudifiable IT services. To give an example for the use of the benefit oriented approach Section 6 presents a scenario and a possible solution. Section 7 concludes the paper and provides an outlook on future work.

2 CLOUD SERVICE BENEFITS

2.1 Promised Benefits

Several research papers and practitioner reports promise cloud service benefits, which are listed in this subsection. Particularly we included the research done by Khajeh-Hosseini et. al., who identified *the benefits and risks of using public IaaS clouds [...] by reviewing over 50 academic papers and industry reports* (Khajeh-Hosseini et al., 2011). The (promised) benefits of cloud computing depend on the enterprises perspective and can be distinguished into cloud service customer benefits and cloud service provider benefits.

Promised Cloud Service Customer Benefits

- B1 DISTRIBUTED ACCESS¹: Enhanced mobile and geographically distributed access enables customers (and consumers) to access cloud services almost anywhere. Cloud services provide (end-user-)device mobility, which improves collaboration, and geo-distribution of similar cloud services, thereby increasing existing backup facilities, availability and continuity capabilities.
- B2 INCREASED SCALABILITY¹: Customers benefit from scale-up/-down capabilities of cloud services by adjusting their cloud service usage according to the existing workload. Under-/ and overprovisioning can be reduced.
- B3 ELASTICITY²: *Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an **autonomic** manner, such that at each point in time the available resources match the current demand as closely as possible* (Nikolas Roman Herbst et al., 2013).
- B4 IMPROVED ORGANIZATIONAL FLEXIBILITY AND AGILITY³: Cloud services can be flexibly adjusted to business needs. A higher IT abstraction level leads to (IT-related) business decisions which focus on core business activities and not on IT details. Organizational changes are less restricted by the local IT environment.
- B5 TERM TRANSFORMATION OF INVESTMENT⁴: The use of services transforms capital expenditures (CAPEX) for investments like hardware or software into operational expenditures (OPEX)(Etro, 2011, p. 12). Thereby longer, unmodifiable investment periods, which can last up to several years, are transformed into several much shorter terms, which can be adjusted on short notice. The frequently occurring charge-per-use or pay-as-you-go accounting-models for cloud services greatly encourages this transformation.
- B6 REDUCED TIME TO MARKET¹: Cloud services enable the faster creation of new or redesigned services or products. The timeframe between idea and service/product is reduced.

¹(Khajeh-Hosseini et al., 2011; Carroll et al., 2011; Phaphoom et al., 2012; Wallraf and Pols, 2014)

²(Khajeh-Hosseini et al., 2011)

³(Khajeh-Hosseini et al., 2011; Carroll et al., 2011; Wallraf and Pols, 2014)

⁴(Etro, 2011; Khajeh-Hosseini et al., 2011; Wallraf and Pols, 2014)

- B7 ENTRY BARRIER REDUCTION⁵: Initial resource demand for new or redesigned services can be decreased. The term transformation of investment *reduces the constraints on entry and promotes business creation* (Etro, 2011, p. 12).
- B8 LOWER ADMINISTRATION COSTS¹: The expenditure for administration, maintenance and general operation is reduced for the customer.
- B9 IMPROVED AVAILABILITY AND PERFORMANCE¹: Both can be improved due to the huge resources of cloud service providers. Quality assurance and control is centralized.
- B10 ENHANCED DATA SECURITY⁶: Professional centralized security management, which is transferred to the service provider, is often better than the security management of customers.
- B11 DISASTER RECOVERY⁷: Centralized disaster recovery and geo-distribution of resources is beneficial for cloud customers.

Promised Cloud Service Provider Benefits

- B12 IMPROVED MANAGEMENT EFFICIENCY⁸: Economies of scale lead to more efficient management and provider automation.
- B13 ENERGY EFFICIENCY⁹: The aggregation of system components should also have a positive effect on the level of power consumed on hardware and software (Rajan and Jairath, 2011).¹⁰
- B14 ECONOMIES OF SCALE²: Because computing, storage and other service needs are aggregated at provider level, cloud services make better use of economies of scale.
- B15 IMPROVED CAPACITY MANAGEMENT¹¹: Providers benefit by an improved capacity management through using otherwise idle system components.

2.2 Unfulfilled Benefits

A study based on the contributions of users, developers, consultants, entrepreneurs and researchers to the Cloud Computing Google Group indicates, that the

⁵(Etro, 2011)

⁶(Carroll et al., 2011; Phaphoom et al., 2012; Wallraf and Pols, 2014)

⁷(Khajeh-Hosseini et al., 2011; Phaphoom et al., 2012)

⁸(Byung et al., 2013; Khajeh-Hosseini et al., 2011; Carroll et al., 2011)

⁹(Rajan and Jairath, 2011; Carroll et al., 2011)

¹⁰Although this needs further proof.

¹¹(Khajeh-Hosseini et al., 2011; Carroll et al., 2011)

promised benefits B8, B9 and B10 have not been fully achieved, while the results on the benefits B2 and B5 demonstrate their fulfillment. (Phaphoom et al., 2012)

Moreover, KPMG and BITKOM state, that the initial goals B1, B2 and B4 have been achieved by the majority of enterprise cloud users. The promised benefits B6, B8, B9, B10 and reduced IT investment costs still lack fulfillment. (Wallraf and Pols, 2014, p. 26)

Obviously the benefits B13, B14 and B15 could not be achieved, because if otherwise, IT investment costs could have been reduced. Additionally, the failure to achieve benefit B6 also indicates the failure in achieving benefit B7.

Although benefit B5 can be considered fulfilled, because of the general construction of IT services, there is no real proof that management efficiency could be improved (B12) by providing cloud services. For the benefit of existing cloud service implementation it can be assumed that B11 is fulfilled, due to the existing experience with legacy IT services. Unfortunately, there is no proof for the (un)fulfillment of benefit B3. Even though the argumentation in this section mainly follows a paper which summarizes contributions of the Google Cloud Computing Group (Phaphoom et al., 2012) and a report which describes the situation in Germany (Wallraf and Pols, 2014) it can be assumed that many of the presented unfulfilled benefits are also unfulfilled worldwide. Otherwise, globalisation and the ubiquitous nature of the cloud would ruin any cloud service provider in the US (Google Cloud Computing Group) and in Germany shortly. An overview of the fulfilled and unfulfilled benefits is given in Table 1.

Nevertheless, eight out of ten enterprise users tes-

Table 1: Fulfilled and unfulfilled cloud benefits.

Fulfilled Cloud Benefits	B1	Distributed Access
	B2	Increased Scalability
	B4	Improved Organizational Flexibility and Agility
	B5	Term Transformation of Investment
	B11	Disaster Recovery
Unfulfilled Cloud Benefits	B6	Reduced Time to Market
	B7	Entry Barrier Reduction
	B8	Lower Administration Costs
	B9	Improved Availability and Performance
	B10	Enhanced Data Security
	B13	Energy Efficiency
Undecided	B14	Economies of Scale
	B15	Improved Capacity Management
	B3	Elasticity
	B12	Improved Management Efficiency

tify that the utilization of a cloud is beneficial for them.(Wallraf and Pols, 2014, p. 24)

3 EXISTING CLOUD TAXONOMY

Almost any big player in the business of software and computer services¹² and additionally several companies in the business of technology hardware and equipment¹³ use the term "cloud" as an integral part of their advertising language. Despite widely used for several IT environments, the terms "cloud" and "cloud computing" are still not used with a commonly accepted understanding.¹⁴

3.1 Characteristics of Existing Cloud Definitions

The ongoing research on cloud computing resides on several incomplete and in parts contradictory definitions, like the definitions of NIST(Mell and Grance, 2011), Forrester(Staten, 2008), Gartner(Cearley, 2010), European Commission(EC)(European Commission, 2010, p. 8), BITKOM(Münzl et al., 2009, p. 16) and BSI(Federal Office for Information Security, 2011, p. 13). These definitions describe the cloud with specific characteristics like access, service type, customer type, business benefits and provisioning features, which are listed in detail in the following.

Cloud Access Characteristics¹⁵

AC1 ubiquitous	AC5 over a public network
AC2 on-demand	
AC3 self-service	AC6 over a broad network only
AC4 over a private network	AC7 API

¹²like Microsoft, Google, IBM, Oracle and SAP, listed in the Financial Times global top 500(Financial Times, 2014),which provides *an annual snapshot of the world's largest companies* (Dullforce, 2014)

¹³like Apple, Intel, Cisco and EMC, listed in the Financial Times global top 500(Financial Times, 2014)

¹⁴The iCloud *connects you and your Apple device[,] safely store[s] all your presentations, spreadsheets, PDFs, images, and other kinds of documents* (Apple, 2014), finds your Apple device and protects it against theft; IBM is *building a smarter planet with IBM SmartCloud cloud computing* (IBM, 2014) and Microsoft brings higher education into the cloud [Microsoft Deutschland GmbH, 2012-10-11].

¹⁵Convenient access is also a possible characteristic which is required by the NIST definition. But since convenience is only measurable subjectively and depends very much on the users point of view, it does not qualify as a cloud characteristic.

Cloud Service Type Characteristics

SC1 IT service	SC4 configurable service parameters
SC2 SaaS, PaaS, IaaS	
SC3 XaaS ¹⁶	

Cloud Customer Characteristics

CC1 public	CC3 community
CC2 private	CC4 hybrid

Cloud Business Characteristics

BC1 measured (priced) service
BC2 specified level of quality
BC3 tailored to a market need
BC4 CAPEX \Rightarrow OPEX

Cloud Provisioning Characteristics

PC1 elastic	PC5 pooling
PC2 scalable	PC6 no management effort or provider interaction ¹⁷
PC3 rapid	
PC4 real-time	

3.2 Comparison of Existing Cloud Definitions

By using the extracted definition characteristics to take a closer look at these cloud definitions, the differences become visible (see Table 2).

Especially if the introduced expected benefits (see Section 2) of the cloud are also considered, the deficits of existing cloud definitions are obvious. A mapping between the expected benefits of the cloud and their counterparts (characteristics, see Table 3) applied to the definitions of the NIST, Forrester, Gartner, EC, BITKOM and BSI clearly shows those deficits (see Table 4). Therefore, a clean definition set for the terms *cloud*, *cloud computing*, *cloud service*, *cloud service provider* and *cloudification* is essential for further scientific work regarding the limits of the cloud and non-cloudifiable services.

4 CLOUD TAXONOMY PROPOSAL

As discussed above, cloud taxonomy (see Figure 1) should be build on top of the definition for the generic

¹⁶besides S/P/IaaS

¹⁷Some definitions use terms like *with minimal management effort* or *minimal provider interaction*. From a theoretical approach this is synonymous with *no management effort* or *no provider interaction*, because minimal is nothing if there are no constraints and with unspecified constraints minimal is meaningless.

Table 2: Comparison of cloud definitions.

Dimension	Characteristic	NIST	Forrester	Gartner	EC	BITKOM	BSI
Access	AC1	✓					
	AC2	✓					
	AC4	✓				✗	✓
	AC5	✓	✓	✓			
	AC6	✓					
	AC3	✓	✓				✓
	AC7	✓					
Service Type	SC1	✓	(✓)	✓		✓	✓
	SC2	✓				✓	✓
	SC3	✗				✗	✓
	SC4	✓			✓		✓
Customer	CC1	✓		✗		✓	
	CC2	✓		✗			
	CC3	✓		✗			
	CC4	✓		✗			
Business	BC1	✓			✓	✓	✓
	BC2				✓		
	BC3				✓	✓	✓
	BC4				✓	✓	✓
Provisioning	PC1	✓		✓	✓	(✓)	✓
	PC2	✓		✓			
	PC3	✓					
	PC6	✓					
	PC4	✓			✓		✗
	PC5	✓					

✓: cloud definition contains characteristic
✗: cloud definition contradicts characteristic

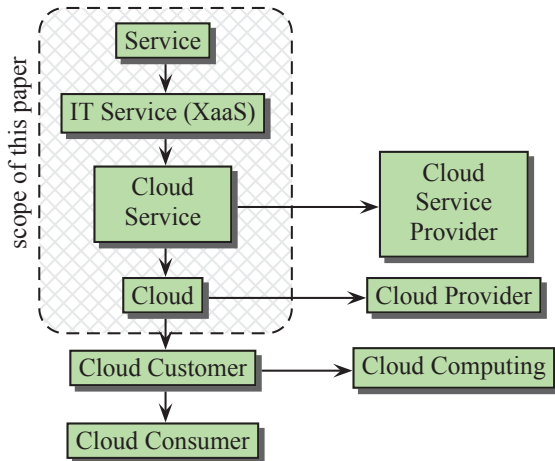


Figure 1: Cloud taxonomy.

service.¹⁸ In this paper only the terms "service", "IT service", "cloud service" and "cloud" will be defined. All other terms listed in Figure 1 can be derived

¹⁸Byung et al.(Byung et al., 2013) also state that the management of some cloud services is more efficient than the management of standard IT services.

Table 3: Benefits targeted by cloud definition characteristics.

	Benefit	Essential Definition Characteristics	Nice-to-have Definition Characteristics
Customer	B1	AC1	AC5
	B2	AC7, PC2, PC3	AC2
	B3	PC1	AC2, AC7, SC4, PC2
	B4	AC2, PC2	CC3, CC4
	B5	BC4	BC1
	B6	AC2, AC3, AC7	BC2, BC3
	B7	AC3, BC4	BC2, AC7, BC3
	B8	AC7, BC3	
	B9	AC1, AC2, BC2, PC1, PC2	AC4, AC5, AC6
	B10	BC2	CC2
	B11	AC1	
Provider	B12	BC3, PC6	
	B13	PC5	
	B14	AC3, AC7, BC1, PC2, PC6	PC5
	B15	BC2, PC2	PC5

Table 4: Benefits targeted by essential cloud definition characteristics.

	Benefit	NIST	Forrester	Gartner	EC	BITKOM	BSI
Customer	B1	✓					
	B2	○		○	✓	(✓)	✓
	B3	✓		✓			
	B4	○		○			
	B5	○	○			✓	○
	B6	○	○			○	
	B7	○	○			○	
	B8	○		○	○	○	✓
	B9	○			○	○	○
	B10	○			✓	○	
	B11	✓					
Provider	B12	○				○	○
	B13	✓					
	B14	○	○	○	○	○	○
	B15	○		○	○	○	○

✓: targeted by at all essential cloud def. characteristics
○: targeted by at least one essential cloud def. char.

easily and are omitted due to space limitations.

4.1 IT Service

According to ITIL (Iqbal and Nieves, 2007, page 16) a service is defined as follows¹⁹:

Definition 1 (Service). A service is a means of delivering value to customers by facilitating outcomes

¹⁹See also ISO/IEC 20000-1:2011; a service is a means of delivering value to customers by facilitating outcomes customers want to achieve (ISO/IEC, 2011, 3.26).

customers want to achieve without the ownership of specific costs and risks.

Definition 2 (IT Service). An IT service is a service, which consists at least partly of IT-related aspects.

Generally spoken, an IT service can be created by a specific business unit utilizing assets of the IT organization to add value to the customers business. These assets A can be segmented into sets like management A^m , organization A^o , processes A^{pr} , knowledge A^{kn} , people A^{ppl} , information A^{inf} , applications A^{app} , infrastructure A^i and financial capital A^{fc} (see (Iqbal and Nieves, 2007, page 39)) and their subsets like $A_i^{app}, i \in \mathbb{N}$, which hold the following:

$$\Sigma^{type} := \{m, o, pr, kn, ppl, inf, app, i, fc\} \quad (1)$$

$$A = \bigcup_{k \in \Sigma^{type}} A^k \quad (2)$$

$$\forall k \in \Sigma^{type} \exists n \in \mathbb{N} : A^k = \bigcup_{i=1}^n A_i^k \quad (3)$$

Definition 3 (XaaS). The value of an IT service is generated by providing X-as-a-Service (XaaS), which (as of today) can be segmented into the following service types:

- Management-as-a-Service (MaaS), like interim management
- Organization-aaS (OaaS), like franchising
- Processes-aaS (PaaS), like franchising
- Knowledge-aaS (KNaaS), which splits into Education-aaS (EDUaaS) and consulting
- People-aaS (PPLaaS), like temporary work
- Information-aaS (INFaaS), like Reuters or Bloomberg
- Application-aaS (APPaaS), which splits into Software-aaS (SaaS) and Platform-aaS (PaaS)
- Infrastructure-aaS (IaaS), which splits into Hardware-aaS (HaaS), virtual-Infrastructure-aaS (vIaaS) and Desktop-aaS (DaaS)
- and Capital-aaS (FCaaS), like a call loan or a credit facility

Mixtures of these service types are also XaaS.

In accordance to the definition of XaaS (Definition 3) an IT service can be characterized by service type and the needed input assets iA^k and output assets oA^k , which can be split up into provider assets PiA^k and PoA^k and customer assets CiA^k and CoA^k .

Furthermore, an IT service can also be created by a specific business unit, additionally utilizing available XaaS of the IT organization to add value to the customers' business. As already discussed, those XaaS S can also be segmented into sets like management S^m , organization S^o , processes S^{pr} , knowledge

S^{kn} , people S^{ppl} , information S^{inf} , applications S^{app} , infrastructure S^i and financial capital S^{fc} and their subsets like $S_i^{app}, i \in \mathbb{N}$

Altogether, these assets and services are the building blocks of a generic IT service. They serve as input parameters to the service function

$$f_s : CiA \times CiS \times PiA \times PiS \rightarrow CoA \times CoS \times PoA \times PoS \quad (4)$$

which creates the service according to a service level agreement (SLA) or an operational level agreement (OLA). These agreements specify the the level, scope and quality of the service in detail.

Definition 4 (Service Level Agreement (SLA)). SLAs are the documents agreed with the customers that specify the level, scope and quality of service to be provided (Lloyd et al., 2007, page 24)).

Definition 5 (Operational Level Agreement (OLA)). OLA are any underpinning agreements necessary to deliver the quality of service agreed within the SLA (Lloyd et al., 2007, page 24)).

Additionally the SLA/OLA may contain specifications about occurring customer obligations. The provided service itself is an entity of the set CoS . Therefore the SLA can be defined as the function

$$f_{SLA} : CiA|_{SLA} \times CiS|_{SLA} \rightarrow CoS|_{SLA} \quad (5)$$

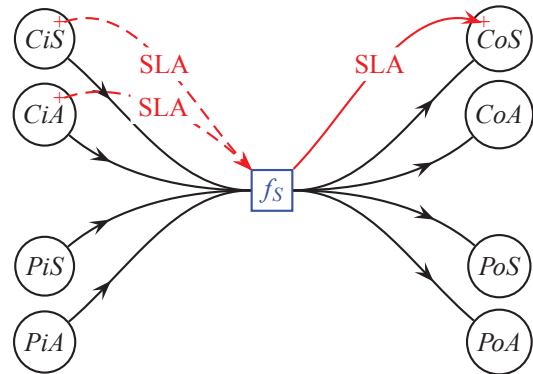


Figure 2: Generic service model.

Finally, the service may contain a configurable service parameter vector p , which leads to

$$f_{S_p} : CiA \times CiS \times PiA \times PiS \rightarrow CoA \times CoS \times PoA \times PoS \quad (6)$$

$$\text{and } f_{SLA_p} : CiA|_{SLA_p} \times CiS|_{SLA_p} \rightarrow CoS|_{SLA_p} \quad (7)$$

4.2 Cloud Service

Based on the discussions above a new cloud definition can be derived, which, on the one hand gives, a solid foundation to fulfill customers/providers expectancy

Table 5: Benefits targeted by cloud definition characteristics without service benefits and characteristics.

	Benefit	Essential Definition Characteristics	Nice-to-have Definition Characteristics
Customer	B1	AC1	AC5
	B2	AC7, PC2, PC3	AC2
	B3	PC1	AC2, AC7, PC2
	B4	AC2, PC2	CC3, CC4
	B6	AC2, AC3, AC7	BC2
	B7	AC3	BC2, AC7
	B8	AC7	
	B9	AC1, AC2, BC2, PC1, PC2	AC4, AC5, AC6
	B10	BC2	CC2
	B11	AC1	
Provider	B12	PC6	
	B13	PC5	
	B14	AC3, AC7, BC1, PC2, PC6	PC5
	B15	BC2, PC2	PC5

and, on the other hand, considers the already listed commonly accepted cloud characteristics. The most important cloud service characteristics can be summarized by the acronym "MOUSETRAPS".

Definition 6 (Cloud Service). *A cloud service is a **M**easured, **O**n-demand, **U**biquitous, **S**calable and **E**lastic IT service **T**ailored to a market need with a specified level of quality (SLA/OLA). It is provisioned, deprovisioned or reconfigured **R**apidly without provider interaction, by using an **A**PI, **P**ooling mechanisms and a **S**elf-service. The service should be accessible over a broad network.*

By utilizing the previously given definition for an IT service (see Definition 1 and 2) benefit B5 and the cloud characteristics SC1, SC2, SC3, SC4, BC3 and BC4 already have been fully included. Thus Table 3 can be significantly reduced as shown in Table 5. Additionally, the inclusion of SC3 already disqualifies the definitions of NIST and BITKOM (see Table 2).

Particularly, the mandatory SLA/OLA of cloud services enable customers to conduct an appropriate risk-analysis for the cloud operation and migration of legacy customer systems. This enables customers to migrate even mission-critical systems into the cloud.

Definition 7 (Cloud). *A cloud is a multitenant electronic marketplace, accessible by a broad network, with strictly defined interfaces for customers and provider(s), where cloud services can be traded²⁰. Access to the cloud can be restricted to a private, community, public or hybrid audience.*

²⁰General linguistic usage: Services do not "leave" the cloud. They "live" in the cloud.

Table 6: Comparison of cloud definitions without service characteristics.

Dimension	Charact.	NIST	Forrester	Gartner	EC	BITKOM	BSI	Def. 7	Def. 10
Access	AC1	✓						✓	✓
	AC2	✓						✓	✓
	AC4	✓				✗		✓	✓
	AC5	✓	✓	✓		✓	✓	✓	✓
	AC6	✓	✓					✓	✓
	AC3	✓	✓				✓	✓	✓
	AC7	✓						✓	✓
Customer	CC1	✓		✓		✓		✓	✓
	CC2	✓						✓	✓
	CC3	✓						✓	✓
	CC4	✓						✓	✓
Business	BC1	✓			✓	✓	✓	✓	✓
	BC2				✓	✓	✓	✓	✓
Provisioning	PC1	✓		✓	✓	✓	✓	✓	✓
	PC2	✓		✓	✓	✓	✓	✓	✓
	PC3	✓						✓	✓
	PC6	✓				✓	✗	✓	✓
	PC4	✓						✓	✓
	PC5	✓						✓	✓

Table 7: Benefits targeted by all essential and nice-to-have cloud definition characteristics.

	Benefit	NIST	Forrester	Gartner	EC	BITKOM	BSI	Def. 7	Def. 10
Customer	B1	✓						✓	✓
	B2							✓	✓
	B3							✓	✓
	B4	✓						✓	✓
	B6							✓	✓
	B7							✓	✓
	B8						✓	✓	✓
	B9							✓	✓
	B10							✓	✓
	B11							✓	✓
	B12							✓	✓
Provider	B13	✓						✓	✓
	B14							✓	✓
	B15							✓	✓

This newly defined cloud definition fulfills all needed characteristics to achieve the promised benefits (see Tables 6 and 7), and therefore gives a better understanding of the cloud. Although the characteristic PC4 could be dropped, because according to Table 6 it is not beneficial to any of the promised benefits, it is included in Definition 6. Obviously, the dependence between AC2 and PC4 cannot be fully denied.

Consequently "cloudification" can be defined as:

Definition 8 (Cloudification). *Cloudification is the transformation by which an IT service becomes a cloud service which is provisioned using a cloud.*

5 LIMITS OF CLOUDIFICATION

Unfortunately, not every IT service can be fully cloudified. IT services like the hardware maintenance of desktop computers or laptops, full service copier lease with ink refill service, education, consulting or other services which at least partly consist of physical work in not properly standardized environments, do not qualify for cloudification, because of their unavoidable manual (and time consuming) work load. Additionally, full service cloudification would be very costly and contradicts benefits which target the overall reduction of costs. By dropping the cloud characteristics AC4, AC5, AC6, PC3 and PC6 an economically more reasonable definition can be found. The new terms, "semi-cloud service", "semi-cloud" and "semi-cloudification" should be used as terms to discuss those "nearly" cloudified services. The most important semi-cloud service characteristics can again be summarized by the acronym "MOUSETrAPS".

Definition 9 (Semi-Cloud Service). *A semi-cloud service is a **M**easured, **O**n-demand, **U**biquitous and **S**calable IT service **T**ailored to a market need with a specified level of quality (SLA/OLA). It is provisioned, deprovisioned or reconfigured with **E**conomically reasonable provider interaction, by using an **A**PI, **P**ooling mechanisms and a **S**elf-service.*

Definition 10 (Semi-Cloud). *A semi-cloud is a multi-tenant electronic marketplace with strictly defined interfaces for customers and provider(s), where semi-cloud services can be traded. Access to the semi-cloud can be restricted to a private, community, public or hybrid audience.*

Definition 11 (Semi-Cloudification). *Semi-cloudification is the transformation by which an IT service becomes a semi-cloud service which is provisioned using a semi-cloud.*

As can be seen in Table 7, only the benefits B1, B2, B9, B12 and B14 are not fully targeted by the semi-cloud as defined in Definition 10. Further investigation shows, that these benefits are only slightly missed:

B1. All essential definition characteristics fulfilled

B2. Only PC3 unfulfilled

B9. All essential definition characteristics fulfilled

B12. Benefit B12 is based on characteristic PC6.²¹

B14. Only PC6 unfulfilled

Altogether, the shortcomings of the proposed definition for the semi-cloud focus around the missing characteristics PC3 and PC6, which were omitted on purpose to encourage the development of designs which balance investment in the semi-cloud systems with economical benefits. Additionally, non-cloudified services which at least partly consist of unavoidable manual tasks, can be semi-cloudified. Therefore the term semi-cloud can be considered a substantial extension to the cloud taxonomy.

6 APPLICATION SCENARIO

To show some real-life aspects of the given approach towards (semi-)cloudification, we introduce the following scenario and observe possible solutions. By the nature of a scenario, the given problems and the solution can not touch the full extend of this work, but give an impression on the change of perspective set by the new cloud taxonomy.

6.1 Scenario

Consider a midsize IT company which started business activities in 1965 by selling copying machines and fulfilling corresponding maintenance contracts. In 1985 the CEO decided to include office printers for personal computers into the product range. Later on the company began selling PCs, servers and network equipment always together with their corresponding maintenance contracts until in 2005 the new CEO introduced a new service oriented strategy. Today the enterprise has around 200 employees, 5 VIP-Customers (three of those since 1965) and around 400 SME customers total. The company mainly provides four services: printing (on-site and off-site), printing and mailing, desktop management and virtual root servers, which are located at a company owned facility. Last year an ISO/IEC 20000 certification was achieved. Most of the employees are booked to capacity and there is little time left for innovative projects. In the last two years the CEO heard about the cloud and its benefits but deferred a project to enable cloud computing for his customers, because of the resources involved with the organizational transformation towards ISO/IEC 20000. Although the budget is tight the CEO expects his new CIO to "cloudify" his company. He privately assumes that the company will especially benefit from the resulting management effi-

²¹which is dropped deliberately

ciency, energy efficiency and the improved capacity management of the cloud. By this, he believes, his company will reach the turning point towards making profit again. With great commitment the CIO initiates the cloud project.

6.2 Problems

In this setting, besides many other, several problems arise which were discussed in this paper.

- 1) Each of the company's employees has a slightly different understanding of the cloud:
 - The customer relationship manager looks forward to the new cloud service, which will be added to the company's service lines and is highly demanded by the company's customers.
 - The infrastructure department head expects benefits by accommodating desperately needed resource demands with an external cloud provider.
 - The hardware maintenance group thinks, that the cloud is a topic for the software guys.
 - The virtualisation specialists state, that cloud computing is already their daily business.
 - Another approach is given by the head of research and development, who proposes to install a software called FreeStack.
- 2) The CEO wants to fully cloudify the company. But the CIO thinks, that many services of the company can not be cloudified. Installing the proposed IaaS software FreeStack for virtual machines provisioning, which also provides interfaces to external cloud providers, could be a solution for the VM service. But who is going to manage the emerging new cloud service, if most of the employees are booked to capacity?
- 3) The CIO's project budget is limited. Therefore she would like to focus on the most important benefits and implement an economically reasonable cloud solution. But which are these and which one of the several available cloud definitions support these benefits?
- 4) Especially the VIP customers might stick to their legacy services and refuse the new cloud services.

6.3 Solution with the Given Approach

By introducing the employees to the benefit oriented cloud taxonomy of Section 4, a commonly accepted understanding of the cloud can be reached (Problem 1). Especially because of the benefit oriented approach, the characteristics in the given definitions can be directly connected to the CEOs desires.

Additionally, a structural analysis and proper decomposition of the company's services according to Subsection 4.1 reveals that the services printing (on-site and off-site), printing and mailing, desktop management can not be cloudified, but semi-cloudified because of their unavoidable manual workload. Moreover, existing services should be transformed into semi-cloud services with at least equal service features one-by-one to free employees from working on legacy services (Problem 2).

An Interview with the CEO based on the given list of benefits in Section 2 reveals, that he is especially interested in gaining the benefits B12, B13 and B15. Semi-Cloudification of the company's services with a special focus on characteristics PC5, BC2 and PC2 will specifically address benefits B13 and B15 (see Table 5), while characteristic PC6 and therefore benefit B12 contradicts the budget constraints (Problem 3). A moderate, cost-conscious automation of service production steps seems to be advisable.

By the transformation of existing services VIP customers can be migrated without loss of the existing service features (Problem 4).

7 SUMMARY AND OUTLOOK

As indicated by our analysis, most of the promised cloud benefits are not addressed by existing cloud definitions. This presents a possible root cause for the unfulfillment of most of those benefits. Based on an analysis of the generic IT service and by identifying beneficial cloud service characteristics, the so-called MOUSETRAPS, a new cloud definition has been derived, which focusses intensely on the desired benefits of the cloud. The terms "semi-cloud", "semi-cloud service" and "semi-cloudification" present a solid base terminology for further discussions of the topic. On the one hand, semi-cloudification can be understood as a midway point towards full cloudification and, on the other hand, many of the existing non-cloudifiable services can be semi-cloudified.

Further studies are needed to analyze the transformation process between the steps non-cloudified, semi-cloudified and cloudified to improve the cloudification of legacy services. Generally cloud research should shift towards a more benefit-oriented approach, especially, when it comes to subjects regarding the service provider and their motives for implementing cloud solutions. Service providers have to realize that the impact of (semi-)cloudification will not only enhance their own services, but also services of their competitors, resulting in a tighter, but better focused service portfolio for each service provider.

REFERENCES

- Andrikopoulos, V., Binz, T., Leymann, F., and Strauch, S. (2013). How to Adapt Applications for the Cloud Environment. *Computing*, 95(6):493–535.
- Apple (2014). iCloud. <http://www.apple.com/icloud/>. 04.12.2014.
- Byung, C. T., Urgaonkar, B., and Sivasubramaniam, A. (2013). Cloudy with a Chance of Cost Savings. *Parallel and Distributed Systems, IEEE Transactions on*, 24(6):1223–1233.
- Carroll, M., van der Merwe, A., and Kotze, P. (2011). Secure Cloud Computing: Benefits, Risks and Controls. In *2011 Information Security for South Africa (ISSA)*, pages 1–9.
- Cearley, D. W. (2010). Cloud Computing: Key Initiative Overview. http://www.gartner.com/it/initiatives/pdf/KeyInitiativeOverview_CloudComputing.pdf. 21.11.2014.
- Dullforce, A.-B. (2014). FT 500 2014 Introduction and Methodology. <http://www.ft.com/intl/cms/s/0/6fdb9d70-fdf5-11e3-bd0e-00144feab7de.html>. 08.07.2014.
- Etro, F. (2011). The Economics of Cloud Computing. *IUP Journal of Managerial Economics*, 9(2):7–22.
- European Commission (25.01.2010). The Future Of Cloud Computing. <http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf>. 02.01.2013.
- Federal Office for Information Security (2011). Security Recommendations for Cloud Computing Providers: Minimum information security requirements. <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Minimum.information/Security.Recommendations.Cloud.Computing.Providers.html>. 2015-01-05.
- Financial Times (2014). Global 500 Companies Ranked by Sector. <http://im.ft-static.com/content/images/70710ff2-fded-11e3-bd0e-00144feab7de.xls>. 08.07.2014.
- IBM (2014). <http://www-07.ibm.com/hk/cloud/cloud-computing/>. 04.12.2014.
- Iqbal, M. and Nieves, M. (2007). *Service Strategy*. ITIL. TSO (The Stationery Office), London, 2 edition.
- ISO/IEC (2011). ISO/IEC 20000-1:2011 - Information Technology - Service Management, Part 1: Service Management system requirements.
- Kaisler, S. and Money, W. (2011). Service Migration in a Cloud Architecture. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–10.
- Khajeh-Hosseini, A., Greenwood, D., and Sommerville, I. (2010a). Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. In *2010 IEEE International Conference on Cloud Computing (CLOUD)*, pages 450–457.
- Khajeh-Hosseini, A., Sommerville, I., Bogaerts, J., and Teregowda, P. (2011). Decision Support Tools for Cloud Migration in the Enterprise. In *2011 IEEE 4th International Conference on Cloud Computing (CLOUD)*, pages 541–548.
- Khajeh-Hosseini, A., Sommerville, I., and Sriram, I. (2010b). Research Challenges for Enterprise Cloud Computing. *CoRR*, abs/1001.3257.
- Lloyd, V., Rudd, C., and Taylor, S. (2007). *ITIL - Service Design: [IT service management practices; ITIL v3 core publications]*. TSO, London.
- Mell, P. and Grance, T. (2011). *The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology: Special Publication 800-145*. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD.
- Microsoft Deutschland GmbH (2012-10-11). Hochschulen in der Wolke: Immer mehr Bildungseinrichtungen setzen auf Cloud Computing. <http://www.microsoft.com/germany/newsroom/pressemitteilung.msp?id=533628>. 05.12.2014.
- Münzl, G., Przywara, B., Reti, M., Schäfer, J., Sondernmann, K., Weber, M., and Wilker, A. (2009). Cloud Computing: Evolution in der Technik, Revolution im Business: BITKOM-Leitfaden. <http://www.bitkom.org/files/documents/BITKOM-Leitfaden-CloudComputing-web.pdf>. 02.01.2013.
- Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner (2013). Elasticity in Cloud Computing: What It Is, and What It Is Not. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, pages 23–27, San Jose, CA. USENIX.
- Paulus, S. and Riemann, U. (2013). An Approach for a Business-Driven Cloud Compliance Analysis Covering Public Sector Process Improvement requirements. *CoRR*, abs/1310.2832.
- Phaphoom, N., Oza, N., Wang, X., and Abrahamsson, P. (2012). Does Cloud Computing Deliver the Promised Benefits for IT Industry? In Männistö, T., editor, *the WICSA/ECSA 2012 Companion Volume*, page 45.
- Rajan, S. and Jairath, A. (2011). Cloud Computing: The Fifth Generation of Computing. In *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, pages 665–667.
- Rivera, J. and Meulen, R. v. d. (2013). Gartner Says the Road to Increased Enterprise Cloud Usage Will Largely Run Through Tactical Business Solutions Addressing Specific Issues. <http://www.gartner.com/newsroom/id/2581315>. 05.12.2014.
- Staten, J. (2008). Is Cloud Computing Ready For The Enterprise? A Client Choice Report: Not Yet, But This Disruptive Innovation Is Maturing Fast. <https://www.forrester.com/Is+Cloud+Computing+Ready+For+The+Enterprise/fulltext/-/E-RES44229?> 05.12.2014.
- Wallraf, B. and Pols, A. (2014). Cloud-Monitor 2014: Cloud-Computing in Deutschland – Status quo und Perspektiven. <http://www.kpmg.com/DE/de/Documents/cloudmonitor-2014-kpmg.pdf>. 05.12.2014.
- Ward, C., Aravamudan, N., Bhattacharya, K., Cheng, K., Filepp, R., Kearney, R., Peterson, B., Shwartz, L., and Young, C. (2010). Workload Migration into Clouds Challenges, Experiences, Opportunities. In *2010 IEEE International Conference on Cloud Computing (CLOUD)*, pages 164–171.

CoMA: Resource Monitoring of Docker Containers

Lara Lorna Jiménez, Miguel Gómez Simón, Olov Schelén, Johan Kristiansson, Kåre Synnes
and Christer Åhlund

Dept. of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden
{lara.lorna.jimenez, miguel.gomez.simon, olov.schelen, johan.kristiansson, kare.synnes, christer.ahlund}@ltu.se

Keywords: Docker, Containers, Containerization, OS-level Virtualization, Operating System Level Virtualization, Virtualization, Resource Monitoring, Cloud Computing, Data Centers, Ganglia, sFlow, Linux, Open-source, Virtual Machines.

Abstract: This research paper presents CoMA, a Container Monitoring Agent, that oversees resource consumption of operating system level virtualization platforms, primarily targeting container-based platforms such as Docker. The core contribution is CoMA, together with a quantitative evaluation verifying the validity of the measurements reported by the agent for three metrics: CPU, memory and block I/O. The proof-of-concept is implemented for Docker-based systems and consists of CoMA, the Ganglia Monitoring System and the Host sFlow agent. This research is in line with the rising trend of container adoption which is due to the resource efficiency and ease of deployment. These characteristics have set containers in a position to topple virtual machines as the reigning virtualization technology in data centers.

1 INTRODUCTION

Traditionally, virtual machines (VMs) have been the underlying infrastructure for cloud computing services (Ye et al., 2010). Virtualization techniques spawned from the need to use resources more efficiently and allow for rapid provisioning. Native virtualization (Type I) (VMware Inc, 2007b) (VMware Inc, 2007a) is the standard type of virtualization behind cloud services. There are several established platforms that offer this type of virtualization such as, Xen hypervisor (Barham et al., 2003), Linux Kernel Virtual Machine (KVM) (Tafa et al., 2011) and VMware (VMware Inc, 2007a). Full-virtualization, Para-virtualization and Hardware-assisted virtualization are different techniques that attempt to enhance the effectiveness of VMs, with varying degrees of success and certain tradeoffs. However, none of these techniques are on par with today's expectations in the cloud computing industry. It has been demonstrated that VMs introduce a significant overhead that does not allow for an optimized use of resources (Xu et al., 2014). The unfulfilled potential for improvement of VMs is where OS-level virtualization comes in.

OS-level virtualization has become popular in recent years by virtue of its resource efficiency. This light-weight type of virtualization executes processes quasi-natively (Felter et al., 2014), (Xavier et al.,

2013). On top of a shared Linux kernel, several of what are generally referred to as “containers” run a series of processes in different user spaces (Elena Reshetova, 2014). In layman's terms, OS-level virtualization generates virtualized instances of kernel resources, whereas hypervisors virtualize the hardware. Moreover, containers run directly on top of an operating system, whereas VMs need to run their OS on top of a hypervisor which creates a performance overhead (Xu et al., 2014). The downside of containers is that they must execute a similar OS to the one that is hosting the containers. There are various implementations of OS-level virtualization, with differences in isolation, security, flexibility, structure and implemented functionalities. Each one of these solutions is oriented towards different use cases. For example, *chroot()* *jails* (Elena Reshetova, 2014) are used to sandbox applications and Linux containers (LXC) ¹ are used to create application containers.

In March 2013, the Docker platform was released as an open-source project based on LXC and a year later, the environment was moved from LXC to libcontainer ². Docker is based on the principles of containerization, allowing for an easy deployment of applications within software containers as a result of its innovative and unique architecture (Felter et al.,

¹<https://linuxcontainers.org/>

²<http://www.infoq.com/news/2013/03/Docker>

2014). Docker implements certain features that were missing from OS-level virtualization. It bundles the application and all its dependencies into a single object, which can then be executed in another Docker-enabled machine. This assures an identical execution environment regardless of the underlying hardware or OS. The creation of applications in Docker is firmly rooted in the concept of versioning (Docker Inc, 2014b). Modifications of an application are committed as deltas (Docker Inc, 2014c), which allows roll backs to be supported and the differences to previous application versions to be inspected. This is an exceptional method of providing a reliable environment for developers. Furthermore, Docker promotes the concept of reusability, since any object that is developed can be re-used and serve as a “base image” to create some other component. Another essential aspect of Docker is that it provides developers with a tool to automatically build a container from their source code.

The main difference between a Docker container and a VM is that while each VM has its own OS, dependencies and applications running within it, a Docker container can share an OS image across multiple containers. In essence, a container only holds the dependencies and applications that have to be run within them. For example, assuming a group of containers were making use of the same OS image, the OS would be common to all containers and not be duplicated contrary to the case of a VM topology.

Docker has become the flagship in the containerization technology arena since its release (Felter et al., 2014) (Docker Inc, 2013). This open-source project has gained much notoriety in the field of cloud computing, where major cloud platforms and companies (e.g. Google, IBM, Microsoft, AWS, Rackspace, Red Hat, VMware) are backing it up. These companies are integrating Docker into their own infrastructures and they are collaborating in Docker’s development. Recently, a few alternatives to Docker have cropped up, such as Rocket³, Flockport⁴ and Spoonium⁵.

An adequate monitoring of the pool of resources is an essential aspect of a cloud computing infrastructure. The monitoring of resources leads to improved scalability, better placement of resources, failure detection and prevention, and maintenance of architectural consistency, among others. This is relevant for VMs, and it is just as applicable to OS-level virtualization. Out of this need to monitor containers, within the paradigm of OS-level virtualization platforms, the following research questions have been addressed in

this paper:

- How could an OS-level virtualization platform be monitored to obtain relevant information concerning images and containers?

This paper presents an investigation of this issue as well as an implementation of a Container Monitoring Agent.

- Is the resource usage information about the running containers reported by our Container Monitoring Agent valid?

This paper details a quantitative evaluation of the validity of the measurements collected by the proposed Container Monitoring Agent.

The rest of the paper is organized as follows. Section 2 presents state-of-the-art research related to the monitoring of containers and virtual machines. Section 3 explains the different components of the monitoring system architecture. Section 4 presents the results obtained. Section 5 discusses the research questions. Finally, Section 6 presents the conclusions of the paper and discusses the possibilities for future work.

2 RELATED WORK

There is an active interest in industry and in research to build monitoring solutions (Ranjan and Tai, 2014). In (Kutare et al., 2010) the term monalytics is coined. Monalytics refers to a deployment of a dynamically configurable monitoring and analytics tool for large-scale data centers, targeting the XEN hypervisor. One of the monalytics’ topologies defined, matches the architecture chosen for the monitoring solution provided in this paper. The research of (Meng et al., 2012) is centered on providing a state monitoring framework that analyzes and mitigates the impact of messaging dynamics. This technique ensures the trustworthiness of the measurements collected by a distributed large-scale monitoring tool on XEN-based virtual machines. Most existing monitoring research for data center management target virtual machines. To the best of our knowledge, at the time of writing this paper, there were no research papers on the topic of monitoring the Docker platform.

When this monitoring solution for Docker was developed, there were a lack of open-source implementations to monitor Docker. However, recently, several other monitoring systems for Docker have appeared.

Datadog agent (Datadog Inc, 2014) is an open-source tool developed by *Datadog Ink* which monitors Docker containers by installing an agent within the host where the Docker platform is running. It is

³<https://coreos.com/blog/rocket/>

⁴<http://www.flockport.com/start/>

⁵<https://spoon.net/docs>

an agent-based system which requires metrics to be pushed to the *Datadog cloud* thereby making the task of monitoring entirely dependent on *Datadog's cloud*. Unlike the *Datadog agent*, the monitoring agent for the Docker platform presented in this paper is not only open-source, but also independent of any particular collector. It can be integrated within different monitoring architectures after the proper configuration is done.

cAdvisor is an open-source project created by Google Inc (Bryan Lee, 2014) to monitor their own *lmtfy* containers (Google Inc, 2014). Support to monitor the Docker platform was later added to it. Therefore, this monitoring tool provides Docker metrics, which are shown in real time but are not stored for more than one minute. This feature may be useful to test container performance but, due to the small data time frame displayed, it is not possible to get a historical of the metrics collected.

The *Host sFlow agent* (InMon Inc, 2014) is an open-source tool to monitor the resource consumption of a host. It has recently incorporated the option to monitor the Docker platform, making it a viable open-source monitoring solution for Docker. However, this agent adheres to the sFlow standard⁶, which enforces constraints on the information it is able to send as there is no dedicated sFlow structure for Docker. By contrast, the solution provided in this paper does not have limitations on the metrics that can be obtained. The monitoring agent presented here can be modified to select which metrics, out of all the available, to monitor.

As shown above, only the solution proposed in this paper manages to provide a monitoring module for Docker that is open-source, does not adhere to a particular collector or monitoring framework provided some configuration is done, and allows for the selection of a certain subset of metrics from all the available ones.

3 SYSTEM ARCHITECTURE

To monitor Docker containers, three separate modules could be employed: our Container Monitoring Agent (CoMA), a metrics' collector and a host monitoring agent. The solution proposed in this paper, to create a distributed Docker monitoring system consists of: CoMA, the Ganglia Monitoring System and the Host sFlow agent.

In Figure 1, a possible layout of the proposed monitoring architecture is shown. This figure repre-

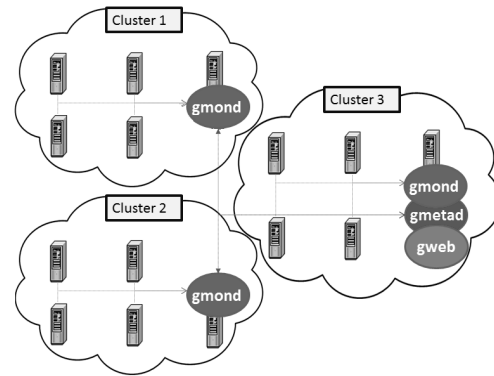


Figure 1: Cloud topology.

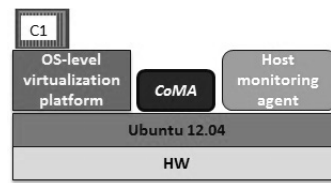


Figure 2: General overview of the host.

sents three clusters. Each one of the monitored hosts represented in Figure 1 have the structure presented in Figure 2.

3.1 CoMA: CONTAINER MONITORING AGENT

We have developed CoMA, the agent that monitors containers of OS-level virtualization platforms such as Docker. CoMA retrieves information about the containers and images in the Docker platform. It also tracks the CPU, memory and block I/O resources being consumed by the running containers. CoMA can be accessed as an open-source project at <https://github.com/laraljj/CoMA>.

The collection of Docker metrics in CoMA is accomplished via two modules. One module makes requests to the platform's Remote API (Docker Inc, 2014a) to collect data about the containers and images. These include: the number of Go routines being executed by the Docker platform; the images that have been created in each host and information about these (e.g. size and virtual size); the number of containers that have been created and from which image they have been built; the status of all the containers in the host (i.e. whether they are running or stopped).

The second module obtains information about resource usage and resource limitations of the running containers. These metrics are obtained by accessing the control groups (cgroups) feature of the Linux kernel (Paul Menage, 2014), which accounts and sets

⁶<http://www.sflow.org/developers/specifications.php>

limits for system resources within different subsystems. The container resources monitored by this module include CPU, memory and block I/O. The number of measurements recorded by CoMA vary according to the number of containers that have been deployed. For the Docker platform as a whole, there are 18 CPU related metrics. Per container, 18 CPU related metrics, 24 memory related metrics, and 100 block I/O related metrics are measured. This means that when a single container is running, there are a total of 160 measurements available, 142 of these are container specific and 18 of these are related to the Docker platform itself. Therefore, when two containers are running there will be 302 measurements, 142 measurements for one container, 142 measurements for the other container and 18 measurements for the Docker platform. The metrics that are being reported by CoMA can be selected according to the needs of each specific deployment, so that only the values of those metrics are dispatched to the collector, instead of all of them.

3.2 Complementary Components

3.2.1 The Ganglia Monitoring System

The Ganglia Monitoring System (Massie et al., 2012) is an open-source distributed monitoring platform to monitor near real-time performance metrics of computer networks. Its design is aimed at monitoring federations of clusters. The Ganglia Monitoring System was selected as collector due to its capacity to scale, its distributed architecture and because it supports the data collection of the Host sFlow agent. However, in the interest of fitting the requirements of a different system, a stand-alone collector could be used instead.

The system is comprised of three different units: *gmond*, *gmetad* and *gweb*. These daemons are self-contained. Each one is able to run without the intervention of the other two daemons. However, architecturally they are built to cooperate with each other.

Gmond is a daemon that collects and sends metrics from the host where it is running to *gmetad*. This is not a traditional monitoring agent, as it does not sit passively waiting for a poller to give the order to retrieve metrics. *Gmond* is able to collect metric values on its own, but *gmond*'s built-in metrics' collection may be replaced by the Host sFlow agent. This setup implies that, for the architecture chosen, *gmond* acts as in-between software layer for the Host sFlow agent and *gmetad*.

Gmetad is a daemon running a simplified version of a poller, since all the intelligence of metric retrieval lays, in our case, with the Host sFlow agent

and *gmond*. *Gmetad* must be made aware of from which *gmonds* to poll the metrics. *Gmetad* obtains the whole metric dump from each *gmond*, at its own time interval, and stores this information using the RRD-tool (i.e. in "round robin" databases).

Gweb is Ganglia's visualization UI. It allows for an easy and powerful visualization of the measurements collected, mostly in the form of graphs. New graphs combining any number of metrics can be generated, allowing the visualization of metrics to be customized depending on individual needs.

3.2.2 The Host sFlow Agent

The Host sFlow agent was selected as the host monitoring agent to track the resources consumed by the OS in the host running the OS-level virtualization platform. Monitoring resources both at the host level and at the virtualization platform level makes it possible to compare the values of the metrics for soundness checks, tracking problems at both levels.

The Host sFlow agent may retrieve information from within an OS running on bare metal or from within the hypervisor if the aim is to monitor virtual machines. This agent can be run in multiple OSs and hypervisors. The agent itself obtains the same metrics as *gmond*'s built-in collector does. The difference between these two solutions is that the sFlow standard, used by the Host sFlow agent to relay metrics, is considerably more efficient than *gmond*. This is because each sFlow datagram carries multiple metric values, which reduces the number of datagrams that need to be sent over the network. For example, monitoring 1,000 servers with *gmond* would create the same network overhead as 30,000 servers with the sFlow protocol (Massie et al., 2012). The sFlow protocol's efficiency justifies the usage of the Host sFlow agent in this monitoring system.

4 EVALUATION

The primary evaluation objective is to assess the validity of the values of the metrics collected with CoMA. Validity in this context means to establish, for the metrics reported by CoMA, whether the measured values reflect the real values. Given the numerous metrics reported about CPU, memory and block I/O, a small subset of these metrics has been selected for the evaluation. This validity assessment is carried out on user CPU utilization, system CPU utilization, memory usage and number of bytes written to disk. User CPU utilization and system CPU utilization refer to the percentage of CPU that is employed to execute

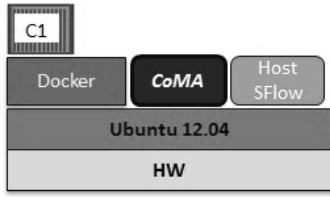


Figure 3: Scenario 1, no workload (baseline).

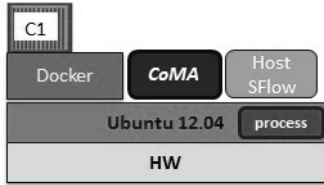


Figure 4: Scenario 2, workload on host OS.

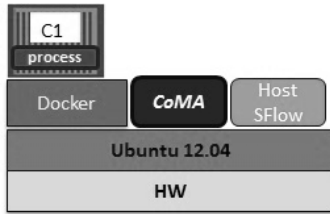


Figure 5: Scenario 3, workload in container.

code in user space and in kernel space, respectively. It should be noted that for all tests, Ubuntu 12.04 LTS and the Docker platform 1.3.1 have been run on the same Dell T7570 computer. This computer runs an Intel Pentium D 2.80 GHz (800 MHz) and 2 GB of RAM at 533 MHz.

The evaluation presented collects for each host: host-specific metrics (reported by the Host sFlow agent) and the metrics from the Docker platform (reported by CoMA). The purpose of collecting the values for both sets of metrics was to compare and contrast the host's resource consumption against the resource consumption of the Docker platform, that is, against the resource consumption of the collection of containers and images within the host. The objective of this comparison is to offer a reliable overview of the system from a resource usage perspective. Comparing both sets of metrics, a system administrator can pinpoint the origin of a particular misbehavior by determining if the issue is due to the Docker platform (i.e. a specific container) or due to some other problem within the host but independent of the containers.

4.1 Validity of CPU and Memory Measurements

Three different scenarios have been set up to assess

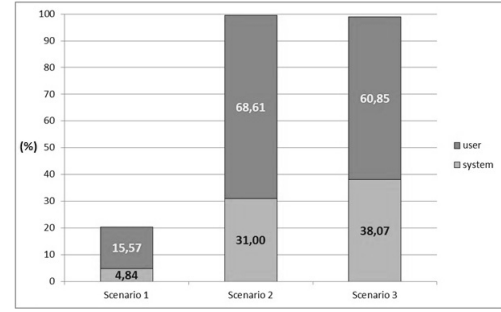


Figure 6: Host CPU utilization reported by the Host sFlow agent.

Table 1: Host CPU utilization reported per scenario. Total CPU is the aggregation of user CPU and system CPU. Standard Deviation (SD).

	CPU system (%)	SD CPU system (%)	CPU user (%)	SD CPU user (%)	Total CPU (%)	SD Total CPU (%)
Scenario 1	4.84	1.79	15.57	5.59	20.42	5.87
Scenario 2	31.00	14.92	68.61	14.91	99.61	20.60
Scenario 3	38.07	11.20	60.85	11.16	98.92	15.29

the collected values of the memory and CPU-utilization metrics. For all scenarios, 6 rounds, each one of 30 minutes have been run. Scenario 1 (Figure 3) presents a baseline of the CPU and memory usage while the OS is executing Docker, which runs a single unlimited container executing `/bin/bash`, the Host sFlow Agent and CoMA. The container was not actively used during this scenario. Scenario 2 (Figure 4) is set up like Scenario 1, except for the fact that a workload generator was executed natively in the OS. This means that the process did not run containerized. *Stress-ng*⁷ has been used to generate load on both CPU and memory. In order to create load on the CPU, two workers were launched so that there would be a worker per core. Each CPU worker executed `sqr(rand())` to generate load. To stress the memory, five workers were started on anonymous mmap, each one of these workers was set to 420MB. Scenario 3 (Figure 5) has been laid out exactly like Scenario 2, the only difference being that the *stress-ng* processes were executed containerized.

4.1.1 CPU: A Single Container

The data obtained from each scenario were processed. Figure 6 shows user CPU, system CPU and total CPU utilization reported at the host level for each scenario. Figure 7 displays CPU utilization pertaining to the process or processes running within that single container deployed in the three scenarios.

In order to determine that the measurements of the CPU metrics collected with CoMA are valid, the data

⁷<http://kernel.ubuntu.com/~cking/stress-ng/>

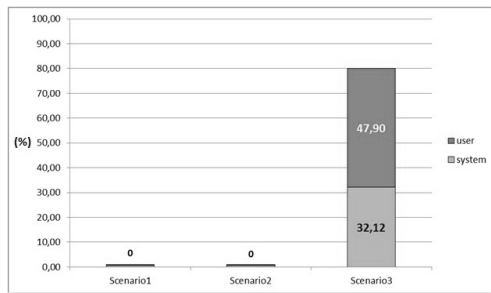


Figure 7: Container CPU utilization reported by CoMA.

Table 2: Container CPU utilization reported by CoMA per scenario. Note that there are no processes running in Scenario 1 and Scenario 2.

	CPU system (%)	SD CPU system (%)	CPU user (%)	SD CPU user (%)	Total CPU (%)	SD Total CPU (%)
Scenario 1	-	-	-	-	-	-
Scenario 2	-	-	-	-	-	-
Scenario 3	32.12	9.44	47.90	9.77	78.41	2.78

obtained from Scenario 3, which can be visualized in Figure 6 and 7, has been compared. The total CPU of Scenario 1 (Table 1), aggregated to the total CPU of the container reported by CoMA in Scenario 3 (Table 2), should resemble the total CPU reported by the host in Scenario 3 (Table 1). The aggregation of those first two values (20.42% and 78.41%) results in a total CPU of 98.83% and a standard deviation of 6.5. The total CPU utilization of the whole host in Scenario 3 is 98.92% with a standard deviation of 15.29. These results verify that the values of the CPU metrics gathered by CoMA are valid.

The CPU utilization data retrieved from this evaluation allows for other noteworthy observations to be made. In Figure 6 and Table 1 a small difference of 0.69% can be ascertained, between running the *stress-ng* processes natively (Scenario 2) or containerized (Scenario 3). The disparity that exists between these two scenarios is due to several reasons. First, the intrinsic variable nature of the data collected has a direct impact on the results attained. However, its irregularity is acceptable as the standard deviations calculated demonstrate, since these are reasonable and valid for these data. Second, the *stress-ng* processes themselves may be accountable for a certain variation.

It can also be noticed that there seems to be a tendency in the way these *stress-ng* processes are executed. When *stress-ng* was run within a container more system CPU utilization was accounted for compared to when *stress-ng* was run natively. The effect is the exact opposite when it comes to user CPU utilization, as can be visualized in Figure 6. This last observation has been verified by computing the correlation coefficient between system CPU utilization and user CPU utilization. A nearly perfect negative corre-

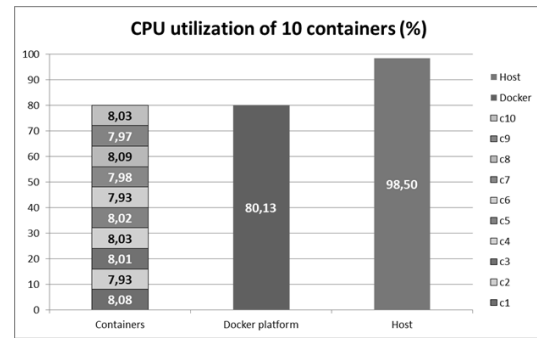


Figure 8: Total CPU utilization of 10 containers, where each container runs the same process generating a symmetric CPU load across all containers. CoMA reports the values for the containers and for the Docker platform. The Host sFlow agent reports the values for the host.

Table 3: Total CPU utilization and standard deviations (SD) for Figure 8.

Total CPU utilization (%)						
Host		Docker platform		Containers		
Total CPU	SD Total CPU	Total CPU	SD Total CPU	Name of container	Total CPU	SD Total CPU
98.50	3.30	80.13	1.10	c1	8.08	0.23
				c2	7.93	0.18
				c3	8.01	0.26
				c4	8.03	0.22
				c5	8.02	0.38
				c6	7.93	0.30
				c7	7.98	0.24
				c8	8.09	0.28
				c9	7.97	0.20
				c10	8.03	0.28

lation of -0.99 was obtained for Scenario 2 and -0.98 for Scenario 3.

4.1.2 CPU: Multiple Containers

These scenarios prove that the CPU utilization retrieved by CoMA, of one container, is valid. However, whether the agent is able to properly report the CPU metrics for multiple simultaneously running containers should also be demonstrated. For this purpose, two tests were carried out. For the first test, ten containers ran the exact same *stress-ng* process to generate load on the CPU with two workers, one per core. In accordance with the default process scheduler in Linux, the Completely Fair Scheduler (CFS)⁸, the expected outcome of this test is for each container to employ a similar portion of the total CPU. As it can be observed in Figure 8 and Table 3, each container is granted an average of around 8% of the CPU. The aggregation of each container's total CPU utilization adds up to 80.07% which almost matches

⁸<http://lwn.net/Articles/230501/>

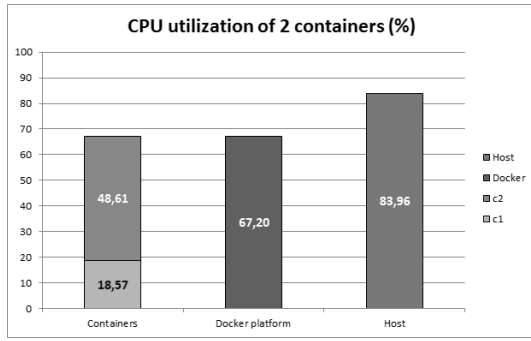


Figure 9: Total CPU utilization of 2 containers with asymmetric processes running in each container. CoMA reports the values for the containers and for the Docker platform. The Host sFlow agent reports the values for the host.

Table 4: Total CPU utilization and standard deviations (SD) for Figure 9.

Total CPU utilization (%)						
Host		Docker platform		Containers		
Total CPU	SD Total CPU	Total CPU	SD Total CPU	Name of container	Total CPU	SD Total CPU
83.96	1.46	67.20	0.65	c1	18.62	0.28
				c2	48.61	0.59

the total CPU utilization of the whole Docker platform (80.13%) as reported by CoMA.

This test shows that each container reports its own set of CPU measurements independently and is able to do so effectively. However, a different test was carried out to verify this by running asymmetric processes in two containers. Each container ran the *stress-ng* process with different settings so as to generate an uneven load across the two containers. As represented by Figure 9 and Table 4, CoMA reported just that. A container used 48.61% of the total CPU whilst the other container employed 18.57% of the total CPU. Both containers together used 67.18%, which resembles the value (67.20 %) reported by CoMA of the total CPU utilization of the Docker platform.

4.1.3 Memory: A Single Container

The memory data captured for all scenarios is displayed in Figure 10. It should be mentioned that there is a greater fluctuation in the memory-reported values than in the CPU values. This phenomenon is due to the manner in which Linux manages its memory. The memory management methodology applied by Linux varies according to the needs of the OS at any given time. This adds another layer of complexity when analyzing the metrics collected. The host's memory usage in Scenario 1 (360.73MB) aggregated to the container's memory usage reported by CoMA in

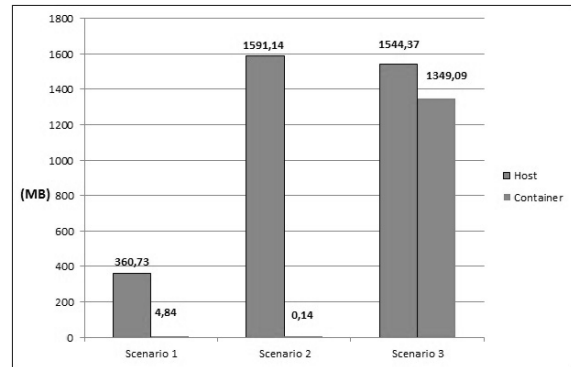


Figure 10: A comparison of the memory usage reported by the host and the container per scenario.

Table 5: Memory usage and standard deviations (SD) for Figure 10.

	Host-reported memory		Container-reported memory	
	Memory usage (MB)	SD Memory usage (MB)	Memory usage (MB)	SD Memory usage (MB)
Scenario 1	360.73	11.35	4.84	0.02
Scenario 2	1591.14	224.43	0.14	0.17
Scenario 3	1544.37	209.47	1349.09	184.57

Scenario 3 (1349.09MB), should be somewhat similar to the host's memory consumption in Scenario 3 (1544.37MB). In this case there is a difference of around 165MB. As it has been explained before, this discrepancy is caused by the memory management enforced by Linux, as well as by the error introduced in the averaging process of the results.

4.1.4 Memory: Multiple Containers

Much like it happens with CPU, the previous scenarios establish that the memory metrics provided by CoMA are valid for a single container. The two CPU tests performed with multiple simultaneously running containers, were also carried out for memory. As it can be observed in Figure 11 and Table 6, when the same process is running in each container, the memory usage value presented by CoMA per container has a greater variability than that observed in the same test for CPU utilization. As it has been previously explained, these fluctuations are due to the changeable nature of how memory is managed by the OS. However, each container's memory usage is close to 152MB. The aggregated memory usage of all 10 containers adds up to 1519.92MB. The Host sFlow agent reports a memory usage of 1773.62MB for the whole host during this test. The difference of 253.70MB between these last two values, represents the memory being employed by the OS to run non-containerized processes. A second test, where two containers were

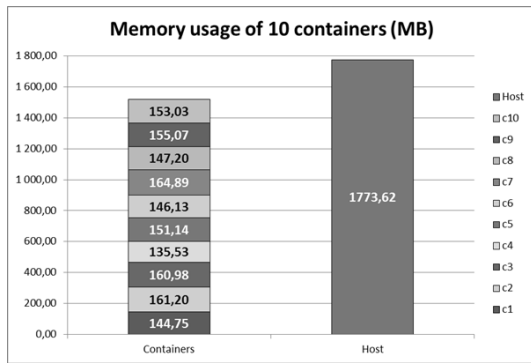


Figure 11: Memory usage of 10 containers, where each container runs the same process. This generates a symmetric memory use across all containers. CoMA reports the values for the containers and for the Docker platform. The Host sFlow agent reports the values for the host.

Table 6: Memory usage and standard deviations (SD) for Figure 11.

Memory (MB)				
Host		Containers		
Memory usage	SD Memory usage	Name of container	Memory usage	SD Memory usage
1773.62	59.57	c1	144.75	20.60
		c2	161.20	20.87
		c3	160.98	11.56
		c4	135.53	15.96
		c5	151.14	11.86
		c6	146.13	14.54
		c7	164.89	14.50
		c8	147.20	12.54
		c9	155.07	20.74
		c10	153.03	27.69

configured to make a disparate use of memory was also carried out. Figure 12 and Table 7 reflect the results obtained, which are consistent with the values gathered when running a symmetric memory load on 10 containers.

4.2 Validity of block I/O Measurements

To evaluate whether the block I/O measurements gathered by CoMA were solid, the I/O tool *fio*⁹ was used to write 1000MB directly to the ext4 filesystem mounted by the host by making use of the `-v` flag (Docker Inc, 2014d) on the container. In order to achieve this, *fio* was configured to initiate five workers, each worker performing random write operations of 200 MB in the shared folder between the host and the container.

The test of writing 1000MB to disk was executed at 12:00 and it finished by 12:07. As Figure 13 shows,

⁹<http://freecode.com/projects/fio>

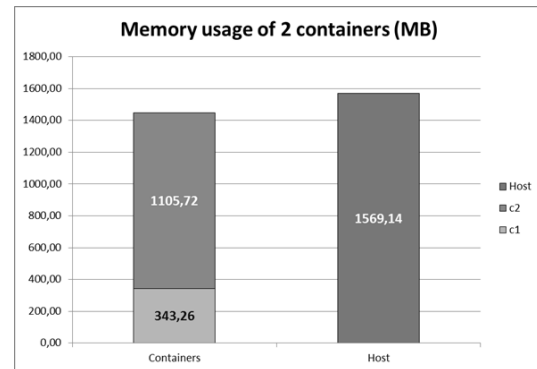


Figure 12: Memory usage of 2 containers with asymmetric processes running in each container. CoMA reports the values for the containers and for the Docker platform. The Host sFlow agent reports the values for the host.

Table 7: Memory usage and standard deviations (SD) for Figure 12.

Memory (MB)				
Host		Containers		
Memory usage	SD Memory usage	Name of container	Memory usage	SD Memory usage
1569.14	194.24	c1	343.26	65.01
		c2	1105.72	130.16

exactly 1000MB were reported to have been written during that time.

A separate test was created, following the same principle previously explained, to write to disk from three simultaneously running containers. *Fio* was configured for each container with a disparate number of workers and file sizes. The first container spawned two workers, each of which had to write 300MB to the shared folder. The second container initiated three workers, each with a file size of 250MB. The third container started five workers, where each worker had to write 200MB to disk. For each container, the number of bytes that CoMA reported were written to disk was exactly right, down to the last byte. The first container took 20 minutes to write the 600MB to disk. The second and third container took around 16 minutes to write 750MB and 1000MB to disk, respectively. The time taken for each container to complete the task of writing these files to memory is closely linked to the number of workers running and the number of containers writing to disk.

5 DISCUSSION

This section discusses CoMA as well as the evaluation results obtained in terms of the research questions proposed.

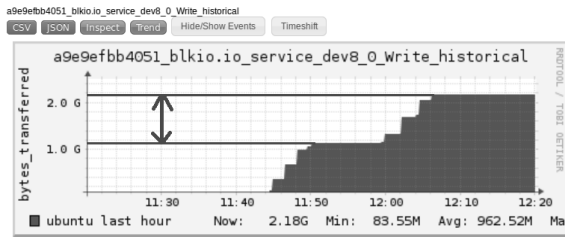


Figure 13: Bytes written to disk by container.

How could an OS-level virtualization platform be monitored to obtain relevant information concerning images and containers?

CoMA retrieves information about the usage of CPU, memory and block I/O of running containers from the Linux kernel’s control groups. It also fetches the data concerning images and containers using Docker’s Remote API.

Is the resource usage information about the running containers reported by our Container Monitoring Agent valid?

The evaluation provides validation across the following three blocks of metrics: CPU, memory and block I/O. The most complex metric to validate was the memory usage of a container. This is due to the way memory is managed in an OS, which causes the memory usage baseline in Scenario 1 to account a slightly overestimated usage. The authenticity of all the measurements that can be collected with CoMA could not be tested because of the number of metrics that CoMA is able to gather. Nevertheless, since the values are being reported by the Linux kernel, assessing at least one metric from each group of metrics is sufficient to establish the validity of CoMA. It should be mentioned that CoMA can be modified to only dispatch a subset of desired metrics.

CoMA’s CPU utilization is dependent on the test-bed that has been set up. This means that CoMA’s resource usage is contingent on the hardware that has been employed, the number of containers that had been deployed, the number of metrics being sent and the sampling rate set for CoMA. This last value can be configured to obtain measurements closer or further away from real-time monitoring, depending on the requirements. There are certain tradeoffs in the selection of the sampling rate. A higher sampling rate would mean obtaining more accurate measurements in terms of time, but more resources would be used in order to monitor the platform. It is worth mentioning that CoMA itself consumes around 15.25% of CPU with a standard deviation of 5.87 for the specific test-bed presented in the evaluation section. This number may seem high, but it is relative to the hardware being employed. An Intel Pentium D 2.8GHz and 2GB

RAM at 533MHz was used in this case. Had conventional cloud computing hardware been used, this percentage would be much lower. Moreover, in this test-bed all available metrics are collected, if fewer of them were collected the percentage of CPU used would decrease. It should also be mentioned that the monitoring solution itself shall be optimized so as to minimize its impact.

It has been previously mentioned that CoMA could be employed to monitor similar OS-level virtualization platforms. For this to happen, said OS-level virtualization platform would have to account resource usage in a similar fashion to Docker, i.e. using the Linux kernel’s control groups. However, the information pertaining to the containers and images that is collected through Docker’s Remote API, is specific to the Docker platform itself.

6 CONCLUSION AND FUTURE WORK

Monitoring the resource consumption of OS-level virtualization platforms such as Docker, is important to prevent system failures or to identify application misbehavior. CoMA, the Container Monitoring Agent presented in this paper, reports valid measurements as shown by our evaluation. It currently tracks CPU utilization, memory usage and block I/O of running containers. CoMA could be configured to gather a subset of the available metrics to suit the monitoring needs of a particular system or application. This paper has presented a possible implementation solution of CoMA to build a distributed and scalable monitoring framework, using the open-source projects Ganglia and the Host sFlow agent.

It would be positive to monitor the network usage of the containers, since this feature has not yet been implemented in CoMA. Moreover, establishing thresholds on certain metrics collected by CoMA to trigger alarms or actions would be beneficial. Also, assessing CoMA’s behavior when numerous containers are deployed on commonly used hardware in data centers is required. This would be a proper test-bed to gauge CoMA’s performance in a realistic cloud computing scenario.

Another area for further research would be to employ machine learning techniques on the values collected, to maximize resource usage by modifying each container’s resource constraints based on the needs of the running containers. There is also the possibility of applying data analytics on the information captured by CoMA to build an autonomous system for container placement within a cloud or across clouds.

There are new and upcoming OS-level virtualizations platforms that could rival Docker, such as Rocket. CoMA could be also employed and evaluated with these recent virtualization platforms.

REFERENCES

- Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177.
- Bryan Lee (Accessed: 2014). cAdvisor monitoring tool. <http://blog.tutum.co/2014/08/07/using-cadvisor-to-monitor-docker-containers/>.
- Datadog Inc (Accessed: 2014). Docker-ize Datadog with agent containers . <https://www.datadoghq.com/2014/06/docker-ize-datadog/>.
- Docker Inc (2013). What is Docker technology ? <https://www.docker.com/whatisdocker/>.
- Docker Inc (Accessed: 2014a). Docker remote API. https://docs.docker.com/reference/api/docker_remote_api/.
- Docker Inc (Accessed: 2014b). Docker working with LXC. <https://docs.docker.com/faq/>.
- Docker Inc (Accessed: 2014c). File sytem architecture of the Docker platform . <https://docs.docker.com/terms/layer/>.
- Docker Inc (Accessed 2014d). Volume system with Docker. <https://docs.docker.com/userguide/dockervolumes/>.
- Elena Reshetova, Janne Karhunen, T. N. N. A. (2014). Security of os-level virtualization technologies.
- Felter, W., Ferreira, A., Rajamony, R., and Rubio, J. (2014). An updated performance comparison of virtual machines and linux containers. *technology*, 28:32.
- Google Inc (Accessed: 2014). Imctfy: Let Me Contain That For You . <https://github.com/google/imctfy>.
- InMon Inc (Accessed: 2014). HostsFlow monitoring tool . <http://host-sflow.sourceforge.net/>.
- Kutare, M., Eisenhauer, G., Wang, C., Schwan, K., Talwar, V., and Wolf, M. (2010). Monalytics: Online monitoring and analytics for managing large scale data centers. In *Proceedings of the 7th International Conference on Autonomic Computing, ICAC '10*, pages 141–150, New York, NY, USA. ACM.
- Massie, M., Li, B., Nicholes, B., Vuksan, V., Alexander, R., Buchbinder, J., Costa, F., Dean, A., Josephsen, D., Phaal, P., and Pocock, D. (2012). *Monitoring with Ganglia*. O'Reilly Media, Inc., 1st edition.
- Meng, S., Iyengar, A. K., Rouvellou, I. M., Liu, L., Lee, K., Palanisamy, B., and Tang, Y. (2012). Reliable state monitoring in cloud datacenters. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, CLOUD '12*, pages 951–958, Washington, DC, USA. IEEE Computer Society.
- Paul Menage (Accessed: 2014). Control Groups (cgroups) Documentation . <https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt>. Available since: 2004.
- Ranjan, R., B. R. L. P. H. A. and Tai, S. (2014). A note on software tools and techniques for monitoring and prediction of cloud services softw: Pract. exper., 44: 771–775.
- Tafa, I., Beqiri, E., Paci, H., Kajo, E., and Xhuvani, A. (2011). The evaluation of transfer time, cpu consumption and memory utilization in xen-pv, xen-hvm, openvz, kvm-fv and kvm-pv hypervisors using ftp and http approaches. In *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, pages 502–507.
- VMware Inc (2007a). Understanding Full Virtualization, Paravirtualization and Hardware Assist. http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf. Accessed: 2014 (white paper).
- VMware Inc (2007b). Virtualization Overview. <http://www.vmware.com/pdf/virtualization.pdf>. Accessed: 2014 (white paper).
- Xavier, M., Neves, M., Rossi, F., Ferreto, T., Lange, T., and De Rose, C. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 233–240.
- Xu, F., Liu, F., Jin, H., and Vasilakos, A. (2014). Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE*, 102(1):11–31.
- Ye, K., Huang, D., Jiang, X., Chen, H., and Wu, S. (2010). Virtual machine based energy-efficient data center architecture for cloud computing: A performance perspective. *IEEE-ACM International Conference on Green Computing and Communications and International Conference on Cyber, Physical and Social Computing*, 0:171–178.

A Survey of Trust Management Models for Cloud Computing

Flavio Corradini, Francesco De Angelis, Fabrizio Ippoliti and Fausto Marcantoni

Computer Science Division, University of Camerino, Via del Bastione 1, 62032, Camerino, Italy
{flavio.corradini, francesco.deangelis, fabrizio.ippoliti, fausto.marcantoni}@unicam.it

Keywords: Cloud Computing, Privacy, Security, Trust Relationship, Trust Model, Trust Management.

Abstract: Over the past few years, cloud computing has been widely adopted as a paradigm for large-scale infrastructures. In such a scenario, new security risks arise when different entities or domains share the same group of resources. Involved organizations need to establish some kind of trust relationships, able to define appropriate rules that can control which and how resources and services are going to be shared. The management of trust relationships represents a key challenge in order to meet high security requirements in cloud computing environments. This allows also to boost consumers confidence in cloud services, promoting its adoption. Establishing trust with cloud service providers supports to have confidence, control, reliability, and to avoid commercial issues like lock in. This paper proposes a survey of existing trust management models addressing collaboration agreements in cloud computing scenarios. Main limitations of current approaches are outlined and possible improvements are traced, as well as a future research path.

1 INTRODUCTION

Over the past few years, cloud computing has been widely adopted in almost every kind of organizations, for providing flexible and on-demand infrastructures, platforms and software as a service. Customers benefit from cloud services in their daily life, sometimes without even being aware that they are using services developed on a cloud computing infrastructure. In addition to the well-known benefits resulting from cloud computing adoption, several issues have emerged during its evolution: most of them relate to security, privacy and trust management. In particular, its proliferation has placed even more attention to trust management, representing one of the key challenges in the adoption of cloud computing technologies.

The speed and flexibility of adjustment to vendor offerings have motivated correct understanding of cloud computing paradigm, but, at the same time, this fact has introduced a higher risk to data privacy and security (Pearson and Benameur, 2010). From the cloud customer point of view, who may be either citizens, businesses or organizations, this represents a crucial concern, especially when entrusting cloud service providers (CSPs) for private or sensitive information, like financial or health data or business-confidential information. The resulting lack of trust is a key inhibitor to cloud adoption in domains where confidential or sensitive information is involved.

Indeed, according to a study presented by researchers at UC Berkeley (Armbrust et al., 2010), trust management and security aspects are ranked among the top 10 obstacles for adopting cloud computing. A more recent survey conducted by KPMG (KPMG International, 2013) affirms that major concerns affecting cloud adoption are about control and data security. In particular, CSPs report that customers main concern over switching to cloud is losing control, an issue voiced by almost half of all respondents. A even more recent white paper by Cloud Industry Forum addressing the UK scenario (Cloud Industry Forum, 2014), confirms that among most significant concerns about cloud adoption there are: data security (61%); data privacy (54%); fear of loss of control/manageability (24%); respectively as first, second and fourth reasons.

Lack of consumer trust is confirmed too from a study about attitudes on data protection and electronic identity in the European Union (European Commission, 2011), where less than one-third of European citizens surveyed trust phone companies, mobile phone companies and Internet service providers. Besides, 70% of them are concerned about their personal data held by companies being used for different aims than the agreed ones.

Main contribution of this work is to present a survey of most relevant approaches of trust models for cloud computing, categorized in different classes. Af-

ter this exhaustive comparison, major limitations of existing models are outlined and possible improvements are traced, as well as a future research path.

The remainder of this paper is organized as follows: the concepts of trust and reputation, starting from their origin to the definition in computer science are described in section 2. Section 3 provides a classification of trust management models and for each group a detailed description of most relevant works in literature. Section 4 presents main limitations of current trust models and possible improvements. Finally, section 5 traces some conclusions and future work to be realized.

2 BACKGROUND

Trust and reputation concepts have their origin in the social sciences that study the nature and behavior of human societies (Gambetta, 1988), basically representing an act of faith. Trust management was originally developed by (Blaze et al., 1996), addressing important issues in network services security: centralized control of trust relationships, inflexibility to support complex trust relationships in large scale networks, and the heterogeneity of policy languages. Moreover, a widely accepted definition of trust, coming from cross-disciplinary set of academic literature, states as follows (Rousseau et al., 1998): "Trust is a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behaviour of another". What arises from these general definitions is that trust is basically an attitude, a form of confidence in another, a belief that the other, despite a capacity to harm, will do the right thing in relation to the trustor (Nissenbaum, 1999). So, dealing with trust presupposes the acceptance of some kind of risk, even if its nature may be unclear or ambiguous. Moreover, trust cannot be just a common value that can be identified by a user and valid for every aspects of cloud services.

Furthermore, trust relates not only to technological aspects, but also social factors like reputation. Reputation is maybe a company's most valuable asset (Nissenbaum, 1999).

2.1 Defining Trust in Computer Science

Trust represents an essential aspect of every system where different entities have to collaborate. For such a complex concept, there is no universally accepted scholarly definition. However, trust relies to the competence of an entity to act dependably, securely and reliably within a specified context, as discussed in

(Grandison and Sloman, 2000). Usually, trust lifecycle composes of three activities, that are: trust establishment, trust update and trust revocation. Moreover, trust is often divided into two classes: direct trust and recommended trust (Zhu et al., 2003). Direct trust represents the trust based on own experience with the other entity. Instead, when two entities have no direct interactions, then trust relationship can be established by another entity's recommendation, called recommended trust.

Another fundamental aspect of trust is the subjectivity, making even more complex its assessment. The term subjective relates to the perception of a subject toward an object. The properties and qualities assigned to an object depend on the subjects perception: for this reason, it may differ from one individual subject to the other (Solhaug and Stølen, 2012).

For these reasons, the notion of trust implies the modeling of trust management systems. A trust management system is a specific technique, normally used in distributed scenarios, able to manage and validate trust relationships agreed between different entities. A trust relationship is a particular kind of relation that defines privileges and restrictions. In this way, a trustor relies upon a trustee according to its ability to perform a specific action or provide a specific service, within a particular context (Grandison and Sloman, 2000). In case the reader is interested, in (Perez et al., 2014) a work providing a taxonomy of trust relationships in authorization domains for cloud computing can be found.

2.2 Trust in Cloud Deployment Models

Cloud services can be deployed in different ways, depending on the organizational structure and the provisioning location. Four deployment models are loosely distinguished, namely public, private, community, and hybrid cloud (Mell and Grance, 2011). The questions related to trust differ across various deployment models (Kumar et al., 2013).

Trust management in a *private* cloud does not represent a main concern if the organization does not rely on third-party CSPs. *Public* cloud model is the most common, but it introduced many risks about security and loss of control over data. *Community* cloud can be owned and managed by the same organizations in the community, a third party, or some combination of them. If there is a third party involved, the problem will occur as well as the corresponding case of the private model. Otherwise the problem is limited to trust relationships discussed and agreed between community subjects. In *hybrid* cloud, a private cloud is involved in the deployment model, besides a public

one. For this reason, trust management issues related to the public model shift to the hybrid one as well. This happens when the private cloud involved needs to scale out relying to the public model: issues about security and privacy become part of the scenario.

2.3 Cloud Transparency Initiatives

CSA (Cloud Security Alliance, 2014c), an international nonprofit organization, provides an important contribution: it aims to promote diffusion and use of best practices for providing security assurance within cloud computing. Among its initiatives, there is one gaining particular attention: "Security, Trust & Assurance Registry" (STAR) (Cloud Security Alliance, 2011). STAR program is a comprehensive set of contribution for cloud provider trust and assurance. It is based upon two components: the "Cloud Controls Matrix" (CCM) (Cloud Security Alliance, 2014a) and the "Consensus Assessments Initiative Questionnaire" (CAIQ) (Cloud Security Alliance, 2014b). CCM is a meta-framework of cloud-specific security controls, referred to leading standards, best practices and regulations. While CAIQ is a set of questions a cloud consumer and cloud auditor may wish to ask of a CSP. It provides a series of "yes or no" control assertion questions which can then be customized to suit each unique customer's demands.

However, despite these efforts, the overall situation shows indecision about if some kind of formal accreditation from a trusted independent organization would be advantageous for the cloud market.

3 TRUST MODELS

Trust modeling is the evaluation process of a system trust, as described in subsection 2.1. This modeling recognizes issues affecting trust of a system and helps in identifying areas where low levels of trust may discredit the system usability (Sanchika Gupta and Abraham, 2013). There exist several classification approaches of trust models for cloud computing present in literature (Firdhous et al., 2012b), (Huang and Nicol, 2013), (Kanwal et al., 2013). (Firdhous et al., 2012b) focuses the categorization according to a specific set of cloud computing parameters the authors have selected. For each trust model analysed in the paper, they analyse some features such as: if an identity management and/or authentication system is involved, which cloud deployment layers are involved or if a Service Level Agreements (SLAs) takes part in the model. (Huang and Nicol, 2013) discusses existing trust mechanisms for cloud, identifying the fol-

lowing categories: reputation based, SLA verification based, cloud transparency, trust as a service, and finally further analysis about formal accreditation, audit, and standards. Furthermore, (Kanwal et al., 2013) proposes a five classes sorting of trust models: Agreement based, Certificate/Secret keys-based, Feedback based, Domain based, and Subjective trust.

Our approach presents a different, simplified classification aiming to reduce the topic complexity, in order to provide a high-level analysis. Following in this section, trust models are categorized, described and briefly analysed upon the following groups:

- Policy Based;
- Recommendation Based;
- Reputation and Feedback Based.

We decided to simplify the classification, avoiding complexity and ambiguity while categorizing specific trust models that might belong to different groups, as it usually happens with some hybrid models. A small overview is also reserved to biological techniques for defining trust models, since they are gaining some attention in the literature.

However, due to limitations of space, we are unable to present all the existing body of literature. For this reason, priority has been assigned to last years' efforts, while less recent papers are in some case cited.

3.1 Policy Based

Trust management models in this group are based on contracts and agreements signed by CSPs for the delivery of their services to customers. The most common agreements are SLA and service policy statements (SPS), providing the basis for trust establishment. In particular, SLA play an important role to make the service trustworthy: it is a negotiation involving from one side CSPs and, from the other one, cloud customers. Various security concerns and quality of service attributes are included in contracts and agreements to establish trust on CSP. A relevant issue of this category is represented by the fact that SLA focuses just on the "visible" elements of cloud service performance (Huang and Nicol, 2013).

(Alhamad et al., 2010). This paper describes the requirements and benefits of using information contained in SLAs, to manage trust in cloud environment, providing a high level architecture capturing major features required, as well as a protocol for the trust model. Aim of the proposed solution is to define reliable criteria for the selection process of CSPs. In other words, its goal is to recommend the "most related and trusted resources" among several CSPs,

meaning that analysed services match all the identified functional requirements. With the term of functional requirements, the authors refer to the detection of the average of several specific dataset or other kinds of data statistical analysis. Whereas, examples of non-functional requirements are represented by the level of privacy to ensure secure data storage or the time used to perform assessment tasks.

(Sato et al., 2010). In the work, the authors introduce the notion of *contracted trust* that check CSPs services, according to contracts and related documents, such as SPS. The fundamental idea is to provide a two levels hierarchy for trust, namely internal trust and contracted trust. The first one is established directly on the cloud platform, if every basic operations are in full control of the customers. Internal trust is achieved via Trusted Platform Module (TPM) that assesses and validates the virtual machine configurations, keeping track of every processes running on cloud platform that assures the process execution control on cloud. On the other hand, the contracted trust is based on SPS, meaning that CSP are involved in this trust layer by negotiating the desired security and QoS requirements of customers.

(Chakraborty and Roy, 2012). The authors show a framework that evaluates trustworthiness of a CSP service using a quantitative trust model. They identified and formalized two classes of parameters, namely *pre-SLA parameters* and *post-SLA parameters*. The first case is the simple one, that is when an initial set of relevant parameters can be obtained directly from SLA statements or other description about the service, available from the CSP. Instead, the second group can be extracted from the session histories or logs. A customer needs to obtain at least one pre-SLA parameter to estimate initial trust value of a CSP. However, measuring trustworthiness based on that is biased toward the single parameter and is not an advantageous solution. For this reason, a user should try to obtain and evaluate as many parameters as possible to obtain a complete trust value about a CSP. In addition, a third party auditor may also be involved in this assessment.

(Marudhadevi et al., 2014). The work presents a *trust mining model* (TMM) to identify trusted cloud services while negotiating an SLA. The challenge for the user is to monitor the services provided from the CSP and check if they meet the conditions mentioned in the agreement. To perform this, the user needs further information such as prior data or knowledge about what is happening on the CSP side, which can help him to better realize the effective QoS. The trust model evaluates the trust degree on the prior data obtained about the service at the time of the SLA. Then, this information is divided into multiple common at-

tributes like the number of service denials, average response time, task success ratio and number of complaints registered by the users. Usually, attributes used to formulate any trust model can be either objective or subjective, while this work uses both types of values. In this way, advantages introduced with this approach are both for CSPs and end users. From one side, the CSP can monitor the performance and improve its services to establish better trust relations with the users. And from the other side, the customer can perceive as secure working with the CSP.

3.2 Recommendation Based

Recommended trust occurs when two entities, the trustor and the trustee have no previous interaction background with each other. In such a scenario, when there is no information that the end user can relate on, the trust relationship can be established by another entity's recommendation, usually a third-party auditor. In this way, end users can have a baseline to evaluate services or providers.

(Kong and Zhai, 2012). The work proposes a particular mechanism, called *Trust-based Recommendation System in service-oriented Cloud computing* (TRSC), which evaluates CSP services based on the trust of them. In TRSC, the resulting trust value is obtained combining direct trust and recommendation trust. Direct trust of an user on a cloud service is computed as usual, that is according on the direct interaction. While the recommended trust is evaluated taking into account opinions coming from users, or other authority of the field, who are trusted by the user, considering that this kind of trust is more reliable.

(Noor et al., 2013). The authors developed a platform for a credibility-based trust management of cloud services, called *Cloud Armor*. The key features of the presented platform are: i) usage of a web crawling approach to automatically discover cloud services; ii) an adaptive and robust credibility model to evaluate credibility of feedbacks; and iii) a trust-based recommender to recommend trustworthy cloud services that suit the users needs. Cloud Armor provides an environment where customers can give feedbacks and request trust assessment for a particular cloud service.

(Rizvi et al., 2014). Aim of the authors is to propose objective trust model, since it involves third-party auditors to develop unbiased trust between CSP and users. In this way, customers have a baseline to assess services and CSPs. In this case, third-party auditor assigns score for each CSP, basing on predetermined criteria significant to trust. More precisely, when a CSP is willing to enter the cloud market, it ap-

plies to be scored by the third-party auditor. The evaluation can be done using different set of criteria, such as those proposed by CSA (Cloud Security Alliance, 2013). However, when scoring a CSP, the customer feedback is taken into account too. For each criteria identified and evaluated by the third-party auditor, the obtained score will be integrated with feedback coming from end users. Like other recommendation-based systems, the approach used in this case is similar to ones adopted by the e-commerce trust models.

(Singh and Chand, 2014). This work proposes a trust evaluation framework able to determine final trust of CSP. The mechanism takes into account, in addition to the user's past experiences, also friends and third party's recommendations. The proposed solution has been simulated through a typical cloud computing scenario.

Similar recommendation based models can be found in (Han et al., 2009) and (Li and Ping, 2009).

3.3 Reputation and Feedback Based

Even if some work in literature discusses about this two groups in a separate way, we prefer to refer to them as a whole class. Because of their similarity, in this way the aim is avoiding ambiguity. The reputation of an entity is the aggregated opinion of a community towards that entity (Huang and Nicol, 2013). Thus, an entity with high reputation is the one trusted by various entities in the community. In this way, an entity that needs to retrieve trust opinion on a trustee, may use the reputation to evaluate the trust level of that subject. The reputation of a CSP helps end users (especially individual users) in choosing a cloud service from many options without particular requirements. A similar approach is defined as "social trust". As already said, this group includes trust models that collect feedback and opinions from other users, evaluating the trust on services and providers. Trust model collects and manages the feedback regarding different QoS and security parameters offered by CSPs. Based on this information, users will prefer the CSP that guarantees all the necessary QoS and security attributes for its customers.

(Krautheim et al., 2010). In this work, a trust model called *Trusted Virtual Environment Module* (TVEM) is presented as a software appliance. For cloud environments already provided with Trusted Platform Module (TPM) virtualization techniques, TVEM introduces better features like improved application program interface (API), cryptographic algorithm flexibility, and a configurable modular architecture. Also a unique Trusted Environment Key is introduced, combining trust from the information owner,

and the CSP to create a dual root of trust for the TVEM that is distinct for every virtual environment and separate from the platforms trust. The TVEM software is protected by hardware enforced memory and process isolation via Intels Virtualization Technology for Directed I/O (VT-d) (Abramson, 2006) and Trusted eXecution Technology (TXT) (Intel Corporation, 2010).

(Habib et al., 2011). This paper describes a trust model based on prepositional logic terms (PLT), called multi-faceted trust management system, to help the cloud service customers to assess trustworthy CSPs. Aim of the proposed solution is to model ambiguity of trust information collected from various sources using a specific set of QoS properties like security, latency, availability, and customer support. The trust model becomes able to integrate two different trust management techniques including reputation and recommendation where logic operators are used.

(Noor and Sheng, 2011). This approach overviews the design and implementation of a Trust as a Service framework. The proposed system is based on a credibility model, responsible for distinguishing between the believable and the malicious trust feedbacks, taking into account the majority consensus of feedbacks too. In addition to the credibility model, the other salient feature of the discussed framework is that it allows trust feedback assessment and storage to be managed distributively, avoiding common drawbacks of centralized architectures.

(Pawar et al., 2012). The authors propose an uncertainty model, which calculates trust values based on different parameters, namely (i) SLA monitoring compliance, (ii) service provider ratings, and (iii) service provider behavior. More in detail, the SLA monitoring defines the opinion about a CSP from the established SLAs about its services. Each of them are provided with a single SLA that includes several common indicators, such as CPU, memory, disk space usage, number of virtual machines. For each indicator of an SLA, a monitor evaluating the compliance/non-compliance of the indicator is provided. Then, CSP ratings are determined with the computation of all ratings, based on consensus and conjunction ratings. To calculate trust values, the model take into account features like belief, disbelief, uncertainty, and base rate.

Similar approaches can be found in (Firdhous et al., 2011a), (Firdhous et al., 2011b), and (Wang et al., 2014). Before the rise of cloud computing, other reputation-based techniques have been developed in the following trust models: *Peertrust* (Xiong and Liu, 2004), *Trummar* (Derbas et al., 2004), *Patrol-F* (Tajeddine et al., 2006), *Patrol* (Tajeddine et al., 2007), and *CuboidTrust* (Chen et al., 2007).

3.4 Biological Techniques

Approaches coming from biological sciences have been recently introduced to improve the definition of trust models for cloud computing. Since this activity is a complex task, the application of this kind of algorithms and techniques that have already been used in fields different from the biological one.

(Wang et al., 2010a), (Wang et al., 2010b). The proposed model discussed in these works is inspired by the biologic gene technique. The discussed solution, called *Cloud Trust model based on Family Gene* (CTFG), is composed of three steps: initialization, identification, and the assignment of the family gene system in the cloud. The work proposes also a formal definition of a model and correlation conception of family gene, cloud family, trust relation, gene identification, and gene assignment.

(Firdhous et al., 2012a). The authors propose that the Bees Algorithm that was used to solve issues in diverse fields could be successfully adapted to address the trust issue in the cloud computing system. The Bees Algorithm is a population based search and optimization algorithm developed based on the food foraging behaviour of honey bees. The work is inspired by some comparative studies carried out on cloud computing and the bees environments.

Another work, referring to trust management in P2P networks, can be found in (Wang et al., 2006) where the authors describe a reputation based trust model inspired by swarm intelligence paradigm.

4 DISCUSSION

As previously discussed, the first step of cloud computing adoption is the end user choice: when a customer needs to decide if he can entrust a particular CSP or not. This becomes way more relevant when the decision involves confidential or sensitive data.

For what concerns policy based models, in particular those which rely on SLAs, a major concern can be identified since SLAs rarely focus on characteristics such as security and privacy, concentrating on elements easier to assess and to monitor too. These last features include the set of service performances such as network bandwidth, services uptime, usage condition of virtual machine, and so forth. Moreover, there is a lack of tools for end users to effectively verify SLA conditions observance. Action that, in many cases, may be performed by a third-party auditor.

About the recommendation based models, some constraints emerge because of the lack of a standardization process: from one side the selection of which

criteria about services provided by a CSP are suitable to be evaluated and then be recommended is tricky; and from the other side, how and by whom a third-party auditor could be professionally certified is not always clear.

For reputation based models, the main limitation is usually the improbable chance to retrieve a huge number of customers to evaluate the CSP, giving a specific rate, for a wide set of complex and detailed criteria. So, in this case, more efforts need to be focused on criteria definition. Moreover, cloud customers do not get any kind of reward for giving their feedback, which is another important challenge for reputation based approaches.

What arises from the presented scenario is that management, mitigation and solving of presented limitations, through the definition of complex trust models, can actually represent the key enabler to boost cloud computing adoption, where constrained because of trust reasons. A correct and wise definition of trust models can surely help customers in the selection process of the CSP that is providing more trustworthy services.

5 CONCLUSIONS

After giving an exhaustive analysis of the origin of trust relationships management and its relevance in the cloud computing scenario, we presented major contributions to address the issue. Actually, cloud computing environment still presents trust issues as an ambiguous area, representing a barrier to cloud adoption for particular real cases. A higher trust can attract customers that currently are avoiding cloud solutions because they are afraid for their data and seeking a greater confidence level. The lack of a commonly reliable and efficient trust evaluation system is to consider a major issue. Several trust models have been proposed and discussed, but what is missing is an accepted criteria to evaluate the effectiveness of such models for a cloud computing scenario.

As future work, it could be of particular interest realize a systematic literature review on trust models, also considering accountability (Pearson, 2011; Jaatun et al., 2014): the work might settle an exhaustive analysis of the trust management scenario for cloud paradigm. Furthermore, another important issue to address is represented by the trust evaluation and definition of trust models for multi-cloud environments. In this case, the assessment of trustworthiness of multi-cloud service providers is more complex and may be achieved with different approaches compared to single-cloud scenario.

REFERENCES

- Abramson, D. (2006). Intel virtualization technology for directed i/o. *Intel technology journal*, 10(3):179–192.
- Alhamad, M., Dillon, T., and Chang, E. (2010). Sla-based trust model for cloud computing. In *Network-Based Information Systems (NBIS), 2010 13th International Conference on*, pages 321–324. IEEE.
- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.
- Blaze, M., Feigenbaum, J., and Lacy, J. (1996). Decentralized trust management. In *Proceedings., 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE.
- Chakraborty, S. and Roy, K. (2012). An sla-based framework for estimating trustworthiness of a cloud. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 937–942. IEEE.
- Chen, R., Zhao, X., Tang, L., Hu, J., and Chen, Z. (2007). Cuboidtrust: a global reputation-based trust model in peer-to-peer networks. In *Autonomic and Trusted Computing*, pages 203–215. Springer.
- Cloud Industry Forum (2014). Cloud uk: Uk cloud adoption snapshot & trends for 2015, the normalisation of cloud in a hybrid it market. http://cloudindustryforum.org/downloads/whitepapers/CIF_WP_14.pdf. Accessed: 2014-12-15.
- Cloud Security Alliance (2011). Cloud security alliance - security, trust & assurance registry (star). <https://cloudsecurityalliance.org/star/>. Accessed: 2014-12-15.
- Cloud Security Alliance (2013). The notorious nine cloud computing top threats in 2013. https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf. Accessed: 2014-12-15.
- Cloud Security Alliance (2014a). Cloud security alliance - cloud controls matrix (ccm). <https://cloudsecurityalliance.org/research/ccm/>. Accessed: 2014-12-15.
- Cloud Security Alliance (2014b). Cloud security alliance - consensus assessments initiative questionnaire (cai). <https://cloudsecurityalliance.org/research/cai/>. Accessed: 2014-12-15.
- Cloud Security Alliance (2014c). Cloud security alliance website. <https://cloudsecurityalliance.org/>. Accessed: 2014-12-15.
- Derbas, G., Kayssi, A., Artail, H., and Chehab, A. (2004). Trummar - a trust model for mobile agent systems based on reputation. In *Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on*, pages 113–120. IEEE.
- European Commission (2011). Attitudes on data protection and electronic identity in the european union. http://ec.europa.eu/public_opinion/archives/ebs/ebs_359_en.pdf. Accessed: 2014-12-15.
- Firdhous, M., Ghazali, O., and Hassan, S. (2011a). A trust computing mechanism for cloud computing. In *Kaleidoscope 2011: The Fully Networked Human? - Innovations for Future Networks and Services (K-2011), Proceedings of ITU*, pages 1–7. IEEE.
- Firdhous, M., Ghazali, O., and Hassan, S. (2011b). A trust computing mechanism for cloud computing with multilevel thresholding. In *Industrial and Information Systems (ICIIS), 2011 6th IEEE International Conference on*, pages 457–461. IEEE.
- Firdhous, M., Ghazali, O., and Hassan, S. (2012a). Applying bees algorithm for trust management in cloud computing. In *Bio-Inspired Models of Networks, Information, and Computing Systems*, pages 224–229. Springer.
- Firdhous, M., Ghazali, O., and Hassan, S. (2012b). Trust management in cloud computing: A critical review. *arXiv preprint arXiv:1211.3979*.
- Gambetta, D. (1988). Trust: Making and breaking cooperative relations.
- Grandison, T. and Sloman, M. (2000). A survey of trust in internet applications. *Communications Surveys & Tutorials, IEEE*, 3(4):2–16.
- Habib, S. M., Ries, S., and Muhlhauser, M. (2011). Towards a trust management system for cloud computing. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 933–939. IEEE.
- Han, S.-M., Hassan, M. M., Yoon, C.-W., and Huh, E.-N. (2009). Efficient service recommendation system for cloud computing market. In *Proceedings of the 2nd international conference on interaction sciences: information technology, culture and human*, pages 839–845. ACM.
- Huang, J. and Nicol, D. M. (2013). Trust mechanisms for cloud computing. *Journal of Cloud Computing*, 2(1):1–14.
- Intel Corporation (2010). Enhanced data protection with hardware-assisted security - intel trusted execution technology. <http://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/malware-reduction-general-technology.html>. Accessed: 2014-12-15.
- Jaatun, M. G., Pearson, S., Gittler, F., and Leenes, R. (2014). Towards strong accountability for cloud service providers. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 1001–1006. IEEE.
- Kanwal, A., Masood, R., Ghazia, U. E., Shibli, M. A., and Abbasi, A. G. (2013). Assessment criteria for trust models in cloud computing. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCoM), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages 254–261. IEEE.
- Kong, D. and Zhai, Y. (2012). Trust based recommendation system in service-oriented cloud computing. In *Proceedings of the 2012 International Conference on Cloud and Service Computing*, pages 176–179. IEEE Computer Society.

- KPMG International (2013). Breaking through the cloud adoption barriers. <http://www.kpmg.com/Global/en/IssuesAndInsights/ArticlesPublications/cloud-service-providerssurvey/Documents/cloud-service-providerssurvey.pdf>. Accessed: 2014-12-15.
- Krauthelm, F. J., Phatak, D. S., and Sherman, A. T. (2010). Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing. In *Trust and Trustworthy Computing*, pages 211–227. Springer.
- Kumar, V., Chejerla, B., Madria, S., and Mohania, M. (2013). A survey of trust and trust management in cloud computing. *Managing Trust in Cyberspace*, page 41.
- Li, W. and Ping, L. (2009). Trust model to enhance security and interoperability of cloud environment. In *Cloud Computing*, pages 69–79. Springer.
- Marudhadevi, D., Dhatchayani, V. N., and Sriram, V. S. (2014). A trust evaluation model for cloud computing using service level agreement. *The Computer Journal*, page bxu129.
- Mell, P. and Grance, T. (2011). The nist definition of cloud computing.
- Nissenbaum, H. (1999). Can trust be secured online? a theoretical perspective.
- Noor, T. H. and Sheng, Q. Z. (2011). Trust as a service: A framework for trust management in cloud environments. In *Web Information System Engineering–WISE 2011*, pages 314–321. Springer.
- Noor, T. H., Sheng, Q. Z., Ngu, A. H., Alfazi, A., and Law, J. (2013). Cloud armor: a platform for credibility-based trust management of cloud services. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2509–2512. ACM.
- Pawar, P. S., Rajarajan, M., Nair, S. K., and Zisman, A. (2012). Trust model for optimized cloud services. In *Trust Management VI*, pages 97–112. Springer.
- Pearson, S. (2011). Towards accountability in the cloud. *Proc. IEEE Internet Computing*, pages 64–69.
- Pearson, S. and Benameur, A. (2010). Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (Cloud-Com), 2010 IEEE Second International Conference on*, pages 693–702. IEEE.
- Perez, J. M. M., Bernabe, J. B., Calero, J. M. A., Clemente, F. J. G., Perez, G. M., and Skarmeta, A. F. G. (2014). Taxonomy of trust relationships in authorization domains for cloud computing. *The Journal of Supercomputing*, pages 1–25.
- Rizvi, S., Ryoo, J., Liu, Y., Zazworsky, D., and Cappeta, A. (2014). A centralized trust model approach for cloud computing. In *Wireless and Optical Communication Conference (WOCC), 2014 23rd*, pages 1–6. IEEE.
- Rousseau, D. M., Sitkin, S. B., Burt, R. S., and Camerer, C. (1998). Not so different after all: A cross-discipline view of trust. *Academy of management review*, 23(3):393–404.
- Sanchika Gupta, P. K. and Abraham, A. (2013). Cloud computing: Trust issues, challenges, and solutions. *Managing Trust in Cyberspace*, page 13.
- Sato, H., Kanai, A., and Tanimoto, S. (2010). A cloud trust model in a security aware cloud. In *Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on*, pages 121–124. IEEE.
- Singh, S. and Chand, D. (2014). Trust evaluation in cloud based on friends and third party’s recommendations. In *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, pages 1–6. IEEE.
- Solhaug, B. and Stølen, K. (2012). Uncertainty, subjectivity, trust and risk: How it all fits together. In *Security and Trust Management*, pages 1–5. Springer.
- Tajeddine, A., Kayssi, A., Chehab, A., and Artail, H. (2006). *PATROL-F - A comprehensive reputation-based trust model with fuzzy subsystems*. Springer.
- Tajeddine, A., Kayssi, A., Chehab, A., and Artail, H. (2007). Patrol: A comprehensive reputation-based trust model. *International Journal of Internet Technology and Secured Transactions*, 1(1):108–131.
- Wang, T., Ye, B., Li, Y., and Yang, Y. (2010a). Family gene based cloud trust model. In *Educational and Network Technology (ICENT), 2010 International Conference on*, pages 540–544. IEEE.
- Wang, T., Ye, B., Li, Y., and Zhu, L. (2010b). Study on enhancing performance of cloud trust model with family gene technology. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 9, pages 122–126. IEEE.
- Wang, W., Zeng, G., and Yuan, L. (2006). Ant-based reputation evidence distribution in p2p networks. In *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*, pages 129–132. IEEE.
- Wang, X., Su, J., Hu, X., Wu, C., and Zhou, H. (2014). Trust model for cloud systems with self variance evaluation. In *Security, Privacy and Trust in Cloud Systems*, pages 283–309. Springer.
- Xiong, L. and Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *Knowledge and Data Engineering, IEEE Transactions on*, 16(7):843–857.
- Zhu, H., Bao, F., and Deng, R. H. (2003). Computing of trust in distributed networks. *IACR Cryptology ePrint Archive*, 2003:56.

Towards Dynamic QoS Monitoring in Service Oriented Architectures

Norman Ahmed^{1,2} and Bharat Bhargava¹

¹*Department of Computer Science, Purdue University, 305 N. University St., W. Lafayette, IN 47906, U.S.A.*

²*Air Force Research Laboratory/RIS, 525 Brooks Rd, Rome, NY 13441, U.S.A.
{ahmed24, bbshail}@cs.purdue.edu*

Keywords: Service Oriented Architecture, Web Services, Aspect Oriented Programming, Web Services Security, Cloud Computing, Quality of Service.

Abstract: Service Oriented Architecture (SOA) is an architectural style that provides agility to align technical solutions to modular business Web Services (WS) that are well decoupled from their consumers. This agility is established by interconnecting WS family of standards specification protocols (commonly referred to as WS-* (WS-star)) to enable security, ease of service interoperability and orchestration complexities when extending services across organizational boundaries. While orchestrating services or chaining services in varying ways to satisfy different business needs, on highly scalable cloud platforms is undeniably useful, it is increasingly challenging to effectively monitor Quality of Service (QoS), especially, service response time. This is due to a) lack of proper formulation of the WS-star interconnections mechanisms, and b) the transient performance behaviour intrinsic to the heterogeneity of the hardware and shared virtualized network and IO resources built on the cloud platforms. We present an analysis of WS-star standards, classifying and discussing their inter-dependencies to provide a basis for QoS monitoring context on protocol formulation. We then illustrate a practical implementation of a dynamic QoS monitoring mechanism using runtime service instrumentation with Aspect Oriented Programming (AOP). Preliminary evaluations show the efficiency of computing QoS on a transient performance cloud platform.

1 INTRODUCTION

Service Oriented Architecture (SOA) is the architectural style that provides agility to align technical solutions to modular business Web Services (WS) that are well decoupled from their consumers in the cloud environment. Built on a Virtualization of heterogeneous hardware and software stack on a SOA-based architecture as its technical foundation, cloud computing is a computing model that enables socio-economic benefits due to its on demand computing resource availability.

In this computing model, service providers and consumers are typically decoupled by means of common universal registries known as Universal Description Discovery and Integration (UDDI) and mediation mechanisms. Service capabilities, interface options, Quality of Service (QoS), and security constraints are described in the Service Level Agreement (SLA) (Overton, 2002) that is typically published in the UDDI.

The SLA document represents a contractual

agreement for obligating the service provider to comply both functional and non-functional parameters of the registered service. The non-functional parameters are QoS attributes, such as service response time and service up time (i.e. 95%-99.999%) that are not known by the consumers before runtime (Erl, 2005) nor by the provider when orchestrating variable services to satisfy different business needs.

To ease the interoperability complexity and security concerns, especially for web services, SOA encourages the use of WS-* standardized specification, referred to WS-star. The forefronts of these protocol specifications are the ones used for data transport (i.e. SOAP/HTTP(s), *WS_Security*, and *WS_SecureConversation*) message level security. Typically, services are developed and deployed by multiple software designers and system integrators without prior knowledge of their effective protocol interconnections when service are orchestrated, the process of chaining services in various ways to satisfy different business needs.

Due to the magnitude of the available standards, chained services have higher chance of overlapping

some functionality, especially security functionalities, that hinder the overall QoS advertised in the published SLA. Coupled with the transient performance behaviour inherent in cloud platforms, further complicates this mixture of standard-based design and contractual compliance requirements to guarantee QoS.

Consider a realistic scenario where two or more orchestrated services deployed in the cloud that implement *WS_Security* to enforce encryption and digital signatures for both inbound and outbound traffic. The overall response time across the chain will be highly impacted due to the potential security functionalities overlap across the services. The main reason is that each service performs encryption and digital signature, which is typically a performance hog. One alternative solution in this case is the use of *WS_SecureConversation*. However, detecting such overlap is increasingly challenging due to the nature of these services' development and deployment by multiple teams in different times. Typically, a Business Process Execution Language (BPEL) is used during orchestration to either determine response time by waiting till response is received or configure it with a proper timeout. Note that these response time evaluations are statically performed in nature.

In addition, there is transient variable performance behaviour of the clouds' VM network and IO interfaces due to multi tenant resource sharing (Mei, et. Al. 2013). For example, over 300 million test cases conducted on nine cloud providers over seven days (Alistair, 2011) have shown *performance time-of-the-day variability* in virtualized environments. Later studies (Zhonghong, 2012) showed such transient performance behaviour is due to the hardware heterogeneity that the cloud is built of. Therefore, it is prudent to dynamically uncover QoS friendly alternatives at runtime to improve service response time, thus, the main objective of our work.

There is a large array of research that addresses WS performance issues; to name a few, some QoS monitoring research have been designed around service selection (Fung, 2005), (Tian, 2004) composition (Mietzner, 2010), (Fung 2005), and dynamic soft QoS guaranteeing (Abdelzaher, 1999). An area that has been substantially overlooked and poorly studied is the understanding of the underlying WS-* standard specification behaviour under the cloud, especially, regarding service response time for web services.

In this work, we propose a dynamic QoS monitoring scheme on SOA-based services on

virtualized shared cloud platforms. The goal is to capture the improper protocol formulation and the underlying platform performance variations to effectively compute service response time without any modification to the service code to improve hard QoS guaranteeing on virtualized environments.

In this paper, we present analysis of WS-* (WS-star) by classifying and discussing their interdependencies to show QoS impacts on improper protocol formulation. We then illustrate dynamic QoS monitoring mechanisms in a widely adopted service container (JBoss). Thus, our contribution is two fold:

- We developed an effective scheme for dynamically monitoring orchestrated services and computing service response time in cloud environments without service code modification or recompilation.
- While the proposed instrumentation scheme is designed for QoS monitoring, it can also be used to detect malicious service in the chain, simply, by instrumenting the method calls that reach beyond its intended service end point.

The rest of the paper is organized as follows. Section 2 gives a brief overview of SOA ecosystem with especial emphasis on web services. We then discuss WS-security protocols and their interdependencies in section 3. We show our proposed approach in section 4 followed by the implementation and experimentation to illustrate the effectiveness of our approach in section 5 and the related work in section 6. Finally, section 7 provides the conclusion and future work.

2 SOA ECO SYSTEM

SOA is an architectural style that promotes a high degree of service decoupling and rapid system development and deployment that span across traditional organization boundaries. The traditional SOA triangle paradigm consists of a service registry (i.e. UDDI), a service provider and a consumer as depicted in Figure 1.



Figure 1: SOA Triangle System Model.

At a high level, web service (WS) is an approach of building web accessible services where the service

providers publish/register their service in the UDDI registry and the service consumers discover and invoke it. The two wide spread paradigms for building services compliant with WS protocols are a Representational State Transfer (REST) (Erl, 2012), referred to RESTful services, and the Simple Object Access Protocol (SOAP)-based services (discussed next).

WS are built on standard specifications to facilitate their integration and secure execution. The core of the WS architecture (WS Architecture, 2002) outlines a set of service characterization that enables these complex functionalities to co-exist. However, the actual specifications of the standards have been collaborated and authored by many organizations such as; W3C, OASIS, OMG, IBM, Microsoft, Oracle, and xmlsoap.org, which makes difficult to fully realize the goals of their interactions.

There has been a considerable research effort that addresses the magnitude of the available standards, their cross-referencing design and development difficulties. For example, in (Gamble, 2011), authors proposed a Security Meta-Language for guiding the formulation of secure messages in WS architecture that model the security relevant portions of the standard for their consistent, comprehensive, and correct applications.

Others have addressed this through the use of enterprise-level integration (i.e. Apache Camel), mediation (i.e. Enterprise Service Bus), and Orchestration (i.e. BPEL) tools. However, dynamically monitoring these critical protocol functionalities over transient performance platforms has not been sufficiently addressed in these tools and in a generic fashion.

2.1 RESTful Services

The RESTful Services paradigm is a lightweight service implementation scheme that avoids preserving service state and the use of the underlying message level security. In other words, the traditional encryption and digital signatures are not employed in this service model due to its computational and bandwidth requirements. RESTful services are stateless services where responding in a timely manner to every service request is critical, thereby widely used in non-critical applications such as; gmail access, facebook updates, amazon consumer interactions, etc.

A transport security layer (TSL) or SSL over HTTP (https) is typically used to secure RESTful services. Such security solutions are sufficient for point-to-point connection oriented where a service

call is authenticated and securely responds to the request. However, this point-to-point security solutions are ill suited in orchestrated/chained service interactions where a service request from a consumer has to reach out to other services in which these services further reach other services in the chain that are possibly in different domains in order to respond to such request.

To remedy these limitations, the use of message-level security is introduced in the standard protocols such as: *WS_Security*, *WS_SecureConversation*, and *WS_Policy*. The key idea of message level security is to structure and wrap the message (both the request and response) by sealing it in an envelope (SOAP) and associating it with security attributes (saml token) to safeguard its access and on transit.

2.2 SOAP-based Services

SOAP-based services provide granular message level security using WS-* family of protocols in which WS-Security is at the forefront. Cryptographic and digital signature techniques are the core of protecting SOAP messages from attacks. As a consequent, this introduces a performance overhead to the services (Liu, 2005). As the services are orchestrated, these performance overheads increase in the order of magnitude due to the overlaps of the security functionalities. Detecting these overlaps of such critical security functionalities to improve QoS is the focus of our work.

In order to effectively illustrate the interconnection of the performance-degrading protocol formulation and avoid hiding the concept in a myriad of protocol standards, we limit our protocol interdependency analysis (discussed next) to only those protocols that impact QoS, specifically policy enforcement and message level security protocols, confidentiality and integrity.

3 WEB SERVICE PROTOCOL INTERDEPENDENCIES

WS decoupling is typically achieved by means of common registries known as Universal Description and Discovery Integration (UDDI). Services deployed in UDDI are discoverable through either WS Application Language (WADL) or WS Description Language (WSDL) standard specifications as depicted in Figure 2 (top left box), and access control protocols (bottom left box).

WADL and WSDL are the two defacto standards for defining web service capabilities. These include: service URI, services, security capabilities, and QoS attributes using *WS_PolicyAttachment* for encryption, signatures, policies, and *WS-Addressing* for end point service response delivery. Discovery and access control protocols have no impact on QoS, therefore, in this work; we only give special emphasis on confidentiality and integrity protocols.

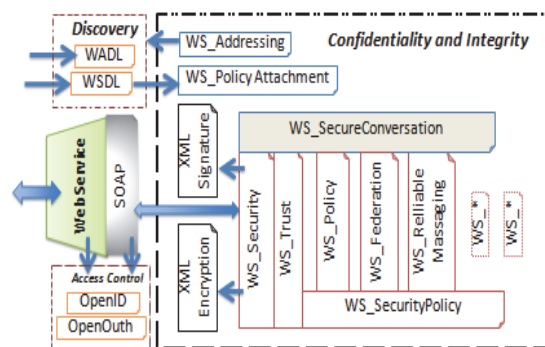


Figure 2: Anatomy of End-to-End Web Service Security Protocols - Service Discovery (top left box), Access Control (lower left box), and Confidentiality and Integrity for message level security (right box).

WS-Security is the core of WS-star protocol for confidentiality and integrity of the service. The WS-Security standard describes the security attributes of service and task delegation between services to facilitate secure authentication, authorization and invocations. Each new security concept or interface specification defined in WS-Security brings additional WS-* family of standards which play a significant role in expressing a web service's security posture.

For example, bridging communication between secure environments require protocols to specify cross-domain access controls. The Security Assertion Markup Language (SAML) provides the authentication and authorization among and across services, even in different security domains (Oasis-open, 2007), and eXtensible Access Control Language (XACML) provides the security policy enforcements for the authorizations that cross the organizational boundaries (Oasis-open, 2012).

Further, WS-Trust is required to broker authentication information, however, WS-Trust does not describe the security functionality of services and its capacity to fulfill the security needs. Instead, it delegates to *WS-SecurityPolicy* to describe the security policy which in turn uses WS-Policy. WS-Policy exchanges policy decisions and enforcement capabilities for every request, introducing more

latency for QoS constraint services, especially if such capabilities deployed in a remote service domains.

In addition, WS-Security defines XML-Signature and XML-Encryption standards for digital signatures and encryption of XML documents to ensure the integrity of the exchanged SOAP messages/envelope. The more security capabilities added the more standard protocols needed. Thus, the SOAP message size increases, which consequently require more bandwidth and computationally intensive operations in encryption, signature, and verifications in which contribute to other QoS issues, especially when services are deployed across cloud domains or consumers with resource constrained devices (mobile).

QoS violations are imminent when improper protocol formulation is coupled with the transient performance behaviour of the underlying platform. A recent study (Zhonghong, 2012) shows that the virtualized heterogeneous hardware built on the cloud has performance variations that can reach up to 60% between instances. Thus, dynamically intercepting and monitoring orchestrated services on such platforms are crucial in order to improve QoS guarantees and consequently prevent SLA violations.

4 SYSTEM MODEL

A motivating example of cross-domain service orchestration scenario is depicted in Figure 3 below.

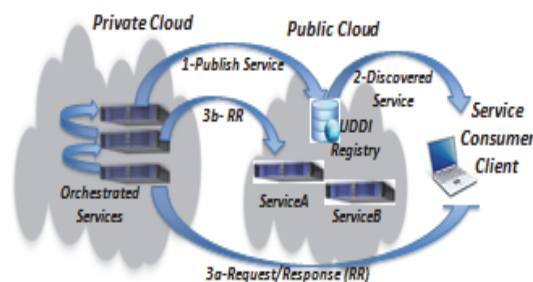


Figure 3: High-level architecture for orchestrated services across private and public clouds.

The high-level architecture above depicts a typical orchestrated service deployment across security domains, public and private. The top arrows marked (1-Publish Service) and (2-Discovered Service) show the service registration flow to the UDDI by the service provider where then the consumer client (depicted as the laptop) discovers that service. The client invokes that service as shown by the arrow-

marked (3a-Request/Response (RR)) and gets back a response. The arrow marked (3b-RR) between the clouds show the cross-domain service invocation that jointly satisfy the consumers' request. Note that all arrows represent a bio-directional data flow.

Chaining services in such environments is typically configured using BPEL. However, once services are deployed in the service container or application servers (eg. Jboss, Oracle, IBM Glassfish, etc.), these configurations are static, and thereby, fail to adapt to the changes of the underlying cloud platforms. Computing service response time in such setting typically requires reconfiguring or even re-designing the services.

We approached this problem by deploying interceptors in the service containers to seamlessly collect service information at runtime and compute response times while considering the transient behaviour of the deployed cloud platform. Service information can then be analyzed by machine learning to predict future QoS attributes, dynamically update SLA in the registry, or even migrate services instances to cloud platforms that are experiencing less performance issues in different regions. In this work, we focus on the detection scheme only.

4.1 QoS Criteria

There are several non-functional QoS metrics categories and service performance attributes in SOA-based WS. In this work, we only consider WS performance, specifically service response time for orchestrated services on cloud platforms.

4.2 Approach Overview

Most QoS attributes in SOA are not a one-size-fit-all for all consumer requests. A priori knowledge of any given QoS attribute for the prospective consumer is difficult to predict (Erl, 2005). Several QoS monitoring approaches offered solutions that improve QoS over the years. However, none have addressed the impact of the overlapping security protocols due to their criticality of the service protection coupled with the performance variability of the underlying cloud platforms.

The basic idea of our approach is to non-intrusively instrument services without introducing overhead. Our design is based on two steps, detection and aggregation. We use Aspect Oriented Programming (AOP), a dynamic application instrumentation framework first introduced in (Kiczales, 1997). AOP allows service code

instrumentation without modifications or recompilation of the code. The instrumented data collected/detected at runtime from each service is then forwarded to the QoS auditor web service (referred as QAudit) to aggregate and then compute response time.

4.3 Service Instrumentation with AOP

Typically, collecting accurate QoS information at runtime is achieved by inserting general purpose logging statements in pre-compile time and during service composition. QoS metrics can then be based on the aggregate of these logs. Such techniques are inefficient and ill suited in cloud computing platforms due to the performance variability behaviour that are not under the control of the service provider. Since accurate QoS attributes cannot be predicted during service registration, dynamic service instrumentation is critical.

We achieve such dynamicity with AOP. A basic AOP model defines two instrumentation primitives known as *pointcut designators* (PCD) and *advice*. The PCD's are typically points in the program where inserting instrumentation is not too hard, for example, method calls are very often used as one of the fundamental PCD. These PCD's are simple instrumentation primitives that can gather critical information without any modifications to the code.

On the other hand, the *advice* is the point where an aspect to be instrumented can be weaved in. The result of PCD and advice generated will then forwarded to externally configured component, in our case, QAudit. QAudit web service evaluates the best QoS metrics under that given cloud platforms performance behaviour or overlapping security functionalities on the services, in which the service provider can take any action necessary such as; either update the SLA for the prospective users, reconfigure security protocols or project future QoS metrics of the given time of the day.

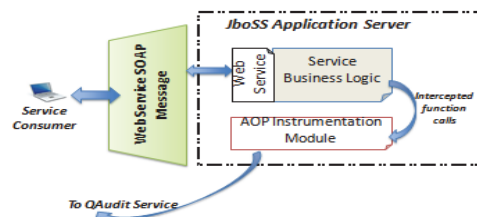


Figure 4: Service Anatomy and AOP Instrumentation Module inside the Jboss service container or app server.

AOP enables user level service interception capabilities within application servers. As depicted

in Figure 4, we used an AOP plugin as a module in the Jboss Application server where our services are deployed, known as JbossAOP (JbossAOP, 2003). JbossAOP instruments services deployed in the service container by intercepting the execution of all aspects of the program, such as specific object on the program, a function parameter values, or method calls within or across program calls.

The performance overhead of the AOP depends on the knowledge of the application (Alexanderson, 2010). For QoS monitoring, the overhead is proportion to the number of the interception points within the services. To limit such overhead, we only intercept WS-Security related function calls, specifically, prior and post encryption, and signature operations in which are negligible when tested in public cloud environment as shown in our previous work (Azarmi, et. al. 2012). Note that one can also instrument communication methods if needed to uncover rogue/compromised service reaching outside its intended endpoints.

4.4 QoS Auditor Web-service

As depicted in service anatomy diagram in Figure 4 above, the service container enables hooks to instrument the services' business logic where the instrumented data can then be sent to the listening service, QoS Auditor (*QAudit*) web service. The QAudit receives the pre and post *WS_Security* function call timing information collected from the diverse orchestrated services under the current performance of the services' environment (VM's). For example, some services are deployed in cloud platforms that are built on different hardware, hypervisor, and possibly running *VM* migration and load balancing algorithms by the cloud provider to accommodate between the tenants.

4.5 QoS Monitoring in Orchestrated Services

The WS Business Process Execution Language (WSBPEL) defines the orchestration of WS standard language for service chaining and execution. Identifying performance bottlenecks in orchestrated services from multiple providers within BPEL engines is a challenging task given the dynamicity of the cloud platforms that's not known a priori.

As described in the previous section, AOP instruments services deployed in the service container by intercepting the execution of all aspects of the program (i.e. method calls) across program calls. Since orchestrated services are also program

calls across domains, AOP can effectively intercept orchestrated WS. We will describe our implementation approach in the next section.

5 IMPLEMENTATION AND EXPERIMENTATION

We are interested in computing service response time for secure web services orchestrated across cloud platforms (public/private) as illustrated in the high-level architecture in Figure 3. In this section, we discuss our prototype and show the preliminary evaluations on private cloud deployments, and the proposed QoS computations scheme.

5.1 Experimental Setup

Our experimental cloud platform uses a private cloud built on *OpenStack*, a cloud management software stack, on a cluster of 4 machines (Dell Z400) with Intel Xeon 3.2 GHz Quad-Core with 8GB of memory. At a high-level, *OpenStack* consists of a controller and computing management applications. We divided our four machines into one controller node and 3 compute nodes. As the name implies, the controller node is to simplify cloud platform management by enabling on demand elasticity, i.e., provision/de-provisioning *VM* instances, adding/removing hardware and instantly making it available in the computing resource pool.

The three compute nodes allow us provisioning 20 virtual CPU's (vCPU) in which we assigned 10 small *VM* instances, 2 vCPU per instance for service deployments. We used a total of 10 VMs with Ubuntu Linux for service consumer (clients) and secure services in all of our experiments.

5.2 Implementation

We developed a CXF-based secure web services (*WS_Security* and *WS_SecureConversation* enabled) and deployed in Jboss application server. The integration of AOP with Jboss container was done using JbossAOP (JbossAOP, 2003) library, a pluggable user specified instrumentation module for Jboss application servers. We leveraged AspectJ (AspectJ, 2001), a stand-alone Java implementation of AOP, as the service instrumentation algorithms for intercepting the WS-Security method calls.

5.3 Preliminary Results

It has been previously reported that performance difference between *WS-Security* and *WS_SecureConversation* in web services are in the order of magnitude higher in *WS-Security* (Liu, et.al. 2005). To illustrate in the context of service orchestration under the transient performance behaviour of the cloud, we configured three secure services that implement *WS_Security* with 3 others that implement *WS_SecureConversation* with the same logic, a secure weather report, deployed in a Linux virtual machines described above.

We orchestrated the 3 services with different configurations while assuring the security of the service. Service configuration is application and domain specific, thus, to illustrate the basic concept; we chained and invoked services in the following format:

$$\begin{aligned} Req^1 &\leftarrow S^{ws} \leftarrow S^{ws} \leftarrow S^{ws} \leftarrow S^{ws} \leftarrow S^{ws} \leftarrow \dots \\ Req^2 &\leftarrow S^{ws} \leftarrow S^{sc} \leftarrow S^{ws} \leftarrow S^{sc} \leftarrow \dots \\ Req^n &\leftarrow \dots \end{aligned}$$

The requests Req^n interacts with service S^{ws} implemented with *WS_Security* and then S^{sc} implemented with *WS_SecureConversation* and so on.

To mimic the performance variability of the cloud platforms, services requests and responses were performed while the system is running *cpu* and memory intensive applications. We observed the system performance using the built in Ubuntu system monitor (*krell*) showing a load over 50%-70% usage of the memory and *cpu*. The service response times received by *QAudit* service, when aggregated, ranged between microseconds to seconds; thus, clearly show QoS impacts on security function overlaps.

These observations show the non-intrusive way of computing QoS in cloud platforms. However, the actual results may vary depending on the service logic and other factors when expended into the public cloud deployments, thereby, considering it in our future work.

6 RELATED WORK

To the best of our knowledge, there is no in-depth analysis of WS-star protocol formulation in SOA in the context of QoS monitoring that reflect the transient performance behaviour of the underlying cloud platforms. Thus, we divide our related work section into two parts; we first discuss works in WS-* performance improvements and next we provide

QoS Management tools and techniques that are relevant to our work.

6.1 WS*- Performance Improvements

There are large volumes of research that employ different methods to address performance improvements on web-services. To name a few: SOAP header envelope reduction techniques, efficient XML parsing and compression methods, and binary and canonicalization techniques.

With the rise of business heterogeneity, orchestrated services pose further callings for selecting and complying with an accurate advertised QoS attributes, especially service response time. As these schemes have set the foundation of WS performance improvement, our approach was inspired by such mechanisms and further extended to dynamically monitor orchestrated services in a virtualized environment.

6.2 QoS Management

QoS management can be classified into three categories: resource allocation, service composition, monitoring and fine-tuning QoS parameters within the services. In this work, we focused on the latter two. It's intuitive to see that an effective resource sharing can aid QoS guarantees; moreover, service composition or selection also plays a critical role in such guarantees.

To highlight some studies in this category, early works, such as (Abdelzaher and Shin, 1999), proposed a virtual service that enables the selection of multiple deployed concrete services depending on the users' QoS interest. A set of cooperative autonomous agents that enable optimal web service composition is proposed in (Brahmi, 2013). Within the context of service selection, similar to *QAudit* approach, Q-Peer (Li, et.al. 2007), a distributed QoS registry is proposed to monitor and collect information on running services to assist consumers for the reliability of the service where as we focus on service response time improvements.

It has been noted that the inaccuracy and violations of QoS in various papers and spurred a wide range of research approaches, to name a few; QoS verifications during service registration (Abdelzaher and Shin, 1999), extending UDDI functionalities (ShaikhAli et. al, 2003), introducing new protocol languages to define SLA (Lamanna, 2003), SLA template adjustments (Spillner and Schill, 2009) and new frameworks for dynamic service monitoring and selections in a realistic

environment (Tian, et. al, 2004). Along the lines of the WS protocol research, a modification of WS-Agreement protocol to enable dynamic run-time renegotiation and SLA adjustments to guarantee QoS when SLA violation is expected to occur is proposed in (Modica, et. al, 2007).

All of the above approaches face adaptability challenges due to the proposed changes required in the protocol standards. Our work can accurately and non-intrusively detects the transient behaviour of the cloud platforms to prevent SLA violations without modifying the service code or the standard protocols. Furthermore, our work will complement the works of fine-tuning QoS parameters for efficient service composition, selection and monitoring schemes to maximize QoS and prevent SLA violations.

7 CONCLUSIONS

Guaranteeing hard QoS on orchestrated web-services in SOA and virtualized cloud platforms are increasingly challenging due to security critical functionality overlaps and the transient performance behaviour of such platforms. In this paper, we developed an effective mechanism to dynamically monitor orchestrated services and compute service response time while considering the underlying performance behaviour of the cloud platforms.

We implemented our proposed approach with Aspect Oriented programming (AOP) and illustrated with a practical scenario to validate our design using three secure services deployed in a private cloud. In our future work, we consider experimental traces over periods of time in our private with public (i.e. Amazon) cloud instances deployed in different geographic locations.

ACKNOWLEDGEMENTS

Authors would like to thank Jim Hanna at AFRL for setting up the experimental platform, and special thanks to the reviewers for their valuable feedback that made this paper more readable.

REFERENCES

- Abdelzaher, T. F., & Shin, K. G., 1999. QoS Provisioning with qContracts in Web and Multimedia Servers. *In the 20th IEEE Real-Time Systems Symposium*.
- Alexanderson, R., Ohman, P., and Karlson, J., 2010. Aspect Oriented Implementation of Fault Tolerance: An assessment Overhead. *In Computer Safety, Reliability, and Security*. Lecture Notes in Computer Science, Volume 6351, pp 466-479.
- Alistair C., 2011. Cloud Performance From the Users Prospective. <http://www.bitcurrent.com/download/cloud-performance-from-the-end-user-perspective/>.
- AspectJ, 2001. <http://eclipse.org/aspectj/>
- Azarmi, M., Angin, P., Bhargava, B., Ahmed, N., et al., 2012. End-to-End Security in Service Oriented Architecture, *In SRDS12, the 31st IEEE Int. Symposium on Reliable Distributed Systems*.
- Brahmi, Z., 2013. QoS-aware Automatic Web Service Composition based on Cooperative Agents. *In WETICE, The 22nd IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*.
- Erl, T., 2005. *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall.
- Erl T., et al, 2012. *SOA with REST*, Prentice Hall. 1st ed.
- Fung, C. et al., 2005. A Study of Service Composition with QoS Management. *In ICWS'05, IEEE International Conference on Web Services*.
- Gamble, R. and Baird, R., 2011. Developing Security Meta-language Framework. *In Proceedings of the 44th Hawaii Int. Conference on System Sciences*.
- JbossAOP, 2003. <http://www.jboss.org/jbossaop>.
- Kiczales, et al., 1997. Aspect-Oriented Programming. *In ECOOP'97, Object-Oriented Programming, lecture Notes in CS. Vol. 1241, pp. 220-242*.
- Lamanna, D., Skene, J., and Emmerich, W., 2003. SLang: A Language for Service Level Agreements. In the 9th IEEE Workshop on Future Trends of Distributed Computing Systems. FTDCS.
- Li, F., et al., 2007. Q-Peer: A Decentralized QoS Registry Architecture for Web Services. *In ICSOC'07, International Conference on Service Oriented Computing*. LNCS 4749, pp.145-156.
- Liu, H., Pallikara, S., and Fox, G., 2005. Performance of Web Service Security. *In Proceedings of the 13th Annual Mardi Gras Conference*.
- Mei, Y. et al., 2013, Performance Analysis of Network I/O Workloads in Virtualized Data Centers. *In IEEE Transactions on Service Computing*.
- Mietzner, et al., 2010. Combining Horizontal and Vertical Composition of Services. *In Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*.
- Modica, G., et al., 2007. Dynamic Re-negotiations of SLA in Service Composition Scenarios. *In SEAA07, EuroMICRO conference of Software Engineering and Advance Applications*.
- Oasis-open.org, 2007. Security Assertion Markup Language (SAML). <https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf>.
- Oasis-open.org, 2012. eXtensible Access Control Language (XACML). https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#CURRENT.

- Overton, C. 2002, On the Theory of Internet SLAs. *Journal of Computer Resource Measurement*, (106): 32-45.
- ShaikhAli, A., Rana, O. F., Al-Ali, R., and Walker, D. W., 2003. Uddie: An Extended Registry for Web Services. *In Proceedings of IEEE Workshop on Applications and the Internet*. pp. 85-89.
- Spillner, J., & Schill, A., 2009, Dynamic SLA Template Adjustments based on Service Property Monitoring. In CLOUD'09. *IEEE International Conference on Cloud Computing*. pp. 183-189.
- Tian, M. et al., 2004. Efficient Selection and Monitoring of QoS-aware with the WS-QoS Framework. *In WI'04, IEEE/WIC/ACM International Conference on Web Intelligence*. pp.152-158.
- WS Architecture, 2002. <http://www.w3.org/TR/ws-arch/>.
- Zhonghong, O., et al., 2012. Exploiting Hardware Heterogeneity within the Same Instance Type of Amazon EC2. *In Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing*.

Reality Vs Hype

Does Cloud Computing Meet the Expectations of SMEs?

Katie Wood and Kevan Buckley

*Department of Computer Science, Faculty of Science and Engineering, Wolverhampton University, Wolverhampton, U.K.
{k.wood, k.buckley}@wlv.ac.uk*

Keywords: Cloud Computing (CC), Security, Risk Management, Small and Medium Enterprises (SMEs), Cloud Service Providers (CSP).

Abstract: Small and Medium Enterprises (SMEs) have become a primary target audience for Cloud Service Providers (CSP), such as Amazon and Microsoft to promote their cloud offering. CSP strong marketing campaigns of 'promised' benefits from using their clouds is an attractive offer for SMEs especially where resources are limited and they wish to become more agile and reduce IT costing to be competitive with larger rivals. This paper argues that once SMEs remove the hype surrounding the concept of cloud computing (CC), the reality of significant benefits do not materialize for SMEs. This paper demonstrates, through working with SMEs considering the options of CC that the challenges and risks associated with cloud might actually hinder the business, rather than providing any real long term value.

1 INTRODUCTION

Cloud computing (CC) has gained increased attention and momentum within a short period of time, even those the technology is still very much in its infancy. The increase interest into this form of distributed system has been fuelled by the strong marketing campaigns from Cloud Service Providers (CSP) that have promoted the promised of benefits. It is frequently reported that CC offers a variety of benefits including cost-saving, agility, efficiency, resource consolidation, business opportunities and Green IT (Chang et al., 2010) Even so, cloud is still a young and evolving paradigm that incorporates the evolutionary development of many existing computing technologies. This paper defines hyper surrounding CC as being the extravagant or intensive positive promotion of CC technologies by CSP and through the media. Such extravagant promotions, has lead to SMEs within this study to consider using CC as significant benefits, especially in term of cost saving and improved performance were expected. However the observations and findings of this study, suggest that in reality such expectations have not be achieved.

This study identified there a lack of understanding surrounding the terminology of CC and the changing variable within CC makes it challenging to alignment with SMEs needs.

Concerns surrounding security are noted as being a majority issue for SMEs in terms of establishing their role and the CSPs in protecting the systems.

The remainder of this paper is organized as follows: Section 2 provides a brief outline of the background and rational for undertaking a research project. Section 3, presents an overview of unique features of cloud where benefits are clamed. The section continues through a detailed discussion of the issues and how these limit the chance of any business value of using cloud technology for SMEs, if not understood or assessed during the decision making process. Section 4 outlines the need for risk assessment to be conducted and awareness of risks associated of CC. Section 5 presents considerations as part of a risk assessment that could be part of assessing CC suitable. Section 6 concludes the paper. Finally, section 7 outlines further work that the authors have planned to continue on this project.

2 PRIMARY OBSERVATIONS

The findings outlined in this paper are from research undertaken with SMEs in the West Midlands, UK. SMEs from different industries where selected to take part in a study to access SMEs understanding of CC and what cloud technologies were currently being used or considered. Initially a questionnaire

survey was used to collect data. 50 SMEs in the West Midlands took part in this stage. Businesses were selected based on the fact they heavily used forms of technology in their operations, but were not deemed as a business in the IT sector. The rationale behind this selection was to access people who have an average level of IT usage and experience of IT, who would understand and interact successfully with cloud technology without the need to be an IT expert. The questionnaire survey allowed the authors to gather valuable information and gauge SMEs general understanding of the concept of CC and establish the level of interest and use from the SME community. From the results of stage 1, several SMEs were then invited to take part in a deeper analysis. During this stage semi-structured, in depth interviews were used and acted as the primary data collection method for the project. The SMEs selected were based on the responses from stage one. SMEs selected for this stage were based on the following criteria:

Table 1: Criteria for SME involved in stage 2 of study.

The business was involved in the initial questionnaire stage of the project.
Staff had a fairly good level of IT skills
There was an interest in understanding more about cloud and some evidence of some form of cloud computing has been used.
The business wanted to consider using PaaS in the future.
The business uses websites, email and database applications, which would be suitable applications to use and store within a form of cloud environment.

Each interview was individually conducted between the author and participant and took around 45 minutes. A combination of closed and open questions were selected relating to the business use of IT, the business rationale for considering CC and the participants understanding of use of CC. Results and transcripts from the interviews have not been shared or compromised at any point during the project or discussed in this paper order to ensure confidentiality and participant's anonymity.

This paper outlines some of the responses from the participants from both stage 1 and stage 2 to highlight growing concerns of issues relating to cloud computing and evidence of limited benefits emerging. The findings overall conclude a growing interest in the topic, however when it came to practically using cloud, SMEs either had experienced problems and disappointment through using the technology, or the complexity led to the decision to migrate to cloud being terminated. This

paper outlines several areas where SMEs stated complexity and where potential benefits promoted by the CSP have not materialized to produce overall benefits.

3 CHALLENGES OF CLOUD

3.1 What Actually Is Cloud Computing?

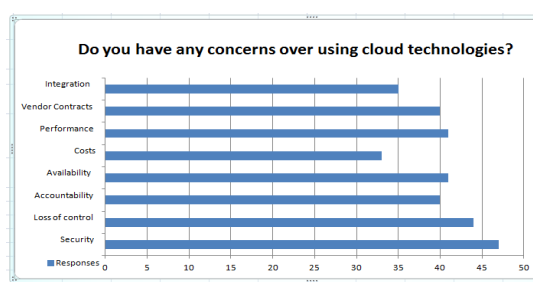
To establish the participants understanding of CC, the first sets of questions asked to participants during the interviews were to assess their understanding of the concept of CC and user experience. All participants involved in this study had heard of the concept CC, so you might wonder why the question "*Do you know what cloud computing is?*" was even asked? According to results from a survey conducted in 2012 (Chang et al., 2010). Participants in that study were asked to explain the concept "the cloud" The majority responded with the view it either referred to an actual cloud (specifically a "fluffy white thing") in the sky or something related to the weather (29 percent). Only 16 percent of participants thought it was related to computer networking to aid storage, access and share data from Internet-connected devices.

It this particular research project with SMEs participants the results from this question concluded that only 30% of the participants could actually provide some clearly definition on what the term means in the context of computing. Surprisingly over half the participants believed they have actual use a form of cloud technology, even those they were not sure what was meant by the term CC. From a security point of view, this statement is concerning as participants are not aware of what technology, services or systems they are actually using and the risk associated.

Given the relatively immature nature of CC and that it is still evolving. It is not surprising that end users and businesses are finding it difficult to understand what is exactly meant by the term cloud computing. There has been work in recent years to establish a benchmark and suitable definition for cloud. Currently however there is still no precise definition (Interworkscloud, 2013) for cloud, which makes it a challenge for businesses to understand the different elements that are required for a successful uptake of CC. This has led to arguments by researchers that the term "cloud computing" is far too broad making it difficult to develop a single and

clear definition. (Wood Katie, 2012) Currently, there are over 20 different definitions. The most regularly used definition is by the National Institute of Standards and Technology (National Institute of Standards and Technologies, 2009) Basically, 'CC' as an umbrella term being applied to different situations and their solutions. It is the next stage in the distributed and shared computing. This form of computing provides a range of computation facilities, storage and scalable functions and services that are accessible anywhere via a connection to the Internet.

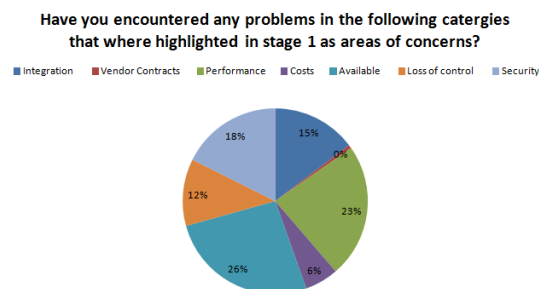
3.2 SMEs Concerns



Graph 1: Results from stage 1 – Questionnaires Shows the range of concerns SMEs have with cloud technologies.

Graph 1 shows the different categories that SMEs had concerns over using cloud technologies. 47/50 participants identified security has their major concern. This was followed by a high proportion of participants stating concerns over loss or control 43/50 and performance 41/50. Concerns highlighted above led to a follow up question directly linked to this during the interview stage in order to see if any of the concerns raised by SMEs have also materialize into real problems for SMEs that had used cloud technologies. Unsurprisingly SMEs had encountered problems in most of the categories previously outline which showed that there is evidence to back up the concerns from SMEs that might still be weighting up the options of cloud. 26% of SME in stage 2 identified that they have encountered security problems and 23% as having experience performance problems. As semi structured questions were used during this stage, some SMEs further explained that the problems they had occurred several times over a short period of time. In once case led to a SME to reconsider using cloud services to store data as the performance of the system led to performance and available problems which was seriously hindering the operational side of the business. Cost of using cloud technologies did not appear to be a concern by SME participants in

this study. There was a consensus from participants that there is a wide option of price plans offered by different CSPs which allowed for flexibly.



Graph 2: Results from stage 2 – Interviews Shows the range of concerns SMEs have with cloud technologies.

3.3 Migration and Section

One of the first challenges for any potential cloud user is dealing with selecting the correct cloud. CC can be classified into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). To further complex the situation, CC can also be offered in different deployment formation - public, private, community and hybrid clouds. Each form of cloud and deployment bring a whole set of different potential benefits and associated risk. Therefore the first challenge is assessing which form of cloud is suitable for that business. In order to benefit from cloud systems and services a business needs to have an understanding of the difference between the forms or clouds as well as looking at several CSPs.

Do you feel there is enough advice, support or training available to SMEs in selecting cloud technologies?



Graph 3: Results from stage 1 – Questionnaires. Shows the percentage of participants that feel there is not enough advice, support or training available for SMEs.

As shown in graph 3. Participants for this study felt strongly that training, education and support on the subject of CC is currently lacking. Participants in this study felt there is not enough advice, support or training in dealing with the starting process over selection and migration of systems and applications and also the legal aspects to ensure their data and rights are protected.

3.4 Compatibility

Linking to the migration to clouds, all businesses in this study highlighted the need for cloud services that are compatible with existing services and application. Participants felt that there is not enough guidelines or support in terms of how suitable clouds are to meet their needs and also their compatibility with exist systems and applications the business might still require. As businesses has already invest in systems prior to considering the option of cloud it is important those systems and applications are not ignored if these are still fit for purpose. There is limited documentation and advice on the matter to aid businesses in finding suitable and available tools and techniques to support this objective. 80% of the participants during stage 1 stated if cloud systems where used, they would still feel current systems and application would have to remain for certain tasks. Therefore this would be increase outgoing costs of maintaining both an existing systems and cloud system, therefore not providing the businesses with any cost saving.

3.5 Security

CSP often claiming that security in the cloud is tighter than in most enterprises (David Binning, 2009) however the following questions need clear and defined answers for CSPs. Will SMEs data be safe in the cloud? What about data protection? What will happen if security threats or breeches occur on the cloud even those the technology has been around for several years now, Gartner warned in 2013 that there are still six major security issues that businesses should tackle when considering cloud adoption. (Warwick Ashford, 2013) Each cloud is unique, offering different benefits and ways to reduce costing. However there are also significant drawbacks of cloud systems, particularly in security. The variation of security threats and possible breaches that the system could encounter, further complicates cloud. [7 Distributed denial of service attacks are on the raise on cloud systems, as clouds host services for different customers on their servers, so it's no surprise these systems are a hot target for cybercrimes. The fact that different businesses and users sharing the cloud space also increases the risk of access errors and leading to data been vulnerable and at risk of being accessed by others. How data is moved across and between CSP and the end user also places greater risk and vulnerabilities.

One major downside of clouds is that the provider has control of the user's data. Users have to

relay and trust their provider will protect their data and privacy. Privacy is an important issue for cloud computing, both in terms of legal compliance and user trust and this need to be considered at every phase of design. (S. Srinivasamurthy and D. Q. Liu, 2010) Privacy has yet to be fully acknowledged as a serious problem by policy makers and CSP. The limited regulations and legislation being enforced on privacy and user protection rights reflect this. According to a recent Cloud Security Alliance Report, insider attacks are the third biggest threat in cloud computing. (Top Threats to Cloud Computing v1.0, 2014) The reasons for this may vary, from users not understanding the system and the configuration processors, through to users who are motivated to create damage and misuse. Administrators and development need to deal with this situation in a more consistent manner across different cloud platforms. Therefore it is essential to access the dynamics of a range of configuration techniques and tools to evaluate and distinguish the impact these issues have on a cloud.

The survey used in stage 1 showed that (75%) of the SMEs stated concerns over employees IT knowledge and felt that employees would have to receive additional education and support in order to use the technology effectively. (50%) of these businesses further stated they are currently not in a position to invest in providing such support for employees at the moment. This further outlines drawback to using cloud technologies that businesses are assuming that high investment in education and training would be required to use these systems, when in fact the role of using clouds could be to simplify certain IT tasks, for example updates security countermeasures.

3.6 Costing

One of most hyped aspect of cloud computing is surrounding cost savings. Yet has stated in early sections of this paper, If SMEs have to continue using existing systems and applications along a cloud system there are no financial benefits. Businesses and individuals considering using cloud, expect appropriately reliable and timely service delivery, easy-to-use interfaces, collaborative support, information about their services, etc.(M. A. Vouk, 2008) Such high exceptions are understandable as the CSP have promoted their cloud service as being able to achieve such goals. All CSP will be affect at some point by downtime, for example during upgrades. There was been recent cases of security breaches in Amazon, Gmail and

Hotmail. The user is often unaware of such problems unless their CSP informs them or they are affected by the security breach. There is currently no legislation in place which states the CSP must inform users of all security breaches on their systems.

3.7 Performance

Clouds promote the benefits of scalability and flexibility for customers as cloud computing shifts everything from local, individual devices to distributed, virtual, and scalable resources, thereby enabling end-users to utilize the systems, storage, and other application resources (which forms the “cloud”) on-demand (Hayes, 2008). The term multi-tenancy refers to the ability to run multiple customers on a single software instance installed on multiple servers. These systems have recently become popular due to the multi-tenancy features within which allows businesses to benefit from reduced costs yet continue to gain access to data and applications. (Wood and Anderson, 2011) Reliance on cloud infrastructure provides issues for the end user in terms of the reliability and availability of the CSP and cloud services. It is crucial that CSPs ensure they meet the privacy requirements of users and legislative requirements. Reports on privacy failure and loss of user data have had serious impact on the creditability of Cloud technology and the overall expansion of cloud services, as well as on the end user. This clearly demonstrates the risk associated with reducing control of own data

4 RISK ASSESSMENT

Given the nature of CC and its key characteristics several risks can be determined. Some of these risks are traditional risk and concerns that are common with any form of networking technology. However, there are also specific risks relating to cloud. Businesses need to understand, analyse, and evaluate important economics and elasticity capabilities of different forms of cloud systems and technologies and providers before making any commitment to CC. Any selection should be based on the suitability for meeting the business requirements and alignment, rather than being motivated with marketing and the desire for the latest technologies. For any business it is important to consider all the options available. In the case of SMEs, it is more critical given their limited budget to get outsider exit advice – i.e. consultancy services to aid the decision

making process, due to financial constraints.

5 CONSIDERATIONS IN TACKLING CLOUD CHALLENGES

5.1 Testing and Evaluating

The development and integration of different system hardware, storage, networks, interfaces, administration and so on, should be carefully planned out. However businesses are not always sure what IT they require in the short and long term. Before committing to any cloud technologies, it is critical that businesses assess and evaluate their business needs as well as exploring the different cloud options available. This might appear time-consuming and costly; however the right decision must be made. There are too many different CSPs available and packages; therefore it is useful to research against the business requirements a few different CSPs to see what is being offered. Several CSPs offer limited access to service as a free trial approach in an attempt to entice users to commit to a contract. These free trials can provide an opportunity for businesses to compare and evaluate different services and functionalities between providers.

5.2 Assurance Measures

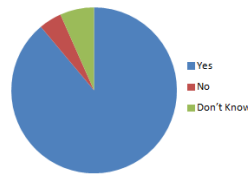
Businesses that wish to explore cloud must seek assurance from the CSP over how potential risks will be prevented. It is important to check CSP reliability reports to determine how often breaches have occurred and the impacts to see if that CSP has a good track record and provide the required level of protection and support. Checking through blogs and internet sites over any reports of major security breaches or problems that other users have experienced can provide valuable insight into the cloud culture.

5.3 Training, Education and Cloud Awareness

Graph 3 shows that SMEs would consider using clouds if more advice, support and training were made available.

There needs to be an inquiry about what the system/services are being offered and the privacy policies that the CSP has. This must be conducted before users commit and hand over their data. A

If there was better advice, support and training available would you consider using this form of technology?



Graph 3: Shows the percentage of SMEs that would consider using clouds if more advice, support and training were made available.

contract which includes the Terms of Service (TOS) and also s Privacy Policies and Service Level Agreement (SLA) will insure a level of assurance for users. This also provides grounds for legal action against the CSP if the provider does not maintain their side of the contract. For example, passing user details on to a 3rd party. It is also essential that users are aware of the data protection laws as their data could be transferred across into regions which are not as strict on data protection. This could result in invasion of privacy. Like all forms of technology, clouds are changing, aspects are being improved and other forms of risks are emerging therefore users need to be aware of changes to their cloud systems. What applications are being updates or removed for example, and how these might impact?

6 CONCLUSIONS

This paper has highlighted that CC often does not meet the expectations of SMEs. Finding suggested this is because SMEs have unrealistic expectations from using CC and the fact that CSP promote a range of benefits which can't be achieved for all. CC is not a one size fit all technology and as every business differ in size, resources and IT experience it is difficult to compare and contracts how CC benefits can be achieved for the masses. Therefore SMEs need to conduct an in depth risk assessment and evaluation of existing systems and CC options in order to access if CC is more suitable to that particular business.

7 FUTURE WORK

Further work and support is required for SMEs to actually deploy a cloud system that can be integration with existing applications and systems. Therefore the author's further work will include explores risks relating to the more technical aspects

of cloud and SMEs role in these, for example configuration management and access rights in a cloud system. Alongside this, a book will be produced which will provide a framework to act as a set of risk and support guidelines for SMEs during the change cycle of migrating to a cloud.

REFERENCES

- Chang, V., Bacigalupo, D., Wills, G. and De Roure, D. (2010) A Categorisation of Cloud Computing Business Models. In: CCGrid 2010, The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 17-20, Melbourne, Australia. pp. 509-512.
- Interworksccloud. (2013) How much do people know about the cloud really? [Online] Available from: <http://blog.interworksccloud.com/how-much-do-people-know-about-the-cloud-really/> [Accessed: 19th June 2012]
- Korri, T (2009) "Cloud computing: utility computing over the Internet" *Seminar on Internetworking 2009*.
- Wood Katie, "Exploring security issues in cloud computing" (2012). *UK Academy for Information Systems Conference Proceedings 2012*. Paper 30. <http://aisel.aisnet.org/ukais2012/30>.
- National Institute of Standards and Technologies; Draft NIST Working Definition of Cloud Computing, May 14, 2009.
- David Binning 24 April 2009 'Top five cloud computing securityissues' <http://www.computerweekly.com/news/2240089111/Top-five-cloud-computing-security-issues#2>.
- Warwick Ashford [Friday 22 March 2013] 'Six security issues to tackle before encrypting cloud data' <http://www.computerweekly.com/news/2240180087/Six-security-issues-to-tackle-before-encrypting-cloud-data>.
- Wood, K (2012) 'Understanding Configuration Management with Cloud Computing' International Conference on Computational Informatics and Technology Enhanced Education (ICCITEE> 2012) Amsterdam, Netherlands.
- S. Srinivasamurthy and D. Q. Liu, "Survey on Cloud Computing Security", *Proc. Conf. on Cloud Computing, CloudCom.'10*.
- Top Threats to Cloud Computing v1.0". Cloud Security Alliance. Retrieved 24/10/2014.
- M. A. Vouk Cloud computing — issues, research and implementations *Journal of Computing and Information Technology - CIT* 16, 2008, 4, 235–246 doi:10.2498/cit.1001391.
- Hayes, B. "Cloud Computing," *Communications of the ACM*, 51(7):9–11 (2008).
- Wood, K and Anderson, M (2011) 'Understanding the complexity surrounding multi-tenancy in cloud computing' 8th IEEE International Conference on e-Business Engineering, Tsinghua University, (ICEBE 2011).

Offline Scheduling of Map and Reduce Tasks on Hadoop Systems

Aymen Jlassi^{1,2}, Patrick Martineau² and Vincent Tkindt²

¹Cyres Group, 19 rue Edouard Vaillant, 37000 Tours, France

²University François-Rabelais of Tours, CNRS, LI EA 6300, OC ERL CNRS 6305, Tours, France
jlassi.aymen@etu.univ-tours.fr, {patrick.martineau, vincent.tkindt}@univ-tours.fr

Keywords: Big Data, MapReduce Model, Hadoop Scheduling Problem, Time Indexed Formulation.

Abstract: MapReduce is a model to manage quantities massive of data. It is based on the distributed and parallel execution of tasks over the cluster of machines. Hadoop is an implementation of MapReduce model, it is used to offer BigData services on the cloud. In this paper, we expose the scheduling problem on Hadoop systems. We focus on the offline-scheduling, expose the problem in a mathematic model and use the time-indexed formulation. We aim consider the maximum of constraints of the MapReduce environment. Solutions for the presented model would be a reference for the on-line Schedules in the case of low and medium instances. Our work is useful in term of the problem definition: constraints are based on observations and take into account resources consumption, data locality, heterogeneous machines and workflow management; this paper defines boundaries references to evaluate the online model.

1 INTRODUCTION

Manage and access efficiently massive data is becoming more and more important for companies. Google (Dean, 2004) introduced the model MapReduce as a distributed and parallel Model for data intensive computing. Every job is composed of a set of “map” and “reduce” tasks, which is executed in a distributed fashion over a cluster of machines. Map tasks have to be executed before reduce tasks. Tasks have to be executed as near as possible to the needed data input. Data output of tasks map are transferred to the reduce tasks using the network. MapReduce model is characterized by its simplicity: users wanting to access to data, create “map” and “reduce” tasks, which are next scheduled by specified middleware. The general idea is to schedule those tasks over nodes, which contain data because moving computation near data is less expensive than moving data where computation units are running. For example, in figure 1, average of input set of integers is calculated.

Hadoop (Hadoop, 2005) is one of the most well-known implementation of MapReduce model. It is based on two main components: Hadoop mapReduce and Hadoop distributed file system. The computation level (mapReduce) is composed of three elements. It assures synchronization over different elements and distributes resources between jobs. The Node Manager (NM) is the responsible for

resources exploitation per slave machine. The Application Master (AM) is responsible for managing the lifecycle of a job; it negotiates with the RM to obtain needed resources (containers) and manages the execution of job’s tasks.

Hadoop distributed file system (HDFS) is composed of NameNode (NN) as a server and DataNode (DN) as a slave. Files in HDFS are from megabytes up to terabytes size. The number of map tasks depends on the number of chunks of data (Zhou, 2012), one map per data block slice. When the scheduler cannot assign tasks to machines where data are stored, bandwidth on the network is allocated to migrate blocks towards. This paper presents an offline model of scheduling problem on Hadoop with mathematical programming based on the time-indexed formulations which received much attention due to its important impact on approximation algorithms and the quality of its linear programming relaxation.

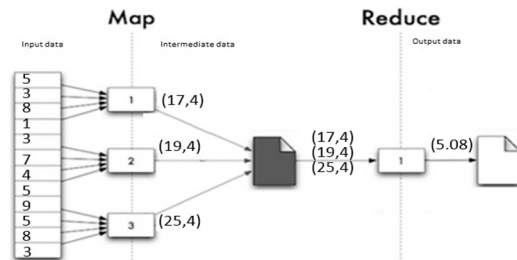


Figure 1: Example of mapreduce job's execution.

It is often used in optimization and approximation for machine scheduling problems. Besides, its linear relaxation yields concise lower bounds than bounds obtained by other integer programming formulations (Queyranne, 1997). Work like (Sousa, 1992) and (Lionel, 2013) argue that scheduling algorithms using LP-relaxation of time-indexed formulations have a constant ratio on their worst-case performance in parallel machine scheduling problems. Researches on the online version of problem suffer from a lack of evaluation: how the efficiency of online algorithms can be evaluated? One way to answer this question is to consider the offline version of the problem, its' optimal solution can be considered as "ideal" reference schedules for online algorithms. In this work, the main motivation is to compute optimal solutions for medium instances of the offline problem.

The remainder is introduced as follows. Section II presents the offline problem of scheduling "map" and "reduce" tasks. In Section III its' mathematical model is introduced. Data generation and model evaluation is presented in Section IV. Section V summarizes the related work. Lastly, Section VI concludes the paper and provides directions of future work.

2 RELATED WORK

The scheduling problem in Hadoop is widely treated in the literature: (Lim, 2014) present a constraint programming formulation of the problem. The objective of the model is to minimize the number of late jobs, which is characterized by its service level agreements (SLA). Authors consider the scheduling of mapReduce jobs comprising an earliest start time, execution time and end-to-end deadline. In this work, authors take into consideration only compute resources (slots), neither RAM nor hard disk are considered. They neglect the relation between data and tasks locations that present a foundation for the map reduce programming model. The work in (Verma, 2012) implements a deadline-based scheduler; it is based on a general model for computing performances bounds on makespan of a given set of n tasks that are processed by k servers (slots). The assignment of tasks to slots is done using an online greedy algorithm; it assigns each task to the slot, which has finished its running task the earliest. (Evrpidis, 2014) and (Lin, 2013) propose models, which aim to minimize the total weighted completion time. The first considers that each job has at least one

map and one reduce task and each job has at most one task pre-assigned to each processor.

Table 1: Used Notations in the Hadoop scheduling problem.

General data:	
M	The number of machines
N	The number of tasks
N^m	Number of map tasks
N^r	Number of reduce tasks
L^m	Set of map tasks
L^r	Set of reduce tasks
A^b	Set of blocks on the cluster
T	The scheduling horizon
For machines	
m_j^s	The number of slots on machine j ($= m_j^{sr} + m_j^{sm}$)
m_j^{sr}	The number of reduce slots on machine j
m_j^{sm}	The number of map slots on machine j
m_j^r	The quantity of RAM of machine j
m_j^h	The hard drive capacity of machine j
$v_{s,j}$	The CPU frequency associated to the slot s of machine j
v_j	The CPU frequency of machine j ($v_j = \sum_{k=1}^{m_j^s} v_{k,j}$)
α_j^r	The cost of the use of one unit of ram (1 Mb) per machine j
α_j^h	The cost of the use of one unit of hard drive capacity (1 Mb) per machine j
$\alpha_{s,j}^c$	The cost of the use of CPU on slot s of machine j
For tasks (map, reduce, Application node)	
n_i^r	The quantity of RAM required by task i
n_i^h	The quantity of hard drive required by task i
n_i^b	The number of data block's manipulated by task i
B_i	List of block numbers manipulated by task i
$b_{i,i'}$	Maximum bandwidth between tasks i and i'
n_i^p	Number of tasks preceding task i
E_i	Set of task numbers that must be completed before task i start.
$p_{i,s}^j$	Estimated processing time of task i if processed on slot s of machine j
For HDFS	
S	The size of a data block in the cluster.
r_b	Number of replication block b .
D_b	Set of machines on which block b is located.
bwd	Bandwidth allocated for migrating a block through the network
For the Network	
$G=(V, E)$	The graph modeling the network
b_{max}	The maximum bandwidth associated to any edge $e_u \in E$
P	A set of paths between machines, a path being a set of edges e_u
P_u	The set of couples of machines (j, j') which use the edge e_u

The second considers task pre-assignment to machines and each machine can execute one task at a time. It models the data transfer from map to reduce tasks and it considers map and reduce dependency. (Kodialam, 2012) express the scheduling problem as an optimization problem using linear programming, they aim to minimize the total weight completion time of jobs, they base their work on a set of assumption: machines can process at most one task at the same moment; tasks can be preempt. Fotakis et al. (Fotakis, 2014) consider the case of unrelated processors with multiple Map and Reduce tasks per job. They consider that tasks can be preempted. They present the first polynomial time approximation algorithm, it minimizes the total weighted completion time. However they neglect the data management aspect and they don't consider multiple tasks execution per machine. In this work we associate resources constraints, network bandwidth management to the data flow management.

3 THE OFFLINE SCHEDULING PROBLEM

We summarize in Table 1 the data used in the scheduling model. It is based on four principal parts: the first describes the information about machines and the cost of every resource's use. The second part describes tasks consumption. The third part gives information about data blocks and the fourth describes networks architecture. We consider non-pre-emptible tasks because, in practice, tasks will not be interrupted in Hadoop and when a task fails, it will rerun as it is newly submitted.

Notice: we assume that bandwidth is booked on the network from the end of map tasks until the end of the reduce tasks. The bandwidth reservation avoids delaying job execution when reduce tasks need to communicate with maps machines to ensure some needs (system files, recovers broken data chunks) (White, 2012).

4 A MATHEMATICAL FORMULATION

This section presents a time-indexed formulation of offline scheduling problem in Hadoop. Let us review the formal definition of the model. We adapt the interval-relaxation method proposed in (Dyer, 1990) in single machine case, and in (Schulz, 2002) in

multiple machines, with the context of MapReduce model. The time horizon T is divided into a set of irregular intervals. These intervals are defined by the potential dates of starting and finishing execution of tasks. For example, in Figure 2, for $\delta \in \llbracket 0, n-1 \rrbracket$, the intervals $(t_\delta, t_{\delta+1}]$ are used to execute tasks, where $t_\delta \in [0, T]$.

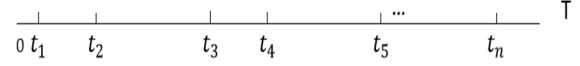


Figure 2: Presentation of the index over time.

We use the following variables:

$$x_{i,s,t_\delta}^j \begin{cases} q, & \text{the amount of time period, the task } i \text{ is} \\ & \text{processed on slot } s \text{ of the machine } j \\ & \text{in } (t_\delta, t_{\delta+1}] \\ 0, & \text{otherwise} \end{cases}$$

Thus $x_{i,s,t_\delta}^j / p_{i,s}^j$ specify that the task is being processed on machine j during the time interval $(t_\delta, t_{\delta+1}]$.

$$y_{b,t_\delta}^{jj'} \begin{cases} 1, & \text{if block } b \text{ is on machine } j \text{ at } [t_\delta, t_{\delta+1}[\\ & \text{after a migration from } j' \\ 0, & \text{otherwise} \end{cases}$$

$$u_{b,t_\delta}^{jj'} \begin{cases} 1, & \text{if block } b \text{ is being migrated from} \\ & \text{machine } j \text{ to } j' \text{ at } [t_\delta, t_{\delta+1}[\\ 0, & \text{otherwise} \end{cases}$$

$$z_{l,l',t_\delta}^{jj'} \begin{cases} 1, & \text{if a map task } l' \text{ is processed on} \\ & \text{machine } j' \text{ and is finished at time } t \text{ and} \\ & \text{a reduce task } l \text{ is processed on machine } j \\ & \text{and finished after } t_\delta. \\ 0, & \text{otherwise} \end{cases}$$

We refer to TST as the total time spent for processing all tasks on the cluster and TRC as the total resource cost induced by the execution. The scheduling problem in Hadoop can be modeled with the objective functions (1) and (2). The TST (1) considers the total execution time of tasks (the first term on the left-hand side of the equation) and the time of data transfer between map and reduce tasks (the second term on the right-hand side of the equation). The TRC (2) considers the resources machines' cost when processing tasks (the first term on the left-hand side of the equation) and the use of resources due to data transfer (the second term on the right-hand side of the equation). The constraints of the model are classified in three categories: resource constraints, processing constraints and the network constraints.

$$\begin{aligned}
& \text{Minimize TST} \\
& = \sum_{t_\delta=1}^T \left[\sum_{i=1}^N \sum_{j=1}^M \sum_{s=1}^{m_j^s} \left(x_{i,s,t_\delta}^j / p_{i,s}^j \right) \right. \\
& \quad \left. + \sum_{i \in L^r, l \in E_i} \sum_{j,j'=1}^M \sum_{b \in A^b} y_{b,t_\delta}^{jj'} (S/b_{l,i}) \right] \quad (1) \\
& \text{Minimize TRC} \\
& = \sum_{t_\delta=1}^T \sum_{i=1}^N \sum_{j=1}^M \sum_{s=1}^{m_j^s} x_{i,s,t_\delta}^j [\alpha_j^r n_i^r + \alpha_j^h n_i^h \\
& \quad + \alpha_{s,j}^c] \quad (2) \\
& + \sum_{t_\delta=1}^T \sum_{j=1}^M \sum_{j'=1}^M \sum_{i=1}^N \sum_{b \in B_i} (y_{b,t_\delta}^{jj'} + u_{b,t_\delta}^{jj'}) (\alpha_j^r n_i^r \\
& \quad + \alpha_j^h n_i^h)
\end{aligned}$$

In the subsection 4.1, constraint (3) guarantees that no more memory than available is used. Constraints (4) and (5) guarantee that the number of reduce (resp. map) tasks running on machine j at time t is less than the number of reduce slots (resp. map slots). Constraint (6) ensures that the overall local disk space used (by the assigned tasks and migrated data) cannot exceed the availability of each machine. In the subsection 4.2, the inequality (7) guarantees the precedence relation between map and reduce tasks associated to the same job are satisfied. If we have many map tasks, reduce tasks are scheduled after the schedule and the end of all map tasks. In figure 1, we compute average of input data, we will have wrong result if reduce tasks start before the end of map tasks. Constraints (7) and (8) ensure that all map tasks (resp. reduce tasks) must be processed.

In the subsection 4.1, the constraints define the policy of data blocks management in Hadoop. The inequality (10) specifies if block b is stored in HDFS on machine j . The constraints (11) and (13) impose the relation between y 's and u 's variables, constraint (13) triggers data migration to ensure that block must be available on the machine before a map task starts and constraint (11) ensures if it is available on a machine after it has been migrated. The Inequalities (12) disable the start of map tasks (imposed by the constraint 8) if the manipulated blocks are not present on the machine on which they have been assigned. The inequalities (15) enable to fix the values of the $z_{l,l',t_\delta}^{jj'}$ variables. When the tasks map and reduce are on the same machine, we don't have network communication and the right part of inequality (15) will be 0.

4.1 Resources Constraints

$$\sum_{s=1}^{m_j^s} \sum_{i=1}^N n_i^r \left(x_{i,s,t_\delta}^j / p_{i,s}^j \right) \leq m_j^r \quad (3)$$

$$\forall j = 1 \dots M, \forall t_\delta = 1 \dots T$$

$$\sum_{i \in L^r} \left(x_{i,s,t_\delta}^j / p_{i,s}^j \right) \leq 1 \quad (4)$$

$$\forall j = 1 \dots M, \forall t_\delta = 1 \dots T, \forall s = 1 \dots m_j^{Sr}$$

$$\sum_{i \in L^m} \left(x_{i,m_j^{Sr}+s,t_\delta}^j / p_{i,m_j^{Sr}+s}^j \right) \leq 1 \quad (5)$$

$$\forall j = 1 \dots M, \forall t_\delta = 1 \dots T, \forall s = 1 \dots m_j^{Sr}$$

$$\begin{aligned}
& \sum_{s=1}^{m_j^s} \sum_{i=1}^N \left(n_i^h x_{i,s,t_\delta}^j / p_{i,s}^j \right) \\
& + \sum_{j'=1, j' \neq j}^M \sum_{i=1}^N \sum_{b \in B_i} S(b) (y_{b,t_\delta}^{jj'} + u_{b,t_\delta}^{jj'}) \leq m_j^h \quad (6) \\
& \forall j = 1 \dots M; \forall t_\delta = 1 \dots T
\end{aligned}$$

4.2 Tasks Constraints

$$\begin{aligned}
& n_k^p * x_{i,s,t_\delta}^j / p_{i,s}^j \\
& \leq \sum_{u=1}^M \sum_{s'=1}^{m_u^{Sm}} \sum_{t'=0}^{t_\delta} \sum_{l \in E_k} \left(x_{l,m_u^{Sr}+s',t'}^u / p_{l,m_u^{Sr}+s'}^u \right) \quad (7) \\
& \forall k \in L^r, \forall t_\delta = 0 \dots T-1, \forall j = 1 \dots M, \forall s = 1 \dots m_j^{Sr}
\end{aligned}$$

$$\sum_{j=1}^M \sum_{s=1}^{m_j^{Sm}} \sum_{t_\delta=0}^{T-1} \left(x_{i,m_j^{Sr}+s,t_\delta}^j / p_{i,m_j^{Sr}+s}^j \right) = 1 \quad (8)$$

$$\forall l \in L^m$$

$$\sum_{j=1}^M \sum_{s=1}^{m_j^{Sr}} \sum_{t_\delta=0}^{T-1} \left(x_{i,s,t_\delta}^j / p_{i,s}^j \right) = 1 \quad (9)$$

$$\forall l \in L^r$$

4.3 Constraints Associated to the Migration of Data Blocks

$$y_{b,t_\delta}^{jj} = \begin{cases} 1, \forall j \in D_b \\ 0, \forall j \notin D_b \end{cases} \quad (10)$$

$$\forall t_\delta = 0 \dots T, \forall b \in A^b$$

$$u_{b,t_{\delta-1}}^{jj'} \leq y_{b,t_\delta}^{jj'} \quad (11)$$

$$\forall b \in A^b; \forall t_\delta = 1 \dots T-1; \forall j, j' = 1 \dots M; j \neq j'$$

$$\leq \sum_{l \in L^m} \sum_{t'=t_\delta}^{T-1} \sum_{s=1}^{m_j^{sm}} \left(\frac{u_{b,t}^{jj'} x_{l,m_j^{sr+s,t'}}^j}{p_{l,m_j^{sr+s}}^j} \right) \quad (12)$$

$$\forall b \in A^b; \forall t_\delta = 0, \dots, T-1; \forall j, j' = 1 \dots M, j \neq j'$$

$$\sum_{s=1}^{m_j^{sm}} \left(\frac{x_{l,m_j^{sr+s,t_\delta}}^j}{p_{l,m_j^{sr+s}}^j} \right) \leq \sum_{j'=1}^M y_{b,t_\delta}^{jj'} + u_{b,t_\delta}^{jj'} \quad (13)$$

$$\forall l \in L^m; \forall b \in B_l; \forall t_\delta = 0, \dots, T-1; \forall j = 1 \dots M$$

4.4 Network Constraint

These constraints define the use of the network in terms of bandwidth. Constraint (14) imposes that all consumed bandwidth (for migration and transfer of data) is less than the maximum bandwidth b_{max} .

$$\sum_{(j,j') \in P_e} \left[bwd \sum_b u_{b,t_\delta}^{jj'} + \sum_{l \in L^r} \sum_{l' \in E_l} \sum_{s=1}^{m_j^s} z_{l,l',t_\delta}^{jj'} \right] * b_{i,l} \leq b_{max} \quad (14)$$

$$\forall e \in E; \forall t_\delta = 1, \dots, T-1$$

$$\sum_{s=1}^{m_j^{sr}} \left(\frac{x_{l,s,t}^j}{p_{l,s}^j} \right) + \sum_{s=1}^{m_{j'}^{sm}} \left(\frac{x_{l',m_j^{sr+s,t'}}^{j'}}{p_{l',m_j^{sr+s}}^j} \right) - 1 \leq z_{l,l',t_\delta}^{jj'} \quad (15)$$

$$\forall l \in L^r, \forall l' \in E_l; \forall t_\delta = 0, \dots, T-2; \forall t' = 0, \dots, t_{\delta-1}; \forall t' = t_\delta, \dots, T-1; \forall j, j' = 1 \dots M, j \neq j'$$

5 EXPERIMENTATION

This article implements a model and tries to find solutions using CPLEX mathematic solver. Face to the multi-criteria property of the problem, the model is concentrated on the time execution aspect and neglects cost execution of the job. It uses an experiment setting for the evaluation of the model using the methodology in (Lionel, 2013). Data input of the model presents an important deal and imitates real world tasks executions. Machine configuration is extracted from AWS (Aws, 2014) and portioned in three categories of machines. Tasks information depends on the size of data input computed by every task. In order to evaluate the persistence of the model, we generate randomly four input data concerning tasks following uniform law: memory, disk consumption, the time execution per task and location of data blocks (Gupta, 2013). We generate also network and cluster configuration details. Table 2 synthetizes values of the expected data input of machines. The first column indicates the category of the machine.

The second column indicates the number of core CPU on the machine. The third one contains the amount of memory per machine. The column number four indicates the quantity of hard disk in Gb. The fifth column contains the frequency of one core CPU on the machine. The sixth column indicates the bandwidth allocated for network communication.

Table 2: Types of generated physical machines.

Category	CPU node	RAM (Gb)	SSD (Go)	CPU freq per core (GHZ)	Bdw (GB)	α_j^r	α_j^h	$\alpha_{s,j}^c$	Slots map	Slots reduce
c3.2xlarge: compute optimized	8	15	160	2.8 Intel Xeon E5-2680v2	1	1	1	2	5	2
i2.2xlarge: storage optimized	8	61	1600	2.5 Intel Xeon E5-2670v2	1	3	5	2	4	3
r3.xlarge: memory optimized	4	30.5	160	2.5 Intel Xeon E5-2670v2	2	2	1	1	2	1

Table 3: Characteristics of used jobs.

Job	Tasks reduce	Tasks map	Type of Job
1	2	3	--
2	2	6	--
3	3	9	--

Columns number seven, eight and nine indicate respectively the unit cost of the memory use (unit = 16Mb), hard disk (unit = 1Gb), and a core of CPU. Despite the evolution in Hadoop, we adopt the principle of separation between slots; the last two columns contain the number of reduce and map cores (slots) per machine. The costs of resources consumption are expressed in columns seven, eight and nine and they depend on the type of machine. We generate the completion time needed to treat tasks; these values depend on the size of the block. We define:

$$P_{l,s}^j = \text{TimeStartUpVM} + S * nt(l) * \left[\frac{vs(l,j,s)}{10 * \text{SpeedProcessorRate}} \right] \quad (16)$$

We take into account the needed time to start up virtual machines TimeStartUpVM, the size of block and the amount of data computed per GHz per unit of time SpeedProcessorRate. We benefit from the last variable to inject the random aspect depending on the categories of machines: for the category “compute optimized”, SpeedProcessorRate $\in [160,320]$ for the other types SpeedProcessorRate $\in [80,160]$. The estimation of memory (n_i^r) and hard disk consumption (n_i^h) depends on the type of the job. Table 4 summarizes used formulas in the generation of data related to the three types of jobs: the number of tasks per job is relatively limited; CPLEX limitation imposes this choice of number of task per job face of the use of one big job. We inject random values at many levels of the data input generation. Face to the large quantity of data generated by the model in time indexed formulation, we consider $S=64\text{Mb}$ and its replication is equal to one. We consider the same size (S) and replication properties of data blocks however we generate randomly the location of the blocks on machines. The network bandwidth for block migration is fixed by the formula $bwd = \min[S * 0.2, 128]$. Network is generated as a binary tree. We repeat the following process: at the main node, we generate a switch; its left child node will be one physical machine selected randomly, the right child will be another switch and so on until all physical machines will be placed on the binary tree.

Table 5 describes scenarios used for the model’s test. For each scenario, we randomly generate 20 instances. The time horizon depends on scenarios and it is divided in intervals. To find the correct value of time horizon, we define an upper bound for every solution using this formula (17). If there is no solution for a particular value of the time horizon, we increment time horizon by a unit of time. We consider that an interval $[t_\delta, t_{\delta+1}[$ from figure 2) is sufficient to transfer data block between machines. In conclusion, we limit bandwidth threshold to migrate blocks and we limit the transfer duration of a block to one interval. To compute the real duration’s value of a schedule per scenario, we define “RealTime” (formula 18) as the real time needed to execute tasks in a solution.

$$T = \text{integer} \left(\frac{N^m}{\text{TotalSlotMap}} + \frac{N^r}{\text{TotalSlotReduce}} \right) + 2 \quad (17)$$

$$\text{RealTime} = \sum_{t=0}^T \max_{\substack{j=1..M \\ s=1..ms(j) \\ l \in L}} x_{l,s,t}^j \quad (18)$$

$$\begin{aligned} \text{RealValueOfTimeHorizonUnit} \\ = \text{RealTime}/T \end{aligned} \quad (19)$$

“RealTime” is a posterior computation, after the compute of the scheduling solution.

Table 4: Basic formulas to generate memory and hard disk consumptions per task.

Type of Job	$n_i^r = n_i^b * S * XY$	$n_i^h = (n_i^b * S * WZ)/1024$
(1) CPU intensive	$XY \in [0.3,0.6]$	$WZ \in [13,26]$
(2) RAM intensive	$XY \in [0.4,0.8]$	$WZ \in [30,46]$
(3) I/O intensive	$XY \in [0.6,1]$	$WZ \in [46,76]$

Table 5: Different scenarios for the generation of tasks, machines and blocks input data.

Scenarios	N1	N2	N3	M1	M2	M3	Blocks	N	M	T
1	1	1	0	1	1	1	10	13	3	3
2	3	0	0	0	2	0	10	15	2	3
3	1	1	1	1	1	1	10	25	3	9
4	3	3	0	0	0	2	10	39	2	15
5	6	0	0	1	1	1	10	30	3	7
6	2	3	1	1	1	0	10	46	2	15
7	3	1	1	0	2	0	10	35	2	6

Table 6: Computational results (20 instances per scenario).

	#InFeas	#Solved	MemLimit	TimLimit	N _{min}	N _{avg}	N _{max}	T _{min}	T _{avg}	T _{max}	Real value of unit of T
Sc1	0	20	0	0	0	7.95	132	0	0.45	1	85.66
Sc2	0	20	0	0	27	42.5	164	0	19.04	174	95.66
Sc3	1	18	0	1	4	5357.6	62168	6	97.75	1044	122.4
Sc4	3	16	1	0	0	6675.15	28365	16	292.36	1313	54.8
Sc5	2	18	0	0	40	132.8	1791	10	66.9	757	94.62
Sc6	4	15	0	1	115	142	389	28	185	1641	126.23
Sc7	0	20	0	0	3	61.25	193	5	10.1	22	70.53

It is used to compute the real duration to execute jobs in a scenario. We define established value as the time Horizon T per scenario; we compute a value of a unit of T as regular time horizon with the formula (19). We enumerate the minimum, maximum and average of the RealTime over iterations and we choose the maximum value to compute the value of a unit of T per scenario. This value is used in the evaluation of the results of solutions.

To test the model, we use a PC with an Intel(R) Core (TM) i5-3360M CPU with 4 cores at 2.8 GHz and 4 Gb of RAM. The linear program formulation has been solved by CPLEX 12.2 with parallel solve (4 threads) and limit time 1800 seconds and memory limit of 2 Gb of RAM. When the time limit or the memory limit is reached, the given solution of the instance will be declared unsolved. Otherwise, CPLEX will return the best solution. For each scenario, table 6 presents: the number of infeasible instances (column #InFeas), the number of instances solved to optimality (column #Solved). The number of instances on which CPLEX stops due to the memory limit (column Mem) and the number of instances on which CPLEX stops due to the time limit (column Time). The columns from number six to number eight provide the minimum, maximum and average number of nodes explored by CPLEX in its branch and cut algorithm while solving the problem. There is no relation between the number of machines and the number of explored nodes. Scenarios 4 and 6 have two machines each, however the number of explored nodes in scenario 4 is largely higher than the number of nodes explored in scenario 6. In the same topic, the number of explored nodes is independent from the number of scenario 7 for example has a number of tasks to schedule higher than scenario 5. However, the number of node explored in scenario 5 is higher than in scenario 7. The columns from number nine to number eleven provide minimum, average and maximum CPU time (in seconds) taken by CPLEX to solve instances. In this topic, we consider only instances, which have infeasible or feasible results.

The result shows that there are large disparities concerning CPU times used to find solution. The last column presents the real value of the time horizon unit; it is used as a comparison reference. It is extracted from the approximate value of the average completion time per scenario. Results of founded schedule time of a scenario argue that it depends on the number of tasks and machines; Scenarios 4 and 6 have largest value of the time horizon. These scenarios have the largest number of tasks to schedule. Scenarios 1 and 2 have the smallest number of tasks and the smallest number of machines in an instance. Results are function of the number of tasks and the number of machines in an instance and some instances take more time to find solution than others. Scenario 6 for example schedules 46 tasks on two machines; it has the largest value of completion time.

6 CONCLUSIONS

In this paper, we propose an offline mathematical model for the scheduling problem in Hadoop. Two kinds of tasks are considered: “map” and “reduce” tasks with dependencies between them. This paper also presents an in-depth study of the major aspects of MapReduce model, such as tasks dependency, network consumption, data flow management and the non-interruptive tasks executions.

It aims at scheduling tasks with the minimum cost of used resources and the minimum total processing duration. We merely focus on a pure scheduling problem; we propose an offline model assuming that all data are known. We present a realistic model, which considers dependence between tasks. We consider data locality and we model data migration and transfer between heterogonous machines. All considered constraints emulate the real world environment in Hadoop. Heterogeneous machines cluster and possibility to execute many tasks per machine are also considered. The proposed model is based on a time-indexed formulation, which despite

its pseudo polynomial number of variables. It has already been shown as an efficient formulation compared to other integer programming formulations. We use the commercial solver CPLEX to find the optimal solution for small and medium size of instances. We give community a boundary to reference with and to evaluate their scheduling algorithms for this size of instances. It turns out that the offline problem is interesting in it self and can be used to design good online strategies. Solution for this model would be a reference for the on-line schedules in smaller dimension to validate first result. Future work will deal with the online aspect concerning the scheduling problem; we plan to propose a heuristic solution and use this work in the evaluation.

Online solution considers at first Total completion time, in a second time we take into account the resources consumption (energy) in a multi-criteria scheduling aspect.

The final solution will be implemented over Hadoop simulation system and evaluated in a large scalability face to default scheduler in Hadoop.

ACKNOWLEDGMENTS

This work was sponsored in part by the CYRES GROUP in France and French National Research Agency under the grant CIFRE n°2012/1403.

REFERENCES

- Aws. 2014. Instances-types. Retrieved from Aws: <http://aws.amazon.com/fr/ec2/instance-types/>
- Dean, J., & Ghemawat, S., 2004. MapReduce: Simplified Data Processing on Large Clusters. In Communications of the ACM.
- Dyer, M. E., & Wolsey, L. A., 1990. Formulating the single machine sequencing problem with release dates as a mixed integer program.
- Evripidis Bampis, V. C., 2014. Energy Efficient Scheduling of MapReduce Jobs. In 20th International Conference.
- Fotakis, D., Milis, I., & Zampetakis, E., 2014. Scheduling MapReduce Jobs on Unrelated Processors. In the Workshop Proceedings of the EDBT/ICDT.
- Gupta, S., Fritz, C., Price, R., Hoover, R., de Kleer, J., & Witteveen, C., 2013. Throughput Scheduler: learning to schedule on heterogeneous Hadoop clusters. In (ICAC '13), International Conference on Autonomic Computing.
- Hadoop Project, 2005. (A. foundation, Producer) Retrieved from <http://hadoop.apache.org/>
- Kodialam, M. S., Lakshman, T., Mukherjee, S., Chanwg, H., & Lee, M. J., 2012. Scheduling in mapreduce like systems for fast completion time. In Patent Application Publication.
- Lim, N., Majumdar, S., & Ashwood-Smith, P., 2014. A Constraint Programming-Based Resource Management Technique for Processing MapReduce Jobs with SLAs on Clouds.
- Lin, M., Zhang, L., Wierman, A., & Tan, J., 2013. Joint Scheduling of Processing and Shuffle Phases in MapReduce Systems. In P. o. Conference (Ed.).
- Lionel, E.-D., Adrien, L., Patrick, M., Ameer, S., Vincent, T., & Denis, T., 2013. A Server Consolidation Problem: Definition and Model. In Proceedings of the 14th conference ROADEF.
- Queyranne, M., & Schulz, A., 1997. Polyhedral Approaches to Machine Scheduling. In Mathematical Programming.
- Schulz, A. S., & Skutella, M., 2002. Scheduling Unrelated Machines by Randomized Rounding. In SIAM Journal on Discrete Mathematics.
- Sousa, J. P., & Wolsey, L. A., 1992. A time indexed formulation of non-preemptive single machine scheduling problems. In Mathematical Programming.
- Verma, A., Cherkasova, L., Kumar, V. S., & Campbell, R. H., 2012. Deadline-based Workload Management for MapReduce Environments: Pieces of the Performance Puzzle.
- White, T., 2012. Hadoop, The Definitive Guide (3rd Edition ed.). O'REILLY. 3rd edition.
- Zhou, W., Han, J., Zhang, Z., & Dai, J., 2012. Dynamic Random Access for Hadoop Distributed File System. In (ICDCSW), Distributed Computing Systems Workshops.

A Generalized Service Replication Process in Distributed Environments

Hany F. El Yamany¹, Marwa F. Mohamed¹, Katarina Grolinger² and Miriam A. Capretz²

¹Faculty of Computers and Informatics, Suez Canal University, Old Campus, Ismailia, Egypt

²Department of Computer Engineering, Western University, London, ON, Canada

{hany_elyamany, marwa_fikry}@ci.suez.edu.eg, {kgroling, mcapretz}@uwo.ca

Keywords: Service Replication, Service-Oriented Architecture, Cloud, Mobile Computing, Replication Process, Quality of Service.

Abstract: Replication is one of the main techniques aiming to improve Web services' (WS) quality of service (QoS) in distributed environments, including clouds and mobile devices. Service replication is a way of improving WS performance and availability by creating several copies or replicas of Web services which work in parallel or sequentially under defined circumstances. In this paper, a generalized replication process for distributed environments is discussed based on established replication studies. The generalized replication process consists of three main steps: sensing the environment characteristics, determining the replication strategy, and implementing the selected replication strategy. To demonstrate application of the generalized replication process, a case study in the telecommunication domain is presented. The adequacy of the selected replication strategy is demonstrated by comparing it to another replication strategy as well as to a non-replicated service. The authors believe that a generalized replication process will help service providers to enhance QoS and accordingly attract more customers.

1 INTRODUCTION

Nowadays, the Web is structured as a mesh of heterogeneous distributed environments including service-oriented architecture (SOA) (Erl, 2008), cloud computing (Erl *et al.*, 2013), and mobile computing (Fling, 2009). Web services have a key role in managing and encapsulating business processes inside such environments. Web services are described, discovered, published, and executed using standard protocols such as WSDL for service description, SOAP for message exchange, and UDDI for service registry and discovery (W3C, 2004; Papazoglou, 2008).

Quality of service (QoS) (Al-Masri *et al.*, 2007; W3C, 2003) is a significant factor in describing and establishing the service level agreement (SLA) among service providers and consumers. In particular, the SLA is an official contract between the provider and the consumer which specifies non-functional requirements, specifically focussing on performance and availability (Michlmayr *et al.*, 2009; Papazoglou *et al.*, 2005). Web service replication is a way of improving WS performance and availability by creating several copies or replicas of Web services which may work either in parallel

or sequentially under defined circumstances and regulations (Salas *et al.*, 2006; May *et al.*, 2009).

This paper introduces a generalized replication process in distributed environments with the objective of helping service providers select and implement an appropriate service replication strategy and consequently increase the quality of service provided. The discussed replication process consists of three main steps: sensing the environment characteristics, determining the replication strategy, and implementing the selected replication strategy. A mobile authentication case study is presented to demonstrate the use of the approach.

The rest of this paper is organized as follows: Section 2 introduces the replication literature review. Section 3 introduces a generalized replication process for distributed environments. Section 4 presents the evaluation case study. Finally, Section 5 concludes the paper.

2 REPLICATION LITERATURE REVIEW

Two main replication types may be distinguished

depending on whether the number and location of replicas change during runtime: static replication and dynamic replication. In *static replication*, the predefined replica communication group does not change during runtime; when a single replica becomes unresponsive, this replica is still considered a member of the communication group. In other words, the number and position of replicas are fixed over time (Guerraoui *et al.*, 1997). *Static replication* is planned at design time (Słota *et al.*, 2005) according to predefined parameters, and implementation is carried out regardless of any changes that may occur during runtime.

Dynamic replication supports a changing number of replicas, changes in physical locations, and selection of running replicas during runtime (Keidl *et al.*, 2003; Mohamed *et al.*, 2013). It is performed in two different styles: *Dynamic replica selection* (Thakur *et al.*, 2012) and *Dynamic replica placement* (Mohamed *et al.*, 2013; Dustdar, 2007). The replication process can be implemented using different techniques depending on the components involved and their characteristics. Salas *et al.* (2006) classified the replication process into three techniques according to the interactions among replicas and requests: *active*, *passive* and *semi-active techniques*. Like Salas *et al.* (2006), May *et al.* (2009) also recognized three categories, but the categories are different: *parallel*, *serial* and *composite techniques*. Zheng *et al.* (2008) expanded the replication techniques from the work of Salas *et al.* (2006) by combining active, passive, and time-replication techniques. Time replication means that a particular service is invoked a finite number of times before its status is declared as failed. Liu *et al.* (2011) took a very different approach and generated a diverse group of replication techniques using a directed acyclic graph (DAG), which are characterized as active, passive, hybrid, active-passive, and passive-active. In their study, they produced a graph model to represent a replication scheme defined as a directed acyclic graph DAG, $G \equiv (V, E)$, where the vertex set V refers to a set of WS replicas and the directed edge set E refers to the replica invocation. In this approach, the directed edges capture a replication schema.

Several researchers have proposed different replica selection strategies and algorithms (Sayal *et al.*, 1998; da Silva *et al.*, 2004; Björkqvist *et al.*, 2012). Sayal *et al.* (1998) described six replica selection algorithms: Fixed, Ping, Hops, Parallel, Probabilistic, and Refresh. Da Silva *et al.* (2004) presented five server selection policies: Random Selection, Parallel Invocation, HTTPing (or Probe),

Best Last, and Best Median. Finally, Björkqvist *et al.* (2012) defined two replica selection algorithms: Distributed Shortest Queue Selection (D-SQ) and Distributed Round Robin Selection (D-RR).

Although extensive efforts have been made in both academia and industry in the area of service replication, we are not aware of any studies that discuss a generalized replication process. The approach introduced in this paper builds on diverse service replication research to create a generalized replication process with the objective of helping service providers increase QoS.

3 GENERALIZED REPLICATION PROCESS

Distributed environments such as service-oriented architecture (SOA), cloud computing, and mobile computing typically use replication technology to improve operational characteristics, including availability and performance. Unfortunately, replication encounters challenges in these environments.

3.1 Generalized Replication Process: Overview

To help practitioners determine the most suitable replication approach for a specific scenario, a generic replication process is needed. This section introduces a generalized replication process for distributed environments based on the replication approaches presented in the previous sections. Specifically, the findings from the reviewed studies are integrated to form a generic replication process.

Figure 1 illustrates the use-case scenario: a client demands a service through a service provider, where the target service may be located and published in a cloud, SOA, or mobile environment. If the service provider (typically the business service provider or service owner) on behalf of the client(s) or consumer(s) detects a delay in answering incoming requests due to technical problems such as resource failures, service(s) overload, or network issues, the service provider should find a solution to speed up the answering process using service replication. How this replication process should be implemented will vary with respect to several metrics, including the characteristics of the host environment.

For example, in the cloud, the service provider could deploy a replica in any location where there are no technical challenges, whereas in a mobile

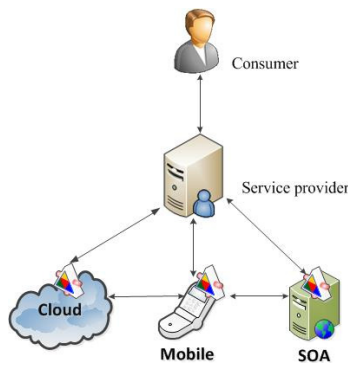


Figure 1: Replication environment interactions.

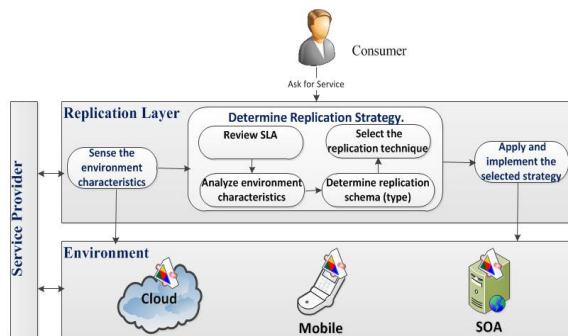


Figure 2: Generalized replication process.

environment, the nearest device or station may need to be selected to host the required replica.

Four major actors must be considered when designing a generic replication process for different distributed environments: consumers, service providers, replication actors, and the environment. As shown in Fig. 2, the replication process is divided into four layers representing these actors and their behaviour:

- *The consumer layer* represents the users who are seeking the published services. A service-level agreement (SLA) is enforced when the consumer binds with the target service. The service provider must work within the signed SLA to achieve the QoS terms listed in the SLA and attain high consumer satisfaction and loyalty.
- *The service provider layer* consists of the services or business owners. It continuously interacts with the replication and environment layers to host, publish, and enhance the business services that it owns or works on. Service providers monitor and update their services to keep current consumers and attract new consumers.
- *The replication layer* maintains the QoS parameters as defined in the SLA by providing additional replicas as needed.

- *The environment layer* could be a cloud, SOA, or mobile environment; in this layer, services or businesses are located and accessed. Moreover, this layer might be called the infrastructure layer because it contains all required physical and logical resources to host the services created by the various providers and targeted by consumers.

3.2 Generalized Replication Process: Steps

Figure 3 represents the basic interconnections between the defined replication process activities in Fig. 2 and the main actors or layers: the consumer, service provider, replication, and environment layers. To achieve a robust replication process, the following steps should be followed, as shown in Fig. 3:

1) Sense the environment characteristics: this activity occurs between the replication and environment layers to obtain the current status of the consumed services in terms of QoS characteristics.

In addition, the capabilities of the available resources in terms of functional characteristics are investigated. For example, this activity may search for a trusted server in a cloud to host a new replica or assign the nearest node to run the new replica in a mobile environment. Examples of the detected characteristics of the various environments are given in Table 1. The information obtained is saved in a database located inside the environment layer to be used in step 2.

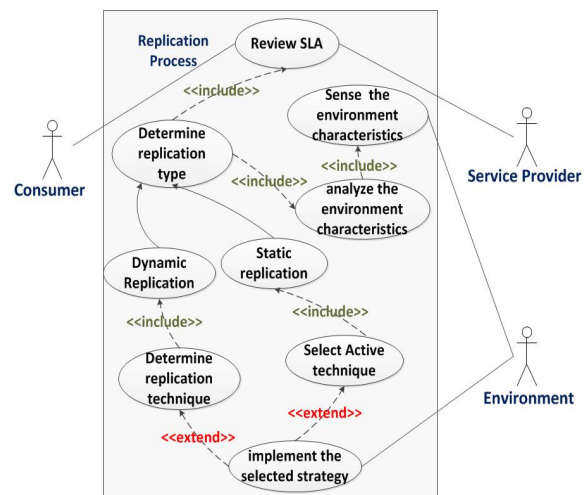


Figure 3: Replication process use-case model.

Table 1: Detected environmental characteristics.

Environment	Functional characteristics	Non-functional characteristics
SOA	Number of hops Number of servers	Server performance Server availability Type of WS: composite or basic.
Cloud	Number of replicas Trusted servers	Service performance Service availability Ensure service consistency. Type of service: deterministic or not
Mobile	Mobile battery power Number of replicas Nearest node	Service availability Network traffic

2) Determine the replication strategy: This step is composed of the four steps as shown in Fig. 2, which can be described as follows:

- a. Review SLA: this activity checks the signed SLA contract periodically or when a change in quality is detected by environmental sensing

with respect to the defined non-functional terms specified in the SLA. The SLA contract could be maintained inside the service provider layer or in the environment layer based on the agreement between providers, consumers, and environment owners. First, the SLA items are reviewed against current environment characteristics to determine the replication target (e.g., availability or performance) on which the replication layer must focus when a violation occurs. Then it notifies the provider and the consumer of the result. In addition, it retrieves the environment and functional capabilities from the first step to determine the candidate host for the new replica.

- b. Analyze Environment Characteristics: this activity examines the environment characteristics collected in Step 1 to match them with the characteristics of the corresponding selection algorithm. For example, as shown in Table 2, if the Hops replication selection algorithm is selected within a dynamic replication strategy, the number of hops should

Table 2: Expected strategy requirements.

Replica Selection Algorithm	Algorithm description	← Replication Type	← Actions
Ping/Probe	The client periodically sends a ping request to all available replicas and then forwards request(s) to the replica with minimal ping round-trip time.	Dynamic	Service performance
Hops	The client sends requests to the nearest replica according to the number of hops between the replica and the client.	Static/Dynamic	Number of hops
Parallel	The client sends requests to all available replicas. The one which works on the incoming request first responds to request and communicate with the consumer directly.	Static/Dynamic	In dynamic case, service availability
Probabilistic	Replica selection is based on a probability that has been calculated and assigned to each replica. Probability is calculated based on SLA.	Static/Dynamic	SLA review
Refresh/Best Last	The client sends requests to the server with the minimal request latency. Latency samples are refreshed periodically.	Dynamic	Service availability and performance
Best Median	Replica selection depends on the lowest median response time among the set of successful invocations recorded for each replica used.	Dynamic	Service availability and performance
Shortest Queue	Service selection depends on the smallest number of queued invocations as determined by locally maintained statistics related to service activities.	Dynamic	Service availability and performance
Round Robin	A list of active replicas is maintained and updated periodically by adding /removing the newly activated /deactivated replicas. Upon service replica selection, the requests are rotated around a list of active replicas.	Static/Dynamic	In dynamic case, service availability
Random	Random replica selection	Static/Dynamic	In dynamic case, service availability

be considered and estimated. At the end of this activity, the defined replication actions are saved in a database to be considered in the implementation activity.

- c. **Determine Replication Type:** Depending on the data collected from the previous activities, the detection process determines the type of replication, as shown in Table 3. This activity obtains the required information from the SLA Review and Analyze Environment Characteristics processes to make a decision about the replication type. Section 2 shows two different types of replication that can be used: static and dynamic replications. In static replication, only the active replication technique can be used. But within dynamic replication, all replication techniques can be used.

Table 3: Data needed to determine the replication type.

Environment	Target	Characteristic	Replication Type
SOA	Availability	Uses a fixed number of replicas Servers have average load performance	Static
	Performance	The number of replicas is needed during runtime Variable servers	Dynamic
Cloud	Availability	Services are deterministic Multicast consumer requests	Static
	Security	Services are deterministic or nondeterministic Multicasting not required	Dynamic
Mobile	Availability	Always runs in a dynamic environment	Dynamic

Table 4: Composition of the replication strategy.

Replication Type →	Replication Target →	Replication Techniques	Replication Strategy
Static	Availability	Active	Parallel
	Performance	Static load balancing	Round Robin Probabilistic
Dynamic	Availability	Active – Passive – Semi-Active	Ping or Probe, Refresh, Best Last, or Best Median.
	Performance	Dynamic load balancing	Distributed Round Robin Selection
	Responsiveness	Dynamic load balancing	Weighted Round Robin

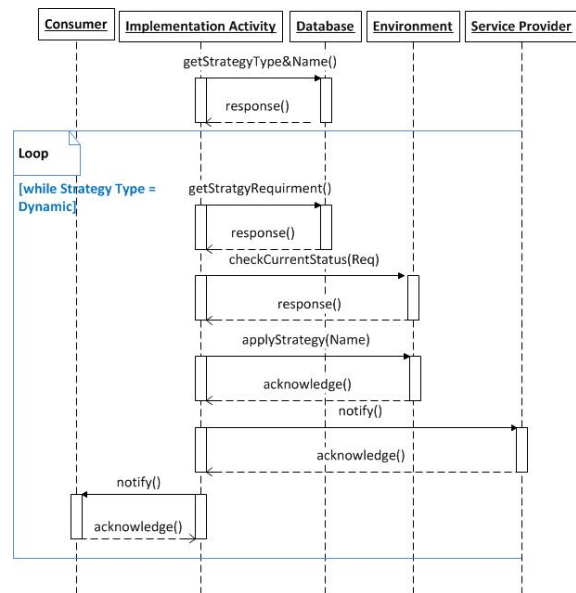


Figure 4: Sequence diagram for the Implement the Selected Replication Strategy activity.

- d. **Select Replication Technique:** Once the replication type and target have been determined, the process moves on to selecting the replication technique. Table 4 shows how the replication strategy is selected and composed (Steps 2a to 2d) based on the replication target, replication type, and selected replication technique.

3) Implement the Selected Replication Strategy: The sequence diagram shown in Fig. 4 illustrates how this process is executed. When a replica is needed, the implementation process obtains the strategy type and name to be used from the database. Basically, it verifies the selected strategy requirements against the current state of the deployed environment to ensure a correct replication process. Once the implementation has been accomplished successfully, a notification is sent to both the provider and consumer that their agreed QoS terms are still being achieved as expected.

4 MOBILE AUTHENTICATION CASE STUDY

This section describes a scenario from a telecommunication company in which a huge number of cell phones are authenticated when connected to the company network through a particular Web service called *mobile authentication service*. Eventually, this service lacks availability

and/or performance during peak times. In this paper, the steps of the introduced replication process are used to overcome the availability and performance challenges for this service. The architecture of the mobile authentication service environment can be divided into three layers: *The mobile layer* represents the users who are seeking mobile authentication service. *The portal layer* contains the replication management middleware which controls the interaction between consumers and the WSs replicas. *The cloud layer* contains all the physical and logical resources required to host the services. In such an environment, the replication management middleware (Portal) will follow the suggested replication steps as described in Section 3:

1) *Sense the environment characteristics*: As outlined in Table 1, the replication management middleware will collect the information; specifically, for this use case, the collected information is listed in Table 5. This case study assumes that three WS replicas are running on one virtual machine, but are installed on different ports. The WS is deterministic because for a given input, it always produces the same output. The user enters username and password, and then the system responds with “successful login” or not.

Table 5: Mobile authentication case study characteristics.

Environmental characteristics	General	Mobile authentication case study
Functional characteristics	Number of replicas	Three replicas used
	Trusted servers	One virtual server with Web services installed on three different ports
Non-functional characteristics	Performance	Required
	Availability	Required
	Ensure service consistency	Not required, because the WSs are retrieving data from a database. The case study makes no changes to data.
	Service type: deterministic or not	Deterministic service.

2) *Determine the replication strategy* consists of the following steps:

- Review SLA: The replica management middleware reviews the SLA to determine the replication target. The target of this case study is ensuring Web service availability and performance.
- Analyze Environment Characteristics: the replication management middleware will

analyze the strategic requirements depending on the environmental characteristics and SLA target. The selection process can be managed by consumers or the service provider. In this case study, the service provider manages the selection process, and therefore the Best Last and Best Median methods are ignored because they depend on consumer preferences; the consumer selects the replicas with lowest response time or median lowest response time depending on the invocation history.

Moreover, in this case study, all Web service replicas are placed on one virtual machine, and therefore the Hops and Probabilistic strategies are removed from the selection list. The Random strategy cannot provide the desired service availability because the failed copy may be selected, and therefore it is also removed. Moreover, the aim of the case study is to forward consumer requests to the best available WS; the load balancing carried out by Round Robin is eliminated. Hence, the selection list contains four strategies: Ping, Parallel, Refresh and Shortest Queue, as shown in Table 6.

Table 6: Analysis of case-study characteristics.

Replica selection algorithm	Strategic requirements	Selection list
Ping/Probe	Service availability	√
Hops	Number of hops	X
Parallel	Service availability.	√
Probabilistic	SLA review, host performance history.	X
Refresh	Service availability and performance	√
Best last	Service availability and performance	X
Best Median	Service availability and performance	X
Shortest Queue	Service availability and performance	√
Round Robin	Service availability	X
Random	Service availability	X

- Determine Replication Type: in this step, there are three options for replication type: static, dynamic service placement, and dynamic service selection. Depending on the cloud row in Table 3, static replication is ignored, and dynamic replication is used. The mobile authentication service deals with a large number of users, so that multicasting of consumers' requests to provide service availability is undesirable because it may cause network failure. Moreover, all replicas are installed on a

private cloud, and therefore dynamic service selection is chosen. Dynamic service replacement is preferable when multiple independent resources are available.

- d. **Select Replication Technique:** according to Table 6, there are four choices: Ping, Parallel, Refresh and Shortest Queue. The Parallel strategy is dropped from the selection list because multicasting of consumers' requests is not supported in this case study. The Shortest Queue strategy is out of consideration because it selects the replica with the lowest load to achieve the shortest response time, which is the same target as the Refresh strategy. The Ping selection process is based on the WS host/port with the lowest ping round-trip time, but the Refresh strategy depends on the WS with the lowest response time. Therefore, the preferred option in this case is the Refresh strategy.

3) **Implementation:** A simulation of this environment was constructed and run using the specifications shown in Table 7.

Table 7: Specifications of simulation environments.

Cloud	Google App Engine (Platform as a Service)
Portal	Apache/2.2.11 (Win32) PHP/5.3.0 Processors: Intel(R) core i3 Memory 4 GB
Mobile	Smart Phone

The implementation scenario can be described as follows:

- The replication management middleware (RMM) ensures that the cloud environment has three Web service replicas. If not, the middleware transfers the required replicas to the cloud environment.
- RMM notifies the users and the telecom company admin(s) to access the service. Users access authentication services. Each user types his/her username and password and presses Enter.
- RMM passes consumer requests to the best available mobile authentication service using the Refresh algorithm.
- The mobile authentication service processes the consumers' requests and sends a response back to replication management middleware. Then RMM forwards the results to the consumer.

The experiments were conducted using ApacheBench Version 2.0.40-dev by passing different consumer loads (1, 3, 5, 8) over 100 times

(100, 300, 500, 800 requests). Then throughput and response time were recorded for three cases: the case without replication and the Ping and Refresh strategies. As shown in Figs. 5 and 6, the Refresh replication strategy provides better throughput rates than non-replication. The Refresh strategy passes consumer requests to the best available service, so that the WS used can be changed during runtime; this leads to a balance in incoming requests between replicas, but not in an equally likely manner.

In the case of the Ping strategy, before every request, a ping was sent to all ports, and the port with minimal round-trip time was selected. The selection depends on the round-trip time of the ping message, not on the service response time, and therefore it is no better than the Refresh strategy. However, it is better than the case without replication for high consumer loads (5, 8) because the port can be changed during runtime, so that load balancing occurs, but not in an equally likely manner.

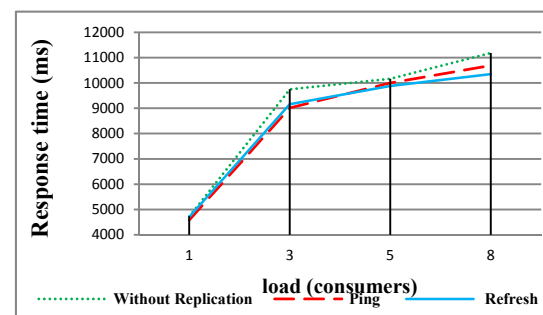


Figure 5: Response time when running 1700 requests.

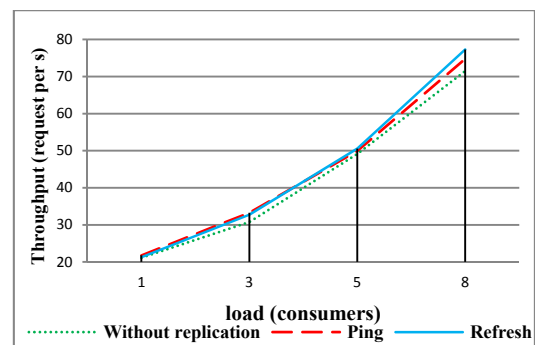


Figure 6: Throughput when running 1700 requests.

5 CONCLUSIONS

In this paper, a generalized service replication process was introduced to manage and control replication inside distributed environments. The

process consists of three steps: sensing the environment characteristics, planning a complete replication strategy, and implementing the selected replication strategy. The application of the generalized process is demonstrated in a case study involving a telecommunication scenario. The selected replication algorithm, the Refresh algorithm, was compared to the Ping algorithm and non-replicated service. Results show that the Refresh algorithm outperformed both Ping and non-replication in terms of throughput and response time.

Future work will include deploying the generalized replication process in a real-world environment and expanding the validation. In addition, it is planned to extend the review of the replication process to cover embedded systems and other distributed environments such as the Internet of Things (IoT) and cyber physical systems.

REFERENCES

- Al-Masri, E. & Mahmoud, Q. H., 2007. QoS-based discovery and ranking of Web services. In *Computer Communications and Networks, 2007 (ICCCN 2007), Proceedings of 16th International Conference*, pp. 529-534. IEEE.
- Björkqvist, M., Chen, L. Y., & Binder, W., 2012. Dynamic replication in service-oriented systems. *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGRID 2012)*, pp. 531-538. IEEE Computer Society.
- da Silva, J. A. F. & das Chagas Mendonça, N., 2004. Dynamic invocation of replicated Web services. *WebMedia and LA-Web, 2004. Proceedings*, pp. 22-29. IEEE.
- Dustdar, S. & Juszczak, L., 2007. Dynamic replication and synchronization of Web services for high availability in mobile ad-hoc networks. *Service Oriented Computing and Applications*, vol. 1, no. 1, pp. 19-33.
- Erl, T., 2008. *SOA: Principles of Service Design*, vol. 1. Upper Saddle River: Prentice-Hall.
- Erl, T., Puttini, R., & Mahmood, Z., 2013. *Cloud Computing: Concepts, Technology & Architecture*. Pearson Education.
- Fling, B., 2009. *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps*. O'Reilly Media.
- Guerraoui, R. & Schiper, A., 1997. Software-based replication for fault tolerance. *Computer*, vol. 30, no. 4, pp. 68-74.
- Keidl, M., Seltzsam, S., & Kemper, A., 2003. Reliable Web service execution and deployment in dynamic environments. In *Technologies for E-Services*, pp. 104-118. Springer, Berlin, Heidelberg.
- Liu, A., Li, Q., & Huang, L., 2011. Quality-driven Web services replication using directed acyclic graph coding. In *Web Information System Engineering (WISE 2011)*, pp. 322-329. Springer, Berlin, Heidelberg.
- May, N. R., Schmidt, H. W., & Thomas, I. E., 2009. Service redundancy strategies in service-oriented architectures. *Proceedings, Software Engineering and Advanced Applications, 2009 (SEAA'09) 35th Euromicro Conference*, pp. 383-387. IEEE.
- Michlmayr, A., Rosenberg, F., Leitner, P., & Dustdar, S., 2009. Comprehensive QOS monitoring of Web services and event-based SLA violation detection. *Proceedings, 4th International Workshop on Middleware for Service Oriented Computing*, pp. 1-6. ACM.
- Mohamed, M. F., ElYamany, H. F., & Nassar, H. M., 2013. A study of an adaptive replication framework for orchestrated composite Web services. *SpringerPlus*, vol. 2, no. 1, pp. 1-18.
- Papazoglou, M. P. & Van den Heuvel, W. J., 2005. Web services management: A survey. *Internet Computing, IEEE*, vol. 9, no. 6, pp. 58-64.
- Papazoglou, M., 2008. *Web Services: Principles and Technology*. Pearson Education.
- Salas, J., Perez-Sorrosal, F., Patiño-Martínez, M., & Jiménez-Peris, R., 2006. WS-replication: a framework for highly available Web services. *Proceedings of the 15th International Conference on World Wide Web*, pp. 357-366. ACM.
- Ślota, R., Nikolow, D., Skitał, Ł., & Kitowski, J., 2005. Implementation of replication methods in the grid environment. In *Advances in Grid Computing (AGC 2005)*, pp. 474-484. Springer, Berlin, Heidelberg.
- Sayal, M., Breitbart, Y., Scheuermann, P., & Vingralek, R., 1998. Selection algorithms for replicated Web servers. *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 3, pp. 44-50.
- Thakur, M. R. & Sanyal, S., 2012. A PAXOS-Based State Machine Replication System for Anomaly Detection. *arXiv Preprint, arXiv:1206.2307*.
- W3C Working Group Note: *Web Services Architecture*, 2004. Available from: <http://www.w3.org/TR/ws-arch/> [14 March 2015].
- W3C Working Group Note: *QoS for Web Services: Requirements and Possible Approaches*, 2003. Available from <http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/> [14 March 2015].
- Zheng, Z., & Lyu, M. R., 2008. A distributed replication strategy evaluation and selection framework for fault tolerant Web services. *Proceedings, Web Services 2008 (ICWS'08) IEEE International Conference*, pp. 145-152. IEEE.

Implementation of Cloud ERP

Moderating Effect of Compliance on the Organizational Factors

Shivam Gupta and Subhas C. Misra

*Department of Industrial and Management Engineering, Indian Institute of Technology Kanpur, Uttar Pradesh, India
shivamgt@iitk.ac.in, subhasm@iitk.ac.in*

Keywords: Cloud ERP, Compliance, Critical Success Factors (CSF).

Abstract: Cloud ERP has changed the way business can be done for Small and Medium Enterprises (SMEs). The two important benefits offered by Cloud ERP are: (a) SMEs can log into the internet from any place to access applications and data services at any point in the time. (b) Pay for the services that are used or needed. Although Cloud ERP has taken the IT world by storm and with all the advancement that has taken place so far, there are still issues and challenges that require to be addressed. This paper relates issues pertaining to Compliance with Organizational factors for successful implementation of Cloud ERP.

1 INTRODUCTION

Cloud ERP is a buzz word in the IT world and with all the advancement that has taken place so far, there are still issues and challenges that require to be addressed. Compliance issues need to be addressed for Cloud ERP implementation. These are discussed in detail.

2 CONCERN IN CLOUD ERP: COMPLIANCE

Compliance plays a vital role in decision making when any business process is moved into the cloud. Companies are not sure about the location of the data stored when it is on cloud. The data which might be safe in one country may not be safe in another country. In America, US Patriot Act gives limitless powers to government and its agencies to access any data. European Union (EU) has enforced strict measures so that cloud service providers can be tried in case of any data theft or breach of laws. The compliance concerns by EU have led to the creation US Safe Harbor Privacy Principles. This insulates European companies from the laws in USA that virtually gave unlimited powers to government agencies to snoop on any data. The issues that revolve around compliance are:

- Cloud based data archiving service should be able to classify, index, search and retrieve data

in a security-rich manner and complying with all government and industry regulations. If the cloud provider is successful in achieving this, then it helps control rising costs of data storage with a utility priced cloud-based service. The cloud user would at all times want to access, search and retrieve data from the cloud. Not able to do so can have an impact on decision making process and operational efficiency (IBM, 2011).

- Cloud based segregation of duties (SOD) can minimize error and fraud occurrences. Any Individual should not have complete security access to a series of transactions which could allow him or her to engage in financial misconduct. Individual can collude with a vendor to (a) receive and pay for fictitious goods or services or (b) pay for services with company's money to be used for personal gain. SOD increases the compliance standards for data and work handling issues but this can negatively impact the business efficiency and also increase costs and staffing requirements. The mission critical elements of the business and sensitive data should be brought under the purview of SOD.
- Global compliance standards and regulations vary from country to country. There is lack of governmental regulations which can impose varying compliance requirements and standards on the industries. In Germany, it is not permitted to relocate auditable information

which can be considered as critical data to a server outside the country. This hosting of information in the cloud outside Germany violates the German laws (Seitz, 2010).

3 CRITICAL SUCCESS FACTORS (CSFS) FOR CLOUD ERP: ORGANIZATIONAL FACTORS

Nguyen (2011) has stated that identification of CSF's is important to attain the desired goals laid down by the business. In relation to CSFs for ERP, it can be explained as conditions that can lead to a successful ERP adoption and implementation (Finney and Corbett, 2007).

The success factors for Cloud ERP implementation from an organizational point of view are considered here. These success factors identified by literature review can be better understood by going through the existing theories which will examine the relationship between CSFs and Cloud ERP.

3.1 Contingency Theory

Suggests that organization culture should be conducive for any change to take place effectively. The manner in which any organization reacts to the change can be critical for the successful adoption and implementation of Cloud ERP. The employees should be given adequate training and guidance so that they do not offer much resistance in transitioning to the usage of Cloud ERP. The independent variable **organization resistance** is better explained by this theory. Literature review (LR) which supports this CSF are (Bingi et al., 1999; Holland and Light, 1999; Ross and Vitale, 2000; Mehrtens et al., 2001; Kumar et al., 2002; Zhang et al., 2002; Abdinnour-Helm et al., 2003; Olson et al., 2007; Saeed et al., 2011; Hasibuan and Dantes, 2012; Utzig et al., 2013).

3.2 Knowledge based View

Lays emphasis on the fact that knowledge can be utilized to achieve competitive advantage. The knowledge that is created over a period of time within an organization acts as a repository which can always be referred or even build upon any newer strategy. Independent variables **communication** (Kumar et al., 2002; Grant, 2003; Mabert et al., 2003; Mandal and Gunasekaran, 2003; Somers and

Nelson, 2004; Yusuf et al., 2004; Nah and Delgado, 2006; Hasibuan and Dantes, 2012) and **implementation strategy** belongs to this theory (Bancroft et al., 1998; Davenport, 1998; Cliffe, 1999; Holland and Light, 1999; Trepper, 1999; Davenport, 2000; Gupta, 2000; O'Leary, 2000; Scott and Vessey, 2000; Motwani et al., 2002; Robey et al., 2002; Mandal and Gunasekaran, 2003; Umble et al., 2003).

3.3 Market based View

Determines the nature of any organization strategy based on the trends of the industry's environment. Rivalry between competitors and threat of new entrant are factors which shape up the nature of strategies implemented by companies. A lot of this is also dependent on the budget of the companies which are doing business or wanting to enter the market. Cloud ERP can offer the services at low rates and this fits well within the financial limitations for a SME. **Project budget** is the independent variable that is best explained by this theory (Bingi et al., 1999; Holland and Light, 1999; Davenport, 2000; Al-Mudimigh et al., 2001; Willis et al., 2001; Ribbers and Schoo, 2002; Trimmer et al., 2002; Somers and Nelson, 2004; Ellis, 2010; Hasibuan and Dantes, 2012).

3.4 Social Capital Theory

Stresses upon the various social capital that exists in an organization like the values and goals. These goals and values coupled with effective leadership can lead to successful implementation of Cloud ERP. **Strategic goals and objectives** as an independent variable explains the importance for successful implementation (Buckhout et al., 1999; Akkermans and Helden, 2002; Al-Mashari et al., 2003; Mandal and Gunasekaran, 2003; Somers and Nelson, 2004; Calogero, 2000; Pabedinskaite, 2010; Hasibuan and Dantes, 2012).

3.5 Strategic Choice Theory

Focuses that the people of an organization can shape the environment around them. If the environment in the organization is nurtured in a way where the existing as well as new projects can be executed without much delay, then the transition to a newer ERP system and its implementation would never pose any problem. For this, the company should be comfortable in shaping up the existing processes in a different manner. The independent variables

considered here are **Business Process Re-engineering (BPR)** (Bingi et al., 1999; Holland and Light, 1999; Bernroider and Koch, 2000; Al-Mudimigh et al., 2001; Kraemmergaard and Rose, 2002; Palaniswamy and Frank, 2002; Trimmer et al., 2002; Zhang et al., 2002; Al-Mashari et al., 2003; Mabert et al., 2003; Muscatello et al., 2003; Bajwa et al., 2004; Hasibuan and Dantes, 2012) and **project management** for successful Cloud ERP implementation (Hoffer et al., 1998; Trepper, 1999; Nah et al., 2003; Akkermans and Helden, 2002; Zhang et al., 2002; Umble et al., 2003; Somers and Nelson, 2004; Bhatti, 2005; Nah and Delgado, 2006; Hasibuan and Dantes, 2012).

4 MEASURING SUCCESS

The biggest beneficiaries out of Cloud ERP implementation will be cloud user. The cloud provider aim will be to beat the competition and generate more and more revenue. This can happen by adding more clients to their existing user base and also convincing companies who have never used ERP solution to switch to an affordable service. But this is directly related to the user acceptance and usage of Cloud ERP services. While there can be different viewpoints about the measure of success, an a-priori framework is proposed which will qualitatively address the research objective. This framework is based on intuition and using previously published literature on ERP and Cloud ERP.

From the viewpoint of a project-manager; time, cost, productivity, and customer satisfaction are the main ingredients of any project's successful completion (Schwaber and Beedle, 2002; PMI, 2004; Parthasarathy, 2007).

Based on balanced scorecard terminology (Kaplan and Norton, 1996), following are the success criteria taken for this study:

- S1 Lower Implementation Cost
- S2 Ease of Use and Reporting
- S3 Lower wait time for consumer
- S4 Increase in Customer Retention
- S5 Increased Ability to meet with Current User Requirements
- S6 Increased Flexibility to meet with Changing User Requirements

These criteria form the constituents of the dependent variable ("Success") in this paper.

5 RESEARCH OBJECTIVE

- RO 1 Development of a framework with the different determinants of compliance for the successful implementation of Cloud ERP.
- RO 2 To establish and verify the role of Compliance on the organizational factors for successful implementation of Cloud ERP.

6 VARIABLES IN THE MODEL

- Independent Variable: Organizational Factors
- Moderating Variable: Compliance
- Dependent Variable: Cloud ERP Successful Implementation

7 CONCEPTUAL MODEL

Moderating Effect of Compliance on the Organizational Factors in Cloud ERP Implementation can be seen below.

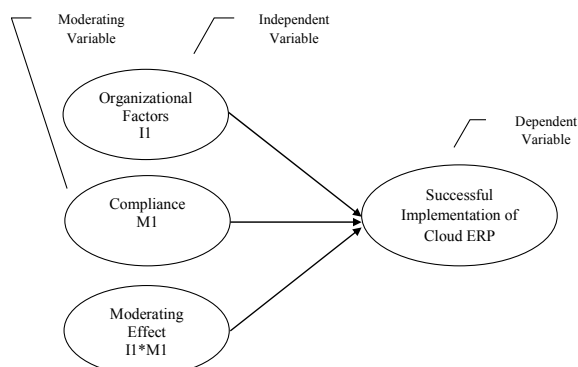


Figure 1: A-Priori Conceptual Model.

8 CONCLUSIONS

This paper brings out a conceptual model based on the literature review as well as various theories which tries to establish the link between CSFs and Cloud ERP implementation. This model can be empirically tested so that it can be used as a tool for assessing the implementation of Cloud ERP.

REFERENCES

Akkermans, H. and Helden, Van (2002), "Vicious and

- virtuous cycles in ERP implementation: a case study of interrelations between critical success factors", *European Journal of Information Systems*.
- Al-Mashari, M., Al-Mudimigh, A. and Zairi, M. (2003), "Enterprise resource planning: a taxonomy of critical factors", *European Journal of Operational Research*, Vol. 146.
- Al-Mudimigh, A., Zairi, M. and Al-Mashari, M. (2001), "ERP software implementation: An integrative framework", *European Journal of Information Systems*, Vol. 10.
- Bajwa, D.S., Garcia, J.E and Mooney, T. (2004), "An integrative framework for the assimilation of enterprise resource planning systems: phases, antecedents, and outcomes", *Journal of Computer Information Systems*, Vol. 44.
- Bancroft, N., Seip, H. and Sprengel, A. (1998), "Implementing SAP", 2nd edn., Manning Publications, Greenwich, CT, USA.
- Bernroider, E. and Koch, S. (2000), "Differences in characteristics of the ERP system selection process between small, medium and large organizations", *Proceedings of the Sixth Americas Conference on Information Systems*, Long Beach, California, USA.
- Bingi, P., Sharma, M.K. and Godla, J. (1999), "Critical issues affecting an ERP implementation", *Information Systems Management*, Vol. 16.
- Bhatti, T.R (2005), "Critical success factors for the implementation of enterprise resource planning (ERP): empirical validation", *The Second International Conference on Innovation in Information Technology (IIT)*.
- Buckhout, S., Frey, E. and Nemec, J. (1999), "Making ERP succeed: turning fear into promise", *Strategy and Business*, Second Quarter Issue 15.
- Calogero, B. (2000), "Who is to blame for ERP failure?", *Sun Server*, June Issue.
- Cliffe, S. (1999), "ERP implementation", *Harvard Business Review*, Vol. 77.
- Davenport, T.H. (1998), "Putting the enterprise into the enterprise system", *Harvard Business Review*, Vol. 76 No. 4.
- Davenport, T.H. (2000), "Mission critical: realizing the promise of enterprise systems", *Harvard Business School Press*.
- Ellis, Simon (2010), "Software-as-a-Service ERP versus on-premises ERP through the lens of total cost of ownership", *Plex Systems*, White Paper.
- Finney, S. and Corbett, M. (2007), "ERP implementation: a compilation and analysis of critical success factors", *Business Process Management Journal*.
- Grant, G.G. (2003), "Strategic alignment and enterprise systems implementation: the case of Metalco", *Journal of Information Technology*, Vol. 18.
- Gupta, A. (2000), "Enterprise resource planning: the emerging organizational value systems", *Industrial Management and Data Systems*, Vol. 100.
- Hasibuan, Z.A. and Dantes, G.R. (2012), "Priority of key success factors (KSFS) on enterprise resource planning (ERP) system implementation life cycle", *Journal of Enterprise Resource Planning Studies*.
- Holland, C.P. and Light, B. (1999), "A critical success factors model for ERP implementation", *IEEE Software*, Vol. 16.
- Hoffer, J.A., George, J.F. and Valacich, J.S. (1998), "Modern systems analysis and design", 2nd edn., Addison-Wesley Reading, MA, USA.
- IBM Data Sheet (2011), "Cloud-based data archiving service", *IBM Global Technology Services*.
- Kaplan, R.S. and Norton, D.P. (1996), "Using the balanced scorecard as a strategic management system", *Harvard Business Review*, Jan-Feb issue.
- Kraemmergaard, P. and Rose, J. (2002), "Managerial competences for ERP journeys", *Information Systems Frontiers*, Vol. 4.
- Kumar, V., Maheshwari, B. and Kumar, U. (2002), "ERP systems implementation: best practices in Canadian government organizations", *Government Information Quarterly*, Vol. 19.
- Mabert, V.A., Soni, A. and Venkataramanan, M.A. (2003), "Enterprise resource planning: managing the implementation process", *European Journal of Operational Research*, Vol. 146.
- Mandal, P. and Gunasekaran, A. (2003), "Issues in implementing ERP: a case study", *European Journal of Operational Research*, Vol. 146.
- Mehrtens, J., Cragg, P.B. and Mills, A.M. (2001), "A model of internet adoption by SMEs", *Information and Management*, Vol. 39.
- Motwani, J., Mirchandani, D., Madan, M. and Gunasekaran, A. (2002), "Successful implementation of ERP projects: evidence from two case studies", *International Journal of Production Economics*, Vol. 75.
- Muscattello, J.R., Small, M.H. and Chen, I.C., (2003), "Implementing enterprise resource planning (ERP) systems in small and midsize manufacturing firms", *International Journal of Operations and Production Management*, Vol. 23 No. 8.
- Nah, F.F.H., Zuckweiler, K.M. and Lau, J.L.S. (2003), "ERP implementation: chief information officers' perceptions of critical success factors", *International Journal of Human-Computer Interaction*, Vol. 16, No. 1.
- Nah, F.F.H. and Delgado, S. (2006), "Critical success factors for enterprise resource planning implementation and upgrade", *Journal of Computer Information Systems*, Vol. 46, No. 5.
- Nguyen, H.V. (2011), "Critical success factors for ERP adoption process: a Vietnamese case approach", *LAP Lambert*.
- O'Leary, D.E. (2000), "Enterprise resource planning system: systems, life cycle, electronic commerce, and risk", *Cambridge University Press*.
- Olson, L.D. (2007), "Evaluation of ERP outsourcing", *Computers and Operations Research*, Volume 34, Issue 12.
- Pabedinskaite, Arnoldina (2010), "Factors of successful implementation of ERP systems", *Economics and Management*.

- Palaniswamy, R. and Frank, T.G. (2002), "Oracle ERP and network computing architecture: implementation and performance, information systems management", Vol. 19.
- PMI (2004), Project Management Body of Knowledge, 3rd edn., Project Management Institute, USA.
- Parthasarathy, S. (2007), "Enterprise resource planning: a managerial and technical perspective", New Age International, New Delhi, India.
- Ribbers, P.M.A. and Schoo, K-C. (2002), "Program management and complexity of ERP implementations", Engineering Management Journal, Vol. 14.
- Robey, D., Ross, J.W. and Boudreau, M-C. (2002), "Learning to implement enterprise systems: an exploratory study of the dialectics of change", Journal of Management Information Systems, Vol. 19.
- Saeed, Imran, Juell-Skielse, Gustaf and Uppstrom, Elin (2011), "Cloud enterprise resource planning adoption: motives and barriers", Advances in Enterprise Information Systems II.
- Ross, J.W. and Vitale, M.R. (2000), "The ERP revolution: surviving vs thriving", Information Systems Frontiers, Vol. 2.
- Schwaber, K. and Beedle, M. (2002), "Agile software development with scrum", Prentice Hall, New Jersey, USA.
- Seitz, Timm (2010), "SAP ERP in the cloud", An Oracle White Paper.
- Scott, J.E. and Vessey, I. (2000), "Implementing enterprise resource planning systems: the role of learning from failure", Information Systems Frontiers, Vol. 2.
- Somers, T.M. and Nelson, K.G. (2004), "A taxonomy of players and activities across the ERP project life cycle", Information and Management, Vol. 41.
- Trepper, C. (1999), "ERP project management is a key to a successful implementation".
- Trimmer, K.J., Pumphrey, L.D. and Wiggins, C. (2002), "ERP implementation in rural health care", Journal of Management in Medicine, Vol. 16.
- Umble, E.J., Haft, R.R. and Umble, M.M. (2003) "Enterprise resource planning: implementation procedures and critical success factors", European Journal of Operational Research, Vol. 146.
- Utzig, C., Holland, D., Horvath, M and Manohar, M. (2013), "ERP in the cloud: is it ready? are you?", Booz and Co.
- Willis, T.H., Willis-Brown, A.H. and McMillan, A. (2001), "Cost containment strategies for ERP system implementation", Production and Inventory Management Journal, Vol. 42, No. 2.
- Yusuf, Y., Gunasekaran, A. and Abthorpe, M.S. (2004), "Enterprise information systems project implementation: a case study of ERP in Rolls-Royce", International Journal of Production Economics, Vol. 87.
- Zhang, L., Lee, M.K.O., Zhang, Z. and Banerjee, P. (2002), "Critical success factors of enterprise resource planning systems", 35th Hawaii International Conference.

User Requirement and Behavioral Aspects in Web Service Discovery

Wala Ben Messaoud¹, Khaled Ghedira¹ and Youssef Ben Halima²

¹*Institute ENSI University of Manouba/SOIE, 41 Liberty Street Bouchoucha, Bardo, Tunisia*

²*ENSI University of Manouba/RIADI Labs, Manouba University Campus, Manouba, Tunisia*

Keywords: Behavioral Aspect, State Chart Scheduling, Web Service, Web Service Discovery, WordNet.

Abstract: In web service (WS) discovery, behavioral aspect has been defined as the sequence of WS operations. The motivation to introduce the behavioral aspect is to offer to the consumer the possibility to choose his WS according to his requirements. The aim is to include the execution manner of WS operations as a new criterion and to apply a selection method if more than one WS candidate is filtered. In this paper, we envision to implement WS discovery approach based on behavioral aspects to fulfill the selection of the precise execution order. This approach ensures an execution order of operations in accordance to consumer needs. The execution manner criterion is defended by state chart as a scheduling method and WordNet as a lexical database. Moreover, semantic equivalences have to be considered in order to solve equivalence between many WS candidates which satisfy consumer needs.

1 INTRODUCTION

The literature on web service discovery is almost common in recognizing the existence of a major problem in the WS consumer's requirements. Current work on service discovery focuses on discovery types not on the analysis of consumer intervention.

WS discovery is the process of satisfaction of a user request according to his requirements. It refers to the process of finding WS that implements the technique of search desired, interviewing service books, to know what WS is available for binding. Our approach unlike the other discovery approaches, allows WS consumer to involve his exigency by entering some sentences as a WS query. The aim is to satisfy WS consumer by analyzing his inputs.

The WS consumer requires a new aspect allowing functional phase (organized operations, free operations) and non functional phase (cost, time, availability). As a solution, we define the behavioral aspect as the execution manner of WS operations. Indeed, a consumer who requires looking for a WS with a tidy list of exigencies may not be satisfied by ordinary WS discovery. Actually, the motivation to introduce the behavioral aspect is to offer to the consumer the possibility to choose his WS according to his needs. The aim is to include requirement criterion as a new aspect and to apply a selection method if more than one WS candidate are

filtered. The behavior aspect should guarantee quality of service (cost, reliability, time ...).

Our approach consists on using Statechart as a scheduling method to highlight the execution order of WS operations and WordNet as a lexical database to prove the semantic part of consumer query. We used BPEL4WS (Business Process Execution Language for Web Services) to specify and execute business process for WS composition and orchestration. More precisely, to explain how WS operations are invoked and executed.

The remainder of the paper is organized as follows: Section 2 presents some concepts used in our approach. Our solution is reported in section 3. Section 4 explains more our approach by an example. In section 5, we feature the related work. Conclusions are presented in section 6.

2 CONCEPTS

Our approach brings answers to many posed issues. It is based on some concepts facilitating the implementation of the different phases of the approach like WordNet, intending to find synonyms to all consumer inputs.

In the goal to order these inputs, we use state charts where the WS operations are the transitions allowing to move from one state to another. Once we have prepared the state chart part, we launch the

semantic aspect of WS discovery, we extract the BPEL sequencing of each WS found and then we select the relevant one.

2.1 WordNet

The goal of WordNet was to develop a system that would be consistent with the knowledge acquired over the years about how human beings process language.

In (Miller, 1995), WordNet is defined as a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

WordNet is a large semantic network interlinking words and groups of words by means of lexical and conceptual relations represented by labeled arcs. WordNet's building blocks are synonym sets (synsets), unordered sets of cognitively synonymous words and phrases (Christiane, 2005).

The authors in (Morato, 2004), define WordNet as one of a series of manually compiled electronic dictionaries, is restricted to no specific domain and covers most English nouns, adjectives, verbs and adverbs. WordNet offers researchers a cost-free use and well-documented open code. It is an ideal tool for disambiguation of meaning, semantic tagging and information retrieval.

In our work, the consumer inputs is a set of keywords that define his requirements. If we don't specify clearly these inputs, we risk falling in linguistic polysemy case. So, we seek to find synonyms to all consumer inputs to properly filter the service concerned. That's why we have chosen to use WordNet as ontology.

2.2 Statecharts - Automaton

In literature, authors define statechart as visual formalism for description of complex systems behaviour. Digital controllers, which act as reactive systems, can be very conveniently modelled with statecharts and efficiently synthesized in modern programmable devices (Łabiak, 2010).

In formal grammar, an alphabet Σ is a finite and not empty set of symbols. Σ^* is the closure of Σ . A language on an alphabet Σ is a subset of the set Σ^* .

A transition diagram allows achieving an operational vision of the concept of language. It is a finite collection of states and transitions.

The statechart notation was developed by David Harel (Harel, 1987). Statechart diagrams are useful

for modelling the lifetime of an object. They are used to describe the system behavior using a finite automaton.

The automaton is represented as a directed graph known as state graph which consists of a finite set of vertices known as nodes, together with a set of directed links between pairs of vertices called arcs. Vertices are represented by circles and arcs by arrows. We can also represent an automaton with a state-transition table (Bhattacharjee, 2014).

The transition table can be associated to the automaton which describes extensively the transition function δ . A column is a character of the alphabet. A line is a state of the automaton (the initial state is preceded by an arrow " \rightarrow ", the final state is preceded by a star "*").

The value $\delta(q, a)$ for $q \in Q$, $a \in \Sigma$ corresponds to the state at the intersection of the row q and the column a . Note that from this table it is easy to find all the statements and the alphabet and thus identify exactly the automaton.

An automaton reads a finite string of symbols a_1, a_2, \dots, a_n , where $a_i \in \Sigma$, which is called an *input word*. The set of all words is denoted by Σ^* . *Accepting word is the word* $w \in \Sigma^*$ which is accepted by the automaton if $q_n \in F$. An automaton can recognize a *formal language*. The language $L \subseteq \Sigma^*$ recognized by an automaton is the set of all the words that are accepted by the automaton. The Deterministic Finite Automata has a finite internal memory available. At each input letter the state of the internal memory is changed depending on the letter scanned.

The previous memory state and the input letter together determine what the next state of the memory is. The word is accepted if the internal memory is in an accepting state after scanning the entire word (Kari, 2013).

In our approach, we use automaton as a scheduling method of WS operations (where we order the WS operations in the aim to respect the non functional/functional properties). Each operation represents a transition from a state to another. The automation generates a language (ordering list of operations) that specify the words to accept.

2.3 WSDL

A WSDL document is, at its simplest, a collection of elements contained within a root definition element. These elements describe a service and how an endpoint implementing that service is accessed (ALBRESHNE, 2009)

Each WSDL includes two parts, the abstract and the concrete. Abstract part describes the messages

sent and received. The operation associates a message exchange pattern with one or more messages.

As for concrete part, it specifies transport and wire format details for one or more interfaces, a port (an endpoint) associates a network address with a binding and a service which groups together endpoints that implement a common interface.

An operation is similar to a function in a high level programming language. A message exchange is also referred to as an operation. Operations are the focal point of interacting with the service.

In our work, we use WSDL for each WS to extract the execution order of operations.

2.4 BPEL4WS

BPEL for WS is a programming language for the execution of business processes. It is based on WSFL (Web Services Flow Language) and XLANG (XML LANGuage), is derived from XML.

According to (Milanovic, 2004) and (OASIS, 2007), BPEL composes web services to get a specific result. The composition result is a named process, partners are defined as services and activities are described by exchanged messages. In other words, a process contains a set of activities and it invokes external partner services using a WSDL interface.

BPEL is used to describe the execution of business process implicating WS. It consists on business BPEL allowing communication with WS, handling XML Data and managing exceptions.

BPEL provides several structure activities:

- <sequence>: define an ordered sequence of WS activities.
- <flow>: define parallel activities.
- <switch>: Case-switch construct for implementing branches.
- <while>: define loops.
- <pick>: select one of several alternative paths.

In WS discovery, there are three layers; operational, organizational and intentional. Intentional layer enables modeling purposes. It is a conceptualization of strategic needs of required modeling by an individual subject, group of individuals, a work unit or organization that involved in the system development process.

In some cases, the WS consumer cannot be a domain expert, he launch his query by entering a list of sentences. These sentences will be transformed to keywords by an algorithm to facilitate the satisfaction process. The consumer inputs can be a list of WS that need to be orchestrated as operations.

3 SOLUTION: BEHAVIORAL BASED APPROACH

The WS consumer seeks his WS with a list of well-defined requirements (functional and non-functional) but he is not satisfied by the ordinary types of discovery.

Syntactic discovery aims to compare between the syntactic query based on keywords and syntactic descriptions of WS. WS consumer launch his query by entering a set of keywords, the result of the comparison between syntactic query based on keywords and syntactic descriptions of the services is a set of WS that don't satisfy the consumer exigencies (the WS name is exactly the keyword entered by the consumer but the content has not the same meaning sought).

The semantic discovery is mainly based on ontology, defined as structured set of terms and concepts representing the meaning of information field, developed to facilitate knowledge sharing and reuse. WS consumer launch his query by entering a set of keywords, the result is a set of WS that don't satisfy the consumer exigencies. For this reason, we intend to define a new WS discovery approach based on behavioral aspect according to our definition. We define the behavior as the execution manner of WS operations. WS discovery approach based on behavioral aspect ensures an execution order of operations in accordance to consumer needs.

The purpose of this paper is to design a discovery technique for choosing the WS with the most relevant behavior. So, the query language to develop should be based on system statechart. It is used to check the compatibility of behavior required by the customer and those of WS found.

The work requires its valuation by an implementation that acquires to client to get his WS with the execution order of the desired operations. This implementation creates a WS automaton according to the operations order. WS automaton accepts only languages that correspond to it. These languages will be generated from BPEL files (Business Process Execution Language) of WS operations searched in WS directories. Some languages will be selected if they are accepted by the WS automaton.

Figure 1 shows that the implementation is done in three phases.

3.1 Phase 1 - Transform Consumer Sentences to Keywords

If the WS consumer is not a domain expert, he may enter his requirements as sentences. An algorithm is defined to transform each sentence in a keyword used in Phase 2.

3.2 Phase 2 - Create WordNet File for Each Keyword - Create Automaton

WordNet is used to define synonyms file for each keyword entered by the consumer. The analysis of consumer inputs is done on a semantic level not on a syntactic level.

In the same time, these keywords allow to create the automaton corresponding to the behavior of searched WS (defining the standard language to accept).

The automaton is defined by an initial state, a list of transitions and a list of final states. A number of states are defined according to the consumer inputs. To switch from a state to another, we must pass through a transition. Consumer keywords define transitions.

3.3 Phase 3 - Extract Sequences from WSDL/BPEL Files of WS Searched - Select the Relevant WS

Consists on searching WS semantically (Semantic Aspect) basing on non functional properties (NFP). For each WS selected, we extract the list of its operations from WSDL (if it is a simple WS) or from BPEL file (if it invoke other WS). The extracted sequencing is transformed to word that will be accepted or rejected by the automaton

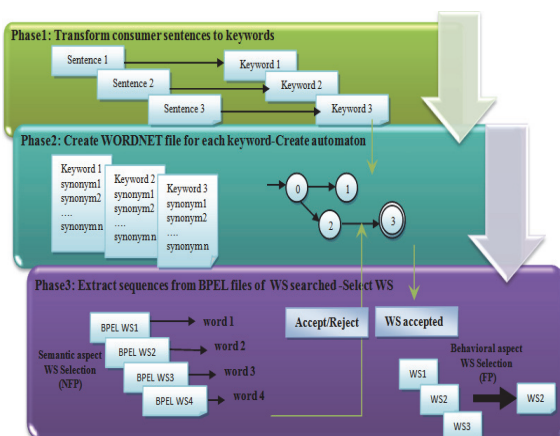


Figure 1: Implementation Steps of Behavioral approach.

already defined.

Accepted words correspond to WS accepted. The last step in Phase 3 is to select the most relevant WS basing on functional properties (FP) that we called (Behavioral Aspect).

4 EXAMPLE: STAY RESERVATION

Let's consider the example named «Stay reservation» in Figure 2, where the consumer wishes to book a plane ticket, rent a car, buy a concert ticket and book a hotel stay (or rent a house) with a heated pool. All these operations should guarantee minimum transfer cost and reduced transfer time.

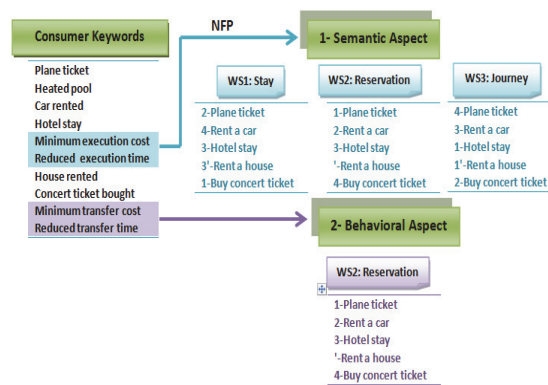


Figure 2: Behavioral WS discovery.

The consumer launches his query by entering a list of sentences such as «Plane ticket», «Heated pool», «Rent a car», «Hotel stay», «Minimum execution cost», «Reduced execution time», «Rent a house», «Buy concert ticket», «Minimum transfer cost» and «Reduced transfer time».

The first step is to specify NFP from FP. NFP are used in semantic aspect as selection properties. In this example, «Minimum execution cost» and «Reduced execution time» are NFP. As a result, we find WS1 named «Stay», WS2 named «Reservation» and WS3 named «Journey». All of this WS satisfy the consumer needs semantically.

For the behavioral aspect, «Minimum transfer cost» and «Reduced transfer time» are chosen as FP to select the most relevant WS.

To explain in more detail, we follow the steps mentioned previously:

- Transform consumer sentences to keywords:

Phase 1 presented in Figure 3 consists on defining an algorithm to transform consumer inputs to keywords

usable in phase 2.

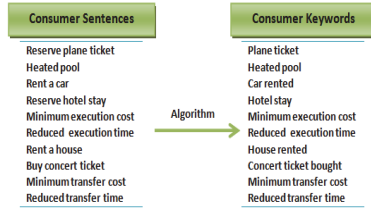


Figure 3: Stay Reservation: Phase 1.

➤ *Create WordNet file for each keyword:*

The system launches WordNet. As a result, we get for each keyword a file with a list of its synonyms. Each row of WordNet file is a keyword synonym. Figure 4 presents two examples respectively bonded to keywords «Plane ticket» and «Car rented».

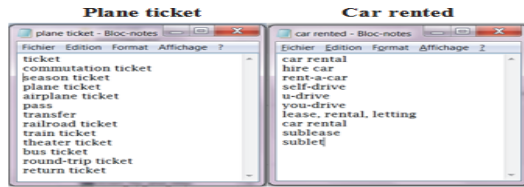


Figure 4: WordNet result.

➤ *Create automaton:*

At the same time, to respect consumer's requirements, we should consider it to create automaton.

The automaton in Figure 5 shows all states that the WS «Stay reservation» can be in during the course of its life. Furthermore, it shows the possible transitions between the states and the events that initiate these transitions.

We follow the automaton to check the compatibility of behavior required by the customer and those of WS found. The following abbreviations are used to develop the automaton.

Firstly, we define the states:

- 0: Stay not reserved
- 1: Plane Ticket reserved
- 2: Car rented
- 3: Hotel Stay reserved
- 4: House rented
- 5: Concert Ticket bought

Secondly, we define the transitions by the following abbreviations:

- Reserve_ticket : P
- Rent_car : C
- Reserve_Hotel : S
- Rent_house : H
- Buy_Concert_ticket : B

The WS automaton $(Q, \Sigma, \delta, q_0, F)$ is defined by: $Q = \{0, 1, 2, 3, 4, 5\}$: the states number depends on operations number. (It is equals to the real operations number +1). The real operations number is calculated by neglecting free operations and for the parallel operations we just count one operation. In our case, we have in general five operations. If we neglect the free operation (Buy_Concert_ticket) and we count just one operation for the two parallel operations (Reserve_Hotel and Rent_House), we will have a number of four real operations. So, the states number equals to five (the real operations number=4 +1).

$\Sigma = \{P, C, S, H, B\}$

$\delta(0, P)=1, \delta(0, B)=5, \delta(1, C)=2, \delta(1, B)=5,$
 $\delta(2, S)=3, \delta(2, H)=4, \delta(2, B)=5, \delta(3, B)=5,$
 $\delta(4, B)=5, \delta(5, P)=1, \delta(5, C)=2, \delta(5, S)=3,$
 $\delta(5, H)=4$

The transitions table is represented by a matrix where the rows are the states and the columns are the operations.

$q_0 = 0$: the initial state is always the 0.

$F = \{3, 4, 5\}$: final state is defined if we look over all the transitions (for the parallel operations, we count just one). In this example, to achieve 3, we pass by PBCS. To achieve 5, we pass by PCSB. To achieve 4, we pass by PBCH.

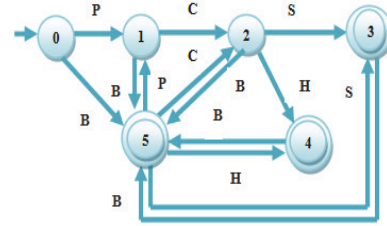


Figure 5: Automaton of WS «Stay reservation».

Then we deduce the transition table to facilitate the extraction of language to accept:

Table 1: Transition table.

	P	C	S	H	B
→0	{1}	∅	∅	∅	{5}
1	∅	{2}	∅	∅	{5}
2	∅	∅	{3}	{4}	{5}
*3	∅	∅	∅	∅	{5}
*4	∅	∅	∅	∅	{5}
*5	{1}	{2}	{3}	{4}	∅

According to the transitions table, the accepted expressions are:

- {PBCS}: 1- Reserve_ticket 2- Buy_Concert_ticket 3- Rent_car 4- Reserve_Hotel.
- {PBCH}: 1- Reserve_ticket 2- Buy_Concert_ticket 3- Rent_car 4- Rent_house.
- {PCBS}: 1- Reserve_ticket 2- Rent_car 3- Buy_Concert_ticket 4- Reserve_Hotel.
- {PCBH}: 1- Reserve_ticket 2- Rent_car 3- Buy_Concert_ticket 4- Rent_house.
- {PCSB}: 1- Reserve_ticket 2- Rent_car 3- Reserve_Hotel 4- Buy_Concert_ticket.
- {PCHB}: 1- Reserve_ticket 2- Rent_car 3- Rent_house 4- Buy_Concert_ticket.
- {BPCS}: 1- Buy_Concert_ticket 2- Reserve_ticket 3- Rent_car 4- Reserve_Hotel.
- {BPCH}: 1- Buy_Concert_ticket 2- Reserve_ticket 3- Rent_car 4- Rent_house.

So, all other execution enchainment of WS operations will be rejected.

- *Extract sequences from WSDL or BPEL files of WS searched:*

The first step is to launch WS discovery semantically basing on NFP «Minimum execution cost» and «Reduced execution time». As a result, we find WS1«Stay», WS2«Reservation» and WS3«Journey». This step is an ordinary WS discovery based on semantic aspect that we can find in many works like (Kritikos, 2007).

The second step is to extract the sequencing of WS operations from the concerning file of each WS filtered. More precisely, to extract words that will be accepted or rejected by the automaton defined previously.

Every WSDL file contains the tag <operation> that defines the list of WS operations. We extract the execution order and transform it to words. Hence, all the WS words are ready to be filtered.

If we have composed WS, every BPEL file contains the tag <sequence> that defines an ordered sequence of WS activities. We extract this WS operations sequencing and transform it to word. Hence, all the WS words are ready to be filtered.

- *Select the relevant WS:*

Basing on FP «Minimum transfer cost» and «Reduced transfer time», we launch the selection phase of the most relevant WS.

The services candidate namely WS1«Stay», WS2«Reservation» and WS3«Journey» satisfy the consumer semantically but only WS2 is selected as the most relevant WS. Calculations are made in order to reduce transfer time and transfer cost.

5 RELATED WORK

The WS behaviour in literature is described by sequences of messages, data types, data constraints and properties that specify time limits within\where messages are exchanged (Elabd, 2011).

Previously, Maamar et al. propose an approach for modelling and specifying behaviours of WS in (Maamar, 2009). This approach sheds the light on two types of behaviours: control (that demonstrate the business logic that supports the functioning of a WS) and operational (that regulates the execution progress of this control behaviour by stating actions to carry out and the constraints to put on this progress). The idea is to coordinate both behaviours at run-time by developing conversational messages and transmit details between these two behaviours. Unlike (Maamar, 2009), our approach treats behaviour as a sequence of ordered operations and abstracts away from considering behaviours conversation.

Another alternative is to first propose a service system by describing the overall behaviour of each consumer, and then to instantiate such consumers retrieving services exposing a behavioural contract which is adequate to the matching given behaviour (Bravetti, 2009).

The work in (Sriharee, 2003) presents an approach based on ontology to improve descriptions of WS that are defined in WSDL with ontology-based behavioural information, the query for services are based on behavioural constraints and have a service ontology linked with each WS. It can benefit from inferring semantics of the service from the service ontology. This work neglects the consumer needs, it doesn't depend on the execution order of WS operations.

The aim of WS discovery based on behavioural aspect prove that WS consumer is the most important part in discovery process, whose role is to specify the WS description as well as its behaviour.

The authors in (Ramollari, 2008) propose an approach where the service provider enhances the WSDL document by means of a formal model of the WS behaviour, expressed in the stream X-machine formalism (SXM). This model is used by the service broker during publication and by the service consumer during discovery.

All these works treat behavioural aspects as conversational messages exchanged between WS and abandon the internal structure of WS execution. That is why the approach that we propose in this paper, involves the importance of sequencing of WS operations to guarantee the quality of service.

6 CONCLUSIONS

We hope you find the information in this template useful in the preparation of your submission.

Service consumers have a choice between different WS candidates that provide similar functions. Accordingly, comparing services requires more sophisticated patterns of discovery.

In this paper, we have proposed a WS discovery approach based on behavioral aspects. We sought to satisfy consumer needs by introducing new criteria based on user requirements. This approach is based on execution manner of WS operations. To fulfill the aim of our approach, we arranged consumer requirements in semantic and behavioral aspect. Hence, we have organized our work in three phases; phase1 consists to transform consumer sentences to keywords if the consumer is a non expert domain. Phase2 is a semantic WS discovery basing on WordNet as lexical database and then create the automaton to put in order WS operations and to extract the language accepted. Phase3 aims to extract sequences from WSDL/BPEL files of WS searched to select the relevant WS.

In future work, we plan implementing our approach by developing a query language that uses the underlying automaton to the required WS. A case study will be set up to explain the objectives of the approach. Also experimentation should be made to highlight the advantage of using the approach. A work is underway to fulfill this objective.

As for Selection phase, we can add other criteria to select the most appropriate service if many services satisfy the desired behavior.

REFERENCES

- G. A. Miller, 1995. *WordNet: A Lexical Database for English*, *Communication of the ACM*, Vol. 38, No. 11.
- F. Christiane, 2005. *WordNet and wordnets*, In: Brown, Keith et al. (eds.), *Encyclopedia of Language and Linguistics*, Second Edition. Oxford: Elsevier, pages 665-670.
- J. Morato et al., 2004. *WordNet Applications. GWC 2004 Global Wordnet Conference (poster session)*. <http://www.fi.muni.cz/gwc2004/>. Brno.
- G. Labiak and G. Borowik, 2010. Statechart-based Controllers Synthesis in FPGA Structures with Embedded Array Blocks. *International Journal of Electronics and Telecommunications*. Volume 56, Issue 1, Pages 13–24, ISSN (Print) 0867-6747, DOI: 10.2478/v10177-010-0002-7.
- D. Harel, 1987. STATECHARTS: a visual formalism for complex systems*, *Science of Computer Programming* 8, 1987, pages 231-274.
- A. Bhattacharjee, B. S. Purkayastha, 2014. A Novel Approach to Construct Deterministic Finite State Automata. *International Journal Of Engineering And Computer Science* ISSN:2319-7242 Volume 3 Issue 1, Page No. 3700-3703.
- J. Kari, 2013. *Automata and formal languages*. Fall semester 2013 University of Turku.
- A. ALBRESHNE et al., 2009. Web Services Orchestration and Composition Case Study of Web services Composition, *WORKING PAPER*.
- N. Milanovic, and M. Miroslaw, 2004. Current Solutions for Web Service composition. *s.l.: IEEE Computer Society*.
- OASIS, 2007. Web Services Business Process Execution Language (WSBPEL).
- K. Kritikos and D. Plexousakis, 2007. OWL-Q for Semantic QoS-based Web Service Description and Discovery, *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, BEXCO, Bussan KOREA*. pages 123-137.
- Elabd, 2011. Conformité de services Web par rapport à des spécifications de haut niveau.
- Z. Maamar et al., 2009. A New Approach to Model Web Services Behaviors based on Synchronization. *International Conference on Advanced Information Networking and Applications Workshops*.
- M. Bravetti, 2009. Foundational Aspects of Service Discovery based on Behavioral Contracts, *3rd International Workshop on Verification and Evaluation of Computer and Communication Systems*.
- N. Sriharee and T. Senivongse, 2003. Discovering Web Services Using Behavioral Constraints and Ontology. *Distributed Applications and Interoperable Systems - DAIS*, pages. 248-259.
- E. Ramollari et al., 2008. Towards Reliable Web Service Discovery through Behavioral Verification and Validation. *Proceedings of the 3rd European Young Researchers Workshop on Service*.

PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services

Yiannis Verginadis¹, Antonis Michalas², Panagiotis Gouvas³, Gunther Schiefer⁴, Gerald Hübsch⁵ and Iraklis Paraskakis⁶

¹*Institute of Communications and Computer Systems, National Technical University of Athens, Athens, Greece*

²*Security Lab, Swedish Institute of Computer Science, Stockholm, Sweden*

³*Ubitech Ltd., Athens, Greece*

⁴*Karlsruhe Institute of Technology, Karlsruhe, Germany*

⁵*CAS Software AG, Karlsruhe, Germany*

⁶*South East European Research Centre, Thessaloniki, Greece*

jverg@mail.ntua.gr, antonis@sics.se, pgouvas@ubitech.eu, gunther.schiefer@kit.edu, gerald.huebsch@cas.de, iparaskakis@seerc.org

Keywords: Data Privacy, Security by Design, Context-aware Security, Symmetric Searchable Encryption, Cloud Computing.

Abstract: The valuable transformation of organizations that adopt cloud computing is indisputably accompanied by a number of security threats that should be considered. In this paper, we outline significant security challenges presented when migrating to a cloud environment and propose PaaSword – a novel holistic, data privacy and security by design, framework that aspires to alleviate them. The envisaged framework intends to maximize and fortify the trust of individual, professional and corporate users to cloud services. Specifically, PaaSword involves a context-aware security model, the necessary policies enforcement and governance mechanisms along with a physical distribution, encryption and query middleware, aimed at facilitating the implementation of secure and transparent cloud-based applications.

1 INTRODUCTION

Until recently, large-scale computing was available exclusively to large organizations with an abundance of in-house expertise. Cloud computing has changed that to the point where any user with even basic technical skills can obtain access to vast computing resources at low cost. In the technology adoption life-cycle, cloud computing has now moved from an early adopters stage to an early majority, where we typically see exponential number of deployments (Santos et al., 2009). Throughout the past few years, many users have started relying on cloud services without realizing it. Major web mail providers utilize cloud technology; tablets and smartphones often default to automatically uploading user photos to cloud storage and social networks; finally, several prominent CRM vendors offer their services using the cloud. In other words, the adoption of cloud computing has moved from focused interest to widely spread intensive experimentation and is now rapidly approaching a phase

of near ubiquitous use.

Enterprises increasingly recognize the compelling economic and operational benefits of cloud computing (Micro, 2010). Virtualizing and pooling IT resources in the cloud enables organisations to realize significant cost savings and accelerates deployment of new applications, simultaneously transforming business and government at an unprecedented pace (CSA, 2013). However, those valuable business benefits cannot be unlocked without addressing new data security challenges posed by cloud computing.

Despite the benefits of cloud computing, many companies have remained cautious due to security concerns. Applications and storage volumes often reside next to potentially hostile virtual environments, leaving sensitive information at risk to theft, unauthorized exposure or malicious manipulation. Governmental regulation regarding data privacy and location presents an additional concern of significant legal and financial consequences if data confidentiality is breached, or if cloud providers move regu-

lated data across national borders (Paladi and Michalas, 2014). The contribution of this position paper is two-fold. First, we present a list of core security requirements and challenges that must be considered when migrating to a cloud environment. These security requirements were derived based on our experience with migrating existing applications to a private Infrastructure-as-a-Service (IaaS) cloud (Michalas et al., 2014). We extend this guide by discussing important attack vector characteristics for cloud environments that will pave the way for providing tighter security when building cloud services. Second, in order to tackle the critical cloud security challenges we present PaaSword, an envisaged framework that will maximize and fortify the trust of individual, professional and corporate users to cloud services and applications. PaaSword achieves that by providing storage protection mechanisms, which improves confidentiality and integrity protection of users' data in the cloud while it does not affect the data access functionality.

The rest of this paper is organized as follows. In Section 2, we further elaborate on the main data security challenges in cloud-enabled services and applications. In Section 3, we introduce a holistic, data privacy and security by design, framework enhanced by sophisticated context-aware access models and robust policy enforcement and governance mechanisms, aimed at facilitating the implementation of secure and transparent cloud-based applications. In Section 4, we briefly discuss relevant work while in Section 5, we conclude the paper by presenting the next steps for the implementation and evaluation of the proposed framework.

2 DATA SECURITY CHALLENGES IN THE CLOUD

According to the Cloud Security Alliance (Alliance, 2013), several top security identified threats refer to information disclosure and repudiation, rendering data security as realised through data protection, privacy, confidentiality, and integrity as top priorities. More precisely, the top four threats identified are: data leakage, data loss, account hijacking and insecure APIs. The externalized aspect of outsourcing can make it harder to maintain data integrity and privacy (IBM, 2011) and organizations should include mechanisms to mitigate security risks introduced by virtualization. Especially when they deal with sensitive data, such as health records, the protection of stored information comes as a top priority. Therefore, data security can be seen as the foundation upon which the entire transition to a cloud architecture

should be based. Multiple risks must be addressed in order for an organization to guarantee the safety of users' records. One of the most important aspects is security of sensitive information. To this end, the deployment must ensure that all sensitive data is stored in encrypted form. Complementary to this, proper key management must ensure that encryption keys are not revealed to malicious users.

Based on this, it becomes evident that the most critical part of a modern cloud application is the data persistency layer and the database itself. As all sensitive information (including user credentials, credit card info, personal data, corporate data, etc.) are stored in these architectural parts, the database-takeover is the ultimate goal for every adversary.

The Open Web Application Security Project¹ foundation has categorized the database-related attacks (SQL injection) as the most critical ones. The importance of this attack vector is also reflected by respective incident reports. According to the Web Hacking Incidents Database², SQL injections represents 17% of all security breaches examined. These injections were responsible for 83% of the total records stolen, in successful hacking-related data breaches from 2005 to 2011. The criticality of the persistency layer is therefore evident. Most of the security fences that are configured in a corporate environment target the fortification of the so-called network perimeter (e.g. routers, hosts and virtual machines). Although existing intrusion detection systems (IDS) and intrusion prevention systems (IPS), try to cope with database-takeover security aspects (like Snort), the fact that, on the one side, automated exploitation tools (e.g. SQLMap) are widely spread, and, on the other side, IPS and IDS evasion techniques have become extremely sophisticated, denote that the risk of database compromise is greatest than ever. Moreover, by using mechanisms that rely on Web Application Firewalls (WAF) an organization can prevent various types of attacks but it is inadequate to protect against today's sophisticated SQL Injection and DoS attacks (Michalas et al., 2010). Additionally, internal adversaries in terms of cloud vendors or even unknown vulnerabilities of software platforms and security components widely adopted in cloud-based development may provide malicious access to personal and sensitive data. A recent example was the Heartbleed flaw³ that constituted a serious fault in the OpenSSL cryptography library, which remained unnoticed for

¹<https://www.owasp.org/>

²<http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database>

³<http://www.infosecurity-magazine.com/news/heartbleed-101/>

more than two years and affected over 60% of Web servers worldwide. Additionally, regarding the post-exploitation phase, things are even worse in the case where a symmetric encryption algorithm has been employed to protect the application data. The already available cracking toolkits that utilize GPU processing power (e.g. oclHashcat) are able to crack ciphers using brute-force techniques with an attack rate of 162 billion attempts per second.

While most of the attack vectors are exposed in any Software-as-a-Service application by the system administrators misconfigurations, the database takeover and the post-exploitation of acquired data is under the sole responsibility of the application developer. The application developer is the one responsible both for sanitizing all HTTP-input parameters that could be used as attack vectors, and for reassuring that compromised data will be useless under the existing brute-forcing and reversing techniques. Nevertheless, even if the application developer follows strict guidelines, the mere utilization of an IaaS provider in order to host a Virtual Machine, or for a Platform-as-a-Service (PaaS) provider in order to develop a cloud application, may by itself spawn a multitude of inherent vulnerabilities. These vulnerabilities cannot be tackled effectively as they typically exceed the responsibilities of an application developer.

3 ENVISIONED FRAMEWORK

In this section, we present PaaSword, a framework that will allow cloud services to maintain a fully distributed and encrypted data persistence layer in order to foster data protection, integrity and confidentiality in the presence of malicious adversaries. To this end, we describe the need for a context-aware security model which will serve as the basis of a fine-grained access control scheme, one which allows the per-user management of access rights. In addition to that, we describe a physical distribution, encryption and query middleware that will be based on a searchable encryption (SE) scheme which will allow legitimate users to directly search on encrypted data, thus ensuring the confidentiality and integrity of stored data.

3.1 Context-aware Access Model

We envision a XACML-based⁴ context-aware access model, which is needed by the developers in order to annotate the Data Access Objects of their applications. This context model should conceptualize the

⁴OASIS eXtensible Access Control Markup Language (XACML). <https://www.oasis-open.org/>

aspects, which must be considered during the selection of a data-access policy. These aspects may be any kind of information which is machine-parsable (Dey, 2001); indicatively they may include the user's IP address and location, the type of device that she is using in order to interact with the application as well as her position in the company. These aspects can be interpreted in different ways during the security policy enforcement. In particular, the context aware access model determines which data is accessible under which circumstances by an already-authenticated user.

Access control models are responsible for deciding if a user has the right to execute a certain operation on a specific object. Objects can be a server, an application, an entire database or even a single field in a table row. The user is considered as the active element and is called subject. A permission associates an object with an operation (e.g. read, write etc.). Access control models provide a list of permissions that each subject has on certain objects.

Commonly used access control models are the Mandatory Access Control (MAC), the Discretionary Access Control (DAC) and the Role-Based Access Control (RBAC) (Ferrari, 2010). In our approach, the process of granting/denying access will be based on dynamically changing parameters, thus our proposed model relies on a DAC model with groups. The context parameters are unique for every single user, so for granting access it is necessary to consider all information associated with a single user. Furthermore, an RBAC model would be inappropriate since for every change of a context parameter the role of each subject has to be changed.

To implement the dynamic change of context parameters in a static access control model, we will use the, so-called, context switches. Depending on the current context, a permission can be granted or denied (switched). This could switch dynamically with every change of the context. Context switches are responsible for managing operational permissions and object permissions. An operational permission gives the right to a subject to perform a specific operation while an object permission gives the right to perform an operation on a specific object.

3.2 Policies Access and Enforcement

Another important aspect of our proposed framework is a middleware that will encapsulate capabilities for maintaining the access policies model, for annotating and managing data access object annotations, for controlling their validity, for dynamically interpreting them into policy enforcement rules and for enforcing

ing them. This envisaged middleware will provide: (a) a transparent key usage for efficient authentication purposes, related to authenticating the origin of the incoming access requests; (b) annotation capabilities in the form of a tool (can also involve an IDE plugin) for allowing developers to declaratively create the minimum amount of rule-set that is needed for security enforcement purposes; (c) the dynamic interpretation of the data access object annotations into policy enforcement rules; (d) the governance and quality control of the annotations and their respective policy rules; and (e) the formulation and implementation of the overall policy enforcement business logic.

In terms of this middleware, we also consider the reuse and proper extension of technologies for developing an appropriate key management mechanism. This mechanism is necessary for the authentication of different parties that will be involved in the encryption and decryption of data. We aim at constituting the key-usage, transparent to the application usage. This involves the key propagation upon authentication of the user, directly to the security enforcement middleware. For efficiency, we will employ a hybrid encryption capitalizing upon the utilization of two different encryption functions. The inner layer will be encrypted with an algorithm that uses a symmetric encryption key K , while the outer layer will use an asymmetric encryption in order to encrypt the symmetric key K . Symmetric encryption allows more efficient schemes but privacy concerns are raised due to the fact that the involved parties must exchange the secret key. However, combining both techniques help to optimize the efficiency of the underlying protocols without sacrificing security. To this end, PaaSWord will rely on both symmetric and asymmetric encryption in order to securely distribute K between legitimate users.

Additionally, we will also employ methods and mechanisms for governance and validity control of the data object annotations. More specifically, we will focus on the application of an ontology-driven governance approach for: *i*) the basic management of data object annotations (i.e. storage, retrieval, deletion, etc.), *ii*) validity checking of the data object annotations (e.g. rejecting any contradicting annotations made by the developer) and *iii*) dependency tracking among data objects annotations.

Another critical aspect of this middleware is the annotations interpretation mechanism. Such a mechanism will be used for dynamically generating access control policies, during application runtime, based on the interpretation of data object annotations. Such a mechanism will implement the essential decoupling between the access decisions and the points of use

(i.e. Policy Enforcement Points (PEP) of the XACML specification). This interpretation is based on an XACML compliant context model and it can augment the offered functionality of any PaaS provider, with a security-as-a-service layer. To do so, we will use the OASIS XACML as it supports and encourages the separation of the access decision from the point of use.

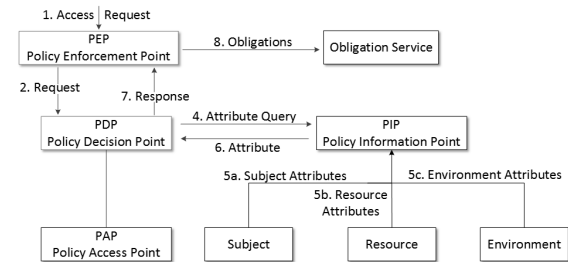


Figure 1: High level view of XACML Components.

3.3 Threat Model, Secure Storage & Query Middleware

In this sub-section, we provide a high level description of the protocol that will be used to effectively protect the stored data from malicious adversaries. To this end, we first describe the threat model under which a cloud application will be considered secure.

Threat Model. Similar to existing works in the area (Paladi et al., 2014; Santos et al., 2009), we assume a *semi-honest* cloud provider. In the semi-honest adversarial model, a malicious cloud provider correctly follows the protocol specification. However, she can intercept all messages and may attempt to use them in order to learn information that otherwise should remain private. Semi-honest adversaries are also called *honest-but-curious*.

Furthermore, for the rest of the participants in the protocol we share the threat model with (Santos et al., 2009), which is based on the Dolev-Yao adversarial model (Dolev and Yao, 1983) and further assumes that privileged access rights can be used by a remote adversary \mathcal{ADV} to leak confidential information. The adversary, e.g. a corrupted system administrator, can obtain remote access to any host maintained by the provider. However, the adversary cannot access the volatile memory of any guest virtual machine (VM) residing on the compute hosts of the provider. This property is based on the closed-box execution environment for guest VMs, as outlined in Terra (Garfinkel et al., 2003) and further developed in (Zhang et al., 2011).

Secure Storage. A basic tenet of PaaSWord is that sensitive data stored on untrusted servers must be always encrypted. This effectively reduces the privacy and security risks since it relies on the semantic security of the underlying cryptosystem, rendering the system relatively immune to internal and external attacks. Having this in mind, we propose a forward-looking design for a cryptographic cloud storage that will be based on a symmetric searchable encryption (SSE) scheme similar to the one proposed in (Kamara and Lauter, 2010). We plan to extend the previous work Cumulus4j (Huber et al., 2013) and MIMOSecco (Gabel and Hübsch, 2014) in which an SSE scheme was presented and it was based on the IND-ICP security notion (Bösch et al., 2014) that hides relations between different data values of a data row and creates the base for secure database outsourcing.

An SSE scheme allows a user to search in encrypted data without learning any information about the plaintext data. Let $\mathcal{DB} = \{m_1, \dots, m_n\}$ be a set of n messages (w.l.o.g. \mathcal{DB} can be considered as a database). For each $m_i \in \mathcal{DB}$ we extract a set of keywords which can later be used for executing queries. This set of keywords is denoted as $\mathcal{W} = \{w_1, \dots, w_n\}$. For each $w_i \in \mathcal{W}$ we calculate $H(w_i)$, where $H(\cdot)$ is a cryptographically secure hash function under a secret key K' . Then, we encrypt the elements of \mathcal{DB} with a secret key $K'' \neq K'$. By doing this, we create a searchable encrypted index I where each index entry, points to an encrypted list of rows that have a certain keyword. The client can use a trapdoor function to search the index and determine whether a specific keyword is contained in the index.

While the above-mentioned scheme is implemented in previous works (Huber et al., 2013; Gabel and Hübsch, 2014) it has a limitation that we tend to cover in our proposed framework. More precisely, the current scheme follows a single write/single read (S/S) architecture, which makes it unrealistic for our cloud scenario. To overcome this limitation, we plan to build an SSE that will support multi write/multi read (M/M) meaning that a group of users based on access rights will be able to both read and write on the encrypted data. To this end, PaaSWord will involve a key distribution algorithm that will extend S/S architecture to M/M. Additionally, a user revocation function will be implemented in order to exclude a user, which either acts maliciously or has no longer access rights. This is a crucial and challenging procedure, if we consider that many of the existing SSE schemes (Bösch et al., 2014) do not support user revocation and thus are susceptible to many attacks.

Query Middleware. In order to successfully support the SSE scheme described above, we aim to develop a persistency layer, called Virtual Database VB (Figure 2), and will be the intermediary that secures client data before it gets uploaded to the cloud. Additionally, this layer will be responsible for processing user queries. In our framework, the VB plays the role of a trusted third party. Consider, for example, the scenario where a user wants to search for a certain data in PaaSWord secured databases. To do so, she will generate a query (q) containing a set of keywords that she is interested in and will send the request to the VB. Upon reception, the VB extracts the keywords from q calculates their hash values and queries the databases where the keywords w_i are stored. If the queries are successful and the keywords exist in one of the tables, the VB will obtain the row from the main table that contains the encrypted original data. Upon reception, the VB will reply to the users request by sending the acquired data.

3.4 Conceptual Architecture

The PaaSWord compliant cloud applications that will be developed will inherit a fully physical distributed and totally encrypted data persistence layer, which will be able to determine on an ad-hoc basis whether an incoming data querying and processing request should be granted access to the target data during application runtime. The transformation process of a traditional application utilizing the PaaSWord framework and the way the transformed application secures and protects the users' sensitive data is presented in Figure 2, which at the same time reveals high level architectural details of the framework.

In this framework, we consider applications that adopt and respect the Model-View-Controller (MVC) development pattern (Krasner and Pope, 1988). As seen in Figure 2 (step 1) the application developer imports an existing or creates a new MVC-based application in her favorite integrated development environment (IDE) for which an IDE-specific plug-in will be provided. During the second step of this process the application developer creates annotations at the DAO of the Controller referring to sensitive data that should be protected, according to the XACML-based model and defines the physical distribution, encryption and access rights scheme for each data object. In the third step, the DAO annotations will be checked for their validity and compiled with the overall application code. This will allow the transformation of the application's controller that has been enhanced with XACML-based DAO annotations, leading to the implementation of a PaaSWord secure ap-

plication. In the fourth step, the persistence layer of the application will be physically distributed and encrypted at the schema and instance level according to the incorporated DAO annotations, imposing the schema and driving query handling capabilities of the VB that augments the actual data persistence layer of the application. At application runtime (step 5), each query and processing request of the end-user is forwarded by the enhanced controller to the query handling mechanism that is responsible for the database proxy queries synthesis and aposynthesis. In step 6 and before the submission of the enhanced query to the VB, the query handling mechanism consults the policy enforcement mechanism to determine whether the incoming request should be granted or not. Upon policies enforcement and access permission, the query handling mechanism submits (step 7) the enhanced query to the augmented persistence layer (virtual database). The database proxy that is aware of the physical distribution scheme of the actual application database realizes the distributed query to the physically distributed and encrypted parts of the actual application database (step 8). Next, the federation of the respective encrypted data from the distributed parts of the database takes place (step 9). The federated data synthesis and ad-hoc decryption utilizing the key of the end-user that is transparently to the application, propagated to the query handling mechanism (step 10). Last, the query handling mechanism delivers the decrypted data to the application controller that forwards them to the end-user through the “view” component of the application.

According to the conceptual view (Figure 2), each end-user is equipped with a Hardware Security Module, such as USB stick or a smart-phone with digital rights management module, which contains a digital certificate (e.g. X.509). Part of the certificate includes keys that can be exported by the PaaS/IaaS provider. These keys upon export and verification will be transparently handed over to the query middleware which will be responsible for interacting with the VB, encrypting and decrypting the targeted data.

4 RELATED WORK

In an attempt to reinforce the security of remote service accesses, researchers introduced the concept of location-aware access control (LAAC), which allows a system to grant, or deny, access to users based on their physical location. LAAC models typically extend the three basic access control models DAC, MAC and RBAC (Decker, 2011). Even though LAAC protocols have been studied extensively (Cleff et al.,

2010), there is a clear lack of schemes that determine user access not only on the basis of the users physical location and credentials, but also on the additional pertinent contextual information.

The work reported in (Covington et al., 2001) was the first to introduce the notion of context-aware access control (CAAC), motivated by applications for intelligent homes. More precisely, the authors introduced a set of services which are enabled based on the location of objects or subjects. The main drawback of the proposed model is the fact that it does not support dynamically generated context, whilst it fails to address important requirements such as multi-granularity of position. Other existing CAAC models are predominantly based on RBAC (Kayes et al., 2013) and typically target a specific domain (Costabello et al., 2012). These models, however, have not been designed to provide fine-grained data access control, e.g. by providing the ability to specify different access rules for different rows of a database.

Regarding the policy management, as shown by a recent survey of methods in contemporary open source registry and repository systems (Kourtesis and Paraskakis, 2012), a major weakness is the lack of proper separation of concerns. The policy definition and policy enforcement are entangled in the implementation of a single software component – the policy checker. The rules that a policy comprises are typically encoded in an imperative manner, as part of the same code that checks for potential policy violations. This has a number of negative repercussions among which is the lack of portability and the lack of explicit representation of policy relationships.

The data distribution and encryption algorithms are also important aspects towards trusted cloud services and applications. In (Gentry, 2009), C. Gentry presented the first fully homomorphic encryption scheme that enables semantically secure outsourcing to the cloud. The cloud provider operates blindly on the encrypted data and yields the correct, encrypted result. Nevertheless, its practicality is in question as the latest implementations are still orders of magnitude slower than just downloading all encrypted data, decrypting, processing and encrypting it locally and finally uploading it again. In another interesting approach (Popa et al., 2011), the concept of onions is used. Onions are managed monolithically by a proxy, acting as an adapter between the user and the storage back-end. Each attribute in a relational table is initially asymmetrically encrypted. If certain queries for an attribute are issued, layers of the onion are peeled off, resulting in another, less secure onion. CryptDB uses a novel scheme for order preserving encryption that leaks no information about the data besides or-

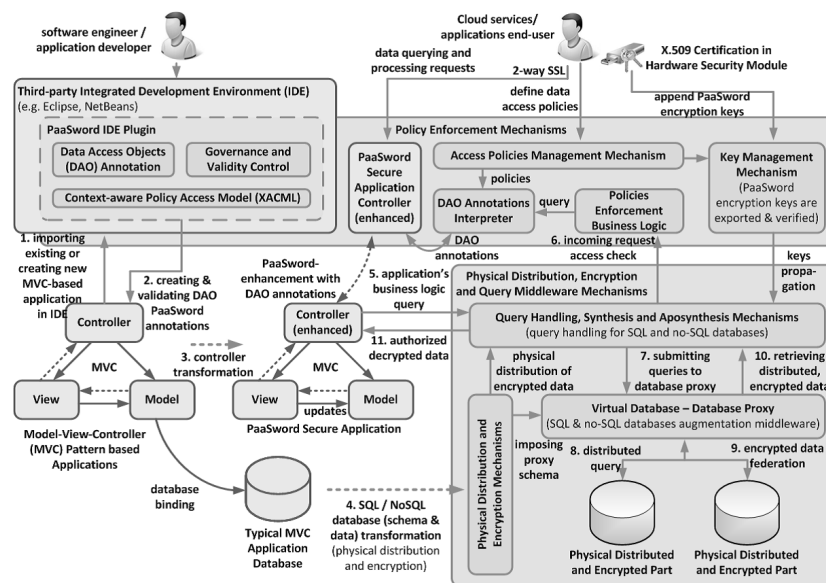


Figure 2: PaaSword Framework Conceptual Architecture.

der and thus allows sorting encrypted data securely. The main drawback of CryptDB is the lack of security guarantees to the client. More precisely, the only guarantee is that an untrusted server will learn only the information that is necessary to process the query. This may cause every attribute to be reduced to the plain text in the worst case. Also, peeling off layers cannot be reversed, so a single query is sufficient to lower the security forever.

5 CONCLUSIONS

In this position paper, we proposed the PaaSword framework that can be exposed as a service at the level of PaaS. This framework can tackle the identified cloud security requirements and challenges that should be considered in order to enhance data protection, integrity and confidentiality in the presence of malicious adversaries. The envisaged PaaSword goes beyond the state-of-the-art and allows cloud services to maintain a fully distributed and encrypted data persistence layer. Our framework involves a context-aware security model, the necessary policies enforcement mechanism along with a physical distribution, encryption and query middleware.

Future work involves the design and implementation of the proposed framework into a fully functional solution which will be validated through the following five pilots in various industrial contexts: *i*) Encrypted persistency as a service in a PaaS provider, *ii*) Intergovernmental secure document and personal data exchange, *iii*) Secure sensors data fusion and analyt-

ics, *iv*) Protection of personal data in a multi-tenant CRM, *v*) Protection of sensible enterprise information in multi-tenant ERP. These pilots will allow us to test PaaSword and validate its added value in a variety of heterogeneous cases.

Finally, an area that will benefit from PaaSword framework is the so called participatory sensing (Michalas and Komninos, 2014). The evolution of this field is driven by the introduction of sensors into mobile devices. The openness of such systems and the richness of user data they entail raise significant concerns for their storage and processing. Protocol designers by having PaaSword framework in hands will be able to incorporate secure cloud computing techniques in order to facilitate the storage and processing of the vast amount of collected data.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644814, the PaaSword project (www.paasword.eu) within the ICT Programme ICT-07-2014: Advanced Cloud Infrastructures and Services.

REFERENCES

Alliance, C. S. (2013). The notorious nine – cloud computing top threats in 2013.

- Bösch, C., Hartel, P., Jonker, W., and Peter, A. (2014). A survey of provably secure searchable encryption. *ACM Comput. Surv.*, 47(2):18:1–18:51.
- Cleeff, A. v., Pieters, W., and Wieringa, R. (2010). Benefits of location-based access control: A literature study. In *Proceedings of the 2010 IEEE/ACM Int’L Conference on Green Computing and Communications & Int’L Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM ’10, pages 739–746, Washington, DC, USA. IEEE Computer Society.
- Costabello, L., Villata, S., and Gandon, F. (2012). Context-aware access control for rdf graph stores. In Raedt, L. D., Bessire, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P. J. F., editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 282–287. IOS Press.
- Covington, M. J., Long, W., Srinivasan, S., Dev, A. K., Ahamad, M., and Abowd, G. D. (2001). Securing context-aware applications using environment roles. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, SACMAT ’01, pages 10–20, New York, NY, USA. ACM.
- Decker, M. (2011). Modelling of location-aware access control rules. In *Handbook of Research on Mobility and Computing: Evolving Technologies and Ubiquitous Impacts*, pages 912–929. IGI Global.
- Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7.
- Dolev, D. and Yao, A. C. (1983). On the security of public key protocols. *Information Theory, IEEE Transactions*, 29(2):198–208.
- Ferrari, E. (2010). *Access Control in Data Management Systems*. Morgan and Claypool Publishers.
- Gabel, M. and Hübsch, G. (2014). Secure database outsourcing to the cloud using the mimosecco middleware. In Krcmar, H., Reussner, R., and Rumpe, B., editors, *Trusted Cloud Computing*, pages 187–202. Springer International Publishing.
- Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., and Boneh, D. (2003). Terra: A virtual machine-based platform for trusted computing. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 193–206.
- Gentry, C. (2009). *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA. AAI3382729.
- Huber, M., Gabel, M., Schulze, M., and Bieber, A. (2013). Cumulus4j: A provably secure database abstraction layer. In Cuzzocrea, A., Kittl, C., Simos, D. E., Weippl, E., Xu, L., Cuzzocrea, A., Kittl, C., Simos, D. E., Weippl, E., and Xu, L., editors, *CD-ARES Workshops*, volume 8128 of *Lecture Notes in Computer Science*, pages 180–193. Springer.
- IBM (2011). Security and high availability in cloud computing environments. Technical report, IBM SmartCloud Enterprise, East Lansing, Michigan.
- Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage. In Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J., Sako, K., and Seb, F., editors, *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer Berlin Heidelberg.
- Kayes, A. S. M., Han, J., and Colman, A. (2013). An ontology-based approach to context-aware access control for software services. In Lin, X., Manolopoulos, Y., Srivastava, D., and Huang, G., editors, *WISE (1)*, volume 8180 of *Lecture Notes in Computer Science*, pages 410–420. Springer.
- Kourtesis, D. and Paraskakis, I. (2012). A registry and repository system supporting cloud application platform governance. In *Proceedings of the 2011 International Conference on Service-Oriented Computing*, ICSOC’11, pages 255–256, Berlin, Heidelberg. Springer-Verlag.
- Krasner, G. E. and Pope, S. T. (1988). A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49.
- Michalas, A. and Komninos, N. (2014). The lord of the sense: A privacy preserving reputation system for participatory sensing applications. In *Computers and Communication (ISCC), 2014 IEEE Symposium*, pages 1–6. IEEE.
- Michalas, A., Komninos, N., Prasad, N. R., and Oleshchuk, V. A. (2010). New client puzzle approach for dos resistance in ad hoc networks. In *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference*, pages 568–573. IEEE.
- Michalas, A., Paladi, N., and Gehrmann, C. (2014). Security aspects of e-health systems migration to the cloud. In *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pages 212–218. IEEE.
- Micro, T. (2010). The need for cloud computing security. In *A Trend Micro White Paper*.
- Paladi, N. and Michalas, A. (2014). “One of our hosts in another country”: Challenges of data geolocation in cloud storage. In *Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE), 2014 4th International Conference on*, pages 1–6.
- Paladi, N., Michalas, A., and Gehrmann, C. (2014). Domain based storage protection with secure access control for the cloud. In *Proceedings of the 2014 International Workshop on Security in Cloud Computing*, ASIACCS ’14, New York, NY, USA. ACM.
- Popa, R. A., Redfield, C. M. S., Zeldovich, N., and Balakrishnan, H. (2011). Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP ’11, pages 85–100, New York, NY, USA. ACM.
- Santos, N., Gummadi, K. P., and Rodrigues, R. (2009). Towards trusted cloud computing. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud’09, Berkeley, CA, USA. USENIX.
- Zhang, F., Chen, J., Chen, H., and Zang, B. (2011). Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 203–216. ACM.

Classifying Security Threats in Cloud Networking

Bruno M. Barros¹, Leonardo H. Iwaya^{1,2}, Marcos A. Simplicio Jr.¹, Tereza C. M. B. Carvalho¹,
András Méhes³ and Mats Näslund³

¹*Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil*

²*Karlstad University, Karlstad, Sweden*

³*Ericsson Research, Stockholm, Sweden*

*bbarros@larc.usp.br, leonardo.iwaya@kau.se, {mjuniior, carvalho}@larc.usp.br,
{andras.mehes, mats.naslund}@ericsson.com*

Keywords: Cloud Networking, Cloud Security, Security Threats, Security Taxonomy.

Abstract: A central component of managing risks in cloud computing is to understand the nature of security threats. The relevance of security concerns are evidenced by the efforts from both the academic community and technological organizations such as NIST, ENISA and CSA, to investigate security threats and vulnerabilities related to cloud systems. Provisioning secure virtual networks (SVNs) in a multi-tenant environment is a fundamental aspect to ensure trust in public cloud systems and to encourage their adoption. However, comparing existing SVN-oriented solutions is a difficult task due to the lack of studies summarizing the main concerns of network virtualization and providing a comprehensive list of threats those solutions should cover. To address this issue, this paper presents a threat classification for cloud networking, describing threat categories and attack scenarios that should be taken into account when designing, comparing, or categorizing solutions. The classification is based on the CSA threat report, building upon studies and surveys from the specialized literature to extend the CSA list of threats and to allow a more detailed analysis of cloud network virtualization issues.

1 INTRODUCTION

The current concept of cloud computing evolved from technologies such as distributed computing and resource virtualization, enabling the utilization of shared computing infrastructures for delivering software, platforms and infrastructures to different customers over the Internet. Nevertheless, cloud computing has other particular requirements such as (Mell and Grance, 2011): on-demand provision of the computing resources; broad network access to configure and request computing capabilities; resources are pooled to be used by multiple customers in a multi-tenant model; the resources should be elastically provisioned and released; and delivered services should be transparently measured for managing and billing purposes. This new model of delivering computing power takes advantage of economies of scale, allowing cloud providers to deliver services for a reasonable cost to several institutions and companies. It also brings advantages to customers, who can pay only for what they consume instead of obliging them to purchase, install and maintain their own equipment.

Unfortunately, however, the advantages brought

by the cloud are also accompanied by threats and security vulnerabilities that discourage its full adoption by many companies. An example is the need of isolating resources, data and communication within the cloud. Public cloud systems utilize a multi-tenant architecture, in which customers should only "see" the cloud resources assigned to them, as if they were the sole user of the infrastructure.

Virtualization technologies play a crucial role in enforcing this isolation, given that they are the main building block in provisioning the customers' infrastructure, including virtual machines (VMs) and virtual networks (VNs). Additionally, a virtualization solution(s) should ensure not only that the VMs operate with isolated resources, but also allow network traffic monitoring and the creation of *secure network domains*. For this reason, enabling SVN in the cloud computing is currently a subject of intense research (Sun and Hu, 2012). Many of the existing proposals rely on open network virtualization solutions such as Open vSwitch for defining virtualized network architectures with security features (Hao et al., 2010; Cohen et al., 2013), inserting security modules inside VMs and virtual switches (Basak et al., 2010;

Barjatiya and Saripalli, 2012), or creating hypervisor-based network controllers (Mattos and Duarte, 2013).

Nonetheless, it is often hard to clearly identify all the threats and vulnerabilities that are addressed by the different network virtualization solutions. Indeed, there is a myriad of issues that can be targeted, such as ensuring traffic isolation, preventing sniffing and address spoofing, as well as detecting and mitigating Distributed-/Denial-of-Service DoS/DDoS and man-in-the-middle attacks (Hao et al., 2010; Basak et al., 2010; Barjatiya and Saripalli, 2012; Cohen et al., 2013; Mattos and Duarte, 2013). However, it is not always the case that solutions proposed in the literature explicitly analyze their (in)ability to cope with each of the existing threats, even if they are truly able to prevent them. This makes comparing and evaluating these solutions a difficult task.

Aiming to address this lack of uniformity in the treatment of network virtualization security proposals, this paper presents a threat classification for SVNs, describing threat classes and attack scenarios that should be taken into account when designing, comparing, categorizing or evaluating solutions. This classification is based on technical reports from cloud standardization organizations such as European Network and Information Security Agency (ENISA, 2013), the Cloud Security Alliance (CSA, 2013), and National Institute of Standards and Technology (NIST, 2011), as well as on scientific papers that reviewed problems in this area (Chowdhury and Boutaba, 2010; Pearce et al., 2013). Given the broad scope of the security challenges described in these reports, they do not (intend to) provide a framework to evaluate security issues directly related to network virtualization in cloud computing. Therefore, the classification proposed herein aims to fill part of this gap by focusing specifically on the technical issues related to virtual networking in the cloud environment.

The rest of this paper is organized as follows. Section 2 reviews the security threat classification for cloud computing proposed by CSA (CSA, 2013). Section 3 presents our proposed threat classification, which builds upon the CSA work. Section 4 then discusses, by means of examples, different cloud virtual networking attacks from each threat category, showing the coverage of the proposed classification. Section 5 presents the conclusion and future work.

2 CLOUD SECURITY THREATS

With the widespread adoption and popularization of cloud-based systems, considerable effort has been made to identify and classify security threats in this

environment. Some relevant examples include the security guidelines for cloud computing provided by the ENISA (Catteddu, 2010; ENISA, 2013), the CSA (CSA, 2013), and NIST (NIST, 2011).

Among these documents, the CSA “Notorious Nine” report (CSA, 2013), which identifies important threats that may occur accidentally or intentionally in cloud systems, is of especial interest: it provides a clear view of the most relevant security threats when deploying and consuming cloud services, ranked according to the industry perspective. Therefore, the report highlights the main security aspects that need to be taken into account by cloud providers for ensuring trust in their services. It is important to notice, however, that (CSA, 2013) is intended as a general guideline of relevant security aspects, not focused on networking issues. Nonetheless, given its importance, it can be seen as an interesting starting point for identifying relevant cloud networking threats, which is exactly the approach adopted in this document. Next, we present a classification method for identify security threats in cloud networking, built upon the CSA classification for cloud security threats.

3 THREAT CLASSIFICATION

CSA “Notorious Nine” (CSA, 2013) and similar-purpose reports (CSA, 2011; ENISA, 2013; NIST, 2011; Gonzalez et al., 2012) are important sources of information about cloud security threats. However, their main goal is to give a high level description of potential problems, not on providing a fine-grained analysis of how each of the many threats identified apply to specific scenarios (e.g., virtual networking). On the other hand, there are works in the literature that investigate SVNs in more depth, such as (Chowdhury and Boutaba, 2010; Schoo et al., 2011; Nataraajan and Wolf, 2012). Unfortunately, since their goal is to survey solutions and to identify challenges in the area, they fail to provide a reusable classification of the virtual networking threats described. Given the relevance of threat modeling to identify security requirements (Myagmar et al., 2005), those works are not ideal for the task of comparing and evaluating different security proposals in virtual networking or guiding the design of comprehensive solutions for the most relevant threats. Aiming to bridge this gap, next, we build upon the CSA “Notorious Nine” threat report for deriving finer-grained threat classification focused specifically on SVNs.

3.1 Extending Current Classifications

For the purpose of building the proposed classification, the CSA “Notorious Nine” threats were decomposed into more specific menaces. The resulting finer-grained list, containing specific attacks and countermeasures, was then analyzed aiming to identify threats that could be associated to virtual networking security issues. The aggregation of the identified threats according to their characteristics then lead to the general categories presented as follows.

Before we present the proposed classification, however, it is important to emphasize that attaining the right level of abstraction is a considerable challenge when trying to create a comprehensive view of virtual networking vulnerabilities in the cloud. Aiming to be both concise and comprehensive, our approach in the proposed classification involves two main requirements: (1) threats should have a detailed enough description to effectively help guide the development of innovative solutions; and (2) the number of threat groups should be small enough to allow the classification to be applied to the analysis and comparison of common solutions. These requirements have an obvious trade-off: a large number of threat classes may lead to an overly detailed classification, but a reduced number of threat groups may lead to a high level description that may be vague and less useful to comparative studies. As a result, our classification proposes a reduced number of categories without ignoring important aspects of virtual networking.

In addition to those basic requirements, we considered the different attack scenarios within the cloud environment, i.e., who the attacker is and who is being attacked. In each scenario, we then try to identify the different threat classes to reflect the concerns already evidenced in the literature.

3.2 Threat Scenarios

The first is the Cloud Provider Network, which includes all the cloud provider private network resources connecting all the data center infrastructure that allows the cloud service provision. The second is the Public Network, which comprises the public Internet that allows users to access the cloud services. Both networks are illustrated in Figure 1. We have limit our scope to threats in the Cloud Provider Network, to identify security issues related only to the cloud computing paradigm and to its technological mechanisms. We can identify three attack scenarios involving different entities of the cloud:

1. **Tenant-to-Tenant.** Threats related to attacks promoted by a legitimate tenant targeting another le-

gitimate tenant. Such attacks are usually performed by exploring vulnerabilities of the cloud provider network infrastructure

2. **Tenant-to-Provider.** Threats related to cloud vulnerabilities that allow a legitimate tenant to disrupt the operation of the cloud infrastructure, preventing the cloud provider from delivering the service in accordance with the service level agreements established with other legitimate tenants.
3. **Provider-to-Tenant.** Threats related to vulnerabilities in the cloud provider infrastructure, which allows malicious insider attacks from employees and other agents with direct access to the cloud infrastructure.

Figure 2 illustrates these three attack scenarios in the network context previously mentioned. Despite the interest of provider-to-tenant threats, for the purpose of this research we consider a reliable cloud provider, focusing our efforts on the tenant-to-tenant and tenant-to-provider threats.

3.3 Cloud VN Threat Categories

Following the method and the requirements discussed in Section 3.1, we identified five general classes of virtual networking security threats, described as follows. For each class, we also present the CSA-related threats and the virtual networking attacks obtained from the decomposing process described in Section 3.1. We note, however, that the inclusion of new threat classes to this classification should be considered a matter of continuous work, following new discoveries in the field of cloud security.

1. **Physical Isolation.** Covers all the vulnerabilities related to the physical resources of the underlying network infrastructure being shared by multiple tenants. Attacks in this class are normally related to capturing and to analyzing data collected from shared resources, but can also involve the exhaustion of resources from shared hardware.
2. **Logical Isolation.** Covers all the vulnerabilities directly related to situations in which logical resources (e.g., vCPUs, vLANs and vSwitches) are not adequately isolated, allowing tenants to access each other’s networking capabilities. Attacks of this class can exploit vulnerabilities in the cloud

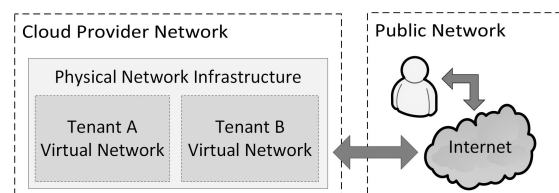


Figure 1: Complementary networks in cloud environments.

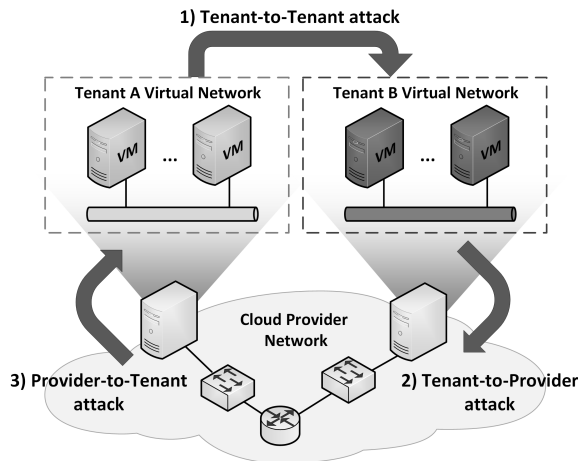


Figure 2: Security attack scenarios in cloud VNs.

virtualized network functions and network management modules.

3. **Authentication.** Covers all the vulnerabilities related to inadequate authentication, which allow attackers to mask their real identities. This can be accomplished by exploiting authentication protocols, acquiring credentials and/or key materials by capturing data traffic, or via password recovery attacks (e.g., brute force or dictionary attacks).
4. **Authorization.** Covers all the vulnerabilities related to authorization problems, allowing granting or scaling rights, permissions or credentials to or from an unauthorized user. The attacker can exploit a vulnerability in the cloud platform authorization modules, or even in the victim computer, to create or change its credentials in order to obtain privileged rights. Similarly to attacks that exploit authentication threats, authorization threats can lead to the leakage of confidential data and access to management modules.
5. **Insecure APIs.** Covers all the threats related to failures, malfunctions and vulnerabilities in APIs that compose the cloud system. Attacks of this class try to exploit insecure interfaces for accessing or tampering with services running in other tenants or cloud administrative tools. This may lead to data loss/leakage, as well as to cause unavailability of services.

3.4 Correlation with CSA Threats

Table 1 summarizes the correlations between CSA threats and the network threat categories herein proposed. Analyzing this table, one can notice that CSA threats related to *shared technology vulnerabilities*, *data breaches*, and *DoS* are the most fre-

quently related to network vulnerabilities. Therefore, they should be the main focus of SVN-oriented solutions, considering both tenant-to-tenant and tenant-to-provider attack scenarios, as discussed in more details in the next section.

Table 1: Correlation of proposed categories and threats with CSA notorious top nine threats.

Category	Threat Examples	CSA Top 9
Physical Isolation	Side-channel attack	DB,
	Guest-hopping attack	DoS,
	Resource exhaustion	STV
	Physical man-in-the-middle	
Logical Isolation	Man-in-the-middle	DB,
	Sniffing	DoS,
	Port scanning	STV
	Replay attacks	
Authentication	Spoofing	DB,
	Compromised-key attack	ASH,
	Password-based attacks	STV
Authorization	Software exploitation	DB, MI,
	Phishing	ACS, STV
Insecure APIs	API exploitation	
	Code and Package injection	IIA

Abbreviations: data breaches (DB), Denial-of-Service (DoS), shared technology vulnerabilities (STV), account or service traffic hijacking (ASH), malicious insiders (MI), abuse of cloud services (ACS), insecure interfaces and APIs (IIA).

4 CLASSIFICATION ANALYSIS

This section correlates the threat scenarios discussed in Section 3.2 with the classification proposed in Section 3.3, presenting threat examples for each proposed category in both Tenant-to-Tenant and Tenant-to-Provider scenarios.

4.1 Physical Isolation

Such attacks can be perpetrated if the attacker's VM is sharing the same host machine or physical network node as the victim's VM. In this scenario, the attacker can engage in: a side-channel attack, subjecting confidential traffic to cryptanalysis; use DoS/DDoS attacks to exhaust or to bring physical network resources down; or to gather information about the services running in the target machine. Sniffing is one of the most common attacks: attackers attempt to access information from other tenants or cloud provider network by reading packets going through the physical NICs.

Threat Examples on Tenant-to-Tenant Scenario.

1) *Side channel*: An encryption algorithm used by

tenants is subject to cryptanalysis based on timing attacks. 2) *Guest hopping*: The attacker inserts a VM in the same host of the victim to exploit shared network components in subsequent attacks. 3) *DoS/DDoS*: Exhaustion of network resources shared by tenants inside the same host.

Threat Examples on Tenant-to-Provider Scenario.

1) *Sniffing*: By monitoring physical network interfaces an attacker may intercept provider management data. 2) *Side channel*: An encryption algorithm used by a provider is subject to cryptanalysis based on timing attacks. 3) *DoS*: Amplification attacks performed from inside the cloud, possibly combined with address spoofing can flood provider network infrastructure.

4.2 Logical Isolation

The cloud provider should ensure isolation of resources among its customers, which includes virtualized network components that compose the cloud logical network and the tenants virtual networks. Tenant-to-tenant attacks involves breaking this logical isolation, allowing a malicious tenant to get access victim's resources. This sort of attack can lead to data leakage, malicious use of other tenants' resources, and/or disruption of network services.

A malicious tenant may also exploit virtual network resources, controllers and other services provided by the cloud provider. Some examples involve the use of port scanning and network reconnaissance mechanisms, so that the attacker can get access to privileged information, such as the network topology, list of virtual networks, or configuration messages. Also, a malicious tenant may use the cloud infrastructure to deploy botnets for launching DoS attacks against the cloud infrastructure, its tenants or any external targets.

Threat Examples on Tenant-to-Tenant Scenario.

1) *Port scanning*: The attackers scans open ports and running services in other tenant VMs. 2) *Network reconnaissance*: Protocols for network configuration and identification can be exploited to discover other virtual networks and/or network topologies pertaining to other tenants (e.g., using ARP requests) 3) *Sniffing*: Sniffing and/or man-in-the-middle attacks due to the lack of isolation between virtual networks. 4) *Malware (worm)*: Malicious software that can replicate itself through and between tenant virtual networks. 5) *Botnets*: Use the cloud infrastructure to deploy botnets, in which groups of VMs running malicious computer programs target other tenants and shared network resources.

Threat Examples on Tenant-to-Provider Scenario.

1) *Network reconnaissance*: Use network reconnaissance mechanisms to discover other network topologies, VNs, and services. 2) *Malware (worm)*: Malicious software that can replicate itself through the provider network infrastructure from a compromised tenant virtual network. 3) *Botnets*: Use the cloud infrastructure to deploy botnets, targeting shared network resources and network controller nodes. 4) *Replay attack*: replication of control messages that can impair network services and cloud operations (e.g., instantiating duplicated VMs)

4.3 Authentication

Attacks that exploit authentication vulnerabilities can be performed either in Tenant-to-Tenant and Tenant-to-Provider scenarios. In both cases the attacker may exploit the authentication protocols by means of compromised-key (e.g., leakage of key material information) and/or password-based attacks (e.g., dictionary and brute force attacks). Also, the authentication and identity management modules can be exploited to force authentication threats. In the case of Tenant-to-Tenant scenario, the attacker wants to gain access to the victim's virtual networks. In Tenant-to-Provider scenario, the attacker may gain access to the whole network services and/or controlling modules, affecting the whole cloud.

Threat Examples on Tenant-to-Tenant Scenario.

1) *Credential replay*: By capturing and replaying a user's credential. 2) *Spoofing*: The malicious tenant masquerades as another tenant by falsifying data and thereby gaining illegitimate access to resources.

Threat Examples on Tenant-to-Provider Scenario.

1) *Credential replay*: The replay of a user's credential might allow impersonation of a cloud administrator. 2) *Spoofing*: The malicious tenant performs a DNS spoofing attack, inserting bogus data into a DNS name server cache database and causing the name server to return an incorrect IP address.

4.4 Authorization

In this case, the attackers try to escalate their privileges in the system. Since network privileges in the cloud usually make sense only with respect to the cloud provider, authorization attacks are reasonable only when we consider Tenant-to-Provider scenario. For instance, a malicious user may fake its credentials to acquire administrative roles, authorization to other customers services, or to gain access to services out of contract.

Threat Examples on Tenant-to-Provider Scenario.

1) *Network Reconnaissance*: A legitimate tenant can

escalate its privileges to run network reconnaissance scripts in the cloud network infrastructure, discovering the cloud provider network topology and using this information for promoting more precise attacks against the network resources. 2) *Cloud Exploit Kits (Malware-as-a-Service)*: Malicious software hosted inside cloud provider infrastructure and available to other tenants to attack the provider services through its network resources. 3) *Network Programmability*: The attackers try to escalate their privileges to get access to program APIs of the network devices (e.g., OpenFlow API).

4.5 Insecure APIs

Attacks to APIs can affect a broad range of cloud modules, e.g., those responsible for resource allocation, authentication and identity management, storage, or accounting. The network modules deployed in VMs, as well as controller, computing and network nodes, are usually distributed along the cloud infrastructure. Therefore, the APIs used for network virtualization and configuration may be target of attacks and vulnerability exploitation. For instance, attacks based on code injection techniques may exploit computer errors caused by processing invalid data, undermining cloud services and databases. This category of attack may target either Providers or Tenants.

Threat Examples on Tenant-to-Tenant Scenario:

1) *Code injection*: The attacker performs SQL injection using a network controller API (e.g., Neutron) to erase a tenant data from the cloud network configuration database.

Threat Examples on Tenant-to-Provider Scenario:

1) *Code injection*: The attacker performs SQL injection using a network controller API (e.g., Neutron) to modify (parts of) the network configuration database.

5 CONCLUSIONS AND FUTURE WORK

The wide variety of threats related to cloud computing network virtualization makes it difficult to compare or to categorize existing solutions focused on securing virtual networks. This paper proposes a threat classification for cloud virtual networks built upon the “notorious nine cloud computing top threats” of CSA. Moreover, the presented classification allows a more detailed view of the network threats discussed in cloud computing literature. This finer-grained approach makes it easier to identify the technologies that might be used to solve different security issues

in cloud networking, facilitating the analysis and design of security solutions.

As future work we plan to employ this threat classification in a literature review of cloud networking security solutions. The result expected is a comprehensive literature survey that allows not only comparing existing solutions, but also the identifying the gaps and challenges in cloud networking security.

ACKNOWLEDGEMENTS

Innovation Center, Ericsson Telecomunicações S.A. (Brazil) and CNPq (grant 305350/2013-7).

REFERENCES

- Barjatiya, S. and Saripalli, P. (2012). BlueShield: A Layer 2 Appliance for Enhanced Isolation and Security Hardening among Multi-tenant Cloud Workloads. *IEEE Int. Conf. on Utility and Cloud Comp.*, pages 195–198.
- Basak, D., Toshniwal, R., Maskalik, S., and Sequeira, A. (2010). Virtualizing networking and security in the cloud. *SIGOPS Oper. Syst. Rev.*, 44(4):86–94.
- Catteddu, D. (2010). Cloud computing: Benefits, risks and recommendations for information security. In Serrão, C., Aguilera Díaz, V., and Cerullo, F., editors, *Web Application Security*, volume 72 of *CCIS*, page 17.
- Chowdhury, N. and Boutaba, R. (2010). A survey of network virtualization. *Comput. Netw.*, 54(5):862–876.
- Cohen, R., Barabash, K., Rochwerger, B., Schour, L., Crisan, D., Birke, R., Minkenberg, C., Gusat, M., Recio, R., and Jain, V. (2013). An intent-based approach for network virtualization. In *IFIP/IEEE INM'13*.
- CSA (2011). Security Guidance for Critical Areas of Focus in Cloud Computing V3.0. Technical report, CSA.
- CSA (2013). The Notorious Nine Cloud Computing Top Threats in 2013. Technical report, CSA.
- ENISA (2013). Threat landscape 2013-overview of current and emerging cyber-threats. Technical report, ENISA.
- Gonzalez, N., Miers, C., Redígolo, F., Jr. Simplicio, M., Carvalho, T., Näslund, M., and Pourzandi, M. (2012). A quantitative analysis of current security concerns and solutions for cloud computing. *JCC*, 1(1):1–18.
- Hao, F., Lakshman, T. V., Mukherjee, S., and Song, H. (2010). Secure Cloud Computing with a Virtualized Network Infrastructure. In *Proc. of the USENIX*.
- Mattos, L. F. D. and Duarte, O. C. M. B. (2013). A Mechanism for Secure Virtual Network Isolation Using to Hybrid Approach Xen and OpenFlow. In *SBSeg'2013*.
- Mell, P. and Grance, T. (2011). The NIST definition of cloud computing (draft). Technical report, NIST.
- Myagmar, S., Lee, A., and Yurcik, W. (2005). Threat modeling as a basis for security requirements. In *SREIS*.
- Natarajan, S. and Wolf, T. (2012). Security issues in network virtualization for the future internet. In *ICNC*.

- NIST (2011). Guide to Security for Full Virtualization Technologies. Technical report, NIST.
- Pearce, M., Zeadally, S., and Hunt, R. (2013). Virtualization: Issues, security threats, and solutions. *ACM Computing Surveys (CSUR)*, 45(2):17.
- Schoo, P., Fusenig, V., Souza, V., Melo, M., Murray, P., Debar, H., Medhioub, H., and Zeglache, D. (2011). Challenges for cloud networking security. In *MNM*.
- Sun, Q. and Hu, Z. (2012). Security for networks virtual access of cloud computing. In *MINES'2012*.

Setting Priorities

A Heuristic Approach for Cloud Data Center Selection

Ronny Hans, David Steffen, Ulrich Lampe, Björn Richerzhagen and Ralf Steinmetz
Multimedia Communications Lab (KOM), TU Darmstadt, Rundeturmstr. 10, 64283 Darmstadt, Germany
ronny.hans@kom.tu-darmstadt.de

Keywords: Cloud Computing, Data Center, Quality of Service, Multimedia, Service, Heuristic.

Abstract: A rising number of multimedia applications with Quality of Service requirements is delivered via cloud computing platforms. To reduce latencies between data centers and customers, providers need to enhance and utilize their cloud infrastructure by providing resources closer to the consumer. For planning such infrastructures and efficiently assigning existing resources, capable algorithms to solve the underlying optimization problem are required. With our priority-based heuristic approach, we are able to reduce the computation time by up to 99.99% compared to an exact approach, while retaining a favorable solution quality.

1 INTRODUCTION

Over the past years, cloud computing has developed into a new paradigm for Information Technology (IT) service delivery. It enables customers to use resources according to their demand, independently of location and time. The amount of services which are provided via cloud data centers grows rapidly. While in 2012, the ratio of overall Internet traffic caused by communication with cloud data centers amounted to 46 %, it has been predicted to reach a share of 69 % in 2017 (Cisco, 2013).

Beside the increasing quantity in demand, the Quality of Service (QoS) requirements also grow. Multimedia applications – such as Desktop as a Service or cloud gaming – require low latencies, for example. Such requirements pose new challenges regarding the service delivery for cloud infrastructure providers. Even in industrial countries such as the United States, with a well-developed cloud infrastructure, only a portion of users could be serviced with sufficiently low latencies to enable services such as cloud gaming (Choy et al., 2012).

Until a few years ago, cloud providers focused on huge centralized data centers in only a few physical locations. With the advent of QoS-aware multimedia services, data centers and compute resources that are located closer to the user gain in importance. For both, the appropriate planning of such extensive compute infrastructures as well as the efficient resource allocation in existing infrastructures, appropriate algorithms are required.

The remainder of this paper is structured as follows: In Section 2, we explain the specific problem. In Section 3, we briefly present our previously published solution approaches including the mathematical model. In Section 4, we introduce our priority-based heuristic approach, which is subsequently evaluated in Section 5. An overview of related work is given in Section 6. Section 7 concludes the paper with a brief summary and outlook on future work.

2 PROBLEM STATEMENT

In this work, we consider a cloud provider who aims to provide the infrastructure for multimedia service delivery. Therefore, a set of (potential or existing) data centers in different geographical locations is assumed. Each data center may provide resource units between a lower and an upper capacity bound. The provider can choose between these data centers. Thereby, for each data center certain fixed costs, e. g., for construction or leasing, accrue. In addition, each provisioned resource unit results in variable costs. For the provided resource, the provider defines a set of relevant QoS attributes and states a QoS guarantee with respect to each user cluster and the defined QoS attribute.

The data centers should serve a set of geographically distributed user clusters. Thereby, a user cluster represents a set of user with certain demand, which is expressed in a standardized resource unit, i. e., number of servers. Regarding the delivered services, a

user cluster has certain QoS requirements with respect to each QoS attribute.

Under the assumption that prices are determined by external market conditions, the problem of a provider is the cost-minimal selection of appropriate resources, as well as setting the respective resource capacity. For the resource allocation to different user clusters, the overall service demands of all user clusters and the QoS requirements must be matched by corresponding guarantees. In our former work, we referred to this problem as *Cloud Data Center Selection Problem* (CDCSP) (Hans et al., 2013).

3 IP-/LP-BASED OPTIMIZATION APPROACHES

In this section, we briefly describe the mathematical model for the CDCSP and previously published solution approaches.

3.1 Mathematical Model

The presented mathematical model is part of our former work (Hans et al., 2013). For the model several formal notations are required. To begin with, we define the basic entities:

- $D = \{1, 2, \dots, D^\#\}$: Set of (potential or existing) data centers
- $U = \{1, 2, \dots, U^\#\}$: Set of user clusters
- $Q = \{1, 2, \dots, Q^\#\}$: Set of considered QoS attributes

Based on these basic entities, the associated parameters can be defined as follows:

- S_u : Service demand of user cluster u
- $K_d^{\min} \in \mathbb{R}$: Minimal capacity of data center d
- $K_d^{\max} \in \mathbb{R}$: Maximal capacity of data center d
- $CF_d \in \mathbb{R}$: Fixed costs of selecting data center d
- $CV_d \in \mathbb{R}$: Variable costs for per server unit in data center d
- $QG_{d,u,q} \in \mathbb{R}$: QoS guarantee of data center d w.r.t. user cluster u for QoS attribute q
- $QR_{u,q} \in \mathbb{R}$: QoS requirement of user cluster u w.r.t. QoS attribute q

Finally, in order to model the CDCSP as optimization problem, we use the following decision variables:

- x_d : Selection of a data center d
- $y_{d,u}$: Number of resource units provided by data center d to user cluster u

Model 1: Cloud Data Center Selection Problem.

$$\text{Min. } C(x, y) = \sum_{d \in D} x_d \times CF_d + \sum_{d \in D, u \in U} y_{d,u} \times CV_d \quad (1)$$

$$\sum_{d \in D} y_{d,u} \geq S_u \quad \forall u \in U \quad (2)$$

$$\sum_{u \in U} y_{d,u} \leq x_d \times K_d^{\max} \quad \forall d \in D \quad (3)$$

$$\sum_{u \in U} y_{d,u} \geq x_d \times K_d^{\min} \quad \forall d \in D \quad (4)$$

$$y_{d,u} \leq p_{d,u} \times K_d^{\max} \quad \forall d \in D, \forall u \in U \quad (5)$$

$$p_{d,u} = \begin{cases} 1 & \text{if } QG_{d,u,q} \leq QR_{u,q} \quad \forall q \in Q \\ 0 & \text{else} \end{cases} \quad (6)$$

$$\begin{aligned} x_d &\in \{0, 1\} \quad \forall d \in D \\ y_{d,u} &\in \mathbb{N} \quad \forall d \in D, \forall u \in U \end{aligned} \quad (7)$$

$$\begin{aligned} x_d &\in \mathbb{R}, 0 \leq x_d \leq 1 \quad \forall d \in D \\ y_{d,u} &\in \mathbb{R}, y_{d,u} \geq 0 \quad \forall d \in D, \forall u \in U \end{aligned} \quad (8)$$

The described objective of the CDCSP constitutes a linear, mixed-integer program, which is formalized in Model 1. In the model, Eq. 1 defines the objective of the problem. Thereby, the total cost C depends on the decision variables x_d and $y_{d,u}$ (Eq. 7). The *binary* variables x_d indicate if data center d will be constructed or leased. $y_{d,u}$ are *integer* variables that denote the number of resource units a data center d provides to a user cluster u . Eq. 2 represents the constraint that the service demand of each user cluster needs to be satisfied by the provided service units. Eqs. 3 and 4 assure that the provided capacity of each data center lies between the given lower bound K_d^{\min} and the given upper bound K_d^{\max} . Further, they functionally link the decision variables x and y . In Eq. 5 and Eq. 6 the variables $p_{d,u}$ restrict the resource allocation between data centers and user clusters, depending on the fulfillment of the QoS requirements. In Eq. 8 the binary and integer decision variables from Eq. 7 are substituted by corresponding natural variables, which is required for the LP-relaxed approach (cf. Section 3.2).

3.2 Solution Approaches

As stated earlier, the described model constitutes an Integer Program (IP) and was published as *Cloud Data Center Selection Problem* (Hans et al., 2013). Such IPs can be solved using off-the-shelf algorithms, such as branch-and-bound (Domschke and Drexl,

2004). This results in an exact (i.e., optimal) solution. However, since branch-and-bound is based on the principle of enumeration (Hillier and Lieberman, 2005), the computation time grows exponentially with the number of decision variables in the worst case. To overcome this drawback, we introduced an initial heuristic approach based on the common concept of LP relaxation (Hans, 2013). Although this approach significantly reduces the computation time, it still needs minutes for large problem instances, which makes it inapplicable for on-demand resource assignments.

4 PRIORITY-BASED HEURISTIC APPROACH

The described CDCSP forms an extension of a capacitive facility location problem. Such problems can be solved by using priority based approaches (Angelopoulos and Borodin, 2002). To efficiently find solutions for the CDCSP, we developed a priority-based heuristic approach, where the user demand is assigned to potential data centers in a stepwise manner, following specific rules regarding user cluster and data center selection. Since the approach calculates an initial solution of the optimization problem, we named it *Priority-based Start Heuristic*, in short *CDCSP-PBSH*. Our approach consists of several phases, which are described in the subsequent sections. Later on, we present a set of prioritization and cost allocation rules in detail. Since we use a generic approach, new rules can be easily added. Finally, we describe the conduction of concrete heuristic approaches.

4.1 Generic Optimization Approach

Our approach is divided into five phases, as illustrated in Figure 1. In the *Selection Phase*, the used data centers are determined. In the *Allocation Phase*, the final resource assignment is done. The purposes of the other phases, namely the *Initialization Phase*, the *Update Phase*, and the *Finalization Phase*, are primarily the preparation of the required data structures and the processing of interim as well as the final results.

4.1.1 Initialization Phase

At the beginning of our procedure, a specific problem instance of the CDCSP is analyzed and the required data structure for the subsequent phases is created. Algorithm 2 shows the corresponding pseudo code. First of all, user clusters U are added to the list for

the residual user clusters U^{res} . For each user cluster appropriate data centers are determined, which are able to provide QoS guarantees $QG_{d,u,q}$ according to the QoS requirements $QR_{u,q}$ of the user cluster for all QoS parameters $q \in Q$ (cf. line 7 - 11). The result is stored in a binary variable $p_{d,u} = \{0, 1\}$ (cf. line 9), which corresponds to the constraint in Eq. 6 in our model. The permitted data centers for each user cluster are stored in the list D_u^{per} (cf. line 12). Further, variables for the residual demand of the user clusters S_u^{res} and for the residual capacities of the data centers K_d^{res} are set (cf. line 3 and 16).

Algorithm 2: Initialization.

Start:

```

1:  $U^{\text{res}} \leftarrow U$ 
2: for all  $u \in U$  do
3:    $S_u^{\text{res}} \leftarrow S_u$ 
4:    $D_u^{\text{per}} \leftarrow \emptyset$ 
5:   for all  $d \in D$  do
6:      $p_{d,u} \leftarrow \text{true}$ 
7:     for all  $q \in Q$  do
8:       if  $QR_{u,q} < QG_{d,u,q}$  then
9:          $p_{d,u} \leftarrow \text{false}$ 
10:      end if
11:    end for
12:    if  $p_{d,u}$  is true then  $D_u^{\text{per}} \leftarrow D_u^{\text{per}} \cup \{d\}$  end if
13:  end for
14: end for
15: for all  $d \in D$  do
16:    $K_d^{\text{res}} \leftarrow K_d^{\text{max}}$ 
17: end for
```

4.1.2 Selection Phase

In this phase a first feasible solution for the CDCSP is determined in a stepwise manner. Algorithm 3 shows the corresponding pseudo code. At the beginning of each selection step, a user cluster $u \in U^{\text{res}}$ with a residual service demand $S_u^{\text{res}} > 0$ as well as a data center $d \in D_u^{\text{per}}$ with a residual capacity $K_d^{\text{res}} > 0$ are selected (cf. line 3 and 4). The selection of a user cluster depends on the priority rule (cf. Section 4.2), which is set at the beginning of this phase. From the set of possible data centers, the one with the lowest cost per service unit – depending on the cost allocation rule (cf. Section 4.3) – is selected (cf. Section 4.3). The assignment of capacities $y_{d,u}$ depends on the residual demand of the selected user cluster S_u^{res} and the residual capacity K_d^{res} of the selected data center (cf. line 5).

Within this phase, a made assignment decision is final and will not be changed in later iterations. According to the assigned capacities, the residual de-

mand and the residual capacity are reduced (cf. line 6 and 7). All selected data centers are stored in the list D^{open} (cf. line 8). If the demand of a user cluster is met or the capacity of a data center is exhausted, it will not be taken into account in the subsequent iterations (cf. line 9 and 12).

Algorithm 3: Determination of an Initial Solution.

Start: $D^{\text{open}} \leftarrow \emptyset$

```

1: while  $|U^{\text{res}}| > 0$  do
2:   if  $|D_u^{\text{per}}| = 0$  then exit without solution end if
3:    $u \leftarrow \text{SelectUserCluster}(U^{\text{res}})$ 
4:    $d \leftarrow \text{SelectDataCenter}(D_u^{\text{per}})$ 
5:    $y_{d,u} \leftarrow \min(K_d^{\text{res}}, S_u^{\text{res}})$ 
6:    $K_d^{\text{res}} \leftarrow K_d^{\text{res}} - y_{d,u}$ 
7:    $S_u^{\text{res}} \leftarrow S_u^{\text{res}} - y_{d,u}$ 
8:    $D^{\text{open}} \leftarrow D^{\text{open}} \cup \{d\}$ 
9:   if  $S_u^{\text{res}} = 0$  then  $U^{\text{res}} \leftarrow U^{\text{res}} \setminus \{u\}$  end if
10:  if  $K_d^{\text{res}} = 0$  then
11:    for all  $u' \in U^{\text{res}}$  do
12:       $D_{u'}^{\text{per}} \leftarrow D_{u'}^{\text{per}} \setminus \{d\}$ 
13:    end for
14:  end if
15: end while

```

4.1.3 Update Phase

In the previous phase, a set of data centers was opened and stored in the list D^{open} . This list serves as an improved information base and is used instead of the initial list D , which included all possible data centers. In the *Update Phase*, all assignments are reset and the required data structure is recreated, whereby the procedure corresponds to the *Initialization Phase*.

4.1.4 Allocation Phase

The *Allocation Phase* is comparable to the the previously described *Selection Phase*. Again, the solution is determined in a stepwise manner based on the prioritization and cost allocation rules. Since all data centers were determined in the *Selection Phase*, at least the fixed costs arise. Thus, the allocation of resources can be improved by focusing on different goals, as implemented by different priority and cost allocation rules.

4.1.5 Finalization Phase

During the *Selection Phase*, the relevant data centers were stored in the list D^{open} . Based on its content, values need to be assigned to the decision variables x_d . Thereby, x_d assumes the value one for all data centers in D^{open} . The amount of assigned service units

is stored in $y_{d,u}$. It assumes the value zero if no service units were assigned between a data center and the corresponding user clusters (cf. Algorithm 4).

Algorithm 4: Finalization of the Approach.

Start:

```

1: for all  $d \in D$  do
2:   if  $d \in D^{\text{open}}$  then
3:      $x_d \leftarrow 1$ 
4:   else
5:      $x_d \leftarrow 0$ 
6:   end if
7:   for all  $u \in U$  do
8:     if  $y_{u,d} = \text{null}$  then  $y_{u,d} \leftarrow 0$  end if
9:   end for
10: end for

```

4.2 Priority Rules

A major challenge of priority based procedures is the determination of the sequence in which the demanders are assigned to the supply locations (Bölte, 1994). Appropriate priority rules are required to sort the demanders in a specific sequence. Beside the demander, the selection of the supply locations can also be supported by priority rules (Angelopoulos and Borodin, 2002). For the *CDCSP* we focus on the following three quantity based prioritization rules which sequences the used clusters w.r.t. the demand, the available capacities, or both.

The *Demand Priority Rule* is used to order the user clusters $u \in U^{\text{res}}$ according to their residual service demand $S_u^{\text{res}} > 0$. The basic idea behind this rule is to prefer user clusters with a higher service demand to ensure a valid solution. Since the assignment takes place in every single step of the procedure, the prioritization of the residual user clusters may change.

In contrast to the previous rule, the *Capacity Priority Rule* focuses on residual capacities K_d^{res} of the suitable data centers $d \in D_u^{\text{per}}$. Thereby, user clusters with a lower total service supply are preferred.

Both rules have a low complexity and those may be able to find solutions with low computational effort. Nevertheless, they may lead to solutions with a lower quality since they take only demand *or* supply into consideration. Thus, the third quantity based prioritization rule, *Buffer Priority Rule*, combines both preceding rules to overcome their disadvantages. Thereby, a service buffer as the margin of residual capacities and residual service demand of each user cluster is calculated. User clusters with a lower service buffer assume a higher priority.

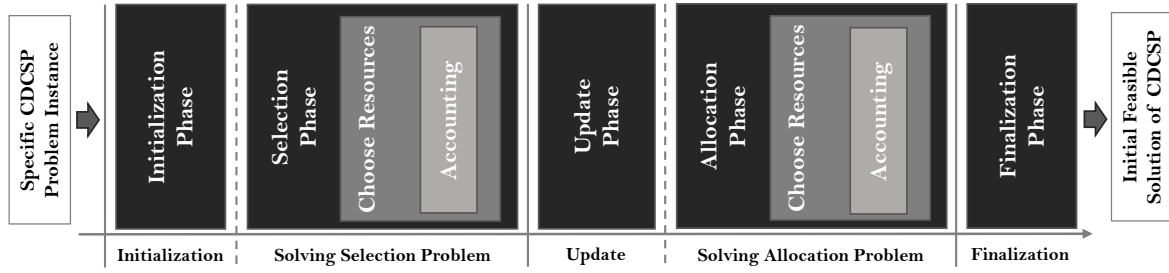


Figure 1: Phases of the Heuristic Approach.

4.3 Cost Allocation Rules

The selection of the data center is based on its costs, which consist of variable and fixed costs. Since the total amount of delivered resources is unknown until the end of the whole assignment procedure, the spread of fixed costs is a challenging task. The individual costs of a service unit $y_{u,d}$ is given by the following function.

$$C(u, d, \text{base}) = CV_d + CF_d * (1/\text{base}) \quad (9)$$

Thereby, the spread of the fixed cost depends on the value of the artificial base parameter. With a larger value of this parameter, the share of fixed costs per service unit decreases. Thus, the strategy for the determination of the base parameter directly influences the service assignment and is given by the cost allocation rules. In the subsequent section, we present two main classes of cost allocation rules.

4.3.1 Static Cost Allocation Rules

Within these rules, the value of the base parameter is determined once at the beginning of a heuristic approach and will not be changed any more. Within the *Max Capacity Cost Allocation Rule*, the maximum capacity of a data center K_d^{\max} is considered. This rule is based on the assumption that a data center is nearly completely utilized. If this is not the case, the total costs of a data center may be underestimated.

If a provider expects a utilization near the minimum capacity of a data center K_d^{\min} , the *Min Capacity Cost Allocation Rule* is more appropriate. Thereby, the minimum capacity serves as the base parameter. In case of a higher utilization, the costs per service unit are overestimated.

To strike a balance, the *Med Capacity Cost Allocation Rule* uses the medium value between the minimum and the maximum capacity of a data center.

For a given set of already opened data centers, another option is to neglect the fixed costs completely. This could be appropriate if the fixed costs arise in any case or if the number of provided service units is

very high. In this case, a sufficiently large value for the base parameter is chosen. The corresponding rule is named *No Fixed Cost Allocation Rule*.

4.3.2 Dynamic Cost Allocation Rules

In contrast to the static rules, the dynamic rules include the already existing assignments in the calculations. The value of the base parameter is calculated in each iteration of our heuristic approach and considers the current utilization of a data center.

The first of our dynamic rules, *Penalize First Cost Allocation Rule*, penalizes a user cluster which tends to open a new data center $d \notin D^{\text{open}}$. In such a case, the full fixed costs are added to the cost function. If a user cluster gets its services from an already opened data center, only the variable costs are included into the calculation. Thus, there is an incentive to use existing data centers, which is especially important during the selection phase.

Another strategy is pursued by the *Prefer Minimal Utilization Cost Allocation Rule*. This rule is based on the assumption that the opened data centers need to reach their minimum capacity constraint. Thus, data centers with an utilization lower than the minimum value get a higher priority. The cost function of such data centers only includes the variable costs, while cost function for data centers with an utilization high than a minimum capacity includes additionally the fixed costs. Especially a scenario with a given set of data centers, like the *Allocation Phase*, benefits from this rule.

The *Current Utilization Cost Allocation Rule* calculates the fixed costs based on the current utilization of a data center. Thereby, the allocation of service units between a data center and an user cluster results from the minimum of the residual demand and the residual capacity. In contrast to the previous two dynamic rules, the value of the fixed costs which is added to the cost function decreases with a higher utilization.

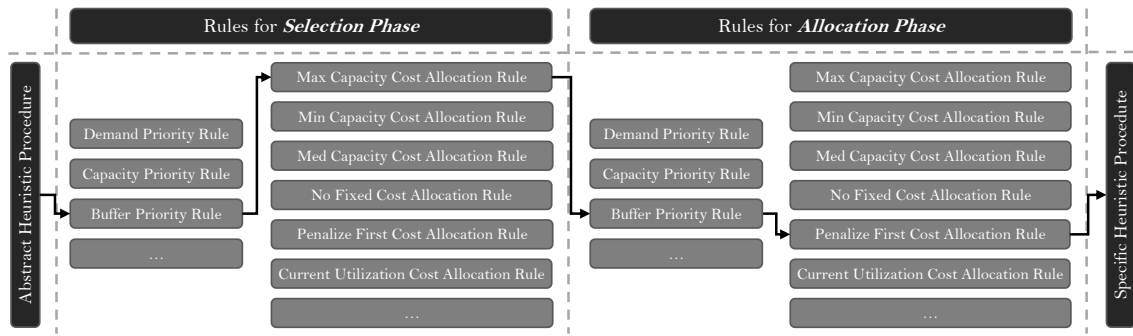


Figure 2: Deduction of Specific Heuristic Approaches.

4.4 Deduction of Specific Heuristic Approaches

In Section 4.1, we presented a heuristic approach as an abstract solution approach for the CDCSP. For both phases, the *Selection Phase* and the *Allocation Phase*, we use priority rules for the user cluster selection and cost allocation rules to choose the corresponding data centers. For both phases, all described rules are equally available. Nevertheless, some of them are more suitable than others, e.g., the selection of data centers with ignoring the fixed costs very likely leads to poor solutions. Figure 2 gives an overview of the previously described rules and shows the deduction of a specific heuristic approach, i.e., the CDCSP-PBSH[1], which is described in detail within the evaluation (cf. Section 5.2).

5 EVALUATION

5.1 Setup

In order to assess the capability of our heuristic approach, we prototypically implemented it in Java 8. As the solver for the exact and the LP-relaxed approach, we used IBM ILOG CPLEX 12.5¹, which was accessed through the JavaILP middleware².

Our evaluation focused on dependent variables, computation time and solution quality, i.e., total costs. As independent variables, we considered the number of data centers and the number of user clusters. These variables directly influence the number of decision variables, and hence, the size of the solution space.

¹<http://www.ibm.com/software/integration/optimization/cplex-optimizer/>

²<http://javailp.sourceforge.net/>

According to our former work (Hans et al., 2013), the problem instance generation was based on the 2010 United States census³. Thereby we set the service demands and different cost parameters according to the population of a randomly selected county and its median income. We focused on latency as our sole QoS parameter and set it corresponding to the requirements of multimedia services. For each *test case*, we created 100 problem instances.

Based on the samples, we subsequently computed the observed mean absolute computation times and the macro-averaged ratio of total cost along with the respective 95% confidence intervals based on a t-distribution (Kirk, 2007). The evaluation was conducted on a workstation, equipped with a Intel Xeon CPU E5-1620 v3 with 3.50 GHz and 16 GB of memory, operating under Microsoft Windows 7.

5.2 Results and Discussion

At the beginning we analyzed the performance and the solution quality, i.e., the ratio cost. We used the exact approach (CDCSP-EXA.KOM) and the LP-relaxed approach (CDCSP-REL.KOM) of our former works and the proposed heuristic approaches (CDCSP-PBSH.KOM) with all combinations of the prioritization and cost allocation rules described in this paper. Due to the large amount of evaluation results, we decided to present only two of them within this paper. The first approach was selected due to its superior solution quality for a large set of test cases, whereas the second was chosen due to its favorable computation time.

- CDCSP-PBSH.KOM [1]: Selection: *Buffer Priority Rule*, *Max Capacity Cost Allocation Rule*; Allocation: *Buffer Priority Rule*, *Penalize First Cost Allocation Rule*

³<http://www.census.gov/geo/maps-data/data/gazetteer.html>

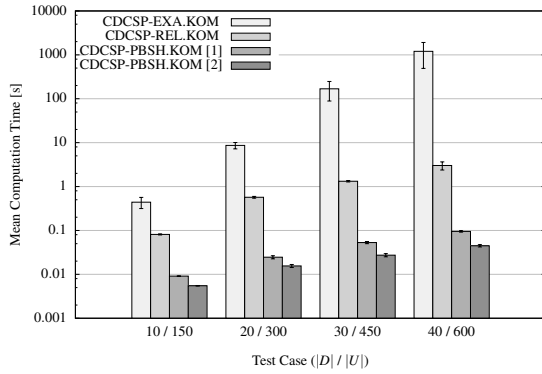


Figure 3: Computation Time (Small Test Cases).

- CDCSP-PBSH.KOM [2]: Selection: *Demand Priority Rule*, Max Capacity Cost Allocation Rule; Allocation: *Demand Priority Rule*, Penalize First Cost Allocation Rule

Figure 3 shows the computation time of the four approaches. The comparison between the exact and the second heuristic approach (CDCSP-PBSH.KOM [2]) shows a statistical significant improvement of 98.75% for the first test case ($|D| = 10 / |U| = 150$) and up to 99.99% for the last test case (40 / 600). The solution quality of the approaches is depicted in Figure 4. The chart shows the ratio of cost compared to the exact approach. In the last test case (40 / 600), the LP-relaxed approach causes 5.27% higher costs compared to the exact approach and our first heuristic approach (CDCSP-PBSH.KOM [1]) causes 5.68% higher costs. The fastest approach delivered the poorest solution quality with a cost increase of 20.68%.

In a second step, we used test cases with a larger amount of potential data centers and user clusters to evaluate the algorithms in large scale environments. Due to the high computational effort, the exact approach is not feasible in such scenarios. Again, we include the previously described heuristic approaches in this setup.

Especially the results for the heuristic approach CDCSP-PBSH.KOM [1] are very interesting. For the chosen number of data centers and user clusters, we are able to reduce the computation time by about 99% compared to the LP-relaxed approach (cf. Figure 5), while retaining the same solution quality, i.e., a cost ratio of one.

Further, for the heuristic approach CDCSP-PBSH.KOM [2], with a less complex prioritization rule, we achieve an even better computation time. However, the solution quality is significantly worse compared to the other approaches, with cost increases ranging from 0.95% to 3.53%.

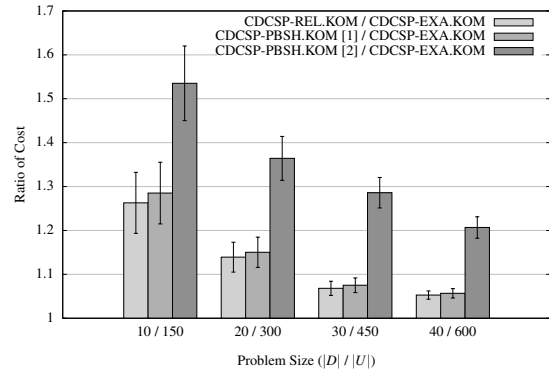


Figure 4: Solution Quality (Small Test Cases).

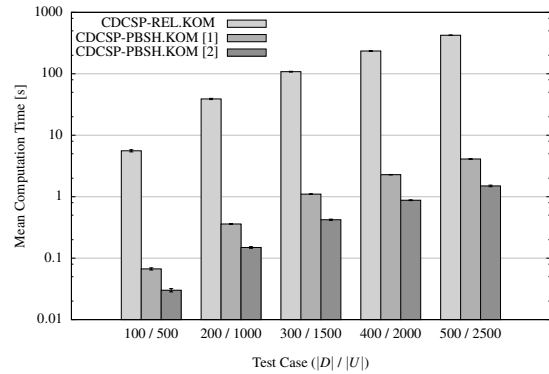


Figure 5: Computation Time (Large Test Cases).

6 RELATED WORK

A lot of work focus on data center placement and resource allocation with different optimization goals, such as the reduction of network latency or the reduction of total cost. Thereby different solution approaches like exact approaches or heuristic such as Tabu Search so Simulated Annealing are used. In this section, we present a set of selected papers, which are most relevant regarding the work at hand.

(Chang et al., 2007) investigated in the consolidation of the server infrastructure for the US army. The authors formulated a optimization problem to minimize the weighted distances between the data centers and users. (Goiri et al., 2011) also analyze the placement of data centers. Thereby, the objective is the reduction of total costs under consideration of quality requirements. The authors formulate an optimization model and solved it with LP relaxation and a simulated annealing heuristic. Both papers focus on data center placement and do not provide algorithms for run time resource allocation.

(Larumbe and Sansò, 2012) formulated an optimization problem for cloud computing which in-

cludes: Location data centers, location of software components, and routing. Therefore, the authors also consider an exact optimization approach, which is primarily appropriated for planing aspects. (Wang et al., 2012) focus on mobile cloud gaming and propose an approach for the minimization of the total costs of a cloud provider taking the individual quality requirements of the users into account. The authors develop a scheduling algorithm for assigning computation and networking resources during run time. In contrast to this work the authors do not formulate an optimization problem.

(Choy et al., 2012) focuses in their work on the availability on cloud gaming in the US. Therefore, the authors analyze the cloud infrastructure provided by Amazon and show that only 70 percent of the population can use services. They propose the use of additional data centers or Edge Server to increase the coverage. In contrast to our work, they does not propose an optimization approach for the efficient placement of such data centers and servers.

In summary, to the best of our knowledge, our work is the first to include a detailed analysis of a priority-based heuristic approach for cost-efficient selection of cloud data centers for QoS-aware services provisioning. In this context, this paper provides a generic heuristic approach, which allows substantial reduction of computation time compared to previously presented approaches.

7 SUMMARY AND OUTLOOK

In this paper, we presented a heuristic approach to a previously introduced optimization problem, the *Cloud Data Center Selection Problem*. From this generic approach, a variety of specific heuristic approaches can be deduced. Depending on the selected prioritization and cost allocation rules, either very fast heuristics approaches or heuristics with an outstanding solution quality can be configured.

Based on the presented approach, we plan two major enhancements in the future. First, we plan to develop a best-of-breed approach, which combines the benefits of multiple heuristics. Second, we plan to develop improvement procedures, such as tabu search or simulated annealing, to further enhance the solution quality of our approach.

ACKNOWLEDGEMENTS

This work has been sponsored in part by the German Federal Ministry of Education and Research (BMBF)

under grant no. 01IS12054, by E-Finance Lab e.V., Frankfurt a.M., Germany (www.efinancelab.de), and by the German Research Foundation (DFG) in the Collaborative Research Center (SFB) 1053 MAKI. The authors are fully responsible for the content of this paper.

REFERENCES

- Angelopoulos, S. and Borodin, A. (2002). On the Power of Priority Algorithms for Facility Location and Set Cover. In Jansen, K., Leonardi, S., and Vazirani, V., editors, *Approximation Algorithms for Combinatorial Optimization*. Springer.
- Bölte, A. (1994). *Modelle und Verfahren zur innerbetrieblichen Standortplanung*. Physica. In German.
- Chang, S.-J. F., Patel, S. H., and Withers, J. M. (2007). An Optimization Model to Determine Data Center Locations for the Army Enterprise. In *IEEE Military Communications Conference*.
- Choy, S., Wong, B., Simon, G., and Rosenberg, C. (2012). The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-User Latency. In *11th Annual Workshop on Network and Systems Support for Games*.
- Cisco (2013). Cisco Global Cloud Index: Forecast and Methodology, 2012-2017. Online Publication.
- Domschke, W. and Drexl, A. (2004). *Einführung in Operations Research*. Springer. In German.
- Goiri, Í., Le, K., Guitart, J., Torres, J., and Bianchini, R. (2011). Intelligent Placement of Datacenters for Internet Services. In *31st Int'l Conf. on Distributed Computing Systems*.
- Hans, R. (2013). Selecting Cloud Data Centers for QoS-Aware Multimedia Applications. In Zimmermann, W., editor, *PhD Symposium at the 2nd European Conf. on Service-Oriented and Cloud Computing*.
- Hans, R., Lampe, U., and Steinmetz, R. (2013). QoS-Aware, Cost-Efficient Selection of Cloud Data Centers. In *6th Int'l Conf. on Cloud Computing*.
- Hillier, F. and Lieberman, G. (2005). *Introduction to Operations Research*. McGraw-Hill, 8th edition.
- Kirk, R. (2007). *Statistics: An Introduction*. Wadsworth Publishing, 5th edition.
- Larumbe, F. and Sansò, B. (2012). Optimal Location of Data Centers and Software Components in Cloud Computing Network Design. In *12th IEEE/ACM Int'l Symposium on Cluster, Cloud and Grid Computing*.
- Wang, S., Liu, Y., and Dey, S. (2012). Wireless Network Aware Cloud Scheduler for Scalable Cloud Mobile Gaming. In *IEEE Int'l Conf. on Communications*.

SERVICES SCIENCE FOUNDATION FOR CLOUD COMPUTING

SHORT PAPERS

Business Process Generation by Leveraging Complete Search over a Space of Activities and Process Goals

Dipankar Deb¹, Nabendu Chaki¹ and Aditya Ghose²

¹*Dept of Computer Science and Engineering, University of Calcutta, Kolkata, India*

²*Decision System Lab, School of Computer Science and Software Engineering, University of Wollongong,*

Wollongong, NSW, Australia

dipankar.deb@gmail.com, nabendu@ieee.org, aditya@uow.edu

Keywords: Business Process Modeling, Process Redesign, Business Goal, Constraints.

Abstract: An efficient and flexible business process not only helps an organization to meet the requirements of the evolving surroundings but also may facilitate a competitive advantage over other companies towards delivering the desired services. This is even more critical for an emerging paradigm like cloud based deployment. In this paper, we introduce a novel mechanism to generate the business process suitable for specific organizations. The approach provides an automated way to build the possible business processes for a given set of tasks that fulfills the goal and satisfies the constraints of an organization. In step 1, we show how to generate the finite space of all possible designs for a given set of tasks. Secondly, we accumulate the effect of each step to deduce the final effect of each possible process design and to ensure that the redesigned set of steps still realizes the service goal. The designs not meeting the service goals are eliminated from the space. In step 3, the rest of the designs are checked for the constraint satisfaction subject to some specific cases. The framework provides a comprehensive, both syntactically and semantically correct, consistent business process generation methodology that adheres to the target business goals and constraints.

1 INTRODUCTION

There is a need to re-design the business processes over the cloud based on the requirements so that services can be offered in an efficient and cost-effective manner. Different business houses, even in the same vertical, often have their own set of distinct goals, policies and constraints. As for example, two different travel agencies may target customers of different strata of the society and can set their goals and constraints accordingly. An appropriate business process model for a particular organization should be tailor-made according to these. Given a set of tasks/activities/services and constraints, this paper aims to construct a business process for an organization. The initial set of activities is referred in rest of the text as capability library. Initially, we generate all possible set of business process designs out of these capabilities.

The manuscript is organized as follows: Section 2 presents a survey in the existing literature followed by a statement on the motivation behind the work. In section 3, syntactic derivation of the exhaustive search space is described. This ensures that no pos-

sible design is dropped out during generation of the intermittent solutions that realize the goals. Section 4 describes the checking for completeness of the proposed algorithms for generation of all possible business process models. Semantic extraction of goal specific business space from the initial syntactically correct search space followed by a running example is presented in section 5. Eliminating redundancy from business space depending on the constraints is described in section 6. In conclusion, we have discussed the effectiveness of the proposed approach in identifying the optimized business process from the reduced business space. Our approach generates all the business process designs irrespective of the business application. However, this is a one-time exercise depending on the number of tasks and may be reused for different business verticals. Subsequently, depending upon the requirements of specific business houses and their application, the goals and policies can be imposed on this exhaustive design space to have client-specific solutions with the most optimal design. The proposed solution follows the methodology of converging to the optimal design from the exhaustive design space as described in figure 1. The

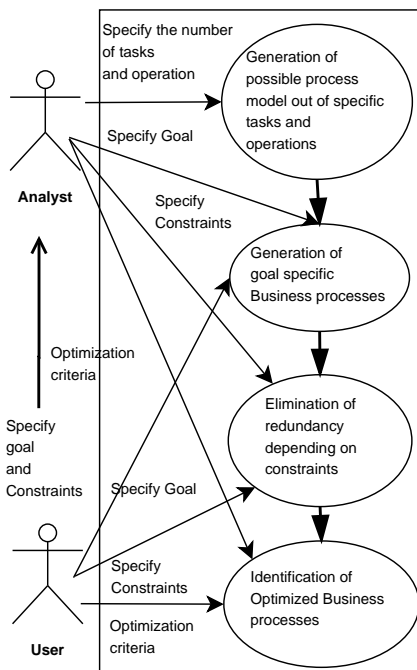


Figure 1: Use case for the proposed business process generation framework.

proposed approach provides an automated way for business process designs as compared to modeling with BPMN. The proposed mechanism may be extended towards deriving an optimal solution based on one or multiple criteria pertinent to specific clients.

2 RELATED WORK

Services can evolve typically due to changes in structure, e.g., attributes and operations; or, in behaviour and policies, e.g., adding new business rules and regulations, or, in types of business-related events; and in business protocols as presented by (Papazoglou, 2008). Thus, the issues of service redesign are very vital. Most of the literatures on business process redesign (Reijers and Liman Mansar, 2005; Liman Mansar et al., 2009; Kumar and Bhat, 2011) do not address the method to arrive at an improved process from the existing business process. A general purpose business process modeling language such as BPMN (BPMN, 2006) or UML activity designs (UML, 2003) are not designed to support enterprise in creating models using their own vocabulary and terminology. A business process modeling framework proposed in (Alotaibi and Liu, 2013) made it easy for IT people to understand and implement. (Lodhi et al., 2014) focuses on the relation between evaluation of business processes and their representation at the

process managerial level.

(Yu et al., 2014) offers a complete methodology for modeling and validating an e-commerce system with a third-party payment platform from the view point of a business process. In another recent work, (Zhang and Perry, 2014) proposes a technique for modeling composite activities by including components of data, human actors and atomic activities and represent business processes with composite activities using process-oriented languages. (Malesevic and Brdjanin, 2013) presents a software tool for the automatic visualization of presents a software tool to automate visualization of the UML activity diagram. Modeling of medical services based on business process model is been described in (Natalia et al., 2013). A new modular workflow modeling language is proposed in (Combi et al., 2014) allows the designer to easily express data dependencies and time constraints. Verification of Business Process Constraints is been demonstrated in (Gao et al., 2013). A synchronization method for change management between process models on different abstraction levels is proposed by (Weidmann et al., 2011). (Macek and Necasky, 2010) derives XML formats for communication links in the conceptual schema of the business process and optimizes them. (Wu et al., 2011) proposed to model the business process based on semantics of business process models and business vocabulary, then used the method to transform a plain text rule statement into BPMN files. The literature survey indicates some limitations and challenges in the domain of business process generation such as fulfillment of user demand, post execution analysis, automated tool support, provisioning of constraint specification and end to end solution to provide a model for the analyst. These motivate us to have an end to end solution to provide a business process design of a specific business logic incorporating the goal, constraints and optimization criteria for the analyst.

3 GENERATION OF EXHAUSTIVE FINITE SPACE

The process starts with a capability library of n tasks for an organization and set of possible model constructs such as XOR-split and AND-split which can be denoted as $\langle T_1, T_2, \dots, T_n, \otimes, \oplus \rangle$ where T_1, T_2, \dots, T_n are the capabilities and \otimes, \oplus denotes an XOR-split and AND-split respectively.

We generate all possible business process designs in the form of tree which is elaborately described in algorithm 1. A business process design tree is a tree for a given fixed capability library in which the

root node is an empty business process design, every leaf node is a syntactically correct business process designs, every non-leaf node is a partial (incomplete) business process design and every child node differs from a parent node by including a single extra process model construct (either an extra activity, or an extra event, or an extra gateway). The algorithm generates the tree considering all possible constraints for complete generation of all possible process models. The business process designs are generated on traversal of the paths of the tree. The complete process models in the tree are all leaf nodes or some intermediate nodes. The leaf nodes representing XOR-split or AND-split are followed by XOR-join or AND-join during the generation of the process tree. A path end with an XOR-split or AND-split is discarded. Such types of constraints are identified during the traversal of the path and are included in the designed algorithm 2. Figure 2 shows the tree view for the generation of process models.

Let us take a domain specific example of Car rental process where the possible subtasks for realizing the above goal are Register Request(**Task1**), Review the request(**Task2**), Reject the request(**Task3**), Allocate Car(**Task4**), Car allocated(**Task5**) and Perform Transportation(**Task6**) Using the exhaustive approach we have all possible business process designs

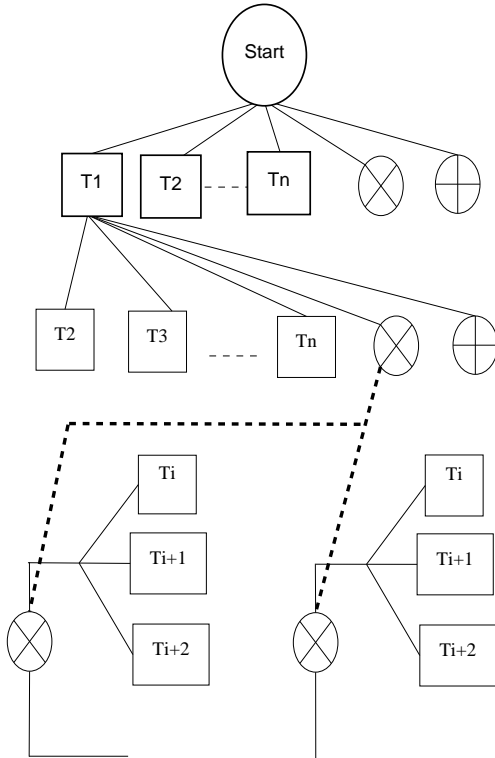


Figure 2: Process Model Generating Tree for n tasks.

with the above identified tasks T1, T2, T3, T4, T5 and T6.

4 COMPLETENESS OF ALGORITHM FOR GENERATING ALL POSSIBLE DESIGNS

The exhaustive set of syntactically correct business process designs refer to the collection of subtrees where all the artifacts are operated for the valid set of operations that are syntactically permissible. We would establish couple of base cases by manual checking for $n=1$, $n=2$ and $n=3$ and shall prove Lemma 1, Lemma 2 and Lemma 3 by the method of induction. In figure 4, we find that all the possible business process designs for $n=2$ where each of the two root tasks at level 1, are having a tree with 4 nodes. Each of these two sub-trees generates two different models. Similarly, by manual checking for $n=3$, we find that the algorithm is generating all the possible business process designs. The corresponding tree is shown in Figure 5 where each root task at level 1 is having 19 nodes in its sub-tree. For brevity we have not shown all the possible business process designs in the figure. Again, by manual checking for $n=4$, we find that the algorithm is generating all possible business process designs where each root task is having tree size=49.

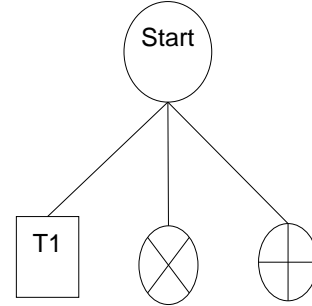


Figure 3: Process design generating tree for a single task.

Lemma 1: If n be the size of the Capability Library, i.e. the number of tasks in the capability library, then the height of the tree is equal to $(3n-4)$, $\forall n \geq 2$.

Proof: We will prove by induction that $\forall n \in \mathbb{Z}$ and $n \geq 2$ the height of the tree $H_n = 3n - 4$.

Base Case: If the size of the Capability Library $n=1$ We have only one task at level 1 of the tree. In other words, at level 1 of the tree, we have $\langle T_1, \otimes, \oplus \rangle$. The next element for a task at intermediate level is

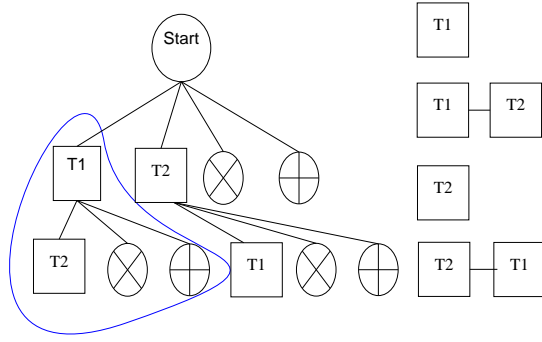


Figure 4: Process Model generating tree where number of tasks =2.

any of $n-k$ tasks where k is used tasks and the value of k ranges from 1 to n . Therefore there will be no expansion for the tree for $n=1$. The tree is shown in figure 3. Thus height of the tree is 1. If size of the Capability Library $n=2$, then its height H_2 will be $[(3 \times 2) - 4] = 2$. Figure 4 shows the height of the tree is 2. If the size of the Capability Library $n=3$, then its height H_3 will be $[(3 \times 3) - 4] = 5$. Figure 5 shows the height of the tree is 5.

Inductive Hypothesis

Assume that the theorem is true for number of tasks $\leq k$.

Inductive Steps: We must prove that the inductive hypothesis is true for $(k+1)$ numbers of tasks. During the expansion of the tree with $(k+1)$ numbers of tasks, we have the nodes of level 1 of the tree as $\langle T_1, T_2, T_3, \dots, T_k, T_{k+1}, \otimes, \oplus \rangle$. The generation of the tree terminates when the number of possible tasks at all level is equal to 1. Therefore starting with $k+1$ number of tasks, the tree expands till used tasks become $(k+1)$ and the number of possible tasks for the next level becomes zero.

We get n numbers of edges for $n+1$ numbers of tasks. Thus the height of the subtree for tasks corresponding to a level with $n+1$ numbers of tasks will be n . For each of the XOR or AND split in the corresponding level with n tasks we have 2 to $[(n+1) - k + 1]$ way split where k is used tasks. It is very obvious that the number of remaining task in the next level for $3, 4, \dots, [(n+1) - k + 1]$ way split will be less than that of 2 way split in the same level. Therefore the height of the subtree for 2 way split will be more than that of subtrees of $3, 4, \dots, [(n+1) - k + 1]$ way split. The maximum height of the subtree for an XOR or AND split will correspond to the level with maximum number of tasks. As the tree expands the number of tasks will be reduced in increasing level of the tree. Therefore the corresponding height of the subtrees for XOR or AND split with lower number of tasks in the corresponding level will be less than that

Table 1: Height of Subtrees.

No of Tasks	No. of Levels	Size of Tree
1	1	1
2	2	2
3	5	5
4	8	8
5	11	11
6	14	14
7	17	17
8	20	20

of subtrees for XOR or AND split with higher number of tasks at corresponding level. Therefore the height of the tree will be equal to height of subtree created with 2 way split at level 2. The height of the subtrees with increasing number of tasks is given in table 1. Suppose L_{T_n} and $L_{T_{n+1}}$ denotes the subtrees created with 2 way split at level 2 with n and $n+1$ number of task respectively. The height of subtree created with 2 way split at level 2 with $n+1$ number of task From the above table it is observed that (by the inductive hypothesis)

$$\begin{aligned}
 L_{T_{n+1}} &= L_{T_n} + 3 \\
 &= (3n - 4) + 3 \\
 &= 3n - 1 \\
 &= 3(n + 1) - 4
 \end{aligned}$$

Therefore, the height of subtree created with 2 way split at level 2 with n number of task is $3n-4$ and hence the height of the tree is $3n-4$.

Lemma 2: All the subtrees generated with $n-1$ capabilities are the subset of the subtrees generated with n capabilities.

Proof: We will prove by induction that $\forall n \in \mathbb{Z}$ and $n \geq 2$. $T_i(n-1)$ is the subtree of $T_i(n)$ where $T_i(n-1)$ and $T_i(n)$ denotes the tree generated with $(n-1)$ and n capabilities. Suppose $T(n)$ be the tree generated with the capability library size n which consists of m number of subtrees denoted by T_i such that

$$\bigcup T_i = T(n)$$

The start node is at level 0 has the vertices $T_1, T_2, \dots, T_n, \otimes, \oplus$ i.e. $n+2$ nodes.

Base Case: If the size of the Capability Library is $n=1$, then there will be only one process model. If the size of the Capability Library is $n=2$ then from figure 4, we find that $T_i(1)$ is the subtree of $T_i(2)$. If the size of the Capability Library is $n=3$ then from figure 5, we find that $T_i(2)$ is the subtree of $T_i(3)$.

Inductive Hypothesis

Assume that the theorem is true for k number of tasks such that $k < n$, i.e. $T_i(k-1)$ is the subtree of $T_i(k)$.

Inductive Steps: We must prove that the inductive

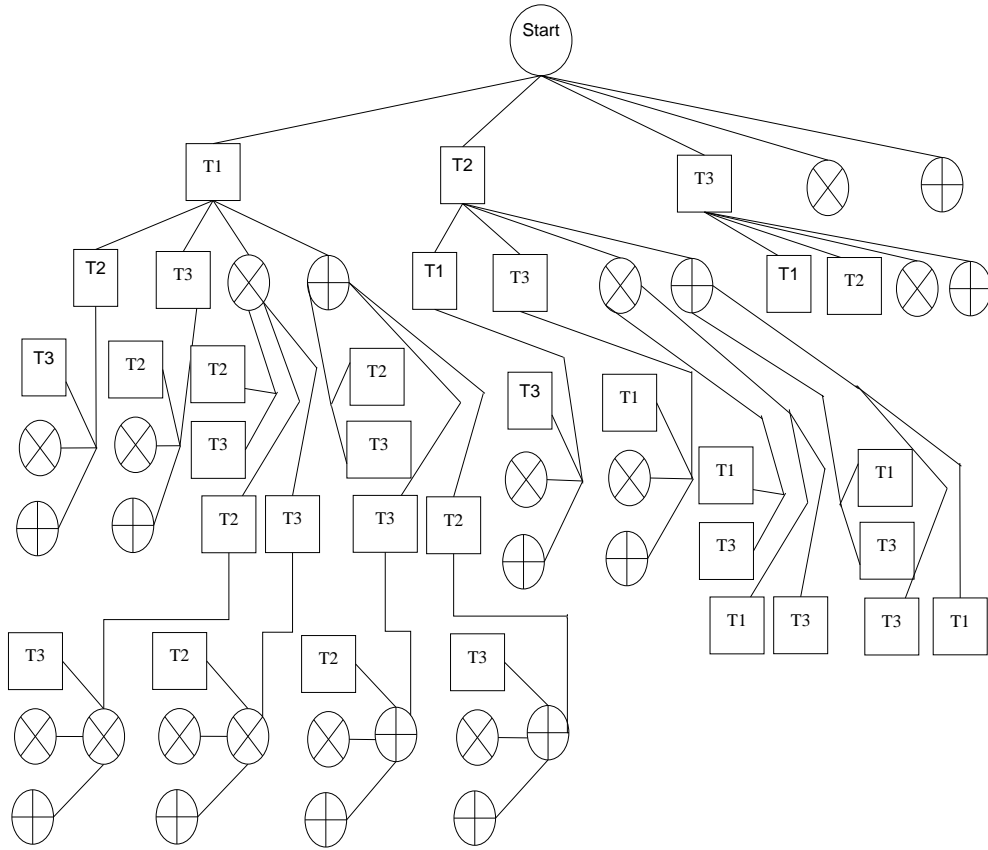


Figure 5: Process Model generating tree where number of tasks =3.

hypothesis is true for $(k+1)$ numbers of tasks, i.e. $T_i(k)$ is the subtree of $T_i(k+1)$. Now, in order to form the tree for T_{k+1} , the start node at level 0 for T_k will have the child vertices $T_1, T_2, \dots, T_k, T_{k+1}, \otimes, \oplus$ i.e., a total $k+3$ nodes will be there below the start node. In turn, node T_k in level 1 of the T_{k+1} tree will have the child vertices $T_1, T_2, \dots, T_k, \otimes, \oplus$. These nodes will again have their decedents as per procedure *ExhaustiveModelGenration()* i.e., the tree with T_k as the root node is essentially a sub-tree of T_{k+1} . Hence, it is proved that if the induction hypothesis holds good for $T_k, k < n$, then it holds good for T_{k+1} as well. Thus the statement of Lemma 2 is proved by induction.

Lemma 3: Tree having the capability library size n can generate all possible process models.

Proof: We will prove by induction that $\forall n \in \mathbb{Z}$ and $n \geq 1$, $T_i(n)$ produces all the possible process models with n capabilities where $T_i(n)$ denotes the tree generated with n capabilities.

Base Case: If the size of the Capability Library is $n=1$, then there will be only one process model. If the size of the Capability Library is $n=2$ then from figure 4, we find that $T_i(2)$ produces all the possible process model. If the size of the Capability Library is $n=3$ then from figure 5, we find that $T_i(3)$ produces all the

possible models.

Inductive Hypothesis

Assume that the theorem is true for k number of tasks such that $k < n$ i.e., $T_i(k)$ produces all the possible process model with capability library size= k .

Inductive Steps: We must prove that the inductive hypothesis is true for $(k+1)$ numbers of tasks, i.e., $T_i(k+1)$ produces all the possible process model with capability library size= $k+1$. As from Lemma 2, it is obvious that $T_i(k)$ is a subtree of $T_i(k+1)$, therefore having capable of generating all possible process models for capability library size k with $T_i(k)$ with height $(3k-4)$, procedure *ExhaustiveModel-Generation()* essentially generate all possible process model for capability library size= $k+1$ with $T_i(k+1)$ with height $(3(k+1)-4)$. Thus the statement of Lemma 3 is proved by induction.

Theorem: The proposed Construction Algorithm generates the exhaustive set of syntactically correct business process models.

Proof: Lemma 1, Lemma 2 and Lemma 3 establish that the algorithm is complete in the sense that it generates all possible business process designs. Hence, the statement of the theorem is correct.

5 EFFECT ACCUMULATION

The primary aim of this work is to redesign the business process. This means replacing the existing process model with an improved one on the basis of better optimization criteria that confirms to the business goals and constraints. So it is quite essential that each of the process models, thus generated are submitted for goal checking done by effect accumulation mechanism. The approach is domain specific. Effect accumulation enables the analyst to provide with immediate effects after each step, so as to be able to calculate the cumulative effect. To accumulate the effects of each step, we focus on the formal effect specifications. Let us define a pair-wise accumulation operator based on one first introduced in (Hinge et al., 2009). As defined $acc(e1, e2)$ to be the set of cumulative effects obtained by executing a step with effect $e2$, given a prior set of effects $e1$. An effect scenario at a given point in a process is one consistent set of cumulative effect of a process if it were to execute up to that point. The first step in effect accumulation is deriving a Scenario level. To obtain effect scenario at a given point in a process the set of scenario level is computed at that point.

Considering the case of Car rental process again all the possible process models generated by our method can also be termed as scenario levels. Out of the automatic generated scenario levels, let us take a particular scenario level $\langle S, T1, G1, T3, G2, T4 \rangle$ where S is the start event.

The effect accumulation stage involves the process of immediate effect annotation for each of the tasks listed in the scenario using a pair-wise operation when the immediate effect of S is combined with the immediate effect of $T1$, the result being the cumulative effect of $T1$. The cumulative effect at $T1$ is then combined with the immediate effect $T2$ resulting in the cumulative effect at $T2$ and so on up to $T6$. We express the effect annotations in conjunctive normal form. Let $e1$ be the cumulative effect annotation and $e2$ be the effect annotation at $t2$ and $t3$ respectively. KB be the knowledgebase which is nothing but a rule set:

$e1 = \text{request registered and request reviewed.}$

$e2 = \text{request accepted.}$

$KB = \text{the request is accepted after the review of the registered request.}$

We express the above informal representation formally in CNL (Control Natural Language) and also can provide an analyst friendly interface by means of a software.

$e1 = \text{request}(x) \wedge \text{request} - \text{review}(x)$

$e2 = \text{accept} - \text{request}(x)$

$$\begin{aligned} KB &= (\text{request}(x) \wedge \text{requestreview}(x)) \rightarrow \text{acceptrequest}(x) \\ &\equiv \neg(\text{request}(x) \wedge \text{requestreview}(x)) \vee \text{acceptrequest}(x) \end{aligned}$$

The cumulative effect of the two tasks consists of the effects of the second task plus as many of the effects of the first tasks. Two alternative effect scenario during the cumulative effect at $T3$ are $\text{request}(x)$, $\text{accept-request}(x)$ and $\text{request-review}(x)$, $\text{accept-request}(x)$. We proceed this way to gather final effect annotation at $T6$. The goal of Car rental process can be decomposed in CNL (Control Natural Language) sentences and may be combined to form a logic sentence. Let F be the set of final effect scenarios after effects are accumulated across all the steps in a service. Let G be the formal representation of the goals associated with the service. We require that the constraint $F \models G$ be satisfied. Methodologically, the redesign of the steps can involve search through a space of alternative sets of steps (including deletion or replacement of existing steps, addition of new ones and so on) provided the constraints are satisfied.

6 CONSTRAINT SPECIFICATION

Relation is an abstract association and connection that holds between two or more conceptual object. A constraint is a special kind of relationship that is restricted or compelled to exist under a given set of conditions. We have to identify the constraint between the business processes. A constraint is said to hold in a given context when the relationship is maintained in the context. In order to verify whether a constraint hold for a process we use temporal logic.

Business rules may be annotated as constraints to specify the behaviors and also to specify the derivation of conditions that affect the execution flow. These rules are forms of conditional operations attached to the process to give data result. The business rules will be restructured when organizations change the data or process to accommodate the varying business needs.

When rules are changed, it would be possible to provide a decision based on the given constraints or based on business requirements. The correct business process could also be verified by evaluating or validating the completeness of the business rule.

However, these rules may lack completeness to determine the computability of business logic. Thus, for a specific client, it is necessary to check whether the rule-set is complete. This can be proved when the rules are interpreted with temporal logic. We propose the following steps for constraint satisfaction checking. First, formally representing the design sets,

secondly, formally representing the constraints and finally use Prover9 first-order logic theorem-prover for performing checking constraint satisfaction. The constraints are considered as the conditions. The user may go through several forms to assign values to different constraints definition by example.

7 CONCLUSIONS

In this paper, we introduce a methodology that supports client-specific constraint checking towards generating goal-oriented, efficient business process designs. Subsequently, one may apply suitable criteria for optimized design. The optimization may consider issues such as delivery time, cost etc. and remains the future prospect of our current work.

The main concern about the proposed approach lies in the computation towards generating the exhaustive set of process designs. However, this step is executed offline and a priori for k numbers of tasks and offered as the template for the analyst.

Our work paves the way for constraint specification and checking. We have done completeness checking for the proposed solution. We are also in the process of developing a tool support with which the analyst can derive the optimized business process as per his/her scope, business goals and identified constraints in the environment.

ACKNOWLEDGEMENTS

This publication is an outcome of the research work undertaken in the CoE on Systems Biology and Biomedical Engineering at University of Calcutta. Authors thankfully acknowledge the support from the CoE.

REFERENCES

- Alotaibi, Y. and Liu, F. (2013). Business process modelling towards derive and implement it goals. In *Industrial Electronics and Applications (ICIEA)*, pages 1739–1744. IEEE.
- BPMN (2006). Business process modeling notation specification. www.bpmi.org. Final Adopted Specification.
- Combi, C., Gambini, M., Migliorini, S., and Posenato, R. (2014). Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(9):1182–1203.
- Gao, J., Chen, W., Wang, Y., Zhao, D., Li, W., and Bo, Z. (2013). Verification of business process constraints based on xyz/z. In *International Conference on Information Technology and Applications (ITA)*, page 479–482. IEEE.
- Hinge, K., Ghose, A., and Koliadis, G. (2009). Process seer: A tool for semantic effect annotation of business process models. In *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*, pages 54–63. IEEE.
- Kumar, M. and Bhat, J. M. (2011). Process improvement by simplification of policy, and procedure and alignment of organizational structure. In *AMCIS'11*.
- Limam Mansar, S., Reijers, H. A., and Ounnar, F. (2009). Development of a decision-making strategy to improve the efficiency of bpr. *Expert Systems with Applications*, 36(2):3248–3262.
- Lodhi, A., Köppen, V., Wind, S., Saake, G., and Turowski, K. (2014). Business process modeling language for performance evaluation. In *47th Annual Hawaii International Conference on System Science (HICSS-47)*. IEEE.
- Macek, O. and Necasky, M. (2010). An extension of business process model for xml schema modeling. In *Services (SERVICES-1), 2010 6th World Congress on*, pages 383–390. IEEE.
- Malesevic, A. and Brdjanin, D. and Maric, S. (2013). Tool for automatic layout of business process model represented by uml activity diagram. In *IEEE EUROCON*, page 537–542. IEEE.
- Natalia, C., Alexandru, M. M. and Mihai, S., Stefan, S., and Munteanu, C. (2013). Medical services modelling based on business process model framework. In *IEEE E-Health and Bioengineering Conference (EHB)*, page 1–4. IEEE.
- Papazoglou, M. (2008). The challenges of service evolution. In *Advanced Information Systems Engineering*, volume 5074 of *LNCS*, pages 1–15. Springer.
- Reijers, H. A. and Liman Mansar, S. (2005). Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4):283–306.
- UML (2003). Uml 2.0 superstructure specification. www.omg.org. Final Adopted Specification.
- Weidmann, M., Alvi, M., Koetter, F., Leymann, F., Renner, T., and Schumm, D. (2011). Business process change management based on process model synchronization of multiple abstraction levels. In *Service-Oriented Computing and Applications (SOCA)*, pages 1–4. IEEE.
- Wu, Z., Yao, S., He, G., and Xue, G. (2011). Rules oriented business process modeling. In *IEEE International Conference on Internet Technology and Applications (iTAP)*, pages 1–4. IEEE.
- Yu, W., Yan, C., Ding, Z., Jiang, C., and Zhou, M. (2014). Modeling and validating e-commerce business process based on petri nets. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 44(3):327–341.
- Zhang, Y. and Perry, D. (2014). A goal-directed modeling technique towards business process. In *IEEE 8th*

International Symposium on Service Oriented System Engineering (SOSE), page 110–121. IEEE.

APPENDIX

Algorithm 1: Algorithm for generating tree for n initial elements.

```

1: procedure GENERATETREE()
2:   Begin
3:   Create root for START EVENT at level 0;           ▷
   initialization
4:   Build level 1 with elements for all of the  $n$  distinct
   tasks, an XOR split and an AND split;
5:    $k \leftarrow 2$            ▷ variable  $k$  represents current level
6:    $T \leftarrow n - k + 1$  ▷  $T$  is the number of remaining tasks
7:   repeat
8:      $\forall$  task at level  $k$ ,  $2 \leq k \leq n + 1$ 
9:     Choose all of the remaining  $n - k + 1$  distinct
     tasks, an XOR split, and an AND split as possible next
     elements;
10:     $\forall$  XOR split at level  $k$ ,
11:    Choose the possible next elements in
12:    for  $i = 2$  to  $n - k + 1$  do
13:      Generate all possible ways of  $i$ -way split
      from  $n - k + 1$  task
14:      if thenumberofsiblingtask  $\neq$  NULL then
15:        the next element is an XOR join
16:      else
17:        return NULL
18:      end if
19:    end for
20:     $\forall$  AND split at level  $k$ ,
21:    Choose possible next elements in
22:    for  $i = 2$  to  $n - k + 1$  do
23:      Generate all possible ways of  $i$ -way split
      from  $n - k + 1$  tasks
24:      if thenumberofsiblingtask  $\neq$  NULL then
25:        the next element is an AND join
26:      else
27:        return NULL
28:      end if
29:    end for
30:     $\forall$  XOR join at level  $k$ ,  $2 \leq k \leq n + 1$ 
31:    Choose all of the remaining  $n - k + 1$  distinct
    tasks, an XOR split, and an AND split as possible next
    elements;
32:     $\forall$  AND join at level  $k$ ,  $2 \leq k \leq n + 1$ 
33:    Choose all of the remaining  $n - k + 1$  distinct
    tasks, an XOR split, and an AND split as possible next
    elements;
34:     $T = T - 1$ ;
35:  until number of remaining tasks  $T < 1$ 
36:  End

```

Algorithm 2: Algorithm for extracting all possible process from the tree for n initial elements.

```

1: procedure EXHAUSTIVEMODELGENERATION()
2:   Begin
3:   mark all nodes in the tree as .NOT. REACHED;
4:    $count = n$  ▷  $count$  stores number of nodes yet to be
   processed
5:   repeat
6:     pick any one of the node, say  $X$ , at level 1 as
     starting node;
7:     mark  $X$  as REACHED;
8:     place  $X$  on READY list;
9:      $count = count - 1$ ;
10:    repeat
11:      pick a node  $A$  from the READY list;
12:      find the child nodes for  $A$ ;
13:      if  $A$  represents XOR or AND then
14:        discard the node;
15:        Break;
16:      end if
17:      if  $A$  and its child node represents two con-
       secutive XOR or AND split then
18:        discard the nodes;
19:        Break;
20:      end if
21:      if node  $A$  representing XOR or AND split
       that do not have siblings then
22:        discard the nodes;
23:        Break;
24:      end if
25:      if the same tasks occurs after XOR or AND
       split or join node then
26:        discard the nodes;
27:        Break;
28:      end if
29:      mark the node  $A$  as REACHED;
30:      add  $A$  to READY list;
31:       $count = count - 1$ ;
32:      print the READY list;
33:    until READY list is empty;
34:  until  $count < 1$ 
35:  End

```

“BPELanon”

Protect Business Processes on the Cloud

Marigianna Skouradaki¹, Vincenzo Ferme², Frank Leymann¹, Cesare Pautasso², Dieter H. Roller¹

¹*Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany*

²*Faculty of Informatics, University of Lugano (USI), Lugano, Switzerland*

{skourama, dieter.h.roller, leymann}@iaas.uni-stuttgart.de, {vincenzo.ferme, cesare.pautasso}@usi.ch

Keywords: Anonymization, BPEL, Workflows, Business Processes.

Abstract: The advent of Cloud computing supports the offering of many Business Process Management applications on a distributed, per-use basis environment through its infrastructure. Due to the fact that privacy is still an open issue in the Cloud, many companies are reluctant to move their Business Processes on a public Cloud. Since the Cloud environment can be beneficiary for the Business Processes, the investigation of privacy issues needs to be further examined. In order to enforce the Business Process sharing on the Cloud we propose a methodology (“BPELanon”) for the anonymization of Business Processes expressed in the Web Service Business Process Execution Language (BPEL). The method transforms a process to preserve its original structure and run-time behavior, while completely anonymizing its business semantics. In this work we set the theoretical framework of the method and define a five management layers architecture to support its realization. We developed a tool that implements the “BPELanon” method, validate its functionality and evaluate its performance against a collection of real-world process models that were conducted in the scope of research projects.

1 INTRODUCTION

In the recent years the Cloud revolutionized many Information Technologies, one of the affected fields is this of Business Process Management. In this case Cloud environments are used to deploy and execute Business Processes (BP)(Amziani et al., 2012) and provide them as a Service (BPaaS)(Wang et al., 2010) that is provisioned through Platform as a Service (PaaS) (Hahn et al., 2014) solutions. The adoption of a Cloud solution can be targeted on public, private or hybrid Cloud solutions (RightScale, 2014). However, when outsourcing the BP to the public Cloud consumers lose the control of their data(Chow et al., 2009). Because of this weakness many companies are reluctant to adopt public Cloud solutions(Ko et al., 2011).

Cloud solutions have been proven more beneficiary for the companies, in comparison to the isolated business model followed by now (Accorsi, 2011). Therefore, privacy issues on the Cloud are currently discussed in the literature (Bentounsi et al., 2012; Jansen, 2011; Doelitzscher et al., 2010; Anstett et al., 2009). To reach these objectives we propose a method and implement a tool to anonymize or pseudonymize Business Processes expressed in the Business Process Execution Language (BPEL). The proposed so-

lution produces an anonymized or a pseudonymized BP for safe deployment and execution on the Cloud. The anonymized/pseudonymized BP will maintain its executional and timing behavior. In the case of pseudonymization the output of the executed BP can also be mapped back to the original, non-anonymized version of the BP.

This work has a focus on the BP, which means that the data and Web Services that surround the BP will be simulated in a “dummy” way. Later on, our solution can be extended or combined with already existing solutions for data(Sedayao, 2012; Zhang et al., 2014) and Web Services(Doelitzscher et al., 2010) anonymization to protect the company’s artifacts to the maximum possible degree. The contributions of this work are as follows:

1. identify the requirements of anonymizing or pseudonymization a BP
2. propose a method (“BPELanon”) that identifies the critical attributes and exports the anonymized BP containing the original BPEL BP without its business semantics, but solely its executable structure
3. provide and explain a tool that implements the method introduced
4. validate the tool’s functionality and evaluate its

performance through a collection of real-world BPEL BP that were conducted under the scope of research projects

This paper extends the work described in (Skouradaki et al., 2014) in terms of the realization, validation, and evaluation of a tool that supports “BPELanon” method. It is structured as follows: Section 2 analyzes the requirements and upcoming challenges of the method to be developed; Section 3 describes the design of the method; Section 4 provides a concrete realization of our approach; Section 5 validates the functionality of the “BPELanon” through case studies, and evaluates its performance against 24 real-world BP; Section 6 discusses related work that has been done for anonymization; and Section 7 summarizes and discusses an outlook to future work.

2 APPROACHING THE PROBLEM

2.1 Requirements

The design of “BPELanon” must address the following initial list of requirements identified during our work in various research projects, and especially during our collaboration with industry partners. The main requirement and purpose of method is to:

[R1:PSEUDONYMIZATION/ANONYMIZATION]

Support both pseudonymization and anonymization of BP upon the user’s choice. Pseudonymization is the technique of masking the data, while maintaining ways to the original data (Federal Ministry of Justice, 1990). On the contrary, anonymization changes the critical data and makes it impossible to trace back the original version of data (Strauch et al., 2012). Providing the option of pseudonymization makes it possible for the originator to trace bugs or inconsistencies found in the anonymized file, and apply changes to the original.

In order to guarantee the satisfaction of [R1:PSEUDONYMIZATION/ANONYMIZATION] a number of other requirements occur. These can be grouped to requirements that stem from the XML nature of BPEL. XML-specific requirements:

[R2:NO SENSITIVE INFO]

Scramble the company’s sensitive information that can be revealed in activity names, variable names, partner link names, partnerlink type names, port type names, message names, operation names, role names, XSD Element names, namespaces, and XPath expressions. The name choice for these attributes is usually descriptive, and reflects the actual actions to which they correspond. So they can reveal a lot of the

BP semantics.

[R3:NO NAMESPACES INFO]

The exported BP must not contain namespace information in incoming links to external web sites that reveal business information (backlinks).

[R4:NO BACKLINK INFO]

The exported BP must not contain names (including activity names, variable names, partner link names, partnerlink type names, message names, operation names, role names, and XSD Element names) with backlinks to business information.

[R5:NO XPATH INFO]

The exported BP must not contain XPath expressions with backlinks to business information. If no custom XPath functions are used, [R5:NO XPATH INFO] is a consequence of requirement [R4:NO BACKLINK INFO].

[R6:NO DOCUMENTATION INFO]

Remove description containers (comments and documentation) that reveal critical information and semantics.

BPEL-specific requirements:

[R7:KEEP STRUCTURE & EXECUTABILITY]

The exported BP must keep the structural information and executability.

[R8:KEEP RUNTIME]

The exported BP must maintain an equivalent run-time behavior.

[R9:KEEP TIMING]

The exported BP must maintain equivalent timing behavior.

The following requirements are related to the renaming method that will be applied:

[R10:PREVENT REVERSE ENGINEERING]

It has to be ensured that the scrambled name prevents reverse engineering to get the original names. For example if data is encrypted with a known function (e.g., RSA, MD5) and we know the used key, then it is easy to obtain the original data.

[R11:AVOID CONFLICTS]

Names must be chosen in a way that conflicts are avoided between the original and exported file. For example an easy name choice would be to change each name with respect to its type followed by an ascending ID. In this case the name of activity “Payment” would have been changed to the name “Activity1”. Nevertheless, this way is not considered safe. “Activity1” could also have been a possible name choice for the original BP as it is a word frequently met in Business Process Management. This would lead to a sequence of conflicts. Which elements named “Activity1” correspond to the anonymized element and which to the one contained in the original BP?

[R12:HUMAN READABLE NAMES]

The names must lead to an human-readable exported file. For example let's assume that we use UUIDs for name choice. That would lead to activity names such as: f81d4fae-7dec-11d0-a765-00a0c91e6bf6. The exported file would not be easy to read for humans.

2.2 Challenges

This section analyzes the challenges that stem from the need to satisfy the requirements described in Section 2.1. Each BP specification is wrapped in a package which is a directory containing all deployment artifacts. At the minimum the directory should contain a deployment descriptor, and one or more process definitions BPEL, Web Service Definition Language (WSDL), and XML Schema Definition Language (XSD) files (Apache Software Foundation, 2013). Many dependency relations among files as shown in Figure 1 increase the complexity of anonymization as small changes in a file may lead to numerous subsequent changes to other BP artifacts [C1:SUBSEQUENT CHANGES]. The complexity is also increased by the need to remove all sensitive information from the BPEL BP package [C2:NO SENSITIVE INFO]. The renaming method also needs to be carefully examined in order to keep timing, prevent reverse engineering of the anonymization, avoid conflicts between the names, and use human-readable names.

The BPEL-specific requirements reveal a new set of challenges that will be more complex to fulfill. How do data and data specific decisions affect the runtime behavior of the anonymized model? [C4:DATA CHALLENGES]. How is BPEL life-cycle affected by anonymization? [C5:BPEL LIFECYCLE]. To what extent will timing behavior be maintained? [C6:TIMING BEHAVIOUR]. We discuss these challenges in Section 5 and intend to further investigate them in future work.

3 DESIGNING THE METHOD

This section describes the method that is used for developing "BPELanon". Elements in a BPEL file can be divided into three groups:

- **Free Elements Group:** Elements that need to be anonymized, but are not bound to changes that occurred in other files.
- **Externally Bounded Group:** Elements that need to be changed because they were bounded with

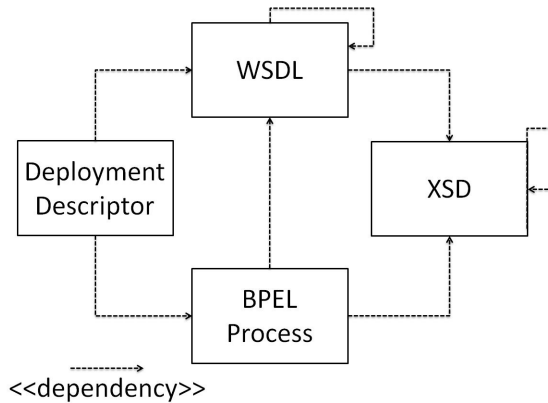


Figure 1: Dependencies of the artifacts of a BPEL BP.

elements that are changed in the WSDL or XSD files.

- **Internally Bounded Group:** Elements that need to be changed because they are bounded to other changed elements within the same file. Internally Bounded Groups can be found in both BPEL, XSD and WSDL files.

The anonymization of "Free Elements Group" is trivial, as it can be reduced to string replacement. However, the anonymization of "Externally Bounded Group" and "Internally Bounded Group" are more complex tasks. For its implementation we need a "Registry of Alterations". This is a registry of metadata that is created during the anonymization of a file and logs the occurring changes. It must contain at least the following information: the element's type, and the corresponding attributes' new and old data.

The main idea of the anonymization is to scan each artifact of the BPEL BP looking for element attributes that might contain semantics (critical attributes) that need to be scrambled. Then add to the "Registry of Alterations" their old and new value. The information on which attributes are critical can be stored with metadata. Next we scan the documents looking for references to the scrambled elements and update their values. Below we describe the anonymization method for the "Externally Bounded Group".

Anonymization starts with the creation of a meta-data schema that reflects the interconnections shown in Figure 1. Next we construct a "Table of References" that shows the relations between a BPEL BP and its WSDL files. This is done by parsing the `<bpel:import>` annotations of the BPEL file. We then process the WSDL files, which contain the definitions for the artifacts that are referenced in BPEL. We run through each one of the WSDL files in "Table of References" and start anonymizing the attributes of the elements step by step. In order to fulfill [R8:KEEP

Algorithm 1: Anonymization process of BPEL-WSDL for “WSDL Bounded Group”.

```

create TableOfReferences by parsing <bpel:import>annotations of BPEL
for all WSDL files W in tableOfReferences do
  for all elements E in W do
     $a \leftarrow \text{getCriticalAttributes}(E)$ 
    for all a do
      updateRegistryOfAlterations( $E.type, a.type, a.data, "old"$ )
      applyAnonymizationPattern( $a.data$ )
      updateRegistryOfAlterations( $E.type, a.type, a.data, "new"$ )
    end for
  end for
for all element E in BPEL file do
   $a \leftarrow \text{getCriticalAttributes}(E)$ 
  for all a do
     $resultType \leftarrow \text{findTypeOfInterconnection}(E.type, a.type)$ 
     $a.data \leftarrow \text{getNewValueOfAttribute}(resultType, a.data)$  {from registryOfAlterations}
  end for
end for
if anonymization then
  delete tableOfReferences
  delete registryOfAlterations
end if

```

RUNTIME] the function of anonymization will pick random words of an English Dictionary (WinEdt, 2000) as we argue that a word of well known human language will lead to more readable results with respect of using random strings as IDs. As discussed in Section 2.1 we only focus on the anonymization of critical attributes as not every attribute needs to be anonymized. By maintaining a “Registry of Alterations”, we apply the subsequent changes to the BPEL. We have created an outer loop that repeats this process for each WSDL file. Another option would be to parse all WSDL files and finally apply the changes to BPEL file in one parse. However, WSDL files might have common names and this would lead to more complex solution. We have therefore chosen this safer although most likely more complex in execution time solution.

At the end of the process “Table of References” and “Registry of Alterations” are destroyed if the tool is set to anonymize and not pseudonymize. Algorithm 1 describes the above procedure. For reasons of simplicity it focuses to the anonymization of a BPEL-WSDL set. However, for the anonymization of the complete set of artifacts presented in Figure 1 a similar process needs to be followed. The complete process of the BPEL BP anonymization is realized through the tool described in this paper.

4 REALIZATION

In this section we present the realization approach of

the “BPELanon” method presented in the previous section. “BPELanon” is implemented on a Java environment. As shown in Figure 3 the architecture of the realization can be separated in five different management layers. The layer “Interaction Management” refers to the part of the implementation that interacts with the user (i.e. the person that want to anonymize their BP); the layer “File Management” is responsible for the managements of the BP files; the “Anonymize Management” for the execution of the BP anonymization; the “Rename Management” to provide the new words to be used and finally the “Registry Management” to log the changes to a registry.

At this point we will move one step further, to the architectural details and see how the components of the different layers interact with each other for “BPELanon” realization. The components of the “Interaction Management” are realized through a graphical interface. With this the user can navigate through the files of the BPEL BP, configure if he needs anonymization or pseudonymization [R1:PSEUDONYMIZATION/ANONYMIZATION], and finally trigger the selected process. This user interface is depicted in Figure 2.

When the user selects the BP, then the buttons “Anonymize” and “Delete originals” become enabled. The user can choose anonymize to scramble the data of a BP (pseudonymize) and “Delete Originals” to lead to complete anonymization of the BPEL BP.

The “Importer” of the interaction layer is responsible for parsing the files, creating the corresponding

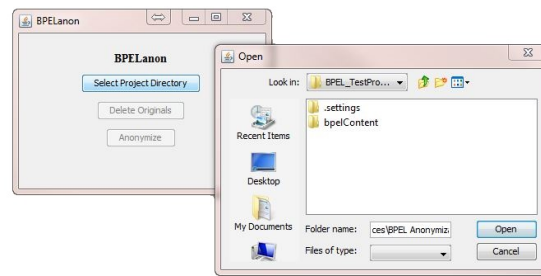


Figure 2: User Interface of the realization of Interaction Management Layer.

Java objects, and calculates the “Table of References” that is used to track down the existing dependencies. With the usage of “Table of References” we are achieving consistency in the exported file [R7:KEEP STRUCTURE & EXECUTABILITY]. The mapping of the dependencies as well as the parsed objects are then given to the “Anonymizer” component, which is basically responsible for the anonymization. In order to calculate the critical elements, their attributes, and their dependencies between the files, the layer has a special anonymizer component (BPEL Anonymizer, XPath Anonymizer, WSDL Anonymizer and XSD Anonymizer) for each one of the BP artifacts. The implementation of these components aims to the satisfaction of the requirements [R3:NO NAMESPACES INFO], [R4:NO BACKLINK INFO], [R5:NO XPATH INFO] and [R6:NO DOCUMENTATION INFO].

The “Anonymizer” component, interacts with the “Name Provider” component that is responsible for fetching and providing random new words to the “Anonymizer”. To accomplish its goal the “Name Provider” interacts with an XML database that realizes an English Dictionary retrieved from (WinEdt, 2000). With this technique we achieve to choose the new names in such way that requirements [R2:NO SENSITIVE INFO], [R10:PREVENT REVERSE ENGINEERING] and [R12:HUMAN READABLE NAMES] are satisfied. The “Anonymizer” interacts also with the “Registry of Alterations” which as discussed is responsible for logging the applied changes. By “Registry of Alterations” we can achieve pseudonymization [R1:PSEUDONYMIZATION/ANONYMIZATION] as the changes have been recorded. If the registry is deleted then we achieve anonymization. Requirement [R11:AVOID CONFLICTS] is also satisfied through the “Registry of Alterations” component as we track the changes, and do the corresponding checks to avoid conflicts. The anonymizer returns the anonymized files to the “Exporter” component that will finally save the anonymized project and notify the user through the user interface.

The last step of our realization is the execution of the anonymized BPEL BP. As expected, the anonymized BPEL BP is searching to invoke services that are anonymized, and thus nowhere implemented. In order to make the anonymized BP executable we need to create dummy services with respect to the new values. This is implemented through the functionality of creating mock-up services, offered by SOAP UI ¹. If the timing information has been initially provided from the provider of the BP, then we can add corresponding timers to the dummy services in order to satisfy [R9:KEEP TIMING]. The demonstration of the executable anonymized BP and evaluation of its time performance are discussed in Section 5.

5 VALIDATION AND EVALUATION

This section validates realization of the “BPELanon” method through case studies. We visualize the presented BPEL BP with the BPEL Designer of Eclipse IDE² and for their execution we have used the Apache ODE³.

During the validation process we had two limitations: a) we are not allowed to publish our real world processes and b) most of the real-world BP that are collected until now are not executable. This is because of the complexity to reproduce their runtime environment. For this reasons we make the first demonstration through an artificial BP. The original artificial BP is shown on the top part of Figure 5. The anonymized version is shown at the bottom of the figure. This BP represents a library BP through which a user can choose to rent or return a book.

Hence, the BP starts with a “Pick” activity (cf. “PickRentOrReturn”) in which the user chooses the desired action. In the case of book rental the user as-

¹<http://www.soapui.org/>

²<https://eclipse.org/bpel/>

³<http://ode.apache.org/>

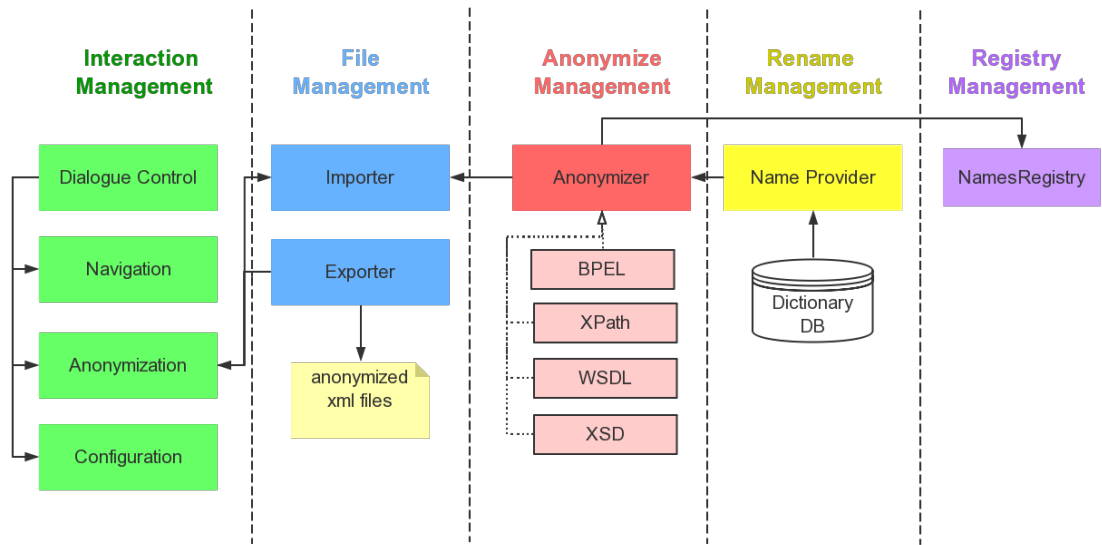


Figure 3: Architecture of “BPELanon” realization.

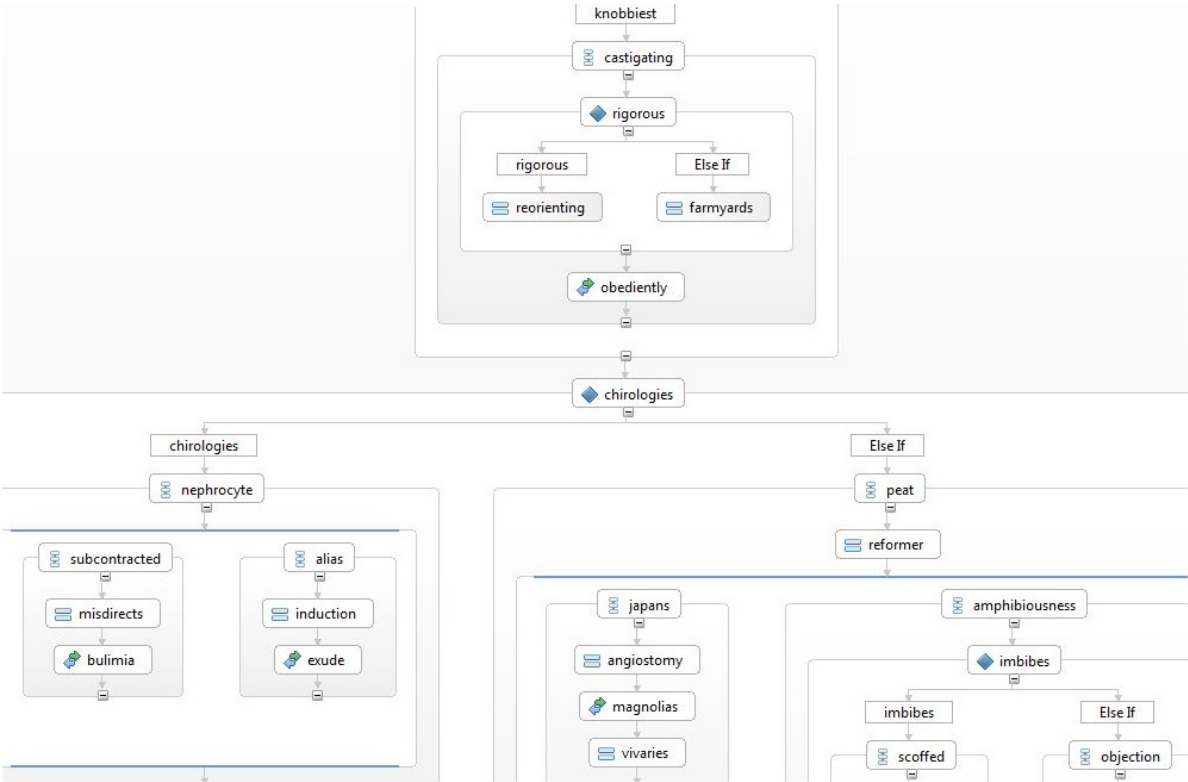


Figure 4: The anonymized real-world BPEL Business Process.

signs the ID of the book to rent and the quantity of copies. The BP waits for some seconds. This is because the “InvokeSearchService” is asynchronously invoked, and combined with a correlation activity. The “Search Service” searches for the book and availability of copies and proceeds to the “InvokeRentService”

for the book rental. In the case where the book does not exist or there is not sufficient number of copies an exception is thrown. The second flow of the BP “returnBookProcess” represents the return of a book to the library. For this the “ReturnService” is invoked (cf. “InvokeReturnService”) and the message is returned to

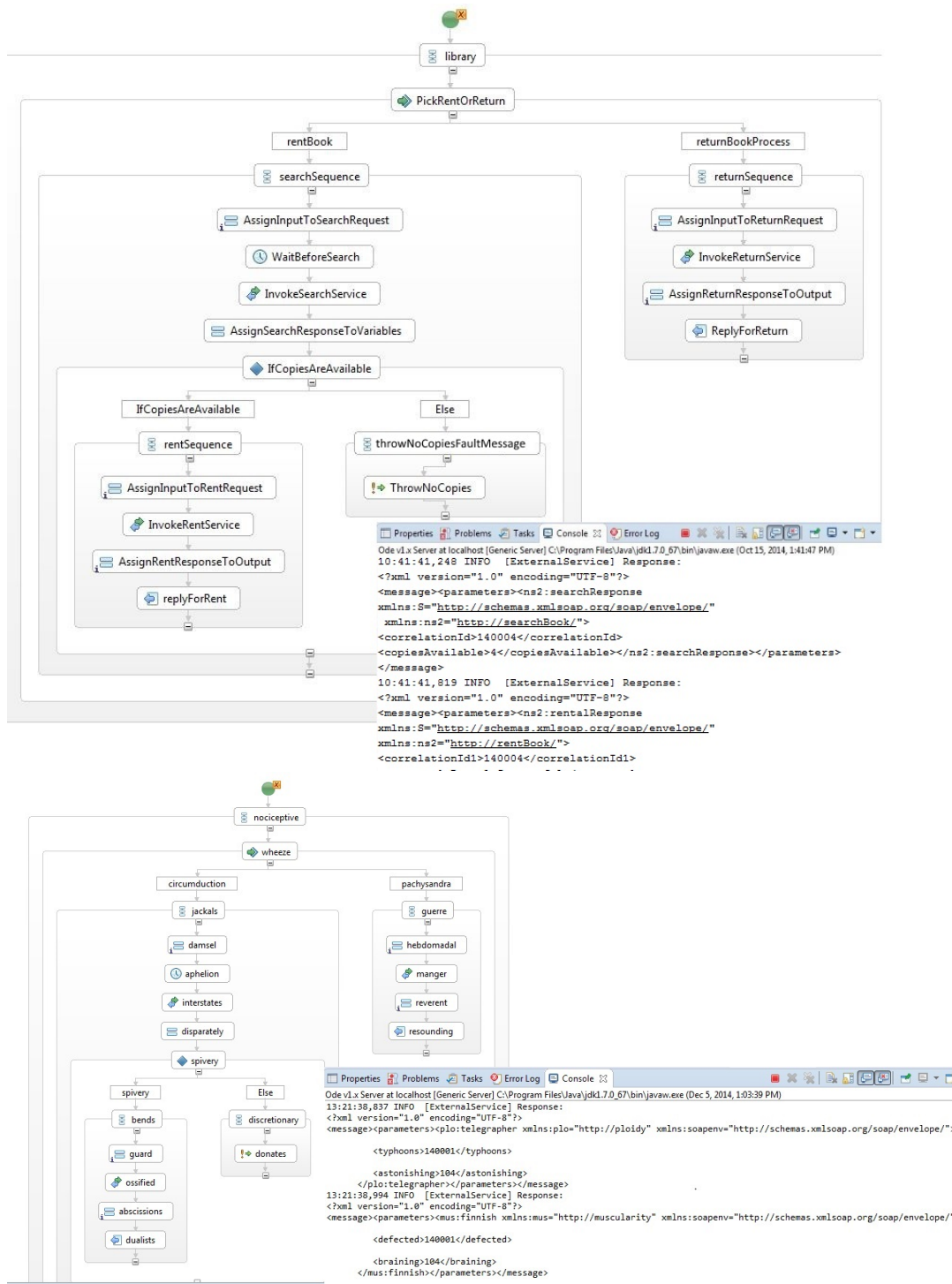


Figure 5: The original and anonymized BPEL Business Process with their execution details.

the user. The full BP package contains the BPEL file, XSD and WSDL files as shown in Figure 6. There are also XPath expressions used in many cases. One example is the “IfCopiesAreAvailable” statement, where the number of copies is compared to 0. The window

at the bottom of the BP shows an execution run where a book rental is chosen.

Moving to the anonymized version of the BP at the bottom part of Figure 5 we can see the scrambled names [R2:NO SENSITIVE INFO] of the various ele-

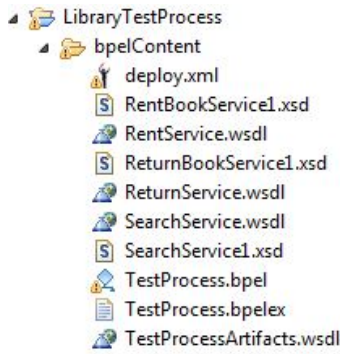


Figure 6: The structure of artifacts of the artificial BPEL BP.

ments. These correspond to the “Free Elements” group as discussed in Section 2.2.

As seen the names are human-readable [R12:HUMAN READABLE NAMES] and they are completely independent of the originals so reverse engineering is prevented [R10:PREVENT REVERSE ENGINEERING]. The anonymization of the other two groups (“Externally Bounded Group” and “Internally Bounded Group”) are basically shown through the executability of the BP. Namely, if they are not anonymized consistently the BP cannot be executed.

As seen in Figure 5 the structural information of the BP and its executability (cf. console to the bottom right corner) are preserved. Concerning executability the user still has the option to pick which BP flow to execute, input some data, and get a response. The timer is also not changed, so the timing behavior [R9:KEEP TIMING] is maintained as the rest of the activities are executed right away in both the original and the anonymized BP. The fact that the BP is executable proves that the files are consistently anonymized and that conflicts between the names in the original and anonymized files were avoided [R11:AVOID CONFLICTS]. The anonymization of namespaces (cf. `xmlns:plo="http://ploidy"` in Figure 5) apply to “Externally Bounded Group” as they need to be applied consistently to all types of files (i.e. BPEL, XSD, WSDL). The anonymization of the name of the element in a complex type in XSD file corresponds to the “Internally Bounded Group” (cf. `<defected>` and `<braining>` elements in Figure 5). Finally, as we show in the console the backlinks in the namespaces are also anonymized [R3:NO NAMESPACES INFO],[R4:NO BACKLINK INFO] and [R5:NO XPATH INFO].

As the example shown in Figures 5 has simple structure, we also validate our implementation with a real-world BP shown in Figure 4. The BP is conducted in the scope of a research project and was originally a scientific workflow. The real-world BP are confidential

and thus it cannot be shown in original format. For this reason we only provide the anonymized version of the model. Figure 4 shows a selected zoomed-out representative part of the model to demonstrate the anonymization. Despite the structural complexity of the BP it is also anonymized consistently. In the case of “Else if” elements, the name has been also anonymized, but the BPEL designer chooses to show by default the “Else if” keyword to indicate the alternative path.

In order to check the algorithm’s performance we ran our experiments on a notebook equipped with Intel Core i7-3520M CPU and 16GB RAM. The machine is running on Windows 7. As we discussed, we realized the algorithm on a Java environment. For the experiments we have used a set of 24 real-world BP that were conducted in the scope of research projects. Anonymization for each model was executed three times, and the corresponding timings were collected. Figure 7 shows how the algorithm performed for the anonymization of the models. The vertical axis shows the corresponding average execution times of the anonymization runs. The horizontal axis shows the total of XML elements of the BPEL BP artifacts and was calculated as defined by Equation 1.

$$\begin{aligned} \sum_{\{XMLElements\}} e = & \\ (\sum_{i \in \{BPEL\}} XMLelement(i) \in BPEL) \forall BPEL file + & \\ (\sum_{i \in \{BPEL\}} XMLelement(i) \in BPEL) \forall BPEL file + & \\ (\sum_{k \in \{WSDL\}} XMLelement(k) \in WSDL) \forall WSDL file + & \\ (\sum_{l \in \{XSD\}} XMLelement(l) \in XSD) \forall XSD file + & \\ \sum_{j \in \{DD\}} XMLelement(j) \in DeploymentDescriptor & \end{aligned} \quad (1)$$

As seen in Figure 7 more than half of the models (14) have up to 1000 XML elements while the rest span from the values 1000 - 5000 XML elements. Concerning the execution times we can see that the algorithm

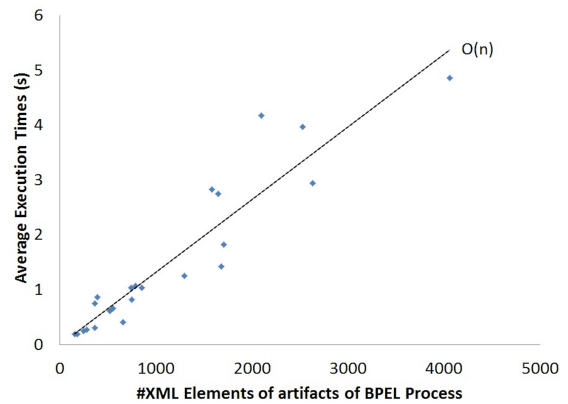


Figure 7: Performance of “BPELanon”.

performs in a linear $O(n)$ complexity where n is the total number of XML elements of the BP. For lower times we can see more points on the execution time trendline. This shows that the execution time is highly related to the number of XML elements. As the BP get more complicated we can see that the points have some distance from the execution trendline. This indicates that the XML elements are not the only factor affecting the execution times. We suspect that the structure of the BP, the total number of files to access, and the total number of the applied replacements also play a role to the performance. However, this assumption needs more data and was left for future work.

6 RELATED WORK

Cloud computing has introduced even more privacy issues, prompting researchers and companies to focus and propose ways to tackle these issues. Some of these issues have been resolved through the use of anonymization of data, Web Services or BP. (Sedayao, 2012) discuss data anonymization in Cloud environments and state that "data anonymization can ease some security concerns, allowing for simpler demilitarized zone and security provisioning and enabling more secure cloud computing". (Zhang et al., 2014) deal with the challenge of guaranteeing privacy on data shared in public Cloud infrastructures. They have a focus on data analysis and propose a privacy-preserving framework based on MapReduce⁴ on Cloud. Most of the approached we were able to find discuss about data anonymization and since the BP deal also with data, they can be seen as complementary to our approach.

In the field of Business Process Management we were not able to find any other approach that describes a distinct method to anonymize BP expressed in BPEL language. Nevertheless, anonymized BP are already used in existing projects. For example in (Kunze et al., 2011) anonymized models are used in a large public collection of BP, but the method followed to anonymize the BP is not discussed, and the BP in this collection are not in an executable format. Bentounsi et al. (Bentounsi et al., 2012) propose a method to publish BP on the Cloud by maintaining privacy. However, this approach is based on fragmenting the BP and sharing some parts of it. The sensitive data of the client are anonymized but the context of the fragment is maintained. Adopting this approach would not serve our goal, since we want to encourage the sharing of the complete BP while completely hiding any business information.

⁴<http://research.google.com/archive/mapreduce.html>

Towards the realization of our method the tools XMLAnonymizer(XMLAnonymizer, 2010) and XMLAnonymizerBean(SAPTechnical.COM, 2007) were found. XMLAnonymizer is a primary approach to anonymization that focuses on changing the attribute value of the XML file ([R4:NO BACKLINK INFO] partially covered). The XMLAnonymizerBean anonymizes elements and attributes by removing the namespaces of an XML file ([R3:NO NAMESPACES INFO] partially covered). Overall, these utilities partially satisfy the requirements of "BPELanon". The "BPELanon" method is a more complex approach since it deals with all the requirements and challenges described in section 2.2.

7 CONCLUSION

In this paper we have proposed a method ("BPELanon") for the anonymization of BPEL BP, that can be valuable when sharing BP on the Cloud, where privacy of personal data, and competitive assets are an open issue. The anonymization of a BPEL BP can be complex due to the numerous artifacts that comprise the BP, and the dependencies that exist among these files. For anonymization of a BP one needs to know the critical elements that need anonymization, and the dependencies between the participating artifacts, in order to track down the sequences of changes that need to be applied. We validated both the method and the tool through case studies of an artificial BP and a real-world BP. We evaluated the method's performance through 24 real-world BP conducted in the scope of research projects.

In future work we will investigate what is the impact of anonymization to the BPEL BP life-cycle and the ways that data and data dependent decisions are influenced by anonymization. For the complete anonymization of a BP we need to combine it or implement also methodologies for Web Service and Data anonymization. It is then essential that the first release of the complete "BPELanon" will then be distributed to companies for evaluation and usage on public Cloud environments.

ACKNOWLEDGEMENTS

This work is funded by BenchFlow project (DACH Grant Nr. 200021E-145062/1). The authors would like to thank B. V. Tahil and N. Siddam for their contribution towards the realization.

REFERENCES

- Accorsi, R. (2011). Business process as a service: Chances for remote auditing. 35th IEEE COMPSACW, pages 398–403.
- Amziani, M., Melliti, T., and Tata, S. (2012). A generic framework for service-based business process elasticity in the cloud. BPM'12, pages 194–199, Berlin, Heidelberg. Springer-Verlag.
- Anstett, T., Leymann, F., Mietzner, R., and Strauch, S. (2009). Towards bpel in the cloud: Exploiting different delivery models for the execution of business processes. ICWS'09, pages 670–677. IEEE Computer Society.
- Apache Software Foundation (2013). Creating a process. <http://ode.apache.org/creating-a-process.html>.
- Bentounsi, M., Benbernou, S., Deme, C. S., and Atallah, M. J. (2012). Anonyfrag: An anonymization-based approach for privacy-preserving bpaas. Cloud-I '12, pages 9:1–9:8, New York, NY, USA. ACM.
- Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., and Molina, J. (2009). Controlling data in the cloud: Outsourcing computation without outsourcing control. CCSW '09, pages 85–90, New York, NY, USA. ACM.
- Doelitzscher, F., Reich, C., and Sulistio, A. (2010). Designing cloud services adhering to government privacy laws. CIT '10, pages 930–935.
- Federal Ministry of Justice (1990). German Federal Data Protection Law.
- Hahn, M., Sáez, S. G., Andrikopoulos, V., Karastoyanova, D., and Leymann, F. (2014). SCE^{MT}: A Multi-tenant Service Composition Engine. SOCA'14, pages 89–96. IEEE Computer Society.
- Jansen, W. (2011). Cloud hooks: Security and privacy issues in cloud computing. HICSS '11, pages 1–10.
- Ko, S. Y., Jeon, K., and Morales, R. (2011). The hybrex model for confidentiality and privacy in cloud computing. HotCloud'11, pages 8–8, Berkeley, CA, USA. USENIX Association.
- Kunze, M., Luebbe, A., Weidlich, M., and Weske, M. (2011). Towards understanding process modeling – the case of the bpm academic initiative. volume 95 of *BPMN 2011*, pages 44–58. Springer Berlin Heidelberg.
- RightScale (2014). 2014 state of the cloud report from rightscale. <http://www.rightscale.com/lp/2014-state-of-the-cloud-report>.
- SAPTechnical.COM (2007). Xml anonymizer bean in communication channel to remove namespace prefix in xml payload. <http://www.saptechnical.com/Tutorials/XI/XMLPayload/Index.htm>.
- Sedayao, J. (2012). Enhancing cloud security using data anonymization. Intel IT, IT@ Intel White Paper. IT Best Practices, Cloud Computing and Information Security.
- Skouradaki, M., Roller, D., Pautasso, C., and Leymann, F. (2014). BPELanon: Anonymizing BPEL processes. ZEUS '14, pages 9–15.
- Strauch, S., Breitenbücher, U., Kopp, O., Leymann, F., and Unger, T. (2012). Cloud Data Patterns for Confidentiality. CLOSER '12, pages 387–394. SciTePress.
- Wang, M., Bandara, K. Y., and Pahl, C. (2010). Process as a service. IEEE SCC '10, pages 578–585. IEEE Computer Society.
- WinEdt (2000). WinEdt Dictionaries. <http://www.winedt.org/Dict/>.
- XMLanonymizer (2010). XMLanonymizer - utility to anonymize data of an xml file. <https://code.google.com/p/xmlanonymizer/>.
- Zhang, X., Liu, C., Nepal, S., Yang, C., and Chen, J. (2014). Privacy preservation over big data in cloud systems. Security, Privacy and Trust in Cloud Systems, pages 239–257. Springer Berlin Heidelberg.

Automated Mapping of Business Process Execution Language to Diagnostics Models

Hamza Ghandorh¹ and Hanan Lutfiyya²

¹Department of Electrical and Computer Engineering, Western University, London, Ontario, Canada

²Department of Computer Science, Western University, London, Ontario, Canada
{hghandor, hlutfiyy}@uwo.ca

Keywords: Web Service Composition Diagnosis, Codebook Technique, BPMN Mapping.

Abstract: This paper illustrates how a specification of a business process can be automatically mapped to a fault diagnostic model. Observed failures at run-time are quickly analyzed through the diagnostic model to determine the faulty service.

1 INTRODUCTION

Web services are loosely-coupled, self-contained, and self-describing modules that perform a pre-determined task. Services can be used in multiple applications and thus are reusable. A service of a particular type can be replaced by another service if necessary. The architectural paradigm for organizing distributed applications based on a composition of web services, which may be under different ownership, is referred to as a Service-Oriented Architecture (SOA) (Papazoglou and Van Den Heuvel, 2007). These compositions can be used to implement a business processes (Lins et al., 2012).

A *fault* (Garza et al., 2007, Alam, 2009) is a defect in either hardware or software that causes a failure. A failure occurs when a service deviates from expected behaviour. To illustrate the relationship between fault and failure consider the following example. A hardware power loss causes a service to become unavailable. The cause of the hardware power loss is the fault and the failure is that the service has become unavailable. In another example an unexpected load may result in a service provider in not providing a response in the expected time i.e., a Quality of Service (QoS) requirement may be violated. The cause of the unexpected load is the fault and the failure is the violation of the QoS requirement.

A fault (or problem) may cause multiple failures (often referred to as *symptoms*). For example, a composition of services could have service WS_i that communicates with WS_j and WS_j communicates with WS_k . If WS_k becomes unavailable then WS_j may not be able to complete a request from WS_i and thus WS_i

observes a failure of WS_j . Another example can be seen in a composition which consists of services WS_i , WS_j , WS_k and WS_l . The first three of these services are clients of WS_l . If the machine that WS_l is hosted on goes down (and thus WS_l is not available) then the other services observe a failure of WS_l . *Fault diagnosis* is used to determine a fault and often includes analysis of notifications of failures (referred to as *events*).

To provide a robust service experience, it is important to have an effective and efficient mechanism for fault diagnosis (Zhang et al., 2012a). Model-based fault diagnosis performs fault diagnosis through models. Some of these, e.g., codebook, have been shown to be effective in practice. Many fault diagnosis models require knowledge of the application configuration. With the sheer number of possible applications there is a need to automate the development of a fault diagnosis model.

This paper proposes an approach to the generation of a fault diagnosis model based on a notational representation of a business process. We show the fault diagnosis model can be used in the management of service compositions.

This paper is organized as follows: Section 2 provides the background, Section 3 presents related work on fault diagnosis, Section 4 presents the proposed approach. Section 5 describes the architecture of management system for a diagnostic module that uses our approach, Section 6 describes the results of the testing of our implementation, and Section 7 concludes the paper.

2 BACKGROUND

This section describes fault diagnosis and a notation for describing a business process.

2.1 Fault Diagnosis

The process of fault diagnosis requires the following: *fault detection*, *fault localization*, and *testing* (Steinder and Sethi, 2004). Fault detection is the process of capturing symptoms (Hanemann, 2007). Detection techniques can be based on active schemes (e.g. polling to determine availability) and/or symptom-based schemes, where a system component indicates that it has detected a failure. Examples of proposed fault detection techniques can be found in Angeli et al (Angeli and Chatzinikolaou, 2004) and Hwang et al (Hwang et al., 2010).

Fault localization typically requires an analysis of a set of observed symptoms. The goal of fault localization is to find an explanation of the symptoms' occurrence. The explanations are delivered in the form of hypotheses. Hypotheses are statements which explain that each observed symptom is caused by one or more designated problems. Based on these hypotheses, a testing step is performed in order to determine the actual problems through the application of a suitable testing mechanism (Steinder and Sethi, 2004).

There are several fault localization techniques. One of these, *event correlation*, attempts to associate one symptom with another symptom in order to infer the relationship between their occurrences (Tiffany, 2002). Through an examination of these associations, a number of possible hypotheses are generated that reflect the symptoms' occurrence. There are several different types of correlations, which are useful for diagnosing problems in a network. One of these is described in 3.

In this work when we say that we are mapping a business process specification to a fault diagnosis model we are specifically referring to a model that supports fault localization.

2.2 BPMN

One standard that can be used to model business processes is referred to as Business Process Modeling Notation (BPMN) (Alonso et al., 2004) (Endert et al., 2007). BPMN has several notational elements. An activity node represents a web service. A link represents different possible flows and is chosen based on the result of the evaluation of a condition of an activity. A gateway represents decision points that represent a workflow's conditions. A sequenceflow repre-

sents a link from a gateway node to an activity node. A pool represents the combination of a composition of flowobjects, gateways, and sequenceflows. A messageflow describes the exchange of messages between pools, and an event describes the start or end point of workflow. A pool may have an activity flowobject that can be represented by another pool. Each pool represents a workflow and a business process is associated with a set of pools. An example of a business processes workflow modelled as a BPMN specification is presented in Figure 1.

3 RELATED WORK

Steinder et al. (Steinder and Sethi, 2004) proposed a classification of fault localization techniques which is derived from graph-theoretic techniques and included techniques such as codebook, context-free grammar, and bipartite causality approaches. Graph-theoretic techniques rely on the use of graphs. The graphs include nodes that represent symptoms and problems, while directed edges are used to model the relationship between the problems and symptoms. Essentially edges represent cause-effect relationships between problems and symptoms or symptoms and other symptoms. An example is seen in Figure 3(a). To create such a model, an accurate knowledge of current dependencies among the system components is required. The rest of this section briefly describes representative work on fault diagnosis based on the relationships between problems and symptoms.

Tighe et al. (Tighe and Bauer, 2010) implemented a distributed fault diagnosis algorithm, proposed by Peng and Reggia and is referred to as *Parimonious Covering Theory* (Peng and Reggia, 1990), in a policy-based management tool called BEAT (Best Effort Autonomic Tool) (Bahati et al., 2007). The algorithm is concerned with the generation of plausible hypotheses or *covers*, based on given information that comes from graph-theoretic models, prior to diagnosis. Hypotheses are delivered and grouped in order to generate disorder-and-manifestation statements that are forwarded to a decision making system for recovery actions.

Zhang et al. (Zhang et al., 2012b) proposed a hybrid diagnosis method to diagnose web services' problems in service-oriented architectures. Their method combines dependency matrix-based diagnosis and a Bayesian network-based diagnosis. Although the authors considered the reduction of the computational complexity of services diagnosis, the hybrid diagnosis method does not cope with the dynamic nature of SOA's services, and Bayesian net-

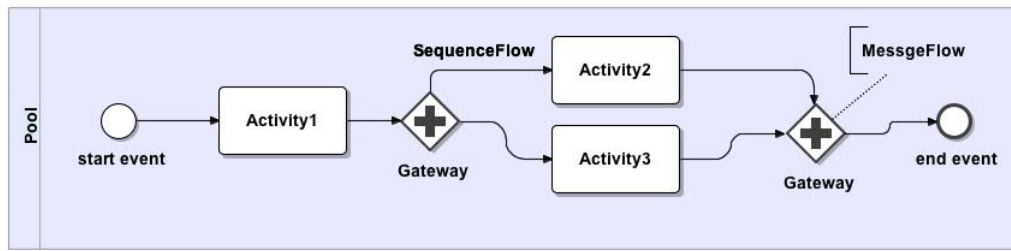


Figure 1: Simple BPMN example.

work diagnosis provides slow measurement.

Ardissono et al. (Ardissono et al., n.d.) proposed a model-based diagnostic framework with autonomous diagnostic capabilities to monitor the state of web services. As a partially distributed approach, the framework includes several local diagnosers, which is attached to a web service or a composition, cooperate with a global diagnostic service. As soon as local diagnosers notice a problem, they raise an alarm to the global diagnostic service to detect it.

Most of the above work focuses on the models. None of the work investigated shows how to automate the development of a fault diagnosis model based on a specification of a business process. However, there is work (e.g., (Morán et al., 2011)) that takes a business process specification and maps it to control rules.

4 PROPOSED APPROACH

This section describes our approach to using the BPMN specification of a business process to a fault diagnosis model.

4.1 Codebook Technique

Earlier we discussed that a fault may manifest itself in the unexpected behaviour of a web service that is observed by other web services. For our fault diagnosis model we use a fault propagation model, which describes which symptoms that may be observed if a specific fault occurs (Kätker and Paterok, 1997). The underlying mathematical structure is typically a graph (Steinder and Sethi, 2004). For this work we chose the codebook technique (Kliger et al., 1995). This technique was implemented in a network fault diagnostic system and the results (Yemini et al., 1996) suggest that this approach is highly scalable.

The codebook technique or coding technique (Steinder and Sethi, 2004) uses a causality graph and problem code (PC) matrix of a web service composition's workflow to locate the source of failures. A causality graph is a bipartite graph whose vertices can

be partitioned into two disjoint subsets V and W such that each edge connects a vertex from V to one from W (Caldwell, 1995). A PC matrix is a matrix representation of a causality graph used to infer the causes of observed symptoms. The PC matrix is built based on the causality graph. An example of the causality graph and the matrix are illustrated in Figure 3(a) and 3(b), respectively. The matrix consists of a column that represents symptoms that problems cause. A matrix entry either has the value of zero or one. For example, the value of one assigned at $PC[1, 1]$ position in PC matrix indicates that symptom S_1 can be observed for problem P_1 . The value of zero assigned at $PC[1, 3]$ position indicates that symptom S_1 can not be observed for problem P_3 .

At run-time a problem will cause one or more symptoms to be generated. From this a string can be formulated. If the i^{th} symptom was observed then the i^{th} position in the string is one otherwise it is zero. This string will be referred to as a *current symptoms vector* (CSV).

The diagnosis process uses the *Hamming distance*. The Hamming distance is the minimum number of substitutions that transforms one string into the another. For example, the Hamming distance between two words "toned" and "roses" is three letters and the Hamming distance between the two strings 1011101 and 1001001 is two bits (MacKay, 2005). Each value in a column in the PC matrix is compared with its corresponding code in a given CSV. If both values are identical (i.e., the value in the column in the PC matrix and its corresponding code in the given CSV are the same), the Hamming distance value is denoted as zero. Otherwise, the Hamming distance is denoted as one. The values are then summed to determine the Hamming distance of the two words. The minimum of the Hamming distance values is an indicator of the corresponding problems as the causative problems. For the PC matrix, if the given CSV is 11000, the Hamming distance is (0,4,4) for columns labeled P_1, P_2 and P_3 respectively. Thus, the causative problem was P_1 . If the given CSV is 11101, the Hamming distance is (2,2,4) for columns labeled P_1, P_2 and P_3 . Thus, the causative problems are limited to P_1 and P_2 .

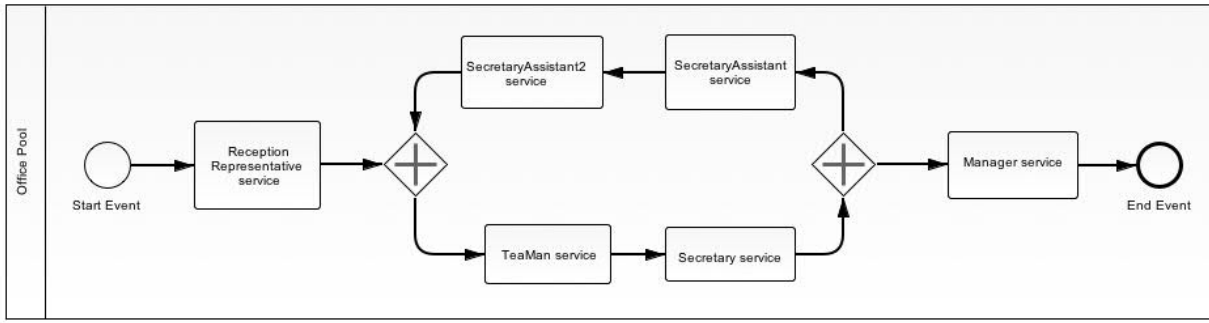


Figure 2: Office Business Process BPMN.

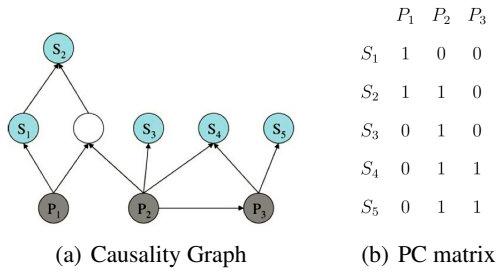


Figure 3: Example of causality graph and PC matrix (Steinder and Sethi, 2004).

4.2 BPMN Mapping

The BPMN mapping is done through the transformation from BPMN graphs to a composition dependency (CD) graph which is done prior to determining the causality graph. For illustration purposes, Figure 2 presents BPMN model for an office business process, which is concerned with delivering only important mails to the manager office through different filters. The transformation from BPMN to a CD graph is performed as follows: assume that a CD graph is represented as (V, E) . Each BPMN atomic activity node is a node in V . If a decision point follows an activity then the node in V representing the activity will have two outgoing edges. Edges represent different possible flows. Figure 4 depicts the CD graph for the office business process, where P_1 represents the Reception-Representative service, P_2 represents the TeaMan service, P_3 represents the Secretary service, P_4 represents the SecretaryAssistant service, P_5 represents the SecretaryAssistant2 service and P_6 represents the Manager service. We note that the granularity of the model is limited to a service. Hence a problem, P_i , corresponds to a service. We will use the notation P_i to refer to both a problem and to a service.

Assume that the CD graph is represented as (V, E) while the causality graph is represented as (V', E') ¹.

¹The causality graph vertices are known in advance

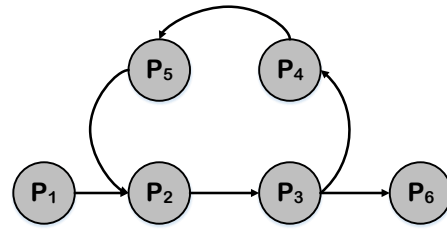


Figure 4: Abstract View of Office Business Process.

The set V' can be partitioned into two sets W, X such that each edge in E' connects a vertex from W to a vertex in X . The set W is the set of potential problems. Since each node in the CD graph represents an activity and any of these activities can be faulty then the set of W is the same as the set V . Let v be a node in a CD graph. This node represents a potential problem. Any node, u , in the CD graph, for which there exists a path from it to the node v , potentially could exhibit a failure condition if v becomes faulty. Any node that could exhibit a failure condition is in set X . For a node u we use the notation P_u to represent u as a problem and S_u to represent u as a symptom. Determining the causality graph of the CD graph requires these two algorithms: *Modified Depth-first Search* (mdfs), and *path-Generator*. The mdfs and pathGenerator algorithms are presented in algorithm 1 and algorithm 2, respectively. The mdfs algorithm takes as input a CD graph and does a depth-first traversal. When all child nodes of node v have been traversed then the pathGenerator algorithm is used to generate all paths from node v to each leaf node. These paths are used to produce the causality graph. The causality graph of the office business process is depicted in Figure 5.

The mdfs algorithm uses two variables: *Vertices-List*, and *BackTrackEdgesList*. VerticesList is a list that keeps track of each node's label. The BackTrackEdgesList maintains a list of backtrack edges. A backtrack edge (v, w) indicates that the mdfs algo-

based on the given information from a client about fault and symptom quantities

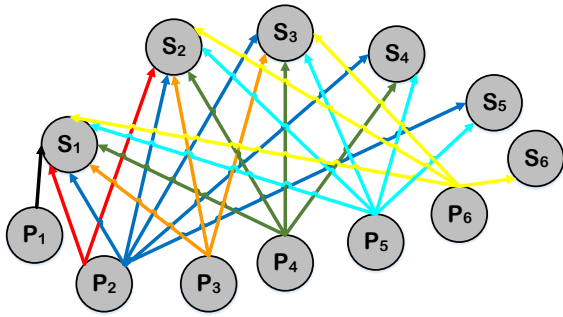


Figure 5: Causality graph of the office business process.

rithm is revisiting node w and that not all of node w 's children had yet been visited. *White* is a label that indicates an unvisited node, which is the initial state for all nodes. *Gray* is a label that indicates a node has been visited but not all of its children have been traversed. *Black* is a label that indicates a node has been visited and all of its children have been processed. When the input CD graph is received, mdfs is triggered (line 1). If the current node being visited is *White*, mdfs will assign the *Gray* label (line 3). The mdfs algorithm examines each outgoing edge (lines 4-5). If the node on the other end of the edge is labelled *White* then this means that the node has not been visited and thus no paths have been generated (lines 6-7). If the node on the other end of the edge is labelled *Gray* then the edge is put in the BackTrackEdgesList (lines 8-9). If there is no unvisited neighbour node for the current node, mdfs executes the pathGenerator algorithm in order to generate paths (line 12).

The pathGenerator algorithm is executed when all nodes on the other end of the outgoing edges of node v have been visited. The pathGenerator uses three variables: *newPath*, *pathsW*, and *Paths*. The *newPath* variable is used to represent a sequence of nodes, and *pathsW* represents a set that contains all the paths from w to all leaf nodes. *Paths* is a container for all possible paths. The pathGenerator algorithm is executed when a current node v is received from mdfs. The pathGenerator looks for outgoing edges of node v . If there are no outgoing edges (line 2), the pathGenerator algorithm creates a new path, appends v node in this path, and adds the path to *Paths* (lines 5-7). If there are one or many outgoing edges (line 8), the pathGenerator algorithm retrieves each path associated with w and creates a new path by putting together v and the path associated with w (lines 10-19).

The execution of the algorithms does not always provide all paths. This happens where there is a cycle. The existence of backtrack edges indicate a cycle. Assume a backtrack edge: (v, w) . The mdfs algorithm will generate all paths from node w to leaf nodes

but the paths generated for node v will not include those paths that start at w . For example, the edge (P_5, P_2) is a backtrack edge in Figure 4. The paths from the root node (P_1) to all nodes in the office CD graph are: $((P_1), (P_1, P_2), (P_1, P_2, P_3), (P_1, P_2, P_3, P_6), (P_1, P_2, P_3, P_4), (P_1, P_2, P_3, P_4, P_5))$. After considering the backtrack edge (P_5, P_2) , the paths will be: $((P_1), (P_1, P_2), (P_1, P_2, P_3), (P_1, P_2, P_3, P_6), (P_1, P_2, P_3, P_4), (P_1, P_2, P_3, P_4, P_5), (P_1, P_2, P_3, P_4, P_5, P_2))$. Paths generated considering backtrack edges are done after mdfs terminates. Let (v, w) be a backtrack node. Let P be the set of paths. For each path that ends with w create a new path that appends v to the path that ends with w .

Algorithm 1: Modified depth-first search(mdfs).

Procedure: mdfs executed on receipt Graph G with root node v

Input : $G = (V, E)$ where
 $E = \{(v, w) | v, w \in V\}$ and node v is a zero indegree edge and all nodes v are initially unvisited.

Variables : VerticesList carries on all nodes, *White* is label for unvisited node state, *Gray* is label for the visited but not finished node state. *Black* is label for the finished node state. BackTrackEdgesList carries on edges resulted from visiting *Gray* nodes.

```

1 mdfs( $G, v$ )
2 if VerticesList[ $v$ ] = White then
3   VerticesList[ $v$ ] = Gray
4   forall the  $e \in G.incidentEdges(v)$  do
5      $w = G.incidentEdges(v, e)$ 
6     if VerticesList[ $w$ ] = White then
7       mdfs( $G, w$ )
8     else if VerticesList[ $v$ ] = Gray then
9       putEdge( $v, w$ , BackTrackEdgesList)
10  VerticesList[ $v$ ] = Black
11  // when there are zero unvisited
   nodes, backtrack
12  pathGenerator( $v$ )

```

4.3 Diagnostic Models

The Codebook technique (Steinder and Sethi, 2004) is used as our diagnostic model. Each path generated starts from a node v and ends at a node w . If a problem occurs in node w then it is possible that symptoms are detected by each node in the path. Thus each path generated is represented in PC matrix as a column. We see this with Figure 5 and Table 1.

Algorithm 2: pathGenerator.

Procedure: pathGenerator executed on receipt
a graph G and node v

Input : Graph G and node v from mdfs

Variables : newPath, pathsW, and Paths

Output : Possible set of paths

```

1 begin
2   if  $G.incidentEdges(v) == null$  then
3     // Create a new path, add  $v$ 
       node in this path, and add the
       path to Paths
4     newPath = null
5     newPath.append( $v$ )
6     Paths = Paths  $\cup$  newPath
7   else
8     forall the  $e \in G.incidentEdges(v)$  do
9        $w = G.incidentEdges(v, e)$ 
10      pathsW = emptySet
11      // Retrieve all previously
        generated paths from  $w$  to
        each leaf node reachable
        from  $w$ 
12      forall the  $p \in Paths.get(w)$  do
13        newPath = null
14        newPath.append( $v$ )
15        newPath.append( $p$ )
16        pathsW.add( $p$ )
17      Paths = Paths  $\cup$  pathsW

```

Table 1: Problem codes matrix for the office business process.

	P_1	P_{2_1}	P_{2_2}	P_3	P_4	P_5	P_6
S1	1	1	1	1	1	1	1
S2	0	1	1	1	1	1	1
S3	0	0	1	1	1	1	1
S4	0	0	1	0	1	1	0
S5	0	0	1	0	0	1	0
S6	0	0	0	0	0	0	1

By apply the mdfs and pathGenerator algorithms on the office CD graph, in Figure 5, since $S4$ can be observed for P_4 , the $PC[4, 4]$ is assigned the value of one. Since symptom $S5$ can not be observed for P_6 , $PC[5, 6]$ has been assigned the value 0. All codes assigned to present the causality relationships in Figure 5 are portrayed in table 1. In table 1, there are two columns representing different patterns that result in symptoms associated with the web service that is associated with problem P_2 . These are represented by (P_{2_1}, P_{2_2}).

Fault diagnosis assumes a vector of symptoms that have been reported. It is assumed that these symp-

toms are generated by a failure detection component located within a composition. The Hamming distance between the vector and each column is calculated. The lower the value of the Hamming distance the more likely that the column explains what is causing the symptoms.

For the office business process assume that the following symptoms are observed: P_1 says that P_2 has timed out, P_2 says that P_3 has timed out, P_3 says that P_4 has timed out, and P_4 says that P_5 has timed out, and P_5 says that P_6 is not responding. For this pattern of symptoms, the CSV is 111110. Based on the PC matrix for the office business process, the result list is depicted at Table 2. From Table 2, the causative web service for the observed symptoms are P_2 and P_5 since they have the minimum values between their peers.

Table 2: Result list of the office business process.

	P_1	P_{2_1}	P_{2_2}	P_3	P_4	P_5	P_6
S1	0	0	0	0	0	0	0
S2	1	0	0	0	0	0	0
S3	1	1	0	0	0	0	0
S4	1	1	0	1	0	0	1
S5	1	1	0	1	1	0	1
S6	0	0	0	0	0	0	1
Σ	4	3	0	2	1	0	3

5 ARCHITECTURE

Section 4 presents an approach to automating the development of a fault diagnostic model. This model is part of the diagnosis module of a third party third party policy-based management system (Hasan, 2011). The management system allows for Service-Level Agreements (SLAs) to be negotiated. These SLAs formalize the QoS requirements. Policies are used for three types of decisions: *service selection*, *SLA violation* and *recovery* policies (Hasan, 2011). The service selection policy is defined by clients to guide choice of services. The violation policy specifies what constitutes a violation of an SLA. The recovery policy is defined by clients that specifies recovery actions to be taken when the management system detects a SLA violation.

5.1 TPA

A key component in the management system is the third party agent (TPA). The TPA carries out these tasks: (1) allows all clients, providers, and provided services to be registered with it; (2) negotiates SLAs,

polices, and keeps track of violated SLAs; (3) generates events to indicate failures and performs recovery actions. An overview of the TPA is presented as Figure 6. The Registration Gate is responsible for (1) forwarding a business process specification to the BPMN Repository, which stores the BPMN specification for each composition being managed by the TPA. This is one of the inputs for the Diagnosis Module. (2) forwarding relevant information about clients and providers to the Negotiator. The Negotiator is responsible for maintaining an agreement (i.e. SLA) between a client and a service provider if both parties have a match between the former's needs and the latter's specification. These agreements are stored in the Contract Repository. The Event Generator relies on the stored information found in logs storage, such as, information related to service invocations. The Event Generator also "uses SLAs and SLA violation policies to generate events that represent SLA violations ... when the number [SLA violations] exceeds what is specified in the SLA violation policy then an event is generated" (Hasan, 2011). The diagnosis module receives the generated events and uses the generated diagnostic model to deliver a diagnostic hypotheses. The Recovery Agent is responsible for analysing the diagnosis module's hypotheses and executing reactive actions.

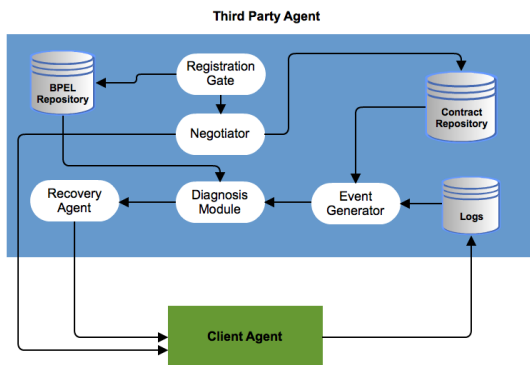


Figure 6: TPA with the Client Agent.

5.2 Diagnosis Module Overview

Our proposed diagnosis module provides a hypothesis about the source of symptoms observed in a composition. The basic module architecture is presented in Figure 7. There are main three components: (1) The Mapper which transforms received BPMN specifications to PC matrix; (2) The Event Coordinator which transforms the generated events to CSV; (3) The Matcher which is responsible for matching PC matrix and CSV to deliver a hypothesis to the Recovery Agent. The Mapper is only used for new appli-

cations or if an application is modified. Otherwise at run-time only the Event Coordinator and Matcher are used. We note that our model narrows the problem to a service. Further tests could be carried out to further narrow down the root cause. However, for recovery purposes it may be sufficient to know the service that is causing failures and the action could be to select another instance of the same type.

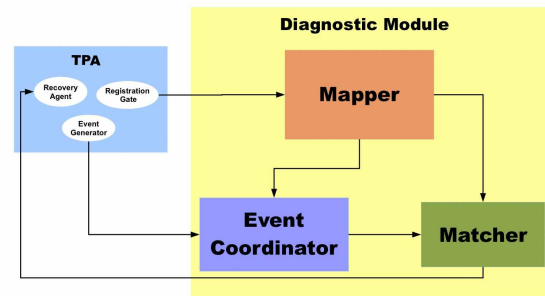


Figure 7: Diagnosis module with the TPA.

6 EVALUATION

After we implemented the Mapper, the Event Coordinator, and the Matcher components, we tested our diagnosis module on composition description graphs to see if the module is able to accurately and correctly determine the source of events. We ran the diagnosis module on a single machine with 2.66 GHz Intel Core 2 Duo processor, Mac OS X 10.6.8, and eight gigabyte 1.07 GHz memory. We used Netbeans 7.0.1 IDE to run tests and create or manipulate CSVs. For the transformation from BPMN to the composition description graphs, we used a tool referred to as the BPMN Modeler, which is an extension of eclipse IDE (Eclipse, 2011). The BPMN Modeler is responsible for creating a BPMN for a business process and forwarding a BPMN textual description to the Mapper component.

We applied our diagnosis module to nine subjects which consists of: single or many joins (i.e. single or many vertices' edges ending in one vertex), single or many splits (i.e. single or many vertices' edges starting from one vertex and ending at an other vertex), single or many cycles (i.e. single or many vertices' edges starting and ending at the same vertex), self cycles (i.e. single vertex' edges is starting and ending at the same vertex), and trees (i.e. single or more vertices are interconnected in a hierarchical manner). For each performed test, we assumed that one fault could happen for each subject. For each subject we did a test for each web service going down. All evaluation results and specifications and execu-

Table 3: Nine CD graphs specifications.

No	CD Graph	Vertices Number	Edges Number	Single Cycle	Self Cycle	Many Cycles	Single Split	Many Splits	Single Join	Many Joins	Diagnosis Time ⁴	Execution Time ⁵
1	Office CD	6	6	•			•		•		3.2	1.68
2	CD 1	7	9		•	•					5.8	1.85
3	CD 2	6	6	•							2	1.65
4	CD 3	10	10					•	•		13.2	2.24
5	CD 4	11	12					•		•	9	2.30
6	CD 5	16	20					•		•	32.8	5.87
7	CD 6	100	114			•		•		•	328.8	16.16
8	CD 7	9	11					•		•	9.4	1.93
9	CD 8	33	34	•				•	•		32.8	5.43

⁴ Time measured in milliseconds⁵ Time measured in seconds

tion time of composition dependencies graphs are presented in table 3. A correct diagnosis was found 100% of the time. In cyclic composition description graphs, the diagnosis module indicates not only the problematic node but also the closest predecessor node to the causative node. The reason is that both the causative node and the predecessor node have the same code in the PC matrix. Thus, any faults occurring in either these nodes will generate the same events in the composition.

7 CONCLUSION

This paper focused on an automated mapping of a business process specification to a diagnostic model. By using our diagnosis module the complexity of diagnosis can be hidden from system administrators by outsourcing this functionality to a third party agent. The proposed approach enhances the automated diagnosis for a large number of compositions. This section briefly discusses the work and possible future work.

Scalability. There are two aspects to this. At run-time there is a need to compare a set of symptoms with each column of the problem code (PC) matrix. There has been considerable work on making this fast as noted in (Steinder and Sethi, 2004) and the work on a network fault management system (Yemini et al., 1996) shows that the use of the codebook can be very effective at run-time. This suggests that this approach will be scalable at run-time for service compositions. The second aspect is the generation of the PC matrix. This requires two algorithms: mdfs and path-Generator. The mdfs algorithm is based on a modified depth-first search algorithm. Although compositions may be large, it is unlikely they will be so large that it would not be feasible to run the algorithms. We

note that the generation of the PC matrix only needs to be done once for a specific composition. If a web service is replaced by another web service there is no need to generate a new PC matrix. If the composition changes then a new PC matrix needs to be generated. However, as future work will look at reusing part of the computation of the PC matrix for an older version of the application in order to reduce the time to create a new PC Matrix if the application topology changes.

Granularity. The granularity of the diagnosis model is limited to each service. If a service is considered to be a problem then a set of tests needs to be carried out to investigate why the service is a problem e.g., is the host down; is the service down. Furthermore it may be possible to use information in error messages to improve the granularity. This will be a topic of investigation for further studies. However, we note that for recovery purposes the level of granularity may often be satisfactory. If a service often violates its SLA then it may be feasible to replace it with another service of the same type. The reasons for SLA violation are not necessarily relevant.

Mappings. This work considered only mapping from a BPMN model to a codebook fault diagnosis model. Further work will look at other businesses processes specifications as well as other fault diagnostic approaches. The current version of the diagnosis module only uses the the codebook technique. Since the coding phase is performed only once, the codebook approach is very fast, robust, and efficient. However, the accuracy of the codebook technique is hard to predict when more than one problem occurs with overlapping sets of symptoms. In addition, since each change of system configurations requires regenerating the codebook, the technique is not suitable for environments with dynamically changing dependencies (Steinder and Sethi, 2004). We will enable the module to use several event correlations techniques

by which the module will be able to regenerate more efficient diagnostic knowledge bases.

ACKNOWLEDGEMENTS

The research for this paper was financially supported by the Ministry of Education of Saudi Arabia, and College of Computer Science and Engineering at Taibah University² and the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

- Alam, S. (2009), Fault management of web services, Master, University of Saskatchewan.
- Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004), *Web Services: Concepts, Architectures and Applications*, 1st edition edn, Springer.
- Angeli, C. and Chatzinikolaou, A. (2004), 'Online fault detection techniques for technical systems: A survey', *International Journal of Computer Science and Applications* 1, 51–64.
- Ardissono, L., Console, L., Goy, A., Petrone, G., Picardi, C. and Segnan, M. (n.d.), 'Towards self-diagnosing web services', <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.75.1874>.
- Bahati, R. M., Bauer, M. A. and Vieira, E. M. (2007), Policy-driven autonomic management of multi-component systems, in 'Proceedings of 2007 the conference of the center for advanced studies on Collaborative research', ACM, pp. 137–151.
- Caldwell, C. (1995), 'Graph Theory Glossary', <http://www.utm.edu/departments/math/graph/glossary.html#b>. Online; accessed 05-June-2011.
- Eclipse (2011), 'BPMN Modeler', <http://eclipse.org/bpmn/>. Online; accessed 17-Oct-2011.
- Endert, H., Hirsch, B., Küster, T. and Albayrak, S. (2007), Towards a mapping from bpmn to agents, Proceedings of the 2007 international workshop and 2007 conference on Service-oriented computing: agents, semantics, and engineering, Springer-Verlag, Berlin, Heidelberg, pp. 92–106.
- Garza, A., Serrano, J., Carot, R. and Valdez, J. (2007), Modeling and simulation by petri networks of a fault tolerant agent node, in 'Analysis and Design of Intelligent Systems using Soft Computing Techniques', Vol. 41 of *Advances in Soft Computing*, Springer Berlin / Heidelberg, pp. 707–716.
- Hanemann, A. (2007), Automated IT Service Fault Diagnosis Based on Event Correlation Techniques, PhD thesis, LMU Mnchen: Faculty of Mathematics, Computer Science and Statistics.
- Hasan, M. S. (2011), Policy Based Third Party Web Service Management, Master, University of Western Ontario.
- Hwang, I., Kim, S., Kim, Y. and Seah (2010), 'A survey of fault detection, isolation, and reconfiguration methods', *Control Systems Technology, IEEE Transactions on* 18(3), 636–653.
- Kätker, S. and Paterok, M. (1997), Fault isolation and event correlation for integrated fault management, in 'Integrated Network Management V', Springer, pp. 583–596.
- Kliger, S., Yemini, S., Yemini, Y., Ohsie, D. and Stolfo, S. (1995), A coding approach to event correlation, in 'Integrated Network Management IV', Springer, pp. 266–277.
- Lins, F., Damasceno, J., Souza, A., Silva, B., Aragão, D., Medeiros, R., Sousa, E. and Rosa, N. (2012), 'Towards automation of soa-based business processes', *International Journal of Computer Science, Engineering and Applications* 2(2), 1–17.
- MacKay, D. J. (2005), Binary codes, in 'Information Theory, Inference & Learning Algorithms', Cambridge University Press, pp. 206–227.
- Morán, D., Vaquero, L. M. and Galán, F. (2011), Elastically ruling the cloud: specifying application's behavior in federated clouds, in 'Cloud Computing (CLOUD), 2011 IEEE International Conference on', IEEE, pp. 89–96.
- Papazoglou, M. P. and Van Den Heuvel, W.-J. (2007), 'Service oriented architectures: approaches, technologies and research issues', *The VLDB journal* 16(3), 389–415.
- Peng, Y. and Reggia, J. A. (1990), *Abductive inference models for diagnostic problem-solving*, Springer-Verlag New York, Inc.
- Steinder, M. and Sethi, A. S. (2004), 'A survey of fault localization techniques in computer networks', *Science of Computer Programming* 53(2), 165–194.
- Tiffany, M. (2002), 'A Survey of Event Correlation Techniques and Related Topics', <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.5339>. Online; accessed 19-Dec-2010.
- Tighe, M. and Bauer, M. (2010), Mapping policies to a causal network for diagnosis, in '6th International Conference on Autonomic and Autonomous Systems', pp. 13–19.
- Yemini, S. A., Kliger, S., Mozes, E., Yemini, Y. and Ohsie, D. (1996), 'High speed and robust event correlation', *Communications Magazine, IEEE* 34(5), 82–90.
- Zhang, J., Huang, Z. and Lin, K.-J. (2012a), A hybrid diagnosis approach for qos management in service-oriented architecture, in 'Web Services (ICWS), 2012 IEEE 19th International Conference on', IEEE, pp. 82–89.
- Zhang, J., Huang, Z. and Lin, K.-J. (2012b), A hybrid diagnosis approach for qos management in service-oriented architecture, in '19th IEEE International Conference on Web Services', pp. 82–89.

²Taibah University site <https://www.taibahu.edu.sa/Pages/AR/Home.aspx>

Cross-layer Service Adaptation

State-of-the-Art, Shortcoming Analysis, and Future Research Directions

Ameni Meskini¹, Yehia Taher², Rafiqul Haque³ and Yahya Slimani¹

¹University of Carthage, INSAT, LISI research Laboratory, Tunis, Tunisia

²Laboratoire PRISM, Université de Versailles/Saint-Quentin-en-Yvelines, Versailles, France

³Laboratoire d'Informatique en Image et Systèmes d'information, Université Claude Bernard Lyon 1, Lyon, France
{ameni.meskini, yehia.taher, yahya.slimani, akm-rafiqul.haque}@gmail.com, prism.uvsq.fr,
akm-rafiqul.haque@univ-lyon1.fr, fst.rnu.tn

Keywords: Web Service, Service based Application, Service Adaptation, Cross-layer Adaptation.

Abstract: In the past few years several cross-layer monitoring and adaptation technologies have been proposed. Although these are cross-layer adaptation technologies, however, in practice they focus on a particular layer. Some solutions involves two layers, yet none of the existing solutions do not consider all the layers during adaptation process. Furthermore, cross-layer adaptation approaches generate incompatibility problems. This is an adaptation coordination problem. Incompatibility refers to the situations where the adaptation is performed in a layer is not compatible with the constraints exposed by the other layers. This survey aims at studying and analyzing current approaches for web services adaptation, discussing their shortcomings and proposing research directions on cross-layer web service adaptation.

1 INTRODUCTION

Service adaptation has drawn enormous research interests in the area of Service Oriented Computing (SOC) (Geihs et al, 2009). Adaptation from the functional point of view can be defined as an ability of a Service Based Application (SBA) (Bucchiarone et al, 2009) to adapt changes or requirements that are needed to guarantee fault-tolerance or to optimize system performances.

While developing an SBA, it may not be possible to capture all functional and nonfunctional requirements because many times the requirements evolve at runtime. Typically, an application is able to carry out the operations (at runtime) that are studied and documented during requirement analysis. The unprecedented requirements that are evolved at runtime may lead to failure. In other words, applications are unable to perform the operations that have not been realized.

This limitation gave rise to the notion of service adaptation. Many new techniques such as service replacement or adding new service have been proposed to build adaptive SBAs. However, there are several challenges that cannot be dealt with efficiently by the existing techniques. One major challenge is handling the impact of adaptation

operations. For instance, service replacement at different layers of SBAs. It is worth noting that an SBA has different layers (Papazoglou et al, 2008). The notion of multi-layer SBAs relies on the Service Oriented Architecture (SOA) (Liu et al, 2011) paradigm. These layers interact with each other.

Thus, if a service is adapted in one layer (e.g., BPM layer), it may affect the other layers (e.g., orchestration layer). This promotes the notion of cross-layer adaptation which is the main focus of this paper. Cross-layer adaptation is a process of adapting a service in different layers of SBAs. It promotes configuration challenges. Several techniques have been proposed to tackle these challenges. This paper aims to investigate all existing technologies, methodologies, and techniques related to cross-layer adaptation. It presents a comprehensive review of the state-of-the-art, summarizes their strengths and weaknesses, and identifies future research direction in this area.

This paper is organized as follows. In Section 2, we present the review of the state-of-the art. We discuss our findings in Section 3. A conclusion is drawn in Section 4.

2 SERVICE ADAPTATION APPROACHES

Although our focus in this paper is cross-layer adaptation, we cover all the existing technologies related to service adaptation. The purpose of this study is to provide a comprehensive understanding of the strength of service adaptation and also to outline the limitations why traditional adaptation technologies are unable to assist in cross-layer adaptation. It is worth noting that although our study mainly covers the service based systems, we try to cover adaptation in the agent based systems as well.

2.1 Interaction-based Adaptation Approaches

Interaction-based adaptation approaches deals with interactions between web services such as, actions required to mediate communications between web services, and adapting the compositions of web services in case of failure of a component. Unlike the approaches that focus on QoS adaptation (of instance composition), interaction-based adaptation approaches are concerned with the changes and the adaptation of interactions within a service composition.

For example, re-engineering services to ensure that they can integrate new components by guaranteeing interoperability. Sometime adaptation operations such as substitute to repair or to optimize QoS are not sufficient for efficient adaptation. The main reason is the emergence of new requirements and additional constraints which may not be possible to handle efficiently by the selected services. For instance, a service may not be able to meet user needs or may fail to handle heterogeneity of the interfaces between services or communication protocols. The objective of adaptation in this case is not only to manage the QoS adaptation, but also to ensure that the adaptation measures do not lead to interaction failures. The composition and the mediation are the most common solutions in such situations.

In interactions-based adaptation, existing approaches realize exchanging messages in service compositions based on pre-defined policies such as the policies proposed in (Baresi et al, 2007). This involves the business processes which are essentially composition of services. We found several services composition-based adaptation approaches which we discuss in this section. Baresi et al. (Baresi et al, 2007) address the problem of substitution of services

and the dynamic binding of the service providers in order to repair failures. Their work targets the adaptation of the workflows (defined using BPEL (IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, 2003) at runtime to select between available alternatives based on nonfunctional requirements, or to retry a service following in the first choice. To enable the deployment and the reconfiguration of service compositions during its execution, the authors used a specification of BPEL process which is enriched by a set of rules and constraints for the discovery or dynamic service binding until the time to execute.

The choice depends on the criteria defined by the user during the establishment process. The proposed framework can also exchange services based on the events collected during the monitoring phase. It relies on three actors: the registry service (DIRE) (Baresi et al, 2007) that can be distributed between the service providers, the runtime environment (SCENE) (Baresi et al, 2007) with the rules of discovery and binding, and monitoring features (Dynamo) (Baresi et al, 2005) and (Baresi et al, 2007) that produce events to reconfigure the processes.

The main limitation of this approach is the web service composition language. The authors in (Ardagna et al, 2007) propose an implicit approach for adaptive composition of services within the flexible processes. This approach is implemented in business process management layer. The main objective is to select the best set of services available at run-time by taking the constraints of business process, users preferences, and execution contexts into account.

The authors introduce a new approach to model the problem of service selection. This approach is effective for large process and in the case of QoS constraints are at extreme. In the proposed model, the problem of service selection is formalized as a mixed problem of linear programming, the loop peeling is adopted in optimization, and the constraints posed by the stateful web services are considered.

2.2 Mediation-based Adaptation Approaches

While composing interactions, services may encounter heterogeneity problems. For instance, interaction types can be different, incompatible communication protocols; different semantics of interactions promote the heterogeneity problem.

These problems may occur in different steps of

composition. Also, the problems may occur while adaptation actions carried out such as, during substitution of a WS by another WS.

The solution of the heterogeneity problem is called mediation which is critical to achieve adaptation and composition of services. However, additional mechanisms are needed for successful interactions between web services and to perform different adaptation actions (e.g., substitution, re-selection, composition, etc.). Adaptation (in this case “mediation”) is an important functionality which enables integration of business services. Generally speaking, mediation resolves conflicts between two actors. In the context of web services, mediation aims to resolve heterogeneity between web services in order to enable successful interactions (Chaffle et al., 2006).

One needs to generate a service that ensures interactions between the two services with two signatures, different protocols or interfaces, in order to guarantee interoperability. The requirements of an adaptation in these approaches stem from two sources: (i) the level of heterogeneity in the upper stack of interoperability (e.g., business level, infrastructure protocols.), and (ii) the diversity of customers, each one of them supports different protocols and interfaces. Mediation can be automatic (Williams et al., 2006) or semi-automatic (Reza et al, 2007).

Taher et al, 2009 (Taher et al, 2009) propose a multilayer software architecture. They propose a framework for transparent and flexible substitution of a service provider by another with respect to an existed consumer. A framework for automatic generation of adapters and service interfaces modelling using automata was adopted to solve the problem of incompatibility in the interaction between two services: a consumer and a new provider. If incompatibilities between these services are detected, an adapter is generated automatically based on the incompatibilities. The generation of the adapter relies on the automata model. The generated adapter contains a sufficient detail of the projected technology called CEP (Complex Event Processing) engine (Luckham et al, 2001).

However, unfortunately, the complex incompatibilities were not considered in this tool. For example, the implementation of several different operations of customer service and a supplier service is not possible by this tool.

The solution proposed in (Hau, 2003) uses OWL (Dean, 2002) to annotate interfaces too. Both solutions (proposed by Syu and Hau) have an abstraction layer called meta-data space. Semantic

annotation is used to describe the methods of services.

Meta-services use these annotations to find appropriate matches between needs and implementations. These solutions differ from the other adaptation approaches. Two distinguished aspects of these approaches are as follows:

- Their locations are dependent on architectures in which they are embedded, and an adoption concerning with the interfaces of web service is often the responsibility of the service provider.
- These approaches are platform dependent such as they are dependent on languages and composition engines

2.3 Cross-layer Adaptation Approaches

The cross-layer adaptation refers to a process of adapting a general system consisting of several layers, where the technology and processes of each layer are integrated and controlled by the same adapter frame. In the context of SOA, this denotes a consistent adaptation through the service interface of different layers and applying a SOA system while maintaining the characteristics such as loose coupling and service autonomy.

The problem of monitoring and adaptation of different types of software systems has gained interests in both the research community and industry. In recent years, these issues have promoted interest in the area of SOA. However, the results and directions are still insufficient. One of the key issues here is that the proposed approaches are very fragmented. They deal only with the problems which are specific to a particular aspect of web service and a particular functional layer, such as business process management layer, service composition and coordination layer, or service infrastructure layer. However, the implementation of various layers of web service can be nested in different artifacts. A layer may contain objects that reside in another layer. However, such cases are ignored by traditional monitoring and adaptation solutions.

Consequently, there is a possibility that these solutions will detect the problems incorrectly which will lead to inaccurate decisions concerning adaptation. This shortcoming of existing solution promotes the need of cross-layer adaptation. In this section, we study the most recent solutions which have been proposed to provide a monitoring and adaptation tools that covers multiple layers. We found that in these solutions, controlling and

adaptation are developed by using various techniques such as, monitoring and event logging, detecting the patterns of events, and correlation and mapping between events and appropriate adaptation strategies, etc.. The solutions proposed in (Gjrven et al., 2008), (Popescu et al., 2010), (Popescu et al., 2012), (Zengin et al., 2011), (Zengin et al., 2011) and (Zeginis et al. 2011) are based on the situation-action mechanism. The situations correspond to a set of events and disparities while the actions are defined as templates for adaptation. These approaches combine the taxonomies of adaptation problems and mechanisms based on the events for guiding the selection process of the adaptation models based on the degree of correspondence between events and disparities of adaptation.

In (Gjrven et al., 2008), a middleware called QuA is presented that provides a multilayer adaptation coordinated by incorporating multiple mechanisms of adaptation in the interface and application layers. However, the proposed middleware is lacking the flexibility because the adaptation logic is predefined and static. A multi-layer adaptation framework is proposed in (Popescu et al., 2010). The authors use taxonomy and adaptation models (patterns) which are created during the design phase to represent the possible solutions to adaptation problems. In this framework, they designed adaptive predefined templates to provide a means for dynamic multi-layers adaptation.

These models define the behaviour of the adaptation processes. However, this approach does not consider the infrastructure layer and the authors do not provide the mechanism for detection disparities. An adaptation manager called CLAM is proposed in (Zengin et al., 2011) to handle adaptive inter-layer and multilayer problems. The authors have classified a group of adaptation paths of an adaptation tree which can be built in any layer of SBAs. The limitation of these approaches is the execution control which is performed in an isolated manner. This does not allow an effective analysis of monitoring data and detected events because events are analysed and processed independently of each other and the critical information are not propagated between layers. This can lead to an incorrect identification of the original source of the problems. Also, some approaches do not realize monitoring in all the layers which affects the final step of adaptations. For example, the actual problem can occur in the infrastructure layer, while it is detected in the composition layer and therefore, it cannot be properly diagnosed.

Additionally, in (Guinea et al., 2011), it is also

argued that monitoring of the web services is not sufficient to allow proper and effective adaptation at runtime. The authors present a framework which uses various techniques for monitoring different layers. Also, it uses a centralized agent of adaptation to collect the events and analyse the violations of KPIs.

Although the cross-layer adaptation approaches designed to identify the sources of problems through analysis and diagnosis that take several layers into account, the works presented in this section have some limitations. Based on our analysis, these approaches can be improved to be more efficient. For instance, since the adaptation approaches do not consider the characteristics and requirements of all the layers of SBAs rather they focus on a specific layer, the activities of adaptation may fail to achieve the desired effects. Furthermore, these approaches may lead to incompatibility problems.

2.4 Adaptation in Agent based Systems

From architectural point of view, there is a similarity between agent and service based systems. This is one of the main reasons we studied the adaptation solutions proposed in this domain. The notion of agent based system is relatively new. We found a few research works on adaptive agent based system.

Qureshi and Perini (Nauman et al., 2008) proposed a methodology called TProcess for seamless self-adaptation in agent based system. The methodology is shaped a triangle that includes three elements include requirement-time, design-time time, and runtime. The authors argue that adaptation should be built on the top of these mutual dependent elements. The critical components of TProcess are goal models which are defined at requirement-time step. The goal models contain QoS parameters, their values and conditions. These are mapped to the implementation platform in the design-time step.

In (Bernon et al., 2003), the author proposed a methodology called ADELFE to guide developers to develop adaptive multi-agent systems. The methodology is based on object-oriented methodologies, follows rational unified process and uses Unified Modeling. In (Ibrahim, 2004), the author proposed a framework for developing intelligent adaptive agents.

In the proposed framework, the agents are defined as systems or machines that utilize inferential or complex computational methodologies to modify or change control parameters, knowledge bases, task plans, problem-solving, methodologies, course of actions, or other objects in order to

successfully accomplish a set of tasks that are of interest to the user. The intelligent adaptive agents are classified into three based on the agent's capabilities on performing external and internal. These categories are listed below:

- Internal adaptation: In this criterion, the internal systems of the agent are adaptive; however, its external actions do not reflect adaptive behaviour.
- External adaptation: It is simply the opposite of internal adaptation. In this the internal systems of agents do not reflect adaptive behaviour.
- Complete adaptation: Internal systems are adaptive and external actions reflect adaptive behavior.

There are a few significant differences between adaptive SBAs and adaptive agents. In SBAs, adaptations are performed in different layers, as these applications rely on multilayer architecture.

However, multilayer adaptation is of the scope of agent based systems. Additionally, none of the adaptive agent based solutions is aware of cross-layer adaptation. However, evidently, the service based systems can be benefited by using the approaches used in adaptive agent based systems. Particularly, the notion of context-awareness and self-adaption can be efficacious for adaptive service based systems.

3 ANALYSIS AND RESEARCH DIRECTIONS

In this section, we summarise our findings and propose a few potential extensions specifically in the area of cross-layer adaptation. We studied various solutions published in the literature. It is worth noting that in this section we limit our discussion in the context of service based systems which is the main focus of this study.

3.1 Analysis

We studied different research initiatives that focus on adaptation problems concerning service interaction in the service composition layer. Specifically, we studied the heterogeneity problems regarding interactions which can be found in the service interface layer. The heterogeneity problem may lead to inconsistency with respect to data exchanged between the services. We found that the main reason for heterogeneity problem is different

formats of the messages exchanged between services. For an effective and adaptation heterogeneity between web services must be dealt with efficiently.

We found mediation-based adaptation approaches deals with heterogeneity. They enable exchanging consistent data between Web services. However, these approaches have limitations. They lack of flexibility and the automation needs to be efficient for a complete and effective adaptation. Moreover, they are limited to technical and structural aspects of a system. They do not cover other aspects. In addition, due to the highly dynamic and evolving nature of the environment and different requirements of service users (infrastructure protocols, and behavior), a manual intervention is required, especially to define the management tasks to handle disparities or to specify or adjust the composition diagram. This is certainly a limitation to carry out adaptation operations efficiently.

In addition, the adaptation mechanisms are not rich enough and deals only with the specific adaptation situations and actions, which does not cover multiple anomalies that may occur in execution environments. The cross-layer adaptation approaches are fragmented and isolated. They do not consider the effects of changes and modifications on all the functional layers of the SBAs. The existing cross-layer, adaptation solutions are designed to adapt a particular functional layer, namely, the business layer, the service composition layer, or infrastructure layer. The realization of different layers of web service can be nested such as different artifacts of a layer can refer to the same objects reside in another layer, while these relationships are ignored by the current monitoring and cross-layer adaptation solutions.

Also, these mechanisms are designed to support quality assurance for adaptation. They deal with the analysis of adaptation activities against the system model, and adaptation measures. Table 1 presents a synthetic summary of the cross-layer adaptation solutions which we studied in this paper. We consider three factors, defined by (Reza et al.2007) adaptation objectives that involves adaptation requirements (repair, optimization, mediation, etc..), adaptation methodology, and the layers covered by the solutions. Also, these mechanisms are designed to support quality assurance for adaptation.

They deal with the analysis of adaptation activities against the system model, adaptation measures, and other adaptations.

Table 1 presents a summary of the approaches found in the literature. We consider three factors

Table 1: Classification of cross-layer adaptation approaches.

Approach	Adaptation Objectives	Methodology of adaptation	Layer affected
(Reza et al, 2007)	Fault tolerance	Proactive	BPM, SCC
(Popescu et al., 2010)	Mediation	Reactive	BPM
(Popescu et al., 2011)	Reparation	Reactive	BPM
(Guinea et al, 2011)	Reparation	Reactive	BPM, SI
(Mos et al. 2009)	Monitoring	Reactive	SI
(Schmieders et al., 2011)	Reparation	Reactive	SCC, SI
(Vidackovic et al., 2009)	Optimistaion	Reactive	BPM
(Gjrven et al, 2008)	Configuration	Proactive	BPM, SCC
(Syu et al, 2004)	Mediation	Reactive	SI

defined by (Reza et al. 2007): (i) adaptation objectives involves adaptation requirements (repair, optimization, mediation, etc.., (ii) Adaptation methodology, and (iii) affected SBA layers which concerns with the change of locations and adaptation progress. From the comparison (shown in the above table) we conclude that none of the current approaches cover all the layers of service based systems. The solutions proposed by Reza et al., Guinea et al., Schmieders et al., Gjerven et al. are relatively more efficient as they cover two layers.

However, cross-layer adaptation solution must cover all three layers of SBAs to deal with various runtime challenges efficient that evolve in current service based system such as cloud service based applications. Remarkably, most of these approaches cover BPM layer, however, to the best of our understanding if an event adapted in the BPM layer, yet it the adaptation has not been propagated to the bottom layers implies that the adaptation has not been realized automatically and may not have done efficiently. This is an important limitation. The current solutions focus on specific layers (e.g., infrastructure layer or Business Process Management layer). One might think of building a hybrid solution which can combine two or more of the existing solutions. However, it will promote a huge complexity. Developing a hybrid solution needs a list of complex tasks include the following:

- Analysis of the affected layer,
- Identification of adaptation actions,
- Aggregation of these actions to check their effects on different layers,
- Launching a coordination system to coordinate adaptation actions,
- Checking whether the adjustment performed at one layer is compatible with the constraints posed by other layers, etc..) which can be costly in terms of response time.

3.2 Research Directions

We identified four critical aspects: context awareness, self-adaptation, completeness, performance, which should be focused in the topic of cross layer adaptation.

Context aware adaptation and self-adaptation have already been studied in agent oriented system. It is worth noting that context awareness and self-adaptation are complementary because self-adaptive system should be aware of the context. Otherwise, self-adaptation can be difficult.

The Table 1 shown in the previous section unearthed a very important shortcoming of cross layer service adaptation technologies. Although these technologies are known as cross-layer adaptation solution, to the best of our understanding, these solutions are complete. These approaches lack the ability to trace incompatibilities that can be triggered through adaptation. Therefore, a solution is needed which can create adaptation loop which runs adaptation process until new requirements or changes are adapted by resolving incompatibilities or conflicts. Adaptation promotes performance challenge. In other words, the system performance can be challenged enormously by adaptation. We found literature reported trade-off between adaptation and performance. An extensive research is necessary to develop a solution that can process adaptation by guaranteeing high efficiency (with respect to processing time).

We plan to develop an intelligent and fault-tolerant solution for cross-layer adaptation that can address the requirements discussed in the above. The proposed solution will enable to perform adaptation process by guaranteeing efficiency and effectiveness. It will be able to perform adaptation in all the layers of SBAs without any incompatibilities or conflicts. The solution will be context-aware and will support self-adaptiveness. This will ensure the autonomic execution of adaptation operations across

the SBA layers.

We strongly believe that the genetic algorithms are potential for our solution especially to optimize the adaptation process. Genetic algorithms are widely used to handle cases such as requirement evolution and performance optimization which are the two most critical issues.

4 CONCLUSIONS

In this paper we studied adaptation technologies particularly the cross-layer adaptation technologies. We discussed the outcomes of our analysis. In particular, we discussed the limitations of different approaches of cross-layer service adaptations.

The major limitation we found is the lack of coordination between adaptation activities that may lead to conflicts or incompatibilities. According to our study, the current solutions do not consider the fact that adaptation in a layer may affect adversely the other layers of service based systems. According to our study, current cross-layer adaptation approaches lack efficient coordination which leads to conflict and incompatibilities. We believe that these problem must be addressed for an efficient cross-layer service adaptation. We presented the results of a brief study on adaptive agent based systems. We found in our study that the agent based adaptive systems have some advanced , features such as context-awareness, self-adaptation, etc.. The adaptive SBAs can be benefited by these features especially, the service based adaptive systems can be more intelligent and autonomous.

Additionally, based on our understanding we presented some research directions in the area of cross layer service adaptations. We strongly believe that the research in this area should focus on context awareness, self-adaptation, and performance etc. to develop highly high-performance solutions. We also presented a proposal of a solution which are currently working on.

There are a few limitations of our study. Firstly, this is merely a literature review. However, the state of the art could be better reviewed or understood by benchmarking the existing solutions. A comparison of adaption technologies in different contexts can be done by following a set of rigorous protocols. This paper is missing such an comparison. In our future work, we plan to conduct an empirical study with the current cross-layer adaptation technologies. Also, we plan to conduct a study by covering more contexts.

REFERENCES

- Ardagna, D. , Pernici, B.(2007). Adaptive Service Composition in Flexible Processes, *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369-384, June 2007.
- Baresi , L. , Guinea, S. , (2005), Dynamo: Dynamic Monitoring of WS-BPEL Processes. In *5th International Conference on Service Oriented Computing*, pages 478–483, 2005.
- Baresi, L., Di Nitto, E. , Ghezzi, C., and Guinea, S. , (2007) . A Framework for the Deployment of Adaptable Web Service Compositions, *Service Oriented Computing and Applications*, vol. 1, no. 1, pp.75-91, 2007.
- Bernon, C., Gleizes, M. P., Peyruqueou, S., Picard, G. (2003). ADELFE: a methodology for adaptive multi-agent systems engineering. In *Engineering Societies in the Agents World III* (pp. 156-169). Springer Berlin Heidelberg.
- Bucchiarone, A., Cappiello, C., Di Nitto, E., Kazhamiakin, R., Mazza, V., and Pistore, M., (2009). Design for Adaptation of Service-Based Applications: Main Issues and Requirements ,*ICSOC/ServiceWave* in page 467–476.
- Chafle, G., Dasgupta, K. , Kumar, A. , Mittal, S.,and Srivastava, B. , (2006). Adaptation in Web Service Composition and Execution, *Proc. IEEE Int'l Conf. Web Services (ICWS '06)*, pp. 549-557, 2006.
- Dean, M., Connolly, D., Harmelen, F., Hendler, J., Horrocks, I., Debo-rah L., McGuinness, Peter F. Patel-Schneider, Andrea Stein, L., (2002). Web ontology language (OWL) reference version 1.0. Technical report, www.w3c.org, 2002.
- Geihs, K., Reichle, R., Wagner, M., Khan ,M., (2009). Service-Oriented Adaptation in Ubiquitous Computing Environments , *International Conference on Computational Science and Engineering*.
- Gjrven, E., Rouvoy, R., Eliassen, F. , (2008). Cross-layer self-adaptation of service-oriented architectures, *Proceedings of the 3rd workshop on Middleware for service oriented computing* 2008.
- Guinea, S. , Kecskemeti, G. , Marconi, A. , Wetzstein, B. , (2011). Multi-layered monitoring and adaptation,
- Hau, J., Lee, W. , Newhouse, S. , (2003) . The ICENI Semantic Service Adaptation Framework. In: *UK e-Science All Hands Meeting* (2003).
- Ibrahim F. Imam: Adaptive applications of intelligent agents. *ISCC 2004*: 7-12.
- IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems, (2003). Business Process Execution Language for Web Services version 1.1. <http://www.ibm.com/developerworks/library/specification/wsbpel/>. 2004 Imam, I. F. (2004, June). Adaptive applications of intelligent agents. In *Computers and Communications*, 2004. *Proceedings. ISCC 2004. Ninth International Symposium on (Vol. 1, pp. 7-12). IEEE*.
- Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., and Leaf, D., (2011). NIST Cloud Computing

- Reference Architecture: Recommendations of the National Institute of Standards and Technology. NIST Special Publication 500-292. pp. 10.
- Luckham, D., (2001), *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman (2001).
- Nauman, A., Qureshi, Anna Perini: *An Agent-Based Middleware for Adaptive Systems*. QSIC 2008: 423-428.
- Papazoglou, M. P., (2008). *Web Services - Principles and Technology*. Prentice Hall. ISBN 978-0-321-15555-9. pp. 1-752.
- Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S., (2010). *Taxonomy-Driven Adaptation of Multilayer Applications Using Templates*, saso, pp.213-222, 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2010.
- Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S. (2011). *A Formalised, TaxonomyDriven Approach to Cross-Layer Application Adaptation* ACM Transactions on Autonomous and Adaptive Systems, ICSOC.
- Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S., 2012. *A Formalised, Taxonomy-Driven Approach to Cross-Layer Application Adaptation* in ACM Transactions on Autonomous and Adaptive Systems, Proceedings of the 9th international conference on Service-Oriented Computing, December 05-08, 2011, Paphos, Cyprus.
- Reza, H., Nezhad, M., Benatallah, B., Martens, A., Curbera, F., Casati, F., (2007). *Semi-automated adaptation of service interactions*, Proceedings of the 16th international conference on World Wide Web, May 08-12, 2007, Banff, Alberta, Canada.
- Schmieders, E., Micsik, A., Oriol, M., Mahbub, K., and Kazhamiakin R., "Combining SLA prediction and cross layer adaptation for preventing SLA violations". In Proceedings of the 2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services, Timisoara, Romania, June 2011.
- Syu, J.-Y., (2004). *An Ontology-Based Approach to Automatic Adaptation of Web Services*, Department of Information Management National Taiwan University, 2004. (<http://www.im.ntu.edu.tw/IM/Theses/r92/R91725051.pdf>).
- Taher, C., Aït-Bachir, A., Fauvet, M., Benslimane, M., *Diagnosing Incompatibilities in Web Service Interactions for Automatic Generation of Adapters*. AINA 2009: 652-659.
- Vidackovic, K., Weiner, N., Kett, H., Renner, T. "Towards business-oriented monitoring and adaptation of distributed service-based applications from a process owner's viewpoint". In: ICSOC/ServiceWave Workshops. pp. 385394, 2009.
- Williams, S.K., Battle, S. A., Cuadrado, J. E., (2006). *Protocol mediation for adaptation in semantic web services*, Proceedings of the 3rd European conference on The Semantic Web: research and applications, June 11-14, 2006, Budva, Montenegro.
- Zeginis, C., Konsolaki, K., Kritikos, K., and Plexousakis, D., (2011). *Ecmaf: An event-based cross-layer service monitoring and adaptation framework*, In 5th Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC'11) co-located with ICSOC 2011. Springer, 2011.
- Zeginis, C., Plexousakis, D., (2010). *Web Service Adaptation: State of the art and Research Challenges*, Technical Report 410, ICS-FORTH, October 2010.
- Zengin, A., Marconi, A., Baresi, L., Pistore, M., (2011). *CLAM: Managing cross-layer adaptation in service based systems*, soca, pp.1-8, 2011 IEEE International Conference on Service-Oriented Computing.
- Zengin, A., Kazhamiakin, R., Pistore, M.: "CLAM: Cross-Layer Management of Adaptation Decisions for Service-Based Applications," icws, pp.698-699, 2011 IEEE International Conference on Web Services, 2011.

The Influence of the Provider's Service Fairness on the Customer's Service Recovery Satisfaction and on Positive Behavioral Intentions in Cloud Computing

Montri Lawkobkit¹ and Roland Blomer²

¹*Faculty of Business Administration, Sripatum University, Bangkok, Thailand*

²*Institute of Biomedical Informatics, UMIT, University for Health Sciences,
Medical Informatics and Technology, Hall in Tirol, Austria
mlawkobkit@gmail.com, roland@blomer.de*

Keywords: Service Fairness, Service Recovery Satisfaction, Behavioral Intentions, Continual Improvement, Structural Equation Modelling.

Abstract: The study shows a statistically significant positive effect between the provider's perceived structural service fairness and the customer's service recovery satisfaction and, in turn, also shows statistically positive regression weights between the customer's service recovery satisfaction and the intention to react positively in three directions: (1) to continue with the software, (2) to propagate a positive word-of-mouth (WOM), (3) to give honest feedback. The influence of the provider's perceived social service fairness on the customer's service recovery satisfaction does not appear to be significant but indicates a positive correlation. The study is based on data collected via a structured questionnaire from qualified users who have subscribed to Business-to-Business customer relationship management software and who use it as Software-as-a-Service in the cloud. Structural Equation Modelling was applied for the data analysis in order to confirm the chosen dependency model. The findings may help service providers to better understand their customers and to stimulate constructive actions to their continual improvement process.

1 INTRODUCTION

Cloud computing has developed to be one of the fastest growing markets with an expected value of approximately US\$68 billion by 2018 wherein customer relationship management (CRM) applications used in software-as-a-service mode (SaaS) will capture a market share of about 25% with a compound annual growth rate (CAGR) of about 12% (Buyya et al. 2009; Dhar 2012; Pang 2014). More than 500 major vendors and service providers compete in this arena in which the customer ultimately decides on the provider's business success or failure. Customer satisfaction (CS) plays the role as the main key performance indicator in service management and represents an important control in the continual improvement process of every service provider.

A particular challenge in service management is presented when expected and agreed service levels are – for whatever reason – not met i.e. in the case of service failure. Disappointed customers will not only

complain and possibly switch provider, but will also disseminate their bad experiences. Negative word-of-mouth (WOM) may reach up to 20 other (potential-)customers and may thus harm the provider's business significantly (Zemke 1999).

A service provider should be well-equipped for service failure recovery so that he can retain customers and maybe regain CS. This should also occur in cases of painful service failure (Johnston 1995). Some authors claim that after effective service recovery, customers might feel higher levels of satisfaction when compared with previous levels (known as “service recovery paradox”) (McCollough & Bhardwaj 1992). Effective service recovery, however, must be part of any service provision concept in order to survive and grow in a highly competitive market.

The main objective of this paper is to design and to test a model which shows the dependencies between the perceived internal structures and processes of a service provider and the service recovery satisfaction (SRS) of the customer and

how, in turn, CS stimulates customer behavioral outcomes in favor of the current and future business of the service provider.

Service recovery, moreover, has been an interesting area for practitioners and marketing scholars for years (Kau & Loh 2006; Zhou et al. 2013).

This study examines the focal determinants of fairness based on Greenberg's (1993) taxonomy of organizational fairness and their influence on SRS. The two distinct fairness dimensions are structural and social fairness. Figure 1 presents the conceptual model and hypothesized relationships in this study.

The service fairness (structural and social) of the provider would then positively impact the SRS of the customer which, in turn, favorably influences the customer behavior intentions in three directions: (1) to continue with the software, (2) to propagate positive word-of-mouth (WOM), (3) to give honest feedback to the provider and external agencies, such as consumer protection organizations. In a previous paper, a similar chain of effects was evident in cases where the service was performed correctly (Lawkobkit & Larpsiri 2014).

2 LITERATURE REVIEW AND HYPOTHESES

2.1 Service Recovery Satisfaction

Levesque and McDougall defined satisfaction as the "overall customer attitude towards a service provider" (Levesque & McDougall 1996, p.14). It means the customer's overall judgment on the service provider (McDougall & Levesque 2000) that a product or service itself, or the product or service feature, is providing a level of under or over fulfilment (Tronvoll 2011). A service failure occurs whenever the service provider fails to deliver his services as expected by the consumer (Kelly & Davis 1994). A service failure is basically a flawed outcome that might indicate a breakdown in reliability (Berry & Parasuraman 1991).

In the computing area, customer SRS can be defined as the end-user's perception when interacting with a specific application, including perception, toward service failures and CS or dissatisfaction with the organization's approach to service recovery (Kwok et al. 2009).

Service failures and recoveries and their determinants have been studied in different contexts such as public and private service delivery (Zhou et

al. 2013) and can enhance service quality and avoid negligence (Kuo et al. 2011).

Previous research studied many factors influencing SRS such as recovery and order (time) (Boshoff 1997), redress and responsiveness (Hocutt et al. 2006), distribution, procedural and interactional justice (Choi & Choi 2014). Past research has used the term 'justice' and 'fairness' interchangeably. Here, the term 'fairness' is used for the purpose of consistency.

Previous research shows that service recovery justice for customers affects their level of satisfaction (Kuenzel & Katsaris 2009). SRS can bring several benefits such as positive WOM and repurchase intention (Tax & Brown 1998).

The literature suggests that fairness could play a significant role in service failure and recovery (Lawkobkit & Larpsiri 2014; Yang & Peng 2009). In service management, perceptions of fairness are important antecedents of recovery satisfaction and lead to recovery satisfaction (Lawkobkit & Kohsuwan 2012).

The level of SRS results from many factors although these are all grounded in the customer's experience of the application, of the services taken and the interaction with their service providers. Therefore, improving the level of CS would be a very important goal to the service provider.

2.2 The Focal Determinants of Service Fairness and Service Recovery Satisfaction

Organizational fairness is one of the important factors that has been widely studied also in the field of organizational behavior (Colquitt et al. 2001). Organizational fairness has also received attention in the context of employee perceptions of fairness in the workplace with regard to matters such as job satisfaction, complaint handling, and human resource management (Folger & Greenberg 1985).

Organizational fairness may be defined as the perception of fairness by an individual in the working environment (Byrne & Cropanzano 2001; Greenberg 1990). Greenberg's (1993) rudimentary taxonomy highlights the distinction between the structural and social determinants of fairness. A taxonomy is formed with two independent dimensions: fairness (procedural and distributive), and focal determinants (structural and social).

One of the major research areas in organizational psychology has been focused on the concept of focal determinants (Cropanzano 1993). Some prior research has discussed focal determinants in the area

of strategic decision making in leadership and ethics (Tatum & Eberlin 2007).

In addition, prior studies have revealed a relationship between social fairness and both managerial performance (Tatum et al. 2002) as well as employee behaviors (Masterson et al. 2000). Social fairness has become one of the important components of outcome fairness. In a transformational leadership study, social fairness had more impact than structural fairness because the leader cares about the needs and well-being of the followers and wants to be open and responsive (Eberlin & Tatum 2005).

Greenberg's (1993) taxonomy positions the focal determinants of fairness as the immediate focus of a just action relative to existing categories of fairness. The two specific determinants of service fairness can be briefly characterised by the following:

1) *Structural Fairness*: This type of fairness refers to the structural elements of the organization and focuses on the environmental context within which interaction occurs (Greenberg 1993).

In cloud service, structural fairness refers to the structural elements of the service provider that allow the involvement of their customers in decision-making and provide a fair distribution of outcomes. The customer is convinced that he and the supplier follow the same agenda. When customers perceive high structural fairness, they will believe that an unfair outcome is merely an accident and will expect that structural fairness will still hold.

Satisfied customers will be less likely to terminate their relationship with their service providers. Moreover, the level of satisfaction will increase if their service providers use technological support to track and monitor their services with on-line and off-line customers. Several results from previous studies support the concept of perceived structural fairness that has impacted directly on outcomes (Tatum & Eberlin 2007). This consideration leads to the following hypothesis:

H₁: Perceptions of structural service fairness are positively associated with SRS.

2) *Social Fairness*: This type of fairness is recognized also as one of the significant sources of fairness perception in Greenberg's study (1993), who proposed a distinguishable fairness in the taxonomy. Social fairness focuses on information exchange on an individual level by "showing concern for individuals regarding the distributive outcomes they receive" (Greenberg 1993, p.85), and "may be sought by providing knowledge about procedures that demonstrate a regard for people's concerns" (Greenberg 1993, p.84).

In cloud service, social service fairness indicates to customers that the service provider cares about their well-being and keeps customers informed before and during changes to the service process.

Information about services is given to customers who have been involved. The CS resp. SRS level will increase when they feel the service provider has treated them with respect, politeness, sincerity and fairness throughout the service process. Once the service providers are truthful in all communication and tailor their explanations to match customer needs, the level of information fairness will always be high. The customers perceive a fair information exchange before, during and after the service process from the perspective of social fairness, and a positive customer outcome can occur. From this, the following hypothesis is developed:

H₂: Perceptions of social service fairness are positively associated with SRS.

These two service fairness factors should have an impact on SRS, and H₁ & H₂ address the question of whether an individual's perception of structural and social fairness is strong enough to influence satisfaction, thus indirectly contributing to continued usage and behavioral intention.

2.3 Service Recovery Satisfaction and IS Continuance Intention

SRS is one of the key factors for IS service scholars (Kassim et al. 2012; Sun et al. 2014; Wu 2013). Several IS researchers have also found that satisfaction is a strong predictor of system usage, IS success, service recovery and continuance behavior (Kim et al. 2012).

Satisfaction is an influential factor in the re-consumption intention of customers. In accord with the study of Bhattacharjee (2001), the post-acceptance model of IS continuance (PAM) views relationship satisfaction as a basis for the continued intention to use IS; satisfaction with prior use has a strong positive impact on customer intentions to continue using the system. The more an individual customer is satisfied with prior usage experience, the greater the chance that the customer will continue to use the system.

Continuance behavior may be defined as explaining user intentions to continue or discontinue using an IS, where a continuance decision follows an initial acceptance decision. Therefore, satisfaction is a main determinant influencing continuance intention as revealed in various research (Zhou 2013) in previous continuance study contexts such as shopping (Chen & Chou 2012), e-learning (Cheng

2014).

This research employs the concept of IS continuance intention and applies the measurement approach from Bhattacharjee (2001). This dimension has three scale items to measure the continued usage of the SaaS application rather than discontinuing its use or using an alternative. Thus, the relationship between satisfaction and continuance intention can be hypothesized as:

H₃: Service recovery satisfaction with IS usage is positively associated with IS continuance intention.

2.4 Service Recovery Satisfaction and Behavioral Intentions

Fishbein and Martin (1975) and Ajzen and Fishbein (1980) developed the Theory of Reasoned Action, which is a model to predict behavioral intention. Behavioral intention measures a person's relative strength of intention to perform a behavior. In this regard, two customer behaviors are WOM and feedback to the service provider, both of which are related to customer retention and the customer's long-term relationship with their providers.

WOM refers to "informal communication between private parties concerning evaluations of goods and services" (Anderson 1998, p.6), which is about valence (positive, negative or neutral). A key motivation for this behavior is a customer's experience with the service. This service experience produces "a tension which is not eased by the use of the product alone, but must be channeled by ways of talk, recommendation, and enthusiasm to restore the balance" (Dichter 1966, p.148). Additionally, WOM reflects a sense of loyalty (Zhang et al. 2010).

WOM behavior is defined in this study to refer to the customer's intention to share favorable information about the service provider and its service among peers. We believe that any positive WOM activity contributes to the viability of a technology with support services (CRM-SaaS) because it influences service fairness and can be exploited by the service provider.

Several previous studies discussed the relationship between recovery satisfaction and WOM (Seawright et al. 2008). Many scholars have revealed the positive relationship between recovery satisfaction and WOM (Wen & Geng-qing Chi 2013); therefore, this study proposes the following hypothesis:

H₄: Service recovery satisfaction related to positive word-of-mouth is positive and strong.

Customer feedback with regard to the second behavior indicates that positive feedback is always

driven by satisfaction (Saha & Theingi 2009). A very interesting finding from Söderlund (1998) was that negative feedback is more likely to be provided by dissatisfied customers because of the compensation involved. However, customers always provide positive feedback without expecting a reward. In the digitized era, customers can provide their feedback in various forms of online feedback mechanism based on the specific category (Liu & Zhang 2010).

In this study of cloud service, feedback refers to the communication from customers as service receivers to their service providers and external agencies (e.g., consumer protection organizations). Customers might use satisfaction as a proxy for the level of service fairness that they should receive. Previous research revealed a positive relationship between feedback and satisfaction (Saha & Theingi 2009; Söderlund 1998). On the basis of the above discussion, the following hypothesis is therefore proposed:

H₅: Service recovery satisfaction related to positive feedback is positive and strong.

This study applies a conceptual model in which the perceptions of the focal determinants of service fairness and satisfaction result from the use of a technology with support services. This then leads to continuance intention and customer behavioral intention including WOM and feedback to their service provider.

3 METHODS, SAMPLE AND DATA COLLECTION

A quantitative study was conducted to assess the relationships between two dimensions of service fairness and SRS and their further propagation on IS continuance intention, WOM and feedback to the service provider.

Previously developed methods have been chosen as guides in this study for their merit and overall utility. However, they have been modified in order to reflect the specific cloud service context, as well as the targeted users. The service fairness items were adapted from a number of works but generally follow (Bies & Moag 1986; Leventhal 1980; Maxham & Netemeyer 2003; Shapiro et al. 1994). Other items were adopted from Maxham & Netemeyer (2002) for SRS, Bhattacharjee (2001) for IS continuance intention, and finally Zeithaml, Berry & Parasuraman (1996) for WOM and feedback.

All items were reworded to relate specifically to

CRM-SaaS. A 7-point Likert-scale was employed for each survey item, ranging from 1 = “strongly disagree” to 7 = “strongly agree”.

In order to acquire and develop the most appropriate pilot version for the questionnaire, an expert panel reviewed the initial draft. These are professionals from both sides of service management: the academics and the industry. The pilot test ($n = 60$) showed good results for all variables on the service fairness concepts, satisfaction, IS continued usage, WOM, and feedback. After the various changes were incorporated and considered, the final version of the survey was then carried out.

SaaS providers in cloud service providing a service together with an application is the context of this study. Individuals from small and medium-sized enterprises (SMEs) were tapped. Those who use business-to-business (B2B) CRM-SaaS formed the population of the study. The pilot and main study focused on respondents who were B2B SRM SaaS-users.

Company databases of full-time employees working in organizations provided the source for prospective panel members. In all, 30,899 recruitment emails were sent. The first response rate was 11.62% (3,589). Four stringent screening questions constraints reduced them to 475 questionnaires, which gives a response rate of 1.54%.

There were 475 sample respondents, and among them, sixty percent were male while the other forty were female. The majority of the respondents were within the age range from thirty to fifty years old, and nearly ninety percent (88.84%) had over five years working experience. As shown in the data, the most common positions were operating staff (17.24%), supervisors (17.05%) and sales representatives (14.54%). Half of the respondents (52.20%) were from organizations employing between fifty and five hundred employees. The business service industry covered the highest percentage of respondents (58.52%).

The sample thus exhibited the following significant characteristics: they are from an experienced working-age group, have responsibility at their present company requiring frequent use of CRM-SaaS software, and interact with the software service provider.

4 RESULTS

The analysis results of the descriptive statistics for

internal reliability of the measures ranged from .961 (structural fairness) to .993 (Social fairness) for the two service fairness dimensions. The other four measures are .909 for satisfaction, .896 for continuance intention, .914 for WOM and .751 for feedback. All the measures included in the questionnaire showed adequate levels of initial internal reliability ($> .70$) (Hair et al. 2009).

Figure 1 and Table 1 present the standardized estimates and standardized regression weights, with all five hypotheses supported. The structural model was accepted and the chi-square was significant (chi-square = 1532.601; $df = 399$, $p = .000$, relative chi-square = 3.841; NFI = .888; GFI = .808; CFI = .907; TLI = .907; RMSEA = .077). The path coefficients for the structural model are shown in Table 1. The relative effect (standardized regression weights) between independent and dependent variables shows a statistical significance for all hypothesized relationships.

A summary of standardized path coefficients and the square multiple correlations (R^2), of the best-fit measurement model are shown in Table 1. The significance of four of five path coefficients to the model is amplified, even though they are positive and statistically significant at $p > 0.05$. Moreover, most of the R^2 values of the observed variables were greater than 0.50, indicating the reasonably good convergent validity of the model.

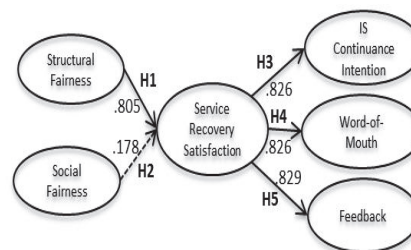


Figure 1: Result of Structural Equation Modelling (SEM).

Table 1: Results of standardized coefficients.

Outcome (R^2)	Determinant (Hypothesis)	Coefficients (P -value)
SRS (.950)	Structural fairness (H ₁)	0.805 (***)
	Social fairness (H ₂)	0.178 (.049)
Contin. (.682)	SRS (H ₃)	0.826 (***)
WOM (.682)	SRS (H ₄)	0.826 (***)
Feedback (.688)	SRS (H ₅)	0.829 (***)

Coefficients - Standardized regression weights (***) P -Value $< .001$

The analysis of path coefficients indicates that four hypotheses are supported. The influence of

structural fairness (coefficient = 0.805) on SRS was significant. Unfortunately, social fairness (coefficient = 0.178) on SRS was only nearly significant ($p = 0.49$). Moreover, the influence of SRS on IS continuance intention was significant (coefficient = 0.826). Similarly the influences of SRS on WOM (coefficient = 0.826) and on feedback (coefficient = 0.829) were significant (see Table 1). The impact of the endogenous variables is indicated by the R^2 values. The highest R^2 appeared in satisfaction (95%) and the next R^2 was shown in feedback (68.8%), and continuance intention and WOM that had the same values (68.2%). (See Table 1) The results of the research model (H1 – H5) show that all five hypotheses are supported, so the model does work well in this context.

5 CONCLUSIONS

One of the key success factors for service management is related to successful service recovery when there has been service failure. The service providers' actions during service failure can influence their customer perceptions and the providers can have lessons to learn in order to be able to manage more effectively in success and failure areas in the future (La & Kandampully 2004).

The analytical results of this study showed that SRS is significantly influenced by the provider's structural service fairness. In other words, CS can be regained by fair and equal treatment of customers. This SRS in turn furthers the customer's intention to continue the service under consideration, to disseminate favourable information about this service (WOM), and to enter into a feedback process with the provider. Other factors that could influence the co-operation between customer and provider after a service failure is trust in the service provider and the commitment of the provider to resolve the failure.

The findings are consistent with previous research which placed greater importance on the information and contact for service recovery in a Korean context (e.g., Park & Kim 2011) and a positive relationship between satisfaction and feedback (e.g., Saha & Theingi 2009).

This study contributes to both academia and practice. In academia, the study builds on previous research on the relationships of service recovery attributes and CS enhancing continuance as well as behavioral intentions. For practitioners, especially for managers, the study provides an insight into the usefulness of service recovery measures to enhance

effectively CS, continued usage, WOM and feedback to the respective service providers.

In summary, this paper suggests that cloud service fairness promises to be a fruitful arena for additional research into the area of customer satisfaction, continued usage and behavioral intentions. Practitioners in the service support area would find additional practices to improve the level of CS during service recovery after a failure. Service support management should consider and must account for these areas.

In regard to the research background, CRM-SaaS was studied. It is suggested to expand the study to other cloud service applications in order to generalize the study by understanding the characteristics of cloud computing and possible deviations from the results of this study. Greater diversity in service recovery would be suggested for further research.

REFERENCES

- Ajzen, I. & Fishbein, M., 1980. *Understanding attitudes and predicting social behaviour*, Englewood Cliffs, NJ: Prentice-Hall.
- Anderson, E. W., 1998. Customer satisfaction and word of mouth. *Journal of Service Research*, 1(1), pp.5–17.
- Berry, L. & Parasuraman, A., 1991. *Marketing Services*, New York: The Free Press.
- Bhattacharjee, A., 2001. Understanding information systems continuance: an expectation-confirmation model. *MIS Quarterly*, 25(3), pp.351–370.
- Bies, R. J. & Moag, J. S., 1986. Interactional justice: communication criteria for fairness. In R. J. Lewicki, B. H. Sheppard, & M. H. Bazerman, eds. *Research on Negotiation in Organizations*. Greenwich, CT: JAI, pp. 43–55.
- Boshoff, C., 1997. An experimental study of service recovery options. *International Journal of Service Industry Management*, 8(2), pp.110–130.
- Buyya, R. et al., 2009. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6), pp.599–616.
- Byrne, Z. S. & Cropanzano, R., 2001. The history of organizational justice: the founders speak. In R. Cropanzano, ed. *Justice in the workplace: From theory to practice*. Mahwah, NJ: Lawrence: Erlbaum Associates, Inc., pp. 3–26.
- Cheng, Y.-M., 2014. Extending the expectation-confirmation model with quality and flow to explore nurses' continued blended e-learning intention. *Information Technology & People*, 27(3), pp.230–258.
- Chen, Y.-T. & Chou, T.-Y., 2012. Exploring the continuance intentions of consumers for B2C online

- shopping: perspectives of fairness and trust. *Online Information Review*, 36(1), pp.104–125.
- Choi, B. & Choi, B.-J., 2014. The effects of perceived service recovery justice on customer affection, loyalty, and word-of-mouth. *European Journal of Marketing*, 48(1/2), pp.108–131.
- Colquitt, J. A. et al., 2001. Justice at the millennium: a meta-analytic review of 25 years of organizational justice research. *Journal of Applied Psychology*, 86(3), pp.425–445.
- Cropanzano, R., 1993. *Justice in the workplace: approaching fairness in human resource management*, Mahwah, NJ: Lawrence Erlbaum Associates.
- Dhar, S., 2012. From outsourcing to cloud computing: evolution of IT services. *Management Research Review*, 35(8), pp.664–675.
- Dichter, E., 1966. How word-of-mouth advertising works. *Harvard Business Review*, 44(6), pp.147–166.
- Eberlin, R. & Tatum, B. C., 2005. Organizational justice and decision making: when good intentions are not enough. *Management Decision*, 43(7/8), p.1040.
- Fishbein, M. & Ajzen, I., 1975. *Belief, attitude, intention, and behavior: an introduction to theory and research*, MA: Addison-Wesley.
- Folger, R. & Greenberg, J., 1985. Procedural justice: an interpretive analysis of personnel systems. *Research in Personnel and Human Resources Management*, 3, pp.141–183.
- Greenberg, J., 1990. Organizational justice: yesterday, today, and tomorrow. *Journal of Management*, 16(2), pp.399–432.
- Greenberg, J., 1993. The social side of fairness: interpersonal and informational classes of organizational justice. In R. Cropanzano, ed. *Justice in the Workplace: Approaching Fairness in Human Resource Management*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 79–103.
- Hair, J. F. et al., 2009. *Multivariate data analysis* Seventh., Englewood Cliffs: Prentice Hall.
- Hocutt, M., Browers, M. & Donavan, D., 2006. The art of service recovery: fact or fiction. *Journal of Service Marketing*, 20, pp.199–207.
- Johnston, R., 1995. Service failure and recovery: impact, attributes and processes. *Advance in Services Marketing and Management: Research and Practice*, 4, pp.211–28.
- Kassim, E. S. et al., 2012. Information System Acceptance and User Satisfaction: The Mediating Role of Trust. *International Conference on Asia Pacific Business Innovation and Technology Management*, 57(0), pp.412–418.
- Kau, A.-K. & Loh, E.W.-Y., 2006. The effects of service recovery on consumer satisfaction: a comparison between complainants and non-complainants. *Journal of Services Marketing*, 20(2), pp.101–111.
- Kelly, S. W. & Davis, M. A., 1994. Antecedents to customer expectations for service recovery. *Journal of the Academy of Marketing Science*, 22(11), pp.52–61.
- Kim, T., Yoo, J.-E. & Lee, G., 2012. Post-recovery customer relationships and customer partnerships in a restaurant setting. *International Journal of Contemporary Hospitality Management*, 24(3), pp.381–401.
- Kuenzel, S. & Katsaris, N., 2009. A critical analysis of service recovery processes in the hotel industry. *TMC Academic Journal*, 4(1), pp.14–24.
- Kuo, Y.-F., Yen, S.-T. & Chen, L.-H., 2011. Online auction service failures in Taiwan: typologies and recovery strategies. *Electronic Commerce Research and Applications*, 10(2), pp.183–193.
- Kwok, D., Land, L. & Stephens, G., 2009. Multi-dimensionality of overall consumer satisfaction – socio-technical perspective. In Proceedings of the Fifteenth Americas Conference on Information Systems. p. 310.
- La, K. V. & Kandampully, J., 2004. Market oriented learning and customer value enhancement through service recovery management. *Managing Service Quality*, 14(5), pp.390–401.
- Lawkobkit, M. & Kohsuwan, P., 2012. Focal determinants of service fairness and service recovery satisfaction in cloud computing. In Chiang Mai: 17th Asia Pacific Decision Sciences Institute (APDSI) International Conference, p. 81.
- Lawkobkit, M. & Larpsiri, R., 2014. The focal determinants of service fairness, satisfaction and behavioral intentions in service management. *APHEIT Humanities-Social Sciences Journal*, 21(1), pp.22–36.
- Leventhal, G. S., 1980. What should be done with equity theory? In K. Gergen, M. Greenberg, & R. Willis, eds. *Social exchange: advances in theory and research*. New York: Plenum Press, pp. 27–55.
- Levesque, T. & McDougall, G. H. G., 1996. Determinants of customer satisfaction in retail banking. *International Journal of Bank Marketing*, 14(7), pp.12–20.
- Liu, R. R. & Zhang, W., 2010. Informational influence of online customer feedback: an empirical study. *Journal of Database Marketing & Customer Strategy Management*, 17(2), pp.120–131.
- Masterson, S. S. et al., 2000. Integrating justice and social exchange: The differing effects of fair procedures and treatment on work relationships. *Academy of Management Journal*, 43(4), pp.738–748.
- Maxham, J. G. & Netemeyer, R., 2003. Firms reap what they sow: the effects of shared values and perceived organizational justice on customers' evaluations of complaint handling. *Journal of Marketing*, 67(1), pp.46–62.
- Maxham, J. G. & Netemeyer, R., 2002. Modeling customer perceptions of complaint handling over time: the effects of perceived justice on satisfaction and intent. *Journal of Retailing*, 78, pp.239–252.
- McCullough, M. A. & Bhardwaj, S. G., 1992. The recovery paradox: an examination of consumer satisfaction in relation to disconfirmation, service quality and attribution based theory. In *Marketing Theory and Applications*. American Marketing Association, p. 119.

- McDougall, G. H. & Levesque, T., 2000. Customer satisfaction with services: putting perceived valued into the equation. *Journal of Services Marketing*, 14(5), pp.392–410.
- Pang, A., 2014. Worldwide cloud applications market forecast 2014-2018. *Apps Run The Cloud*. Available at: <https://www.appsruntimecloud.com> [Accessed December 28, 2014].
- Park, M. & Kim, M., 2011. A cross-cultural analysis of online satisfaction, service failure and recovery: an E-A-S-Qual approach. *Journal of the Korean Society of Clothing and Textiles*, 35(6), pp.700–711.
- Saha, G. C. & Theingi, 2009. Service quality, satisfaction, and behavioral intentions: a study of low-cost airline carriers in Thailand. *Managing Service Quality*, 19(3), pp.350–372.
- Seawright, K. K. et al., 2008. An empirical examination of service recovery design. *Marketing Intelligence & Planning*, 26(3), pp.253–274.
- Shapiro, D. L., Buttner, E. H. & Barry, B., 1994. Explanations: what factors enhance their perceived adequacy? *Organizational Behavior and Human Decision Processes*, 58(3), pp.346–368.
- Söderlund, M., 1998. Customer satisfaction and its consequences on customer behaviour revisited: the impact of different levels of satisfaction on word-of-mouth, feedback to the supplier and loyalty. *International Journal of Service Industry Management*, 9(2), pp.169–188.
- Sun, H., Fang, Y. & Hsieh, J. P.-A., 2014. Consuming information systems: an economic model of user satisfaction. *Decision Support Systems*, 57(0), pp.188–199.
- Tatum, B. C. et al., 2002. Organizational justice and performance as measured by 360-degree feedback. In 18th Annual Convention of the Association for Psychological Science.
- Tatum, B. C. & Eberlin, R. J., 2007. Leadership, ethics, and justice in strategic decision making. *Business Strategy Series*, 8(4), p.303.
- Tax, S. S. & Brown, S. W., 1998. Recovering and learning from service failure. *Sloan Management Review*, 40(1), pp.75–88.
- Tronvoll, B., 2011. Negative emotions and their effect on customer complaint behaviour. *Journal of Service Management*, 22(1), pp.111–134.
- Wen, B. & Gengqing Chi, C., 2013. Examine the cognitive and affective antecedents to service recovery satisfaction: a field study of delayed airline passengers. *International Journal of Contemporary Hospitality Management*, 25(3), pp.306–327.
- Wu, I.-L., 2013. The antecedents of customer satisfaction and its link to complaint intentions in online shopping: an integration of justice, technology, and trust. *International Journal of Information Management*, 33(1), pp.166–176.
- Yang, H.-E. & Peng, K.-H., 2009. Assessing the effects of service recovery and perceived justice on customer satisfaction with SEM. In Management and Service Science, 2009. MASS '09. International Conference on 20-22 Sept. 2009. pp. 1 – 4.
- Zeithaml, V. A., Berry, L. L. & Parasuraman, A., 1996. The behavioral consequences of service quality. *Journal of Marketing*, 60(2), pp.31–46.
- Zemke, R., 1999. Service recovery: turning oops into opportunity. In *Best Practices in Customer Service*. New York, NY: AMA Publications, pp. 279–88.
- Zhang, Z. et al., 2010. The impact of e-word-of-mouth on the online popularity of restaurants: a comparison of consumer reviews and editor reviews. *International Journal of Hospitality Management*, 29(4), pp.694–700.
- Zhou, T., 2013. Understanding continuance usage of mobile sites. *Industrial Management & Data Systems*, 113(9), pp.1286–1299.
- Zhou, Y. et al., 2013. Recovery strategy for group service failures: The interaction effects between recovery modes and recovery dimensions. *European Journal of Marketing*, 47(8), pp.1133–1156.

Context-aware Security@run.time Deployment

Wendpanga Francis Ouedraogo¹, Frederique Biennier¹, Catarina Ferreira Da Silva²
and Parisa Ghodous²

¹ *Université de Lyon, INSA-Lyon, Villeurbanne Cedex, France*

² *Université de Lyon, Université de Lyon 1, LIRIS, CNRS, UMR 5205,
20 Avenue Albert Einstein, 69621 Villeurbanne Cedex, France*

{wendpanga-francis.ouedraogo, frederique.biennier, catarina.ferreira-da-silva, parisa.ghodous}@liris.cnrs.fr

Keywords: Context Aware Security, Execution Context, Security Patterns, Security Policy, Security as a Service.

Abstract: Taking advantage of the agility and interoperability provided by Service Oriented Architecture (SOA), Web 2.0 and XaaS (Anything as a Service) technologies, more and more collaborative Business Processes (BP) are set "on demand" by selecting, composing and orchestrating different business services depending on the current need. This involves re-thinking the way information, services and applications are organized, deployed, shared and secured among multi-cloud environment. Fitting this de-perimeterized and evolving execution context requires organising the service protection in a dynamic way in order to provide an up to date and consistent protection. To fit this goal, we propose to integrate the different protection requirements defined according to the business environment in a single security policy. Then we plug a context-aware security deployment architecture on the cloud service middleware to analyse both the security policy and the execution context to select, compose and orchestrate the convenient protection means. A proof of concept built on Frascati middleware is used to evaluate the impact of this "on-line" security mediation.

1 INTRODUCTION

In a highly competitive environment, enterprises are more and more involved in collaborative strategies to provide complex services and outstanding products fitting the customers requirements. Service-Oriented Architecture (SOA) and Cloud platforms provide agile and interoperable supports to compose, share and deploy on the fly complex service-based workflows. The development of such a de-perimeterized and agile Information System challenges the development of dynamic protection strategy, as threats and vulnerabilities evolve continuously depending on to both organizational and deployment platforms contexts.

The security policy model provided by the OASIS allows outsourcing security mechanisms from business services. Nevertheless, this model may lead to multiple security policies enactment as services are "replicated" while they are composed to set a new business process. This may lead to inconsistent protection compared to the up-to-date corporate protection strategy.

To overcome this limitation, we propose to extend this security policy model to integrate context information, avoiding replicating the policy depending on

the business and/or execution context. Then, this unified security policy is used as a Model@run.time by a mediation service to select, compose and orchestrate on the fly the security services deployment depending on the business context and execution environment.

After presenting a motivation example and the state of the art in section 2, we introduce our context-aware security model focusing on the business resource protection in section 3. We detail its implementation in section 4 and evaluate its performance in section 5.

2 CONTEXT AND MOTIVATION EXAMPLE

Motivations for defining a unified security policy have been found in previous projects in Collaborative Networked Organisation (CNO) and Dynamic Supply Chain environments. The reduced time-frame and the fast changing environment of these CNOs call for an opened and agile IT support. This is partly fulfilled thanks to the composition / orchestration / elastic deployment mechanisms provided by SOA, Web 2.0,

XaaS (Anything as a Service) technologies. Developing such cloud-based business service reusing ability requires to-rethink the way business service and data are protected to fit their a priori unknown execution context.

2.1 Motivation Example

This CNO dynamic protection context can be illustrated with a use case picked from a previous project where a mechanical engineering SME is involved in different CNOs. This SME uses a key business service to validate mechanical specifications of new products. This business service may be invoked in different business processes (BPs):

- The corporate *Computer Aid Design (CAD)* modelling system can invoke this service to check the intermediate specification consistency (75% of the invocations),
 - The *Product Lifecycle Management (PLM)* system can invoke the service to check the product information before integrating data in its database (10% of the invocations),
 - The service can also be used by some of the partners to validate the engineering requirements they send. In such case, authorized partners can invoke the service from their own *CAD* system (15% of the invocations).
- As the data produced and acceded by the validation service has a high patrimonial value for the enterprise, different protections must be deployed:
- Strong authentication to support specification traceability, i.e., knowing who has achieved/worked on the specifications.
 - Restricted access control to allow only people from the enterprise or some authenticated partners to accede the service.
 - Exchanged data protection with cryptographic algorithm.

These protection requirements lead to identify 3 execution contexts related to the validation service:

- *Context 1* - Internal specification checking: This service is invoked by the corporate CAD services under the control of members of the enterprise in a safe environment. In this context, no extra protection is required as the calling service can be identified.
- *Context 2* - Certified requirement checking: This service is invoked by the PLM service under the control of members of the enterprise to check engineering data validity before storing them.

As it is used to implement a certification, non-repudiation mechanism is required, involving to capture user identity, check input and output information integrity.

- *Context 3* - Collaborative specification checking: This service is invoked by a partner from its CAD system to validate the specifications sent as engineering requirements. Due to business constraints, authentication, authorization and non-repudiation are necessary, whereas the open execution platform requires data encryption.

Most of the time, the business service and the protection means are orchestrated before being deployed, without checking the global consistency of the different policies on a given asset. To overcome this limitation, we propose to adapt the security policy dynamically so that the protection means fit the (may be new) business context.

2.2 State of the Art

To overcome the "lack of trust" and "unsuitable security policy deployment" pointed out by different surveys (Heiser and Nicolett, 2008) (Ban et al., 2010) regarding Web-based collaborative organizations, security requirements must be integrated in process models. Some annotation-based solutions have been developed to integrate security services in BPMN (Business Process Model and Notation)¹ descriptions (Rodríguez et al., 2007) (Ouedraogo et al., 2013). As these annotations are related either to a particular BPMN object or connector, extensions are required to define a global and consistent end to end process protection.

More recently, the OASIS Service Reference Architecture² provides a set of models to "outsource" the security services (namely Confidentiality, Integrity, Availability, Authentication, Authorization, Non-Repudiation) from the business service. Moreover, security protocols and standards such as WS-*, XACML³ or SAML⁴ have been defined to support interoperable implementation of the necessary security mechanisms.

Different works are based on the OASIS outsourced security policy model to integrate security issues in services composition. Some use security requirements as constraints while selecting and com-

¹<http://www.omg.org/spec/BPMN/2.0/>

²<http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html>

³<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>

⁴<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html>

posing services (Bartoletti et al., 2005) whereas other define various calculus algebra to decide if a right can be granted or not (Bartoletti et al., 2006). Paying attention on Business Process (BP) protection requirements, (Lang and Schreiner, 2009) has proposed a Model-Driven Security approach (MDS) to generate security policies. The generation process is achieved according to a static environment vision (perimeterized process and well-known deployment platform), leading to define different policies depending on the business context. This makes the policy management complex and limits a consistent protection evolution as modifications are achieved locally. Furthermore, while some earlier works focus mostly on access control (Wolter et al., 2009), extensions are now defined to capture protection requirements while designing BP (Lucio et al., 2014). In former works, we have proposed a simplified risk analysis knowledge base to capture protection requirements used to generate the convenient policy assertions (Ouedraogo et al., 2013). Even if this user-oriented security engineering strategy allows to identify protection requirements and mitigation means adapted to the collaboration context, it can lead to inconsistent protection as each context-dependent protection policy is defined separately. To overcome this limitation, the different protection strategies must be gathered in a single reference policy attached to each business service and deployed in a business context-aware mode to mitigate the security risks associated to contextual vulnerabilities and threats.

3 CONTEXT AWARE SECURITY SPECIFICATION

We focus on business service protection while cloud-based deployment allows reusing these protections in different business contexts. To avoid inconsistent protection specification, we extend the resource model introduced in the MDS approach to integrate context related information so that a unique security policy can be set. To maintain up-to-date protection, security patterns associated to risks mitigation best practices are also introduced.

3.1 Resource Model

A resource may be a service, some information used by the service or even a part of a process. Each resource (Res_k) can be characterized by a name ($ResN$), a type ($ResT$) which can be business service/service data or a data, a resource application layer ($ResL$) (business, service, or infrastructure). A resource can

include sub resources ($SubResN$) (for e.g. a business service can include set of others business services (sub services) and each sub service handles a set of data (see eq. (1)). As a consequence resource are defined in a recursive way.

$$Res_k = (ResN; ResT; ResL; \{SubRes_s\}) \quad 0 < s \leq N_s; \quad (1)$$

Where " Res_k " is the resource, "k" the resource number, " $SubRes_s$ " the name of the sub resource, "s" the sub resource number, and " N_s " the total of the sub resources.

We enrich this traditional resource definition with a "use case" part related to a particular business extension context. This allows to define the different mitigation means associated to a particular business context (see Fig. 1).

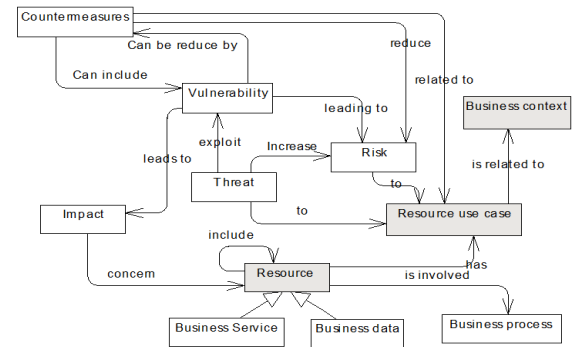


Figure 1: Security concepts associated to business service protection.

3.2 Execution Context Specification

We define the execution context ($ExCtx$) as a set of functional ($FntCtx$), organizational ($OrgCtx$) and technical ($TecCtx$) specifications, which determine the choice of security countermeasures to perform (see eq. (2)).

$$ExCtx = (FntCtx, OrgCtx, TecCtx) \quad (2)$$

The *Functional Context* ($FntCtx$) describes the resource set of Functional Requirement ($FntReq$) i.e. the type of information (strategic, personal, financial...) used or handled by the resource and the resource information sensitive level (top secret, secret, confidential, restricted, unclassified) related to its importance (see eq. (3)).

$$FntCtx = \{FntReq_f\} \quad 0 < f \leq N_f \quad (3)$$

Where " $FntReq_f$ " is a functional requirement, "f" requirement number, and " N_f " the total of the functional requirements. Each $FntReq$ is defined as a tuple (eq. (4)):

- *FntReqId*: is the id of functional requirement,
- *FntReqT*: is the information type (strategic, personal, financial...) used or handled by the resource,
- *FntReqV*: is the sensitive level (top secret, secret, confidential, restricted, unclassified) indicating its importance based on the risk analysis method.

$$FntReq = (FntReqId, FntReqT, FntReqV) \quad (4)$$

The *Organizational Context* (*OrgCtx*) is mostly focused on access control (authentication and authorization), availability and non-repudiation. It allows defining the obligations and the restriction constraints in terms of access control, providing answers to the following questions: Who can access and interact with the resource? From which locality and at which time? This *Organizational Context* (*OrgCtx*) includes a set of organizational requirement (*OrgReq*) (see eq. (5)).

$$OrgCtx = \{OrgReq_o\} \quad 0 < o \leq No \quad (5)$$

Where *OrgReq_o* is an organizational requirement, "o" the requirement number, and "No" the total of the organizational requirement. Each *OrgReq* has a tuple (see eq. (6)):

- *OrgReqId*: is the organizational requirement id,
- *OrgReqT*: is the type of organizational requirement (Who, When, FromWhere).
- *OrgReqW*: defines the way that the requirement is defined ("UserGroup" or "individually" for the "who" requirement, "date" or "dateTime" for "when", and "country" or "region" or "IPAdress" for "FromWhere").
- *OrgReqV*: is the value related to *OrgReqW*, i.e. the specific user or service which can invoke the resource, its localization and the period the resource can be invoked.

$$OrgReq = (OrgReqId, OrgReqT, OrgReqW, OrgReqV) \quad (6)$$

The *Technological Context* (*TecCtx*) includes a set of Technical Requirements (*TecReq*) related to the network, devices and deployment platform involved in the user / resource interaction at runtime (see eq. (7)).

$$TecCtx = \{TecReq_t\} \quad 0 < t \leq Nt \quad (7)$$

Where *TecReq_t* is a technical requirement, "t" the requirement number, an "Nt" the total of the technical requirement.

Each *TecReq_t* defines which technical mean (networks, devices) is used to access the resource. It is defined as a Tuple including (eq. (8)):

- *TecReqId*: is the technical requirement id.

- *TecReqT*: is the type of technical element (device, network, protocol, etc.) authorized to access or interact with the resource.
- *TecReqV*: is the value (Smartphone, PC, IP address) related to type of technical element.

$$TecReq = (TecReqId, TecReqT, TecReqV) \quad (8)$$

To support a context-aware security deployment, we integrate this context information in the resource model (see eq. (9)):

$$Res_k = (ResN; ResT; ResL; \{SubRes_s\}, \{ExCtx_e\}) \quad 0 < e \leq Ne \quad (9)$$

Where *ExCtx_e* is a particular execution context, "e" is the execution context number, and "Ne" the total of the execution number.

Based on this model, listing 1 presents the *ValidateSpec* resource associated to the specification checking service from our motivation example.

Listing 1: XML-based syntax representation of the *validateSpec* resource from our motivation example.

```

1 <res resN="/MechanicalSystemSpec/validateSpec" resT="
  Service" resL="Service">
2   <exCtx id="1">
3     <fntCtx>
4       <fntReq fntReqId="2" fntReqT="strategic"
5         fntReqV="Secret"/>
6     </fntCtx>
7     <orgCtx>
8       <orgReq orgReqId="1" orgReqT="who" orgReqW="
9         UserGroup" orgReqV="{CADService}"/>
10      </orgReq>
11    </orgCtx>
12    <tecCtx>
13      <tecReq tecReqId="2" tecReqT="Network/DNS"
14        tecReqV="{mechanicalCompagny.com}"/>
15    </tecCtx>
16  </exCtx>
17  <exCtx id="2">
18    <fntCtx>
19      <fntReq fntReqId="2" fntReqT="strategic"
20        fntReqV="Secret"/>
21    </fntCtx>
22    <orgCtx>
23      <orgReq orgReqId="1" orgReqT="who" orgReqW="
24        UserGroup" orgReqV="{PDMSservice}"/>
25    </orgReq>
26    </orgCtx>
27    <tecCtx>
28      <tecReq tecReqId="2" tecReqT="Network/DNS"
29        tecReqV="{mechanicalCompagny.com}"/>
30    </tecCtx>
31  </exCtx>
32  <exCtx id="3">
33    <fntCtx>
34      <fntReq fntReqId="1" fntReqT="strategic"
35        fntReqV="Secret"/>
36    </fntCtx>
37    <orgCtx>
38      <orgReq orgReqId="1" orgReqT="who" orgReqW="
39        UserGroup" orgReqV="{CADService, ..
40        CADService}"/>
41    </orgReq>
42    </orgCtx>
43    <tecCtx>
44      <tecReq tecReqId="2" tecReqT="Network/DNS"
45        tecReqV="{mechanicalCompagny.com}"/>
46    </tecCtx>
47  </exCtx>
48 </res>

```

This listing describes the *validateSpec* resource (Line 1) and its different execution contexts: Lines 2-13 for Context 1, Lines 14-25 for Context 2 and Lines

26-37 for Context 3. Access control features are defined for each context (see line 7 for Context 1, line 19 for Context 2 and line 31 for Context 3).

3.3 Context Aware Pattern Model

To generate the adapted protection means, the adapted countermeasures are selected from the OASIS security taxonomy. Our countermeasure pattern model is a tuple (eq. (10)):

$$Pat_p = (PatN, PatG, PatL, PatM, \{PatCtx_c\}, PatR, \{PatSet_s\}, PatCsq) \quad (10)$$

where $0 < (p, c, s) \leq (Np, Nc, Ns)$

- *Pattern Name (PatN)* is a unique name that refers to one element of our security taxonomy.
- *Pattern Goal (PatG)* defines the reason for using the pattern.
- *Pattern layer (PatL)* defines the layer of the pattern (Business, Service, Technique).
- *Pattern Metric (PatM)* is the protection level provided by pattern (Very High, High, Medium, and Lower) to fit risk sensitivity level (Top Secret, Secret, Confidential, Restricted).
- *Pattern Context (PatCtx_c)* defines the set of contexts in which the pattern is applicable. Each *PatCtx_c* includes a set of optional conditions split into functional conditions (*FntCdt*), organizational conditions (*OrgCdt*) and technical conditions (*TecCdt*) related to this context (see eq. (11)).

$$PatCtx_c = (\{FntCdt_f\}, \{OrgCdt_o\}, \{TecCdt_t\}) \quad (11)$$

$0 < (c, f, o, t) \leq (Nc, Nf, No, Nt)$

The *FntCdt* includes the information type (*FntCdtT*) that the resource has to handle and sensitive level value (*FntCdtV*) of this information for which the pattern will be useful (eq. (12)).

$$FntCdt = (FntCdtT, FntCdtV) \quad (12)$$

For example data encryption pattern can be used if the information sensitive level is secret, requiring a high protection security. The *OrgCdt* (eq. (13)) defines among 3 organizational conditions which type of conditions (*OrgCdtT*) are necessary to make the pattern applicable:

- The “*who*” condition defines the type of user authorized to access the resource.
- The “*where*” condition defines the authorized location of the user.
- The “*when*” condition defines time constraints to accede to the resource.

Condition applicability (*OrgCdtW*) specifies the control strategy. For example the XACML protocol can be used if the organizational condition is defined and the access control mode is by “User-Group”.

$$OrgCdt = (OrgCdtT, OrgCdtW) \quad (13)$$

The technical condition (*TecCdt*) (eq. (14)) refers to platform-dependent protection constraints. It is used to select a pattern according to the type of the technical component that is implemented (*TecCdtT*) and the related value (*TecCdtV*).

$$TecCdt = (TecCdtT, TecCdtV) \quad (14)$$

- *Pattern related (PatR)* defines a set of related patterns. For e.g., Authentication pattern can require a set of different patterns (integrity and encryption patterns) to secure authentication token.
- *Pattern Setting (PatSet)* (eq. (15)) describes the set of parameters (*Set*) necessary to use a pattern. These parameters are initialized according to the execution context.

$$PatSet = \{Set_s\} \quad 0 < s \leq Ns \quad (15)$$

Where “*s*” is the parameter number and “*Ns*” the total number of parameters. Each parameter (*Set_k*) is defined as a tuple (eq. (16)):

$$Set = (Setkey, SetValue) \quad (16)$$

Where *Setkey* is the parameter name and *SetValue* the value of the parameter.

- *Pattern Consequence (PatCsq)* describes the results or actions implemented by the pattern. The set of patterns is defined by (eq. (17)):

$$Pats = \{Pat_j\} \quad \text{where } 0 < j \leq N_j \quad (17)$$

Where “*j*” is the pattern number and “*Nj*” the total number of patterns.

Based on the protection requirement to fulfill and on information on the current deployment context, the convenient mitigation pattern can be selected to generate a security policy rule.

3.4 Reference Security Policy Model

Our formal security policy model, defined as a tuple (eq. (18)), extends the OASIS policy model. For each security criteria defined in the OASIS model, a set of policy rules is defined and related to policy context. By this way, the security mechanisms that must be deployed to provide a consistent protection can be tuned depending on the execution context.

$$Pol_x = (PolR, PolT, PolG, PolL, PolRls) \quad 0 < x \leq Nx \quad (18)$$

Where

- *Policy Resource (PolR)*: is the resource (Business service, data, etc.) involved in the policy.
- *Policy type (PolT)*: refers to the security service taxonomy (Authentication, Confidentiality, Integrity, Non-repudiation, Availability, etc.)
- *Policy Goal (PolG)*: defines the policy aims.
- *Policy Layer (PolL)*: is the layer (Business, Service or Infrastructure) of the involved resource for this policy.
- *Policy Rules (PolRls)*: is a set of policy rules (eq. (19)). Each policy rule ($PolRl_r$) defines the security mechanism to apply according to the resource execution context.

$$PolRls = \{PolRl_r\} \quad 0 < r \leq N_r \quad (19)$$

Where $PolRl_r$ is a policy rule, "r" the policy rule number, and "Nr" the total of policy rules. Each policy rule is also defined as a tuple (eq. (20)):

$$PolRl = (RlId, RlP, RlM, RlCtx, \{RlSet\}) \quad (20)$$

Where:

- *Policy Rule Id (RlId)*: is the policy rule id,
- *Policy Rule Pattern (RlP)*: is the name of the pattern which matches the resource execution context.
- *Policy Rule Metric (RlM)*: is the pattern associated metric which matches the resource execution context. It is the protection level provided by the pattern.
- *RICtx*: includes the part of the pattern context which matches the resource execution context.
- *RlSet*: is the initialized pattern settings allowing to invoke the security mechanism.

Then all the policies associated to a resource are gathered in a single set (eq. (21)):

$$Pols = \{PolR_x\} \text{ where } 0 < x \leq N_x; \quad (21)$$

Where "x" is the policy number and "Nx" the total number of policy (for all resources).

Lastly, policy attached to any resource R_k can be defined by selecting (σ) the policy in the Pols set where the policy resource (PolR) matches with resource " R_k ":

$$Pols(R_k) = \sigma_{(Pols.PolR=R_k)}(Pols) \quad (22)$$

This security policy is used to generate a security policy file attached to the resource description (WSDL⁵ or WADL⁶ file) that contains all the security assertions and their application context. Listing 2

shows the security policy of the specification validation operation (*validateSpec*) of the Mechanical System specification business service, namely authentication (Lines 2-14) if the calling service is not the corporate *CADService*, authorization (Lines 27-39), non repudiation (Line 15-26) and communication channel protection (Lines 40-54).

Listing 2: validateSpec service security policy expressed using XML syntax.

```

1 <pols>
2 <pol polId="1" polR="/mechanicalSystemSpec /
  validateSpec" polT="Authentication">
3   <polR rId="1" RIP="LoginPWD", rIM="0.5">
4     <rICtx>
5       <orgCdt orgReqT="who" orgReqW="UserGroup"
6         orgReqV="!{CADService}" />
7       <tecCdt tecReqT="Network" tecReqV="Network /
8         DNS" tecReqV="!{mechanicalCompagny.com}"
9       />
10      </rICtx>
11    </polR>
12  </pol>
13 <pol polId="2" polR="/mechanicalSystemSpec /
  validateSpec" polT="NonRepudiation">
14   <polR rId="1" RIP="Log", rIM="0.5">
15     <rICtx>
16       <orgCdt orgReqT="who" orgReqW="UserGroup"
17         orgReqV="!{CADService}" />
18       <tecCdt tecReqT="Network" tecReqV="IP" />
19     </rICtx>
20     <rSets>
21       <rSet setkey="logFile" value="log.log" />
22     </rSets>
23   </polR>
24 </pol>
25 <pol polId="3" polR="/mechanicalSystemSpec /
  validateSpec" polT="Authorization">
26   <polR rId="1" RIP="ACL", rIM="0.5">
27     <rICtx>
28       <orgCdt orgReqT="who" orgReqW="UserGroup"
29         orgReqV="!{CADService,PDMSERVICE}" />
30       <tecCdt tecReqT="Network" tecReqV="Network /
31         DNS" tecReqV="!{mechanicalCompagny.com}"
32       />
33     </rICtx>
34     <rSets>
35       <rSet setkey="ACLFile" value="acl /
36         AccessControlList.xml" />
37     </rSets>
38   </polR>
39 </pol>
40 </pols>

```

4 CONTEXT AWARE SECURITY DEPLOYMENT

Taking advantage of the loosely coupled strategy, we use the middleware online interception capability to intercept each service/middleware interaction and routes this interaction to our context aware architecture (Fig. 2) built as an add-on component. It consists in:

⁵<http://www.w3.org/TR/wsdl>

⁶<http://www.w3.org/Submission/wadl/>

- a *Context Aware Deployment* composite which is the core component to achieve the dynamic security deployment. This component is split into three sub components:
 - The *Context Aware Security* component analyses the calling service requests intercepted by the middleware and encapsulated into a Request object. It invokes the policy manager and the context manager to get the security policy rules associated to the business services fitting the context before invoking the required security services.
 - The *Context Manager* analyses security policies associated to the services and identifies the different policies to be applied according to the current context.
 - The *Policy Manager* identifies the list of policies fitting the context from the global policy file associated to the resource.
- The *Security as a Service* (SecaaS) composite gathers implementation of various security services (authentication, authorization, integrity controls, etc.) such as:
 - The *SecaaS* component is the composite entry point. It analyses the policies sent by the Context Manager component to identify the security services (authentication, authorization, etc.) to call.
 - The security components (*Authentication, Authorization, Integrity, Encryption, Non-Repudiation Availability*) invoke the security service depending on the pattern and parameters specified in the policy.

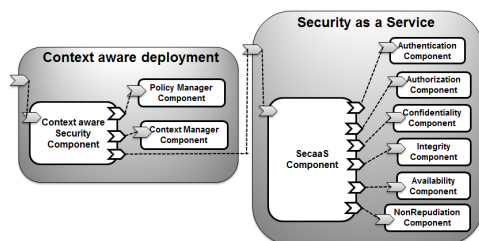


Figure 2: Context aware security architecture.

5 EVALUATION

To evaluate the impact of our Context Aware Deployment, we implement a Proof of Concept plugged on the Frascati middleware (Merle et al., 2011) extending our Model Driven Policy generator presented in (Ouedraogo et al., 2013). We then use this prototype

to support various experiments on BP protection policies specification depending on shared collaborative context.

5.1 Performance Evaluation

We assume that the cloud oriented service middleware provides a unified environment to invoke business services dispatched on a multi-cloud platform. As a consequence, we set a test on a single machine environment using Frascati version 1.6 with Oracle Java Virtual Machine 1.7.0 51 on Microsoft Windows 7 Professional (32 bit) using a 2,54GHz processor Intel(R) Core(TM) 2 Duo CPU with 4Go of memory to compare the dynamic security deployment execution time with the business service and the protection service execution times.

In our example, the *"validateSpec"* of the *"mechanicalSystemSpec"* service and the related information must be protected in the different contexts. The confidentiality requirement impacts both application layer, which is in charge of the access control, traceability, and transport layer. The systematic composition of the authorization, traceability and confidentiality services may be costly (see the different execution times reported in Table 1).

Table 1: *ValidateSpec* Service execution time according to the three contexts.

Context	Security services involved mechanical Specification	Execution time (ms)
Context 1	No security mechanism is required	64
Context 2	Authentication and Non Repudiation	80
Context 3	Context 2 security services + Authorization and Confidentiality	84

The contextualized security service orchestration allows invoking only the necessary security mechanisms according to the context, avoiding both the costly over protection and the risky under protection. This dynamic protection does not impact the daily used BP and simplifies the security policy consistency controls as new protection requirements are introduced once in the service policy specification and implemented for the different collaborative BP without impacting the business process deployment.

Other performance simulations have first been conducted in order to evaluate the performance overhead involved by the context analysis at run time. We extend this benchmark to different services (see Table 2) to compare the cost of our context-aware architecture with the business service execution time. We set two reference execution contexts, one requiring no security deployment (Context 1) whereas the other requires the maximum protection (i.e systematic protection required in Context 3 from our motivation example). Total execution times are measured

Table 2: Execution times for a panel of 1000 invocations of business services where the "no protection" rate evolves from 0% to 100%.

	Systematic protection	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
S1 (10ms)	32000	38000	35800	33600	31400	29200	27000	24800	22600	20400	18200	16000
S5 (50ms)	72000	78000	75800	73600	71400	69200	67000	64800	62600	60400	58200	56000
S10(100ms)	122000	128000	125800	123600	121400	119200	117000	114800	112600	110400	108200	106000

for 1000 invocations split among different occurrence rate for context 1 (see Table 2). The ratio comparing the context-aware secured business service execution time with the systematically secured business service execution time presents a maximum of 4,92% overhead for the bigger service to 18,75% for the smallest one when these services invocation requires the highest protection (0% of occurrence of the "no protection context"). On the opposite, our context aware security deployment can save from 50% up to 86,89% of execution time when all invocation do not need any protection. These results show that the overhead involved by our architecture can be rather neglected (from 4,92% to 18,75% of the service execution time overhead when the highest protection is always required) compared to the large overhead introduced by the systematic invocation of (often) useless security services provided that the "no protection" invocation context rate is greater than 30% as shown in Fig. 3.

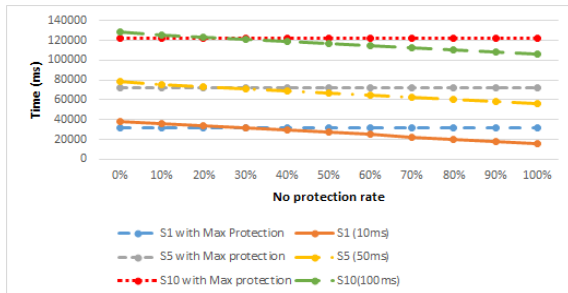


Figure 3: Variation of context aware security execution cost for three business services.

6 CONCLUSION

To secure business services used in collaborative environment, enterprises have to adapt the protection according to the execution context. To this end, we propose a context aware security model and architecture used to select and orchestrate security services at run-time. This architecture, tested on the Frascati middleware, shows that the dynamic security mediation has a rather low impact on the performance level compared with a systematic deployment of costly over-protection.

Further works will focus on the integration of more detailed platform models and on vulnerabilities

monitoring loops so that our coarse-grained vision of the execution context will be refined to increase the protection efficiency.

REFERENCES

- Ban, L. B., Cocchiara, R., Lovejoy, K., Telford, R., and Ernest, M. (2010). The evolving role of it managers and cios.
- Bartoletti, M., Degano, P., and Ferrari, G. (2005). Enforcing secure service composition. In *Computer Security Foundations, 2005. CSFW-18 2005. 18th IEEE Workshop*, pages 211–223.
- Bartoletti, M., Degano, P., and Ferrari, G. (2006). Security issues in service composition. In Gorrieri, R. and Wehrheim, H., editors, *Formal Methods for Open Object-Based Distributed Systems*, volume 4037 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg.
- Heiser, J. and Nicolett, M. (2008). Assessing the security risks of Cloud Computing. *Technical report*, Gartner.
- Lang, U. and Schreiner, R. (2009). Model Driven Security Management: Making Security Management Manageable in Complex Distributed Systems. In *Workshop on Modeling Security (MODSEC08) - International Conference on Model Driven Engineering Languages and Systems (MODELS)*.
- Lucio, L., Zhang, Q., Nguyen, P. H., Amrani, M., Klein, J., Vangheluwe, H., and Traon, Y. L. (2014). Chapter 3 - Advances in Model-Driven Security. In Memon, A., editor, *Advances in Computers*, volume 93, pages 103 – 152. Elsevier.
- Merle, P., Rouvoy, R., and Seinturier, L. (2011). A Reflective Platform for Highly Adaptive Multi-Cloud Systems. In *International Workshop on Adaptive and Reflective Middleware (ARM'11) - 12th ACM/I-FIP/USENIX International Middleware Conference*, pages 14–21. ACM.
- Ouedraogo, W. F., Biennier, F., and Ghodous, P. (2013). Model driven security in multi-context. In *International Journal of Electronic Business Management*, volume 11 No. 3, pages 178–190.
- Rodríguez, A., Fernández-Medina, E., and Piattini, M. (2007). A BPMN Extension for the Modeling of Security Requirements in Business Processes. *IEICE - Trans. Inf. Syst.*, E90-D(4):745–752.
- Wolter, C., Menzel, M., Schaad, A., Miseldine, P., and Meinel, C. (2009). Model-driven business process security requirement specification. *Journal of Systems Architecture (JSA)*, pages 211–223.

Choreography-based Consolidation of Interacting Processes Having Activity-based Loops

Sebastian Wagner¹, Oliver Kopp^{1,2} and Frank Leymann¹

¹IAAS, University of Stuttgart, Universitaetsstr. 38, Stuttgart, Germany

²IPVS, University of Stuttgart, Universitaetsstr. 38, Stuttgart, Germany
firstname.lastname@iaas.uni-stuttgart.de

Keywords: BPEL, Choreography, Process Consolidation, Loops.

Abstract: Choreographies describe the interaction between two or more parties. The interaction behavior description might contain loops. In case two parties want to merge their behavior to gain competitive advantage, the contained loop constructs also have to be merged. This paper presents a language-independent discussion on loop-structure pairing in choreographies and possible merging strategies. Thereby, the focus is turned on loops grouping child activities to be iterated. We show the feasibility of the merging strategies by applying them to BPEL-based choreographies.

1 INTRODUCTION

Business process consolidation (also called “process merge”) integrates two or more complementing and hence often interacting business processes into a single process. From a business perspective, process consolidation is applied by companies to regain control of outsourced business functions (“business process insourcing”). In the scenario in Fig. 1, for instance, a manufacturer integrates the process of its supplier in its own process. Beside the business perspective, there exist also technical drivers for consolidating processes. Especially in instance-intensive interaction scenarios, where hundreds or thousands of process instances interact with each other, consolidating the interacting processes may lead to significant performance gains (Wagner et al., 2013). They result from avoiding the costly message transfer steps, i. e., the message serialization at the sender side, the actual message transfer and the message deserialization at the receiver side. Since usually complex XML-based protocols such as SOAP are used to exchange messages between processes the message transfer becomes even more resource intensive (Ng et al., 2004). Another advantage of consolidating interacting processes is the decreased number of process instances that have to be managed by the workflow engines. As typically the pay-per-use model is applied in Cloud environments, these performance savings result also in lower costs for enacting a choreography on a workflow engine being hosted in the cloud.

To facilitate process consolidation an approach (Wagner et al., 2012) was developed that automatically consolidates complementing acyclic processes, whose interaction behavior is specified by a choreography, into a single process. The approach ensures that the consolidated process, called P_{Merged} in the following, generates the same set of traces of basic activities as the original choreography. To accomplish that, the approach also adds additional control links to P_{Merged} to relate activities originating from the different processes to be consolidated. So far, the consolidation approach is just capable to merge acyclic processes. If processes are merged that interact via activity-based loops, i. e., loops that contain activities to be iterated in their loop body, the consolidated process P_{Merged} becomes invalid. This is due to the fact, that the additional control links created by the current consolidation approach may cross loop boundaries. However, workflow languages that support activity-based loops, such as BPEL (OASIS, 2007) and BPMN (Object Management Group (OMG), 2011), do not allow control links crossing loops boundaries. For instance, in Fig. 1 the consolidation created an invalid process because the generated control link connects the activities “Syn3” and “Syn4” that are located in different loops.

This work extends the consolidation approach to support the consolidation of processes that interact via activity-based loops. For this purpose, we discuss different interaction patterns involving activity-based loops communicating with other loops (e. g., graph-

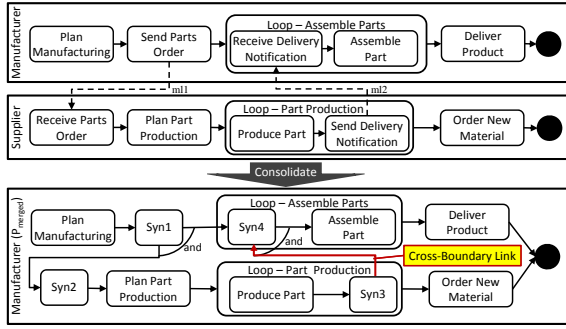


Figure 1: Consolidation of Interacting Processes.

based loops) by means of a language-independent workflow meta-model. For each pattern it is discussed how a consolidation can be performed that keeps the execution order between basic activities that was defined in the original choreography. We developed a tool for consolidating interacting BPEL processes. Therefore, to validate the approach, we will show to what extend the language-independent patterns can be mapped to BPEL in order to implement them in the consolidation tool.

The remainder of the paper is structured as follows. In Sect. 2 we give a brief overview about the process consolidation. In Sect. 3 the meta-model of the workflow language used in this work is defined. Section 4 describes patterns to resolve the cross-boundary violations of activity-based loops. In Sect. 5 the patterns are validated to BPEL and the prototype that implements the patterns is presented. Section 6 gives an overview about the related work and in Sect. 7 the work is concluded.

2 PROCESS CONSOLIDATION APPROACH

The actual business functions of processes, e. g., human tasks, data manipulations etc. is implemented by basic activities, i. e., by activities that do not contain other activities. The possible set of execution traces of basic activities during choreography runtime is determined by the control flow constructs (e. g., control links, loops etc.) and interaction patterns defined in the choreography. Thus, for being correct, P_{Merged} must be able to generate the same set of traces of basic activities (without communication activities that are removed during consolidation) during runtime as the original choreography, where P_{Merged} was created from. The same set of traces can be only generated if P_{Merged} keeps the execution orders between the basic activities. The execution order between two basic

activities a_i and a_j defines that a_i must be either performed before, after or parallel to a_j . To preserve the execution order, the consolidation operation performs the following steps:

At first, a single process named P_{Merged} is created. Then the activities of all interacting processes of the choreography along with their incoming and outgoing control links are copied into P_{Merged} . The basic activities are left in their parent activities (e. g., “Produce Part” stays in “Part Production”). This ensures that in P_{Merged} the originally modeled execution order between the activities originating from the same process is preserved.

P_{Merged} still contains communication activities used by the processes to interact with each other. These activities are replaced by synchronization activities that inherit the control links from the communication activities replaced by them. For instance, in Fig. 1 “Send Parts Order” and “Receive Parts Order” are replaced by “Syn1” and “Syn2” respectively. If the data flow of the workflow language is modeled by control flow constructs such as in BPEL, the synchronization activities can be used to emulate the message transfer. For instance by copying the former message content from the data object that was read by the sending activity, to the data object where the message content was copied to by the receiving activity. In BPMN the synchronization activities just act as sources or targets for the materialized control links but they do not perform any operations. In a choreography the execution order between basic activities originating from different processes is implicitly defined by the interaction specification, i. e., by the message links ($m1$ and $m2$ in the example). The asynchronous interaction between “Send Parts Order” and “Receive Parts Order” via message link $m1$ implies that activity “Plan Manufacturing” is always performed before “Plan Part Production”. To keep this execution order “control-flow materialization” is performed, i. e., based on the interaction type new control links are created. To replace an asynchronous interaction the link originates at the synchronization activity that replaced the sending activity and ends at the synchronization activity that replaced the receiving activity. These new control links may cause cross-boundary violations, i. e., they cross the boundaries of the activity-based loops, as shown in the example in Fig. 1, which is not permitted in BPMN or BPEL.

3 PRELIMINARIES

Definition 1 (Process). A process is defined as a directed single entry single exit (SESE) graph $P = (A, E)$.

The set A denotes the set of activities and set E denotes the set of control links of the graph. The set of control links is defined as $E \subseteq A \times A \times C$. C denotes the set of link conditions. Conditions are logical expressions that can be evaluated at runtime of P to true or false.

An activity of a process has a set of incoming control links $E^{\rightarrow}(a) = \{(a_i, a, c) \mid (a, a_i, c) \in E\}$ and a set of outgoing control links $E^{\leftarrow}(a) = \{(a, a_i, c) \mid (a, a_i, c) \in E\}$. The activity where $|E^{\rightarrow}(a)| = 0$ is called “entry activity” a_{entry} of P and the activity where $|E^{\leftarrow}(a)| = 0$ is called “exit activity” a_{exit} of P . The set of directly preceding activities of an activity a is denoted by $\bullet a$ and the set of directly succeeding activities of a are denoted by $a \bullet$.

The function $\text{PreDom} : A \rightarrow 2^A$ returns all activities that are (pre-)dominated by activity a and a itself (Koehler et al., 2005). An activity a dominates another activity b if every path from the entry activity to b goes through activity a . All activities that are post-dominated by activity a and a itself are returned by the function $\text{PostDom} : A \rightarrow 2^A$ (Koehler et al., 2005). An activity a post-dominates another activity b if every path from b to the exit activity goes through activity a .

The control flow of a process model follows the token semantics of BPMN (Object Management Group (OMG), 2011). The entry activity of P propagates a token to each of its outgoing control links. A link that receives a token consumes it and evaluates its link condition. If the link condition evaluates to *true* the link is activated, i. e., it produces a token and passes it to its subsequent target activity. An activity is started (consumes a token) when at least one of its incoming links is activated and no more upstream tokens may reach the activity. Informally, this also holds for OR-joins. Formally, OR-joins have to be Q-enabled to start. Q-enabledness is defined by Völzer (Völzer, 2010). After the activity is completed, one token is passed on to each of its outgoing links. The exit activity just consumes tokens. A process is completed when there are no other upstream tokens.

Definition 2 (Choreography and Message Links). A choreography $C = (\mathcal{P}, \mathcal{ML})$ consists of a set of processes \mathcal{P} and message links $\mathcal{ML} \subseteq A \times A$. Each message link $ML \in \mathcal{ML}$ connects two activities a_i and a_j from different processes $P_1, P_2 \in \mathcal{P}$. In a message link $ML = (a_i, a_j)$ the source activity a_i is the sending activity and the target activity a_j the receiving activity of a message. An activity must be only source or target of exactly one message link. A message link is activated, when a_i is started and a_j cannot complete until the link is activated, i. e., a_j “hangs” until the link is activated. Note, that a_i sends a message m in a send and forget manner, i. e., a_i completes, even if a_j was not started yet. We refer to all activities that are source or target

of a message link as communication activities.

In the following different types of loops are defined that are provided by the most workflow languages (van der Aalst et al., 2003). These loop types are used, to define the patterns for solving cross-boundary violations introduced in 4.

Definition 3 (Activity-based Loop). An activity-based loop L is a special type of activity defined as $L = (A_L \subset A, E_L \subset E, c \in C, \text{eval}_c = \{\text{pre}, \text{post}\})$. The loop body is a SESE graph consisting of the activities A_L and the control links E_L . No control link must cross the boundary of the loop, i. e., $\forall e \in E_L : \pi_1(e), \pi_2(e) \in A_L$, where π_i projects to the i th element of a tuple. eval_c is set to *pre* if the loop condition must be evaluated before the first iteration of the loop body (pre-test loop) or set to *post* if the loop condition must be evaluated after the first iteration of the loop body (post-test loop). The function $\text{Body} : L \rightarrow 2^A \times 2^E$ returns the graph in the loop body. The loop condition is returned by function $\text{Cond} : L \rightarrow C$.

Activity-based loops can be subdivided in “static activity-based loops” and “dynamic activity-based loops”. For static activity-based loops, the maximum possible number of iterations can be determined during design time by using data-flow analysis techniques (Heinze et al., 2012; Kopp et al., 2008). For dynamic activity-based loops, this is not possible at design time but only at runtime. The function $\text{Max} : L \rightarrow \mathbb{N} \cup \{\perp\}$ returns the maximum number of iterations of L and returns “ \perp ” in the case of dynamic activity-based loops.

The loop body of an activity can be thought of as a subprocess because it has the same operational semantics as a process. An activity-based loop that is started, passes a single token to its entry activity and the loop completes after all produced tokens were consumed by its exit activity.

Definition 4 (Graph-based Structured Loop). A graph-based structured loop $L = (A_L \subseteq A, E_L \subseteq E)$ is a subgraph of P , such that there is an entry node $a \in A_L$ and an exit node $b \in A_L$ (which can be the same), such that every path starting from a visits b and may visit the loop entry a again and thus forms a cycle. Hence, all nodes in A_L are reachable from a .

In this paper, we focus on structured loops and do not tackle unstructured loops. Finally, we provide a definition for interacting loops.

Definition 5 (Interacting Loop). Two loops $L1$ and $L2$

are called interacting loops, if

$$\begin{aligned} \exists ml \in \mathcal{ML} : \\ & (\pi_1(ml) \in \Pi_1(\text{Body}(L1)) \\ & \wedge \pi_2(ml) \in \Pi_1(\text{Body}(L2))) \\ & \vee (\pi_2(ml) \in \Pi_1(\text{Body}(L1)) \\ & \wedge \pi_1(ml) \in \Pi_1(\text{Body}(L2))) \end{aligned}$$

Informally, this means that $L1$ contains a communication activity a_i and that is related to a communication activity a_j in $L2$ via a message link ml . Note, that Π_i returns the set of i th elements from the given set of tuples, here the activities of the body of $L1$ and $L2$.

4 SOLVING CROSS-BOUNDARY VIOLATIONS IN ACTIVITY-BASED LOOPS

This section describes different patterns to solve cross-boundary violations created in P_{Merged} by the control-flow materialization while keeping the execution order between basic activities. The patterns focus on scenarios with two interacting loops $L1$ and $L2$, however, they can be also applied to more interacting loops, as discussed at the end of this section. One of the loops must be an activity-based loop as cross-boundary violations do only occur if activity-based loops are involved.

In the following, we refer to the set of links that are crossing boundaries of a loop as $E_{CB} \subset E$. The source and target activities of a link $e_{CB} \in E_{CB}$ are referred to as synchronization activities. For interacting graph-based loops $L1$ and $L2$ P_{Merged} does not have to be adapted, as graph-based loops do not have a loop body. Hence, cross-boundary violations cannot occur. The pattern to be applied depends on the type of $L1$ and $L2$ receptively and also on the types of loops supported by the workflow language.

The *context* of a pattern defines, for what types of interacting loops it can be applied. The *solution* describes how the cross-boundary violation can be resolved. *Variations* discusses different variants of the pattern. The *discussion* describes how the pattern preserves the control flow order between the atomic activities that was originally defined in C . The patterns can be only applied to choreographies that are deadlock free.

Pattern 1: Activity-based Loop Unrolling

Context. A static activity-based loop $L1$ is related to another static activity-based loop $L2$ via one or more cross-boundary links as shown in Fig. 2. $L1$ and $L2$

are not forced to have the same number of iterations. The workflow language does not support graph-based structured loops.

Solution. As the number of iterations of $L1$ and $L2$ is known at design time *loop unrolling* (also called loop unwinding) can be performed on the two loops to resolve the cross-boundary violations. Algorithm 1 implements loop unrolling for an activity-based loop L . In line 3 the first iteration of L is unrolled into P_{Merged} . All other iterations of L (if any) are unrolled in line 7.

To perform the actual duplication of the loop body graph the loop unrolling algorithm calls the function Duplicate that is shown in Algorithm 2. The function creates one copy a' of each activity of the given graph G (line 5). The incoming and outgoing links of each original activity a are also duplicated. These link copies become the incoming and outgoing links of the corresponding copy of a , i. e., a' (lines 10 to 15). This also includes the cross-boundary links. In lines 4 and 8 Algorithm 1 adds the created activity and link copies to the process graph. As the duplication is performed n times, where n denotes the max. number of iterations of L , n subgraphs G_1, \dots, G_I are created in P_{Merged} . Subgraph G_{L1}^1 represents the first iteration of $L1$, G_{L1}^2 the second iteration, etc. For instance, the example loop $L1$ in Fig. 2 is unrolled into two subgraphs G_{L1}^1 and G_{L1}^2 . Hence, the function Duplicate is called twice by the loop unrolling algorithm. The first call creates the activity copies $a2^1 - a5^1$ and second call $a2^2 - a5^2$ along with the corresponding link copies.

To preserve the control flow order between the unrolled iterations of L , G_1, \dots, G_n have to be linked sequentially with each other by a new set of $n - 1$ control links (lines 11 to 14 in Algorithm 1). Therefore, each exit activity of the subgraphs G_1, \dots, G_{n-1} is connected to an entry activity of G_2, \dots, G_n by a new control link e_{next} . To emulate the behavior, that another iteration of L is only performed if the loop condition of L evaluates to *true*, the loop condition of L is also added to each link e_{next} . To skip all other iterations if the loop condition evaluates to *false* after the execution of an iteration G_i , each exit activity of G_1, \dots, G_{n-1} is linked to all direct successor activities of L via a set of links e_{skip} (lines 11 - 14). This means, that each of these links replaces a link from the set of outgoing links of L ($E^{\leftarrow}(L)$). Note, that each of the links from $E^{\leftarrow}(L)$ may have also a link conditions assigned. Hence, each link e_{skip} must be only activated if the loop condition evaluates to *false* and if the link condition of the link from $E^{\leftarrow}(L)$ it replaces, evaluates to *false*. The last iteration, i. e., G_n is related to the successor activities of L in the same way by calling Algorithm 4.

To ensure that the direct predecessor activities of

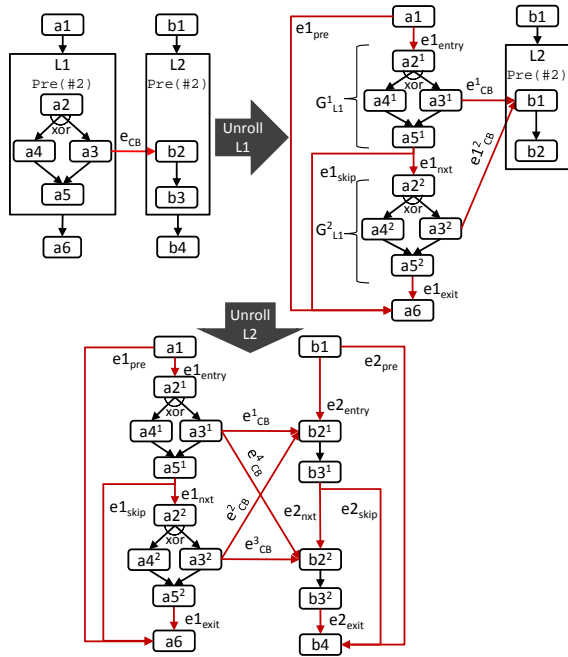


Figure 2: Unrolling of Activity-based Loops.

L become the predecessors of the unrolled iterations of L G_1, \dots, G_n . Algorithm 3 is called. The algorithm links the entry activity of G_i to the direct predecessor activities of L . For each former incoming link of L (i.e., $E^{\rightarrow}(L)$) a new entry link e_{entry} is created (lines 4 and line 11 in Algorithm 3). If L is a post-test loop, the link conditions of the entry links inherit the link conditions of the incoming links $E^{\rightarrow}(L)$ (line 11 in Algorithm 3). This ensures the originally modeled behavior, that the first iteration is always started, if at least one of the incoming links of L is activated.

If L is a pre-test loop the first iteration of the loop body must be only started if at least one of the incoming links of L is activated and if the loop condition evaluates to *true*. To emulate this behavior, the link conditions of the entry links are concatenated with the loop condition of L . To skip the execution of the unrolled loop if the pre-test loop condition evaluates to *false* another set of links $E_{pre} \subset E$ is created between all direct predecessor and successor activities of L .

To guarantee that an activity, target of a copy of a cross-boundary link e_{CB}^i , is performed at most once a Boolean flag is added to P_{Merged} and accessed by the link condition of all copies e_{CB}^i of a cross-boundary link e_{CB} (due to space reasons not shown in the presented algorithms). This flag carries the name of the original cross-boundary link and it is set from *true* to *false* when a copy e_{CB}^i was activated. Thus, any another copy e_{CB}^j of e_{CB} ($i \neq j$) cannot be activated anymore, which prevents its target activity from be-

ing executed again. For instance, if in Fig. 2 the path $\langle a2^1, a3^1, a5^1 \rangle$ is taken, e_{CB}^1 is activated and $b2^1$ is executed which causes the flag to be set to *false*. If the execution continues on the path $\langle a2^2, a3^2, a5^2 \rangle$ the link condition of e_{CB}^2 deactivates this link and prevents $b2^1$ from being started again.

Algorithm 1: Loop Unrolling.

```

1: procedure UNROLL-LOOP( $L$ )
2:    $i \leftarrow 1$ 
3:    $G_i \leftarrow \text{DUPLICATE}(\text{Body}(L))$ 
4:    $P_{Merged} \leftarrow P_{Merged} \cup G_i$ 
5:    $\text{ADD-TO-LOOP-PREDECESSORS}(G_i, L)$ 
6:   while  $i < \text{Max}(L)$  do
7:      $G_{i+1} \leftarrow \text{DUPLICATE}(G_i)$ 
8:      $P_{Merged} \leftarrow P_{Merged} \cup G_{i+1}$ 
9:      $e_{next} = (\text{Exit}(G_i), \text{Entry}(G_{i+1}), \text{Cond}(L))$ 
10:     $\text{ADD-LINK}(P_{Merged}, e_{next})$   $\triangleright$  Add link to  $P_{Merged}$ 
11:    for all  $e_{succ} \in E^{\leftarrow}(L)$  do
12:       $e_{skip} = (\text{Exit}(G_i), \pi_2(e_{succ}),$ 
13:         $(\pi_3(e_{succ}) \wedge \neg \text{Cond}(L)))$ 
14:       $\text{ADD-LINK}(P_{Merged}, e_{skip})$ 
15:     $i \leftarrow i + 1$ 
16:  end while
17:   $\text{RELATE-TO-LOOP-SUCCESSORS}(G_{i-1}, L)$ 
18: end procedure

```

Algorithm 2: Graph Duplication.

```

1: function DUPLICATE( $G$ )
2:    $A = \Pi_1(G)$   $\triangleright$  Original activities
3:    $A' = \{\}; E' = \{\}$   $\triangleright$  Activity and link copies
4:   for all  $a \in A$  do
5:      $a' = \text{DUPLICATE-ACTIVITY}(a)$ 
6:      $A' \leftarrow A' \cup a'$ 
7:   end for
8:   for all  $a' \in A'$  do
9:      $a = \text{GET-ORIGIN}(a')$   $\triangleright$  Get original activity
10:    for all  $e_{in} \in E^{\rightarrow}(a)$  do
11:       $E' \leftarrow E' \cup \text{ADD-LINK}(\pi_1(e_{in}), a', \pi_3(e_{in}))$ 
12:    end for
13:    for all  $e_{out} \in E^{\leftarrow}(a)$  do
14:       $E' \leftarrow E' \cup \text{ADD-LINK}(a', \pi_2(e_{out}), \pi_3(e_{out}))$ 
15:    end for
16:  end for
17:  return  $G' = (A', E')$ 
18: end function

```

Variations. For resolving cross-boundary violations between two interacting static activity-based loops $L1$ and $L2$, both loops have to be unrolled. The order in which $L1$ and $L2$ are unrolled by Algorithm 1 is

not relevant because the loop unrolling technique neither considers nor changes the structure of the other loop. Also if $L1$ and $L2$ have a different number of maximal iterations the unrolling can be performed as the cross-boundary links are duplicated for each iteration. Which, in turn, keeps the control flow relations between each unrolled iteration of $L1$ and $L2$.

A dynamic activity-based loop $L1$ without synchronization activities on alternative paths in its loop body can be unrolled, if it interacts with a static activity-based loop $L2$ that does not have synchronization activities on alternative paths either. The number of unrolled iterations of $L1$ is implied by the maximum number of iterations of $L2$. Each execution of a synchronization activity a from $L1$ connected to a synchronization activity b from $L2$ via a link $e_{cb}(a, b, true)$ results in an execution of b and vice versa. Hence, the maximum number of iterations of $L2$ implies the maximum number of iterations of $L1$.

Algorithm 3: Add Loop Predecessors.

```

1: procedure ADD-TO-LOOP-PREDECESSORS( $G, L$ )
2:   for all  $e_{pred} \in E^{\rightarrow}(L)$  do
3:     if  $eval_C(L) = pre$  then
4:        $e_{entry} = (\pi_1(e_{pred}), Entry(G),$ 
5:                  $(\pi_3(e_{pred}) \wedge Cond(L) = true))$ 
6:       ADD-LINK( $P_{Merged}, e_{entry}$ )
7:       for all  $e_{succ} \in E^{\leftarrow}(L)$  do
8:          $e_{pre} = (\pi_1(e_{pred}), \pi_2(e_{succ}),$ 
9:                  $(\pi_3(e_{pred}) \wedge \pi_3(e_{succ}) \wedge \neg Cond(L)))$ 
10:        ADD-LINK( $P_{Merged}, e_{pre}$ )
11:       end for
12:     else
13:        $e_{entry} = (\pi_1(e_{pred}), Entry(G), \pi_3(e_{pred}))$ 
14:       ADD-LINK( $P_{Merged}, e_{entry}$ )
15:     end if
16:   end for
17: end procedure

```

Algorithm 4: Add Loop Successors.

```

1: procedure ADD-TO-LOOP-SUCCESSORS( $G, L$ )
2:   for all  $e_{succ} \in E^{\leftarrow}(L)$  do
3:      $e_{exit} = (Exit(G), \pi_2(e_{succ}),$ 
4:               $(\pi_3(e_{succ}) \wedge \neg Cond(L)))$ 
5:     ADD-LINK( $P_{Merged}, e_{exit}$ )
6:   end for
7: end procedure

```

Discussion. The consecutive execution of the iterations of an unrolled loop L is emulated by duplicating the loop body of L n times to the subgraphs

G_L^1, \dots, G_L^{n-1} and by linking these subgraphs sequentially. The control relations within the activities of the duplicated loop bodies are preserved as no new activities or control links are introduced in G_L^i . The behavior, that no further iteration of L is performed when its loop condition evaluates to *false*, is emulated by the set of additional control links E_{skip} connecting the exit activity of each unrolled subgraph G_L^i with the successor activities of L .

The loop unrolling technique keeps also the control flow relations between the activities of $L1$ and $L2$ implied by C as the links E_{CB} are also duplicated. Assume, for instance, that $L1$ in Fig. 2 can be iterated up to six times and $L2$ only up to two times. Hence, $L1$ has to be unrolled six times and $L2$ two times. If C is deadlock free, the path $\langle a2, a3, a5 \rangle$ in $L1$ must be taken exactly in two iterations to execute $b2$ and $b3$. However, it can not be determined at design time, which iterations take this path. The duplication of the cross-boundary link e_{CB} ensures, that taking this path is possible in each unrolled iteration G_{L1}^i . At the same time, multiple executions of the same activity are avoided by using the flag that tracks, if a copy of a cross-boundary link was already performed. This also implies that a target activity of one or many cross-boundary links does not run into deadlocks. If it ran into a deadlock, none of the source activities of their incoming cross-boundary link is performed. As each cross-boundary link represents a former message exchange in C , this, in turn, would mean that the target activity would wait forever for an incoming message. Hence, C would not be deadlock free which contradicts to our prerequisite.

Pattern 2: Transforming Activity-based Loops to Graph-based Structured Loops

Context. A dynamic activity-based loop L is related to another static or dynamic graph-based structured loop via a set of cross-boundary links as shown in Fig. 3. The workflow language supports graph-based structured loops.

Solution. To resolve the cross-boundary violations, L can be transformed to a graph-based structured loop L_G by copying the loop body G_L of L to P_{Merged} and by creating a control flow cycle between the exit and entry activity of G_L . Algorithm 5 describes the transformation in detail.

The actual duplication of the loop body of L is done in line 2. To realize the repetitive execution of G_L a control flow cycle between the exit activity a_{exit} and the entry activity a_{entry} is created in line 5 by adding the new control link e_{loop} . For instance in Fig. 3, the

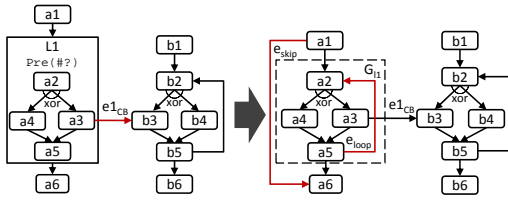


Figure 3: Activity-based Loops to Graph-based Structured Loops.

link e_{loop} connects the exit activity $a5$ with the entry activity $a2$. To incorporate G_L into the process graph of P_{Merged} Algorithm 5 calls Algorithm 3 and 4.

As described in pattern 1, Algorithm 3 is used to ensure that the set of direct predecessor activities of L become predecessors of the entry activity of G_L by creating a set of entry links (denoted as e_{entry}). This means, that if L is a post-test loop, the entry links inherit the link conditions of the original entry links of L . If L has a pre-test condition, it has to be guaranteed, that the first iteration of G_L is only performed if the loop condition evaluates to *true*, otherwise G_L must be skipped.

Algorithm 4 is called, to relate the direct successor activities of L with the exit activities G_L by creating for each outgoing link of L ($e \in E^-(L)$) a corresponding exit link e_{exit} . To start the successor activities if and only if all iterations of G_L completed, the link condition of each exit link is concatenated with a negation of the loop condition of L .

Algorithm 5: Loop Transformation.

- 1: **procedure** LOOP-TRANSFORM(L)
 - 2: $G_L \leftarrow \text{DUPLICATE}(\text{Body}(L))$
 - 3: $P_{Merged} \leftarrow P_{Merged} \cup G_L$
 - 4: $\text{ADD-TO-LOOP-PREDECESSORS}(G_L, L)$
 - 5: $e_{loop} = (\text{Exit}(G_L), \text{Entry}(G_L), \text{Cond}(L))$
 - 6: $\text{ADD-LINK}(P_{Merged}, e_{loop})$
 - 7: $\text{ADD-TO-LOOP-SUCCESSORS}(G_L, L)$
 - 8: **end procedure**
-

Variations. Transforming a static structured loop L to an unstructured loop instead of unrolling it (as described in Sect. 4), may be useful if L has a high maximum of iterations. This avoids P_{Merged} to be “polluted” with unrolled iterations of L .

Discussion. Transforming an activity-based loop L to a graph-based structured loop L_G removes the loop boundaries of L while preserving all control flow constraints implied by C . The control link e_{loop} enables iterations of the loop body G_L to be consecutively executed, as long as the loop condition evaluates to *true*. As the loop condition is not changed, L_G is iterated

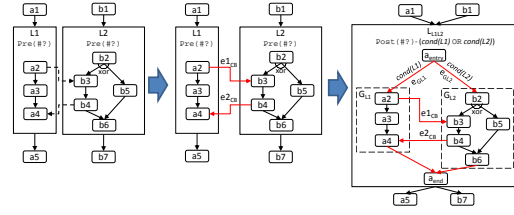


Figure 4: Merging Two Dynamic Activity-based Loops.

as often as L (under the same data assignment). If the loop condition evaluates to *false*, link e_{loop} is deactivated and the exit links $E^-(exit)$ are activated. This ensures the original behavior, that the successors of L are started after all iterations of L completed. The original control relations between $L1$ and other loops are also preserved since the cross-boundary links between are not changed either.

Pattern 3: Merging Two Dynamic Activity-based Loops

Context. A dynamic activity-based loop $L1$ is related to another dynamic activity-based loop $L2$ via a set of cross-boundary links as shown in Fig. 4. The workflow language does not support graph-based structured loops.

Solution. Loop unrolling cannot be performed as the number of iterations of $L1$ and $L2$ is unknown at design time. To resolve the violations, the source and target activities of these links are moved into the same loop, referred to as L_{L1L2} . Thus, the activity graphs G_{L1} and G_{L2} have to be merged into a new single loop L_{L1L2} as shown in Algorithm 6.

The algorithm creates the new loop L_{L1L2} in line 4. The loop body G_{L1L2} of this loop, which consists of the loop bodies of $L1$ and $L2$, is created in line 5. Additionally, an entry activity a_{entry} and an exit activity a_{exit} is added to the loop body. These activities precede and succeed the entry and exit activities of G_{L1} and G_{L2} . They are added to keep the SESE property and to ensure that G_{L1} is only performed, if the loop condition of $L1$ becomes *true* and that G_{L2} is only performed, if the loop condition of $L2$ becomes *true*. For this purpose, the transition condition of the entry link pointing from a_{entry} to G_{L1} gets the loop condition of $L1$ assigned and the entry link pointing from a_{entry} to G_{L2} gets the join condition of $L2$ assigned. Moreover, the link condition of the entry link is conjoined by an additional flag $firstIt \mapsto \{true, false\}$ if the loop body G_{L1} or G_{L2} originates from a post-test loop (lines 10 and 13). If the first iteration of L_{L1L2} is executed the flag is set to *true*, otherwise it is set to *false*. For instance, in Fig. 4 the entry link for e_{GL2} would be

defined as $(a_{entry}, b2, \text{Cond}(L2) \vee 'firstIt = true')$. The value of the flag is hold in the variable *firstIt* that is declared in P_{Merged} (line 8). After the first iteration *firstIt* must be set to *false* (not depicted in Algorithm 6).

L_{L1L2} is always a post-test loop (see below), no matter if $L1$, $L2$ or both are pre-test loops. The loop condition of L_{L1L2} is set to the logical disjunction of the loop conditions of $L1$ and $L2$ (line 16). This ensures that G_{L1L2} is also executed if the condition of $L1$ evaluates to *false* while the condition of $L2$ evaluates to *true* (or vice versa). In Fig. 4, for instance, under a certain data assignment $L1$ may iterate two times and $L2$ five times. To emulate this behavior, L_{L1L2} must be iterated five times which ensured by its loop condition. Since the link condition of the entry link $(a_{start}, a2, \text{Cond}(L1))$ is set to the loop condition of $L1$ its body G_{L1} is just performed twice. L_{L1L2} is wired into P_{Merged} by connecting it to the predecessor and successor activities of $L1$ and $L2$ (lines 17 and 18).

Algorithm 6: Merging Activity-based Loops.

```

1: function MERGE-LOOPS( $L1, L2$ )
2:    $G_{L1} \leftarrow \text{DUPLICATE}(\text{Body}(L1))$ 
3:    $G_{L2} \leftarrow \text{DUPLICATE}(\text{Body}(L2))$ 
4:    $L_{L1L2} = \text{ADD-LOOP}(\emptyset, \emptyset, \text{eval}_C \leftarrow \text{post})$ 
5:    $\text{Body}(L_{L1L2}) \leftarrow G_{L1} \cup G_{L2} \cup \{a_{exit}\},$ 
      $\{(\text{Exit}(G_{L1}), a_{exit}, \text{true}), (\text{Exit}(G_{L2}), a_{exit}, \text{true})\}$ 
6:    $\text{condEntry}_{L1} \leftarrow \text{Cond}(L1)$ 
7:    $\text{condEntry}_{L2} \leftarrow \text{Cond}(L2)$ 
8:   DECLARE( $P_{Merged}, firstIt$ )  $\triangleright$  Adds variable
9:   if  $\text{eval}_C(L1) = \text{post}$  then
10:     $\text{condEntry}_{L1} \leftarrow \text{condEntry}_{L1} \vee 'firstIt = true'$ 
11:   end if
12:   if  $\text{eval}_C(L2) = \text{post}$  then
13:     $\text{condEntry}_{L2} \leftarrow \text{condEntry}_{L2} \vee 'firstIt = true'$ 
14:   end if
15:    $\text{Body}(L_{L1L2}) \leftarrow \text{Body}(L_{L1L2}) \cup \{a_{entry}\},$ 
      $\{(a_{entry}, \text{Entry}(G_{L1}), \text{condEntry}_{L1}),$ 
      $(a_{entry}, \text{Entry}(G_{L2}), \text{condEntry}_{L2})\}$ 
16:    $\text{Cond}(L_{L1L2}) \leftarrow (\text{Cond}(L1) \vee \text{Cond}(L2))$ 
17:    $E^{\rightarrow}(L_{L1L2}) \leftarrow E^{\rightarrow}(L1) \cup E^{\rightarrow}(L2)$ 
18:    $E^{\leftarrow}(L_{L1L2}) \leftarrow E^{\leftarrow}(L1) \cup E^{\leftarrow}(L2)$ 
19:   return  $L_{L1L2}$ 
20: end function
```

Variations. If a dynamic activity-based loop, with synchronization activities on alternative paths in its loop body, is related to a static activity-based loop, its number of iterations cannot be determined from the max. number of iterations of the static loop (refer to pattern 1). Hence, the dynamic loop cannot be unrolled and has to be merged with the static loop in the same way as described in the solution.

Discussion. Merging $L1$ and $L2$ into L_{L1L2} keeps the original control flow order between the activities of

G_{L1} and G_{L2} as the control links are not changed. The loop condition of L_{L1L2} and the link conditions of the entry links ensure that the same number of iterations of G_{L1} and G_{L2} are executed as in C (under the same data assignment). However, the activities of G_{L1} and G_{L2} become *iteration-dependent* on each other, i. e., another iteration $i + 1$ of the activities in G_{L1} cannot be performed until all activity iterations i in G_{L2} completed (and vice versa). This becomes especially an issue, if the execution times of the activities within G_{L1} and G_{L2} are very different from each other. It also postpones the execution of the successor activities of L_{L1L2} . For instance, in 4 *a5* cannot be started until all iterations of L_{L1L2} completed, i. e., compared to C its execution is postponed until all iterations of G_{L1} completed. Note, that the postponed execution resulting from the loop merge may increase the time until the business outcome is reached, but it does not affect the overall completion time of P_{Merged} compared to C . The successful completion of all activities of an instance of P_{Merged} takes as long as completing all activities of C (assuming the same data are used for an instance of P_{Merged} and C).

Combining the Patterns

In the following two algorithms are proposed that make use of the patterns to solve cross-boundary violations in P_{Merged} . If graph-based structured loops are supported by the workflow language, the set of all dynamic or static activity-based loops being source or target of a cross-boundary link (denoted as \mathcal{L}_{CB}) are transformed into graph-based structured loops by Algorithm 7. The transformation preserves all sets of basic activity traces implied by the original loop and it just adds the two new links e_{skip} and e_{loop} to P_{Merged} . The runtime of the algorithm is $O(n)$, where $n = |\mathcal{L}_{CB}|$.

Algorithm 7: Transformation to Graph-based Loops.

```

1: procedure SOLVE-CB-VIOLATIONS( $\mathcal{L}_{CB}$ )
2:   for all  $L_{CB} \in \mathcal{L}_{CB}$  do
3:      $\text{LOOP-TRANSFORM}(L_{CB})$ 
4:      $\text{REMOVE-ACT}(P_{Merged}, L_{CB})$ 
5:   end for
6: end procedure
```

If graph-based loops are not supported Algorithm 8 must be applied. It tries to unroll as many loops within \mathcal{L}_{CB} as possible (pattern 1). All loops that cannot be unrolled, i. e., all interacting dynamic activity-based loops and all static activity-based loops that interact with dynamic activity-based loops, are merged into activity-based loops (pattern 3). Applying pattern 1 decreases the readability of P_{Merged} as it may signif-

icantly increase the number of activities and control links in P_{Merged} . Especially if a large number of iterations is unrolled. However, as pattern 3 causes the iteration-dependency issue, pattern 1 is always preferred to pattern 3.

Algorithm 8: Merging and Unrolling.

```

1: procedure SOLVE-CB-VIOLATIONS( $E_{CB}, \mathcal{L}_{CB}$ )
2:   for all  $L1_{CB} \in \mathcal{L}_{CB} \mid \text{MAX}(L1_{CB}) = \perp$  do
3:      $A_L = \Pi_I(\text{BODY}(L1_{CB}))$ 
4:     for all  $e_{CB} \in E_{CB}$ 
5:        $\mid (\pi_I(e_{CB}) \cup \pi_2(e_{CB})) \cap A_L \neq \emptyset$  do
6:          $L1_{CB} = \text{PARENT}(\pi_I(e_{CB}))$ 
7:          $L2_{CB} = \text{PARENT}(\pi_2(e_{CB}))$ 
8:          $L_{merged} = \text{MERGE-LOOPS}(L1_{CB}, L2_{CB})$ 
9:          $\text{ADD-ACT}(P_{Merged}, L_{merged})$ 
10:         $\mathcal{L}_{CB} \leftarrow (\mathcal{L}_{CB} \cup L_{merged}) - (L1_{CB} \cup L2_{CB})$ 
11:         $A_{L1} = \Pi_I(\text{BODY}(L1_{CB}))$ 
12:         $A_{L2} = \Pi_2(\text{BODY}(L2_{CB}))$ 
13:         $E_{CB}^{L1L2} = \{e_{CB}^{L1L2} \mid \forall e_{CB}^{L1L2} \in E_{CB} : \\
          ((A_{L1} \cup A_{L2}) \cap \pi_I(e_{CB}^{L1L2})) \neq \emptyset \\
          \wedge ((A_{L1} \cup A_{L2}) \cap \pi_2(e_{CB}^{L1L2})) \neq \emptyset\}$ 
14:         $E_{CB} \leftarrow E_{CB} - E_{CB}^{L1L2}$ 
15:         $\text{REMOVE-ACT}(P_{Merged}, L1_{CB})$ 
16:         $\text{REMOVE-ACT}(P_{Merged}, L2_{CB})$ 
17:         $L1_{CB} \leftarrow L_{merged}$ 
18:         $A_L = \Pi_I(\text{BODY}(L1_{CB}))$ 
19:      end for
20:       $\mathcal{L}_{CB} \leftarrow \mathcal{L}_{CB} - L_{merged}$ 
21:    end for
22:    for all  $e_{CB} \in E_{CB}$  do
23:       $a_{L1} = \pi_I(e_{CB}); a_{L2} = \pi_2(e_{CB})$ 
24:       $L1_{CB} = \text{PARENT}(a_{L1})$ 
25:       $L2_{CB} = \text{PARENT}(a_{L2})$ 
26:       $\text{UNROLL-LOOP}(L1_{CB})$ 
27:       $\text{UNROLL-LOOP}(L2_{CB})$ 
28:       $\mathcal{L}_{CB} \leftarrow \mathcal{L}_{CB} - (L1_{CB} \cup L2_{CB})$ 
29:    end for
30: end procedure

```

Algorithm 7 is trivial and not further discussed. Algorithm 8 is explained by using the example scenario in Fig. 5. In this scenario P_{Merged} contains four dynamic and two static activity-based loops. The control flow materialization created six cross-boundary links. To resolve the violations the algorithm is called with the parameters E_{CB} and \mathcal{L}_{CB} . Thereby, the set E_{CB} denotes the set of cross-boundary links, here $e1_{CB}$ to $e4_{CB}$. \mathcal{L}_{CB} contains those loops of P_{Merged} that are source or target of one or more cross-boundary links, i. e., in the scenario $L1$ to $L4$. In a first step, those loops within \mathcal{L}_{CB} that are related to a dynamic activity-based loop via a cross-boundary link are merged. For this purpose the algorithm selects a dynamic activity-based

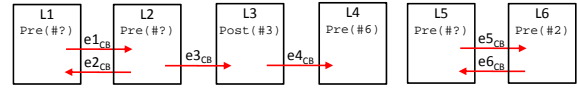


Figure 5: Multiple Interacting Activity-based Loops.

loop from \mathcal{L}_{CB} (line 4), e. g., $L1$. All loops containing activities that are related to activities within the selected loop via a cross-boundary link (line 4) are pairwise merged with the selected loop. For instance, $L1$ is first merged with $L2$ and the resulting loop L_{merged} is added to P_{Merged} (line 9) while $L2$ and $L1$ are removed from P_{Merged} (lines 15 and 16). As the cross-boundary violations between the merged loops are resolved, all cross-boundary links between them are removed from the set E_{CB} in line 13 (but not from P_{Merged}). Hence, after $L1$ and $L2$ were merged $e1_{CB}$ and $e2_{CB}$ are removed. Then all loops that are related to the merged loop via cross-boundary links are merged with L_{merged} , i. e., in our example the static loop $L3$ is merged with L_{merged} . The new merged loop consisting of the loop bodies of $L1$, $L2$ and $L3$ is then, in turn, merged with $L4$. As this new loop has no cross-boundary links to other loops, it is removed from the set \mathcal{L}_{CB} (line 20) and another dynamic loop whose activities are source or target of cross-boundary links is selected (if any). In our example this would be $L5$ or $L6$, which are also merged with each other.

After the dynamic loops were merged with other dynamic or static loops, P_{Merged} contains only cross-boundary links between static loops. These loops are unrolled and added to P_{Merged} . Our example does only contain static loops that are transitively connected to dynamic loops. Hence, no loop unrolling is performed here. The runtime of the algorithm is also $O(n)$ (where $n = |\mathcal{L}_{CB}|$), even though it has two nested for-loops. However, the for-loop in line reduces the iterations of its parent for-loop (line) by removing cross-boundary loops from the set \mathcal{L}_{CB} .

5 VALIDATION: CONSOLIDATION OF BPEL PROCESSES WITH INTERACTING LOOPS

To validate the process consolidation approach we developed a tool (Dadashov, 2013) that merges interacting BPEL processes being part of a BPEL4Chor choreography (Decker et al., 2009) into a single process. So far, the prototype was not capable to merge BPEL processes with interacting loops. To support these interaction scenarios, we applied the patterns of Sect. 4 to BPEL and extended the prototype ac-

cordingly. The prototype gets a ZIP file as input, that contains an XML representation of BPEL4Chor choreography along with the processes to be merged. To perform the consolidation, the prototype creates P_{Merged} and copies all activities of the processes within the BPEL4Chor choreography into P_{Merged} . Based on the message links and the communication activities in the BPEL4Chor choreography the prototype performs the control flow materialization.

5.1 Control Flow Semantic of BPEL

Besides control links BPEL uses join conditions to determine if an activity can be started. A join condition specifies which links have to be activated to start this activity. This requires the status of all links to be known before the join condition is evaluated. Moreover, BPEL is using a dead path elimination control flow semantics (DPE) that determines all activities in the control flow that cannot be executed anymore. Due to the DPE semantics and the fact that all links of an activity have to be evaluated before it can be started, BPEL does not support graph-based loops.

To enable repetitive execution of activities, BPEL offers three types of activity-based loops, the pre-test loops `while` and `forEach` and the post-test loop `repeatUntil`. `while` and `repeatUntil` loops are defined in the same way as the activity-based loops defined in Sect. 3, i.e., they consist of a Boolean loop condition and an activity graph in their loop body. For simplicity reasons, we assume that the loop body is a SESE graph, even though this not stipulated by BPEL. The `forEach` activity has no Boolean expression as loop condition but a `From` and `To` attribute, representing the start and end value of the iteration counter. The attribute `Counter` provides the name of the counter variable that is increased by one in each iteration. The attribute values can be determined at runtime but they must be constant during the execution of the `forEach`. All iterations have to be executed in order to complete the `forEach` loop. The `completionCondition` attribute for modeling at-least-n-out-of-m semantics is not considered here.

5.2 Applying the Patterns to BPEL

This section describes how patterns 1 and 3 from Sect. 4 are applied to BPEL. Pattern 2 is not further considered here as BPEL does not support graph-based structured loops.

Activity-based Loop Unrolling. To determine the maximum iterations of a BPEL loop the data flow analysis techniques introduced by Heinze et al. (Heinze

et al., 2012) are used. However, the pattern cannot be applied to arbitrary static activity-based loops. Because BPEL's link semantic requires that all incoming links of an activity are evaluated before it is started. However, if a loop is unrolled also the cross-boundary links E_{CB} are duplicated. This results in n multiple copies of e_{CB} targeting the same activity, where each copy has its source in one of the unrolled iterations G_L^1, \dots, G_L^n of the loop L . Hence, the source activity of each copy of e_{CB} must be performed before the target activity of e_{CB} can be executed. In Fig. 2, for instance, activity $b2^1$ is not started until $a3^1$ and $a3^2$ completed. In the example, this just leads to a postponed execution of $b2^1$ compared to C where the first iteration of $a3$ can complete after the first iteration of $a3$ is performed (if we assume that messages are instantly delivered). If $L2$ would contain an activity that is source of a link e_{2CB} targeting an activity within the unrolled loop $L1$, this would even lead to a deadlock.

This issue is circumvented by ensuring that the activities A_{L1} and A_{L2} in the unrolled loops G_{L1} and G_{L2} have at most one incoming cross-boundary link e_{CB}^i :

$$\forall a \in A_{L1} \cup A_{L2} : |(E^{\rightarrow}(a) \cup E^{\leftarrow}(a)) \cap E_{CB}| \leq 1$$

This, in turn, requires the source and target activities of E_{CB} to be performed during each iteration. Thus, there must be no potential alternative paths in $L1$ or $L2$ preventing synchronization activities from being executed during an iteration of $L1$ or $L2$:

$$\begin{aligned} \forall a_{L1} \in A_{L1} : (E^{\rightarrow}(a_{L1}) \cup E^{\leftarrow}(a_{L1})) \in E_{CB} : \\ \text{PreDom}(a_{L1}) \cup \text{PostDom}(a_{L1}) = A_{L1} \end{aligned}$$

$$\begin{aligned} \forall a_{L2} \in A_{L2} : (E^{\rightarrow}(a_{L2}) \cup E^{\leftarrow}(a_{L2})) \in E_{CB} : \\ \text{PreDom}(a_{L2}) \cup \text{PostDom}(a_{L2}) = A_{L2} \end{aligned}$$

If the aforementioned prerequisite is fulfilled, a copy of a cross-boundary link has to be connected to a source and target activity part of the subgraph G_{L1}^i and G_{L2}^i in the same iteration, i.e., the source and target activities must dominate the same number of sync. activities A_{syn} :

$$\begin{aligned} \forall e_{CB}^i \in E_{CB} : \\ & |\text{PreDom}(\pi_1(e_{CB}^i)) \cap A_{CB}| \\ &= |\text{PreDom}(\pi_2(e_{CB}^i)) \cap A_{CB}| \\ A_{CB} &= \bigcup_{e \in E_{CB}} \pi_1(e) \cup \pi_2(e) \end{aligned}$$

The set A_{CB} denotes the set of activities of $L1$ and $L2$ having an incoming or outgoing cross-boundary link.

Figure 6 shows a `while` loop $L1$ ($\text{Max}(L1) = 2$) interacting with a `repeatUntil` loop $L2$ ($\text{Max}(L2) = 2$) that meet the aforementioned properties (in the figure

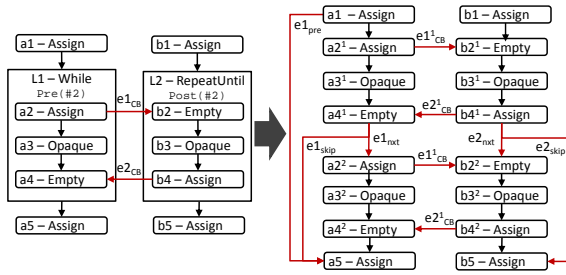


Figure 6: Unrolling Two Static BPEL Loops.

the opaque activities encapsulate some business logic). As there is no alternative flow in $L1$ and $L2$ they have to exchange messages during each iteration. This also implies that they always perform the same number of iterations. Hence, the copies of the cross-boundary links ($e1_{CB}^1$, $e1_{CB}^2$ and $e2_{CB}^1$, $e2_{CB}^2$) between the unrolled loop G_{L1} and G_{L2} must only connect activities from the same iterations. As shown in Figure 6, while or repeatUntil loops are unrolled in the same way as described in the pattern, i.e., the loop bodies are unrolled and added to the container activity of $L1$ and $L2$ (not depicted here). The set of links E_{next} , E_{skip} , E_{entry} , E_{exit} connects the unrolled iterations with each other and with the direct predecessor and successor activities of L . The link e_{pre} has to be only added for the pre-test loops while or forEach. In contrast to the pattern, for an unrolled forEach loop no loop condition is assigned to e_{next} as a sequential forEach cannot be interrupted, i.e., the maximum number of iterations is always performed. Thus, the set of links E_{skip} is not required either.

Merging Two Dynamic Activity-based Loops. If the interacting loops $L1$ and $L2$ are while loops the pattern can be directly applied. Then L_{LIL2} is also a while loop whose loop condition is the disjunction of the loop conditions of $L1$ and $L2$. The loop body of L_{LIL2} is created as described in the pattern. If one of the entry links of G_{L1} or G_{L2} evaluates to *false*, DPE ensures that the activities within G_{L1} or G_{L2} are not performed.

If one of the loops $L1$ or $L2$ is a repeatUntil loop the variation of the pattern for post-test loops has to be applied, i.e., L_{LIL2} must be also a repeatUntil loop.

If a forEach loop interacts with a while loop the merged loop L_{LIL2} must be a while loop because the loop condition of a forEach loop cannot specify complex logical expressions such as disjunctions. To specify the loop condition of L_{LIL2} , the interval defined by the From and To attributes of the forEach is transformed to a logical expression on the counter variable. In a forEach the counter variable is automatically increased with each iteration. This has to be emulated in

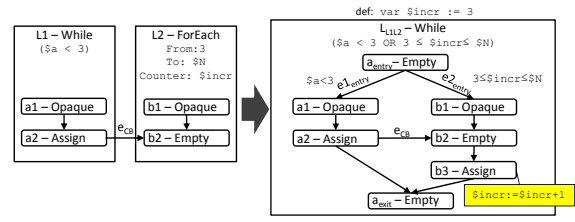


Figure 7: Merging While Interacting with ForEach.

L_{LIL2} by defining the counter variable before L_{LIL2} is started and by initializing it with the value of the From attribute. The counter can be increased by using an assign activity that increments the counter when L_{LIL2} completes. Figure 7 shows how the while loop $L1$ and the forEach loop $L2$ are merged into L_{LIL2} . The counter variable *incr* is initialized with the value 3 from the From attribute. The loop condition of the forEach is transformed to the expression $3 \leq \$incr \leq N$ and forms a disjunction with the loop condition of $L1$. assign *b3* increments the counter variable *incr* at the end of the L_{LIL2} . As the name of the counter variable is kept the activities of the loop body of the former forEach accessing the counter variable do not have to be adapted. Analogously, L_{LIL2} must be a repeatUntil if the forEach interacts with a repeatUntil loop.

For two interacting forEach loops $L1$ and $L2$ the values of the From and To attributes may be unknown at design time and the name of the counter variables used may be different. Hence, they cannot be merged into a forEach loop as only one counter variable can be declared there. Instead they can be also merged into a while. Thereby, two counter variables have to be defined before L_{LIL2} , one for counting the iterations of G_{L1} and another one for counting the iterations of G_{L2} . In BPEL, a single assign activity can perform multiple assignments, i.e., an assign following G_{L1} and G_{L2} can iterate both counter variables.

6 RELATED WORK

Existing approaches focus on merging semantically equivalent processes, which is different from our approach that merges *complementing* processes into a single process. For instance, Küster et al. (Küster et al., 2008) discuss how different variants of the same original process can be merged into a single process by employing change logs. Mendling and Simon (Mendling and Simon, 2006) describe an approach for merging Event Driven Process Chains (EPC) (Scheer et al., 2005) where semantically equivalent elements of an EPC have to be defined manually and based on this semantic mapping, the EPCs are merged.

Loop unrolling and merging is widely discussed in the area of compiler theory (Muchnick, 1997), especially, for optimizing parallel or embedded systems (Qian et al., 2002). Similar to our approach, in these works also loop conditions have to be combined and data analysis has to be performed to determine if a loop can be unrolled or merged (Darte, 1999). In contrast to our approach loop unrolling and merging in this context is employed for optimizing the runtime of short running programs in embedded systems and the language constraints of a programming language are different from those of a workflow language.

Kiepuszewski et al. (Kiepuszewski et al., 2013) also discuss activity and graph-based loops. However, in their work they focus on the transformation of graph-based loops into activity-based loops to structure unstructured workflows.

We investigated how cross-boundary link violations can be solved in parallel BPEL `forEach` loops (Wagner et al., 2014). But there we neither considered sequential `forEach` activities nor other BPEL loop types and the approach described there focuses only on BPEL workflows.

7 CONCLUSION AND OUTLOOK

In this work we extended the existing approach to support the automatic consolidation of processes interacting via activity-based loops. The focus was turned on activity-based loops, as the boundaries of these loops must not be crossed by the control links created by the control-flow materialization. To be universally applicable, the patterns for merging the loops were described independently of a concrete workflow language and different types of loops were considered that might be supported by a workflow language. All merge patterns keep the originally modeled execution order between the basic activities of the processes to be merged. However, when two dynamic activity-based loops are merged (pattern 3), additional control flow constraints are implicitly added to the merged process. The new constraints adhere to execution order defined the choreography but they may increase the time until a business outcome is reached. Hence, if the workflow language supports graph-based loops, interacting dynamic activity-based loops should be always transformed to graph-based structured loops. This also prevents the resulting process from getting too complicated in terms of number of activities and links (e. g., if loop unrolling is performed).

We also discussed the applicability of the patterns to the executable workflow language BPEL. As BPEL has a different control flow semantics compared to

the workflow meta-model, we used to describe the patterns, the loop unrolling pattern had to be restricted in order to avoid deadlocks in BPEL processes.

In future works we have to investigate how to solve the link violations for activity-based loops interacting with activities in an acyclic graph. We also have to discuss how nested activity-based loops have to be treated. This includes the description of a formal algorithm that applies the patterns to activity-based loops interacting with several (nested) loops.

ACKNOWLEDGEMENTS

This work was partially funded by the BMBF project ECHO (01XZ13023G) and the BMWi project NE-MAR (03ET40188).

REFERENCES

- Dadashov, E. (2013). Choreography-based Business Process Consolidation in One-To-Many interactions. Master thesis, University of Stuttgart.
- Darte, A. (1999). On the complexity of loop fusion. In *Parallel Architectures and Compilation Techniques, 1999. Proceedings. 1999 International Conference on*, pages 149–157.
- Decker, G., Kopp, O., Leymann, F., and Weske, M. (2009). Interacting services: From specification to execution. *Data & Knowledge Engineering*, 68(10):946–972.
- Heinze, T., Amme, W., and Moser, S. (2012). Control flow unfolding of workflow graphs using predicate analysis and SMT solving. In *ZEUS*.
- Kiepuszewski, B., ter Hofstede, A. H. M., and Bussler, C. (2013). On structured workflow modelling. In *Seminal Contributions to Information Systems Engineering*, pages 241–255.
- Koehler, J., Hauser, R., Sendall, S., and Wahler, M. (2005). Declarative techniques for model-driven business process integration. *IBM Systems Journal*, 44(1):47–65.
- Kopp, O., Khalaf, R., and Leymann, F. (2008). Deriving Explicit Data Links in WS-BPEL Processes. In *IEEE International Conference on Services Computing*. IEEE.
- Küster, J., Gerth, C., Förster, A., and Engels, G. (2008). A tool for process merging in business-driven development. In *Proceedings of the Forum at the CAiSE*.
- Mendling, J. and Simon, C. (2006). Business process design by view integration. In *BPM Workshops*. Springer.
- Muchnick, S. (1997). *Advanced Compiler Design and Implementation*. Morgan Kaufmann.
- Ng, A., Chen, S., and Greenfield, P. (2004). An Evaluation of Contemporary Commercial SOAP Implementations. In *AWSA*.
- OASIS (2007). *Web Services Business Process Execution Language Version 2.0 – OASIS Standard*.

- Object Management Group (OMG) (2011). *Business Process Model and Notation (BPMN) Version 2.0*. OMG Document Number: formal/2011-01-03.
- Qian, Y., Carr, S., and Sweany, P. H. (2002). Loop fusion for clustered vliw architectures. In *LCTES-SCOPES*, pages 112–119.
- Scheer, A.-W., Thomas, O., and Adam, O. (2005). *Process Aware Information Systems*, chapter Process Modeling Using Event-Driven Process Chains. Wiley-Interscience.
- van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, A. P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51.
- Völzer, H. (2010). A new semantics for the inclusive converging gateway in safe processes. In *BPM 2010*.
- Wagner, S., Kopp, O., and Leymann, F. (2012). Towards Verification of Process Merge Patterns with Allen’s Interval Algebra. In *ZEUS*, Bamberg. CEUR.
- Wagner, S., Kopp, O., and Leymann, F. (2014). Choreography-based Consolidation of Multi-Instance BPEL Processes. In *CLOSER*. SciTePress.
- Wagner, S., Roller, D., Kopp, O., Unger, T., and Leymann, F. (2013). Performance optimizations for interacting business processes. In *IC2E*. IEEE.

Key Requirements for Predictive Analytical IT Service Management

Architectural Key Characteristics for a Cloud based Realization

Christopher Schwarz, Hans Peter Bauer, Lukas Blödorn and Erwin Zinser

Institute of Information Management, FH Joanneu, Gesellschaft mbH, Alte Poststrasse 147, Graz, Austria
{christopher.schwarz, erwin.zinser}@fh-joanneum.at

Keywords: IT Service Management, Predictive Analytics, Business Analytics, Service Oriented Architecture, Service Bus, Semantic Web Technologies, Semantic Reasoning, Controlled Natural Language, Cloud Computing.

Abstract: While trying to maintain sustainable competitive advantage, IT service providers are challenged with tremendous service complexity and a low level of flexibility caused by the lack of transparency, constrained scalability and the missing ability to identify needed service measures proactively. For overcoming these challenges, this paper presents a well-evaluated set of identified key requirements for a feasible realization of a highly scalable cloud based architecture that supports predictive analytics in several domains of IT Service Management. This presented concept goes far beyond traditional approaches and pertinent state-of-the-art software solutions by focusing on business analyses based on knowledge creation and domain-independent knowledge sharing. The proposed approach is based on profound analyses of related work as well as modern service oriented design and business analyses paradigms. It provides semantic complexity handling, structured and multi-layered service interaction, cloud-enabled scalability management as well as predictive business analyses based on semantic reasoning, decision-making support and pattern recognition. The derived results eventually provide solution architects with a feasible and technical independent fundament for architectural implementation decisions. It ultimately enables IT service providers to cope with modern flexibility needs and complexity challenges and therefore to continuously satisfy customers to gain competitive advantage.

1 INTRODUCTION

A company's success fundamentally depends on its ability to develop sophisticated solutions, perfectly fitting to customers' requirements. Nevertheless, to guarantee sustainable success, bearing up customers' satisfaction is necessary, especially if requirements change. Hence, a flexible and agile service network is needed, adaptable to changes, but most of all adaptable to customers' requirements.

According to Porter's Five Forces, competitive rivalry in the IT area is significantly higher than in many other branches, primarily caused by a high threat of new entry and substitution (Fung, 2013, pp. 19,20). The internet and cloud computing eliminate many physical barriers of entering a new market (Fung, 2013, pp. 19,20). Due to the variety of IT services with similar functionality it is easy for customers to change to substitutes, whereas IT service providers have to deal with a high range of competitive companies, serving those customers (Fung, 2013, pp. 19,20). Thus, albeit it is comparatively easy to provide IT services, it requires

a lot of effort to keep up with others by continuously satisfying customers and managing their requirements not just reactively, but in a proactive and predictive way for effective decision-making.

The ability for successfully managing services over a long-time period distinguishes an effective from an ineffective service provider. The key to success is flexibility in IT Service Management (ITSM), by constantly aligning and adjusting the service offers to the actual market need. Managing services means managing a network of interdependent components that have to be completely coordinated. Disability of coping with multitenant service dependencies leads to missing transparency in the service environment, which affects calculation and billing of services and makes analytics for business relevant financial decisions extremely difficult, or even impossible (Schwarz, et al., 2013, p. 1).

The strong influence of ITSM on Financial Management indicates the need for balance and flexibility over multiple domains, which is still an unhandled problem of current software solutions

(Schwarz, et al., 2013, p. 1). IT is the most pervasive factor in business, affecting every single process and decision (Addy, 2007, p. 20). Therefore the discipline of ITSM is applicable for nearly all business domains relying on any kind of IT based service. Moreover, it is necessary to control all IT components from a holistic view, for constantly aligning the business strategy with the IT strategy, especially when environmental changes appear (vom Brocke, et al., 2013, p. 1).

Nevertheless, complexity is considered as the major challenge in ITSM. Complexity is referred to the management of service data and its relations (Benedettini and Neely, 2012, pp. 5,6). Interdependencies and data flows between services have to be determined easily and fast, so that possible effects of changes and events can be recognized ideally in real-time. Thus, efficient ITSM requires transparency of service-to-customer, as well as service-to-service interaction. Besides that, transparency is necessary between services and its underlying components, to ensure appropriate service functionality. In the same way profound knowledge of service relations is needed for defining and fulfilling cost-efficient service levels and consequently to ensure a particular Quality of Service (QoS) level. Constantly changing requirements necessitate QoS management techniques that are far beyond static provisioning of network resources (Kourtesis, et al., 2014, pp. 306,307). Higher elasticity and dynamic provisioning is needed, based on real-time decisions, reasoning and inference (Kourtesis, et al., 2014, pp. 306,307).

Second, the maintenance and continuous improvement of a service infrastructure, tightened influenced by IT outsourcing and cloud-oriented design, define a new level of complexity (Benedettini and Neely, 2012, pp. 5,6). Changing requirements often demand changing functionality, seamlessly integrated in the working service environment.

Up to a certain level, IT services seem to be manageable easily. With growing complexity, however, a point will be reached where the effort of complexity handling is higher than the services' benefit, leading to crucial cost inefficiency (Josuttis, 2007, p. 2). Consequently, the service infrastructure has to be designed and structured in a way that supports effective management of both mentioned complexity types on the one hand and enables seamless integration of new services on the other hand. The high level of complexity forces IT service providers to think of new ways for automatically managing a variety of data, variables and parameters, necessary for the operation of services and its

resources (Kourtesis, et al., 2014, p. 308).

2 RELATED WORK

In the past years, different approaches have been presented in the area of ITSM, all with the overall goal to enhance efficiency, primary by using the Semantic Web concept. They provide sophisticated solutions with a precise goal for dealing with complexity in IT Service Management.

It is beyond doubt that these solutions provide value in the very specific field they are used, but the question arising is, if they are adaptable to other correlating areas as well at an adequate level of effort. Referring back to the de-facto ITSM standard, the Information Technology Infrastructure Library (ITIL), IT Service Management is defined as the discipline to deal with all processes in the service lifecycle (van Bon, et al., 2007, pp. 24-26, 42). Efficient ITSM needs a holistic view, accomplishing a platform that allows making use of the Service Oriented Architecture (SOA) and analytical benefits in any ITSM domain. The ability of using semantic knowledge must not be limited to one specific implementation, but has to be realized on a shared base. The semantic Wiki-based approach of Kleiner et al. (Kleiner, et al., 2012) perfectly fits to the issue of complexity in Incident and Problem Management and thus, it is closely aligned to exactly these ITIL processes. Although the semantic Wiki allows the integration of other applications, the platform is still limited to the information stored in the semantic Wiki. Jantscher et al. (Jantscher, et al., 2014) focus on reducing negative business impacts caused by wrong incident prioritization. They developed an example for analytical ITSM, the Incident Prioritizer, which is an AHP decision support system for the prioritization of incidents based on their business impact. For the prioritization process, relevant incident data is provided by an ontology, defining an ITIL-compliant service catalogue. Valiente et al. (Valiente, et al., 2012) deal with the service management problem of integrating service management processes, which are often specified in natural language. The paper aims to translate ITSM relevant information, expressed in natural language, to a computer understandable format, by using semantic technologies. Thus, Onto-ITIL is presented in this work, as an OWL based ontology, tailored to be used in ITSM, to overcome the gap between natural language process definitions and IT services, or more generally the gap between business and IT. Otherwise, the very generic approach of Freitas et al. (Freitas, et al., 2008) defines

insufficient structure do be applicable in this stage for operative use. The idea of a generic ontology, usable for different domains, has definitely potential in theory, but not enough alignment to the ITIL processes for effective ITSM.

What is needed is an overall platform allowing the seamless integration of ITIL process solutions throughout the whole service lifecycle. The platform architecture has to be designed to support flexibility and to make services adaptable for changes. The approach of El-Gayar et al. (El-Gayar & Deokar, 2013) with its distributed model environment presents the key feature of using a service bus for enabling flexible changing of distributed models, based on specific problems.

Extending this approach of high flexibility and easy sharing and reusing of information to the field of ITSM, a platform can be realized that allows the integration of process solutions throughout the whole lifecycle and sharing and using knowledge in the whole environment.

3 RESEARCH QUESTIONS AND OBJECTIVES

When focusing on the implementation of existing concepts in the related work, the need for further development has been identified, missing a holistic approach for seamlessly integrating them. Thus, the presented approach in the upcoming sections tries to answer the following research questions:

- What are the key requirements for a scalable architecture to support predictive analysis in ITSM to be able to cope with the complexity of service interdependencies and heterogeneity as well as the lack of transparency?
- Is there a way for seamlessly integrating services and making use of provided functionality and commonly used data?
- Is it possible to provide a convenient way for scalability and extensibility management that allows flexible and on-demand use of resources?

4 PROPOSED APPROACH

In the first step, a long-term evaluation process was necessary to identify the major characteristics for predictive-analytical ITSM. They will be introduced in this paper as the eight key requirements of IT Service Management as they combine structure and process-oriented service management of ITIL, the

architectural advantages of a SOA and the centralized integration of semantic technologies for handling service complexities. They have been identified as follows:

1. Structured Service Interaction
2. Centralized Service Orchestration
3. Multi-layered Software Architecture
4. Scalable Computing Architecture
5. Domain-independent Architecture
6. Common Information Integration
7. Predictive Analyses Integration
8. Natural Language Interface

These key requirements do not specify any technological implementations and thus provide a technology-independent, holistic view on an ITSM concept, tailored to be generic but structured, scalable and extensible, and applicable for several domains of ITSM. This overall concept provides an ITSM environment highly adjustable to any business needs for a clear alignment to the business processes and a strong focus on shared processable knowledge throughout the whole environment. The motivation for defining these eight key requirements is to provide a common foundation for the development of any ITSM approach with a focus on effective and predictive analytics. This foundation is not just applicable for dealing with one particular ITSM problem, but constitutes helpful practices at the starting point of any effective ITSM development.

4.1 Structured Service Interaction

Extensive service interaction is an indicator for a well-structured and working service environment. Each service provides defined functionality that can or has to be used by other services. Thus, service communication is inevitable for requesting and returning service provided information. Service interaction must not be avoided. It is an indicator for sophisticated capsulation of functionality and conforms to the concept of information sharing and reusing. The only condition for effective service interaction is to accomplish a structured and consistent way of communication. Communication between humans works as long as they understand each other. The same obtains for communication between services.

However, a network of services often consists of heterogenic technologies for service development. It is nearly impossible, or just manageable with high effort, to keep a service environment homogeneous, which is not the goal for effective service

communication. Rather, it is necessary, in a network of various services, to define and enforce a communication standard valid for all service interaction.

In combination with a generic language, the service interaction has to be structured with defined communication endpoints. Interfacing the communication ensures that information exchange is consistent in the whole architecture, which facilitates the maintenance and extension of the service environment and supports the Service Transition phase. Using consistent communication interfaces, it is clearly defined how new services have to provide and how to retrieve information from others, which reduces the risk of incompatibilities in the Release Management.

4.2 Centralized Service Orchestration

Well-defined interfaces for service requests and responses are inevitable for service consistency and service integration and maintenance, but they do not prevent service networks of reaching a level of unmanageable complexity. Service communication has to be orchestrated over a centralized communication manager, a service bus, to provide a single point of contact for all service communication. Instead of directly addressing, services contact the centralized service bus, which handles the further processing of the service messages, based on a consistent and clear addressing schema for services.

However, effective communication management is not just relaying messages from service A to service B, without considering possible service downtimes or overutilization. Effective communication management has to accept the responsibility of managing message queuing and load balancing, to ensure a stable environment that can dynamically react on service failure. Hence, the service bus can perform message forwarding, without knowledge of the actual service location. The message sender and receiver can be located anywhere, as long as they reach the service bus communication interfaces, which enables the possibility for changing service location, for instance a transformation to the cloud, without losing connectivity to the service environment. This flexibility in service providing, in combination with low maintenance effort, makes complex service networks manageable, even if process requirements change. Decisions for outsourcing of service functionality do not depend on the service interdependencies anymore and can be performed completely based on cost and compliance reasons.

4.3 Multi-layered Software Architecture

Functionality has to be separated into services to provide a manageable structure and to be flexible for changes. Basically, service functionality comprises the ability to store and retrieve data, to process the data, if needed based on workflows, and to present the processed information, which can be described as the four service functionality layers. For sure, these layers can be developed for each service independently, but a structured separation in a standardized layer design, based on interface connectivity definitely supports the architecture's structure and consistency and prevents unnecessary heterogeneity. If all services store and retrieve data based on the same standardized platform for data storage, maintenance of service data is also limited to this platform and does not require skills in multiple technologies.

In addition, the integration of new services or the replacement of service functionality can be performed with lower risk of incompatibilities, if storage communication is accomplished over specified interfaces to a standardized storage platform. On the same way, service logic and service processes have to be implemented. Referring back to the structured and consistent service interaction, communication to the data storage and to other services over the service bus has to be accomplished over standardized interfaces, independent from the logic technology.

The coordination of providing information through the data, logic and process layers, and the presentation of this information, in the presentation layer, is essential for bringing the service value to the customer. The challenge of this layer is to retrieve information over the services bus and to bring it into a specific view. The clear separation between information processing and information presentation allows the interaction of different presentation views with one underlying service module and vice versa. This structure comes up to the service catalogue concept, which allows the combination of service items to different service packages, adjusted to a customer's need and flexible for change. Besides that, this clear separation and the communication management of the service bus, allow the easy development of a presentation view for different devices, without the necessity of changing the service logic. The described aspects enable value-focused development of presentation views tailored to the customers' needs in all aspects, collecting the information needed and presenting it in the most

convenient form. The outcome can be a mobile application for a maintenance worker or a classic desktop program for the controller, both accessing collected and tailored information.

4.4 Scalable Computing Architecture

Maintaining and improving an ITSM environment demands flexibility in scalability management. Dynamic reaction on changes is required, at best close to real-time, to ensure service operation without any difficulty. Thus, cloud platforms can be used, which allow dynamic on-demand resource allocation and flexible scalability as well as location independent service communication provided by a service bus.

Moreover, the layered design of a service allows the cloud-transformation of each layer independently from the other layers. Consequently, the service data storage could be transferred to the cloud, while the service logic is still located on premise. This layer based service design enables scalability management at component level and thus, provides the maximum of flexibility in resource provisioning. The emerging "Big Data" topic necessitates the ability for handling large datasets and vast amount of data for supporting predictive decision processes by extracting the maximum value of data (Kourtesis, et al., 2014, p. 310). Thus, highly scalable systems are needed to process such large volumes of data in real-time (Kourtesis, et al., 2014, p. 310).

Besides the advanced scalability management, the support for various presentation devices is a key feature of this cloud-enabled design. The driver for cloud integration is not just improved scalability management and multi device support. Moreover, elementary, financial and compliance aspects play a major role for switching to cloud resources. Thus, it is important that the ITSM architecture has the potential to switch to cloud resources with low effort and low risk of failure. In an effective ITSM environment, cloud decisions should only depend on financial and compliance aspects, but definitely must not be dependent on the technical ability to switch to the cloud.

4.5 Domain-independent Architecture

Ontology models are, like all models, limited to a specific area, domain or region. But for providing an efficient and holistic ITSM environment, limitations of ontology models and consequently limitations of knowledge are not eligible. Since it is not possible to define a model without boundaries, the ITSM environment has to provide the possibility to define

various ontologies or service modules, respectively, representing all domains of expertise in ITSM. Nevertheless, a centralized administration of multiple models is nearly impossible, because the definition of each ontology model requires profound knowledge in this specific area as well as high maintenance effort. Therefore, a decentralized approach is needed, allowing clients to define ontology models on their own and sharing the reusing semantic knowledge over the service bus. Consequently, a system is required, which overtakes the management of the ontology creation, the update of ontologies and rules and the querying of ontologies.

4.6 Common Information Integration

Service functionality depends on processing data. As already mentioned, service related data has to be stored in the service data layer. Each service has its own specific data, only accessible by the service itself, independently from other services' data. Basically, all information needed by a service can be stored independently and separated from others, but this strict separation has one disadvantage. In an ITSM environment, many services depend on information that is related to the field of ITSM in general and commonly used. Thus, a strict separation of all service data leads to multiple storage of the same information, which is unnecessary. A better approach is to divide service related information and common information, accessible by all services. Besides the prevention of multiple data storage, a centralized common data library provides the possibility of effective maintenance and improvement of commonly used information without changing each service implementation.

4.7 Predictive Analysis Integration

For effective IT Service Management, an architecture that allows flexible collecting and tailoring of information for a specific customer is definitely needed and plays a major role for the successful value creation of a service. Nevertheless, the ability to provide information does not imply that the given information is useful for a specific purpose. It is not the ability of providing information, but the ability of providing knowledge that makes ITSM powerful - knowledge in the sense of unknown and implicit information and its combination and classification based on rules. Furthermore, this kind of knowledge defines a new level of predictiveness by solving complex dependency constructs and revealing behavioral patterns and further provides the base for

proactive pattern recognition processes.

This new level of knowledge creation supports decision-making processes based on advanced analyses and the integration in decision support systems. Therefore both, service logic and service presentation need access to computable knowledge they can process, in other words, access to semantic information, which provides predictive knowledge, based on inference and reasoning. By using semantic technologies, the information value provided by services is extended to a maximum. A maintenance worker can definitely perform more efficient, if the given information reveals unknown relations and supports problem solving, not just reactively but also proactively, based on semantically processed analysis, pattern recognition and decision support systems.

Providing predictive knowledge and analyses in form of semantic ontology processing is a key requirement for effective ITSM and has to be permanently available for all services modules and presentations. Thus, a connection to the service bus has to be arranged, which allows all services in the service environment to access semantic information. The semantic ontology is also defined as a service, providing the ability for other services to query knowledge of a specific domain.

4.8 Natural Language Interface

Predictive knowledge is the key for all advanced business and service analyses and consequently for relevant decision making processes in any ITSM domain. Thus, central availability of knowledge was already defined as one of ITSM core characteristics. Nevertheless, providing the formal representation of semantic knowledge is not applicable in a multi domain ITSM environment. Knowledge has to be detached from the technology behind, by providing Controlled Natural Language (CNL) interfaces, allowing non-technical users to retrieve pure knowledge independent from formal language expressions. Predictive knowledge on-demand is the process of transforming natural language query statements of domain specialists into a query language for RDF like the SPARQL Protocol And RDF Query Language (SPARQL), returning knowledge, processible for analyses and decision making processes.

5 CONCLUSIONS

Flexibility is the major goal and complexity the major

challenge of IT Service Management to continuously satisfy customers and to gain competitive advantage. ITIL, SOA, CNL and the capability for predictive analyses play a major role for effective ITSM, but have to be applied correctly to disclose their full potential, which is still unhandled sufficiently in the related work. Hence, this paper identifies the key requirements for effective and predictive analysis in ITSM, to increase the level of flexibility and make complexity manageable and knowledge available on-demand. This set of requirements is defined from a technical independent view, as a profound and generally feasible fundament for architectural implementation decisions regarding a holistic and scalable predictive-analytical ITSM approach. The consequent step is the conceptual realization of a sophisticated and detailed architectural design, based on these key requirements, including technical details and the implementation process.

REFERENCES

- Addy, R., 2007. *Effective IT Service Management - To ITIL and Beyond!*. Berlin Heidelberg: Springer.
- Benedettini, O. & Neely, A., 2012. *Complexity in services: an interpretative framework*. Chicago, s.n.
- Breitman, K., Casanova, M. A. & Truszkowski, W., 2007. *Semantic Web: Concepts, Technologies and Applications*. London: Springer.
- El-Gayar, O. & Deokar, A., 2013. A semantic service-oriented architecture for distributed model management systems. *Decision Support Systems* (55), 4.pp. 374-384.
- Freitas, J., Correia, A. & Brito e Abreu, F., 2008. *An Ontology for IT Services*. Gijón, s.n.
- Fung, H. P., 2013. Using Porter Five Forces and Technology Acceptance Model to Predict Cloud Computing Adoption among IT Outsourcing Service Providers. *Internet Technologies and Applications Research*, pp. 18-24.
- Jantscher, M., Schwarz, C. & Zinser, E., 2014. *Development of an innovative AHP-based decision support system in the field of IT Service Management*. Washington D.C., s.n.
- Josuttis, N. M., 2007. *SOA in Practice: The Art of Distributed System Design*. Sebastopol, CA: O'Reilly Media, Inc..
- Kleiner, F., Abecker, A. & Mauritzat, M., 2012. *Incident and Problem Management using Semantic Wiki-enabled ITSM Platform*. Vilamoura, s.n.
- Kourtesis, D., Alvarez-Rodriguez, J. M. & Paraskakis, I., 2014. Semantic-based QoS management in cloud systems: Current and future challenges. *Future Generation Computer Systems* (32), pp. 307-323.
- Krafzig, D., Banke, K. & Slama, D., 2005. *Enterprise SOA: Service-oriented Architecture Best Practices*. New Jersey: Prentice Hall Professional.

- Schwarz, C., Schmidt, S., Sellner, A. & Zinser, E., 2013. *Service Oriented Cost Accounting - Utilization-based accounting and charging of IT service costs*. Los Angeles, s.n.
- Schwitzer, R., 2010. *Controlled Natural Language for Knowledge Representation*. Beijing, s.n., pp. 1113-1121.
- Valiente, M.-C., Garcia-Bariocanal, E. & Sicilia, M.-A., 2012. Applying an ontology approach to IT service management for business-IT integration. *Knowledge-Based-Systems* (28), pp. 76-87.
- van Bon, J. et al., 2007. *Foundations in IT-Service-Management Based on ITIL v3*. Zaltbommel: Van Haren Publishing.
- vom Brocke, J., Braccini, A. M., Sonnenberg, C. & Spagnoletti, P., 2013. Living IT infrastructures - An ontology-based approach to aligning IT infrastructure capacity and business needs. *International Journal of Accounting Information Systems*.
- Wang, Y. H., Cao, K. & Zhang, X. M., 2013. Complex event processing over distributed probabilistic event streams. *Computers and Mathematics with Applications* (66), pp. 1808-1821.

BPMN Extensions for Decentralized Execution and Monitoring of Business Processes

Jonas Anseeuw, Gregory Van Seghbroeck, Bruno Volckaert and Filip De Turck

Department of Information Technology, Ghent University, B-9050 Ghent, Belgium
{jonas.ansaeuw, gregory.vanseghbroeck, bruno.volckaert, filip.deturck}@intec.ugent.be

Keywords: Cloud Computing, Business Process Management, Monitoring, Business Process as a Service.

Abstract: Software-as-a-service (SaaS) providers are further expanding their offering by growing into the space of business process outsourcing (BPO). Therefore, the SaaS provider wants to administer and manage the business process steps according to a service level agreement. Outsourcing of business processes results in decentralized business workflows. However, current business process modeling languages, e.g. Business Process Execution Language (BPEL), Business Process Model and Notation (BPMN), are based highly on a centralized execution model and current BPMN engines offer limited constructs for federation and decentralized execution. To guarantee execution of business processes according to a service level agreement, different parties involved in a federated workflow must be able to inspect the state of external workflows. This requires advanced inspection interfaces and monitoring facilities. Current business process modeling languages must thus be extended to support monitoring in the specification, support modeling and support deployment of decentralized workflows. In this paper, correlation and monitoring extensions for BPMN are described. These extensions to BPMN are described such that the existing specification can still be used as is in a backwards compatible way.

1 INTRODUCTION

Software-as-a-service oriented software companies want to add value to their offering by providing systematic and controlled delegation of many of the steps of a company's business process, also known as business process outsourcing (BPO). The companies thus administer and manage the business process steps according to a service level agreement. SLA can be an agreement on QoS parameters, but this is also an agreement on the different interfaces between partners.

For example, the process of designing a product needs both business activities and simulation activities (often in an interleaved sequence). Business activities represent mainly data in and output (e.g. constraints, design parameters, etc.) tasks. Simulation activities take models and parameters as input, and analyze usually characteristics that are mentioned in the user-defined constraints and that must be met. Business activities form a high-level view on the process with branches, loops and concurrent activities, referred as the *business workflow*. Simulation activities are usually needed in the course of a business workflow execution in order to validate design param-

eters early, referred as the *simulation workflow*. Figure 1 shows business and simulation workflows and their relation to each other. The simulation workflows are pluggable in the business workflow. As depicted in figure 1, Company A has control over the business workflow (including the simulation workflows from Company B and C).

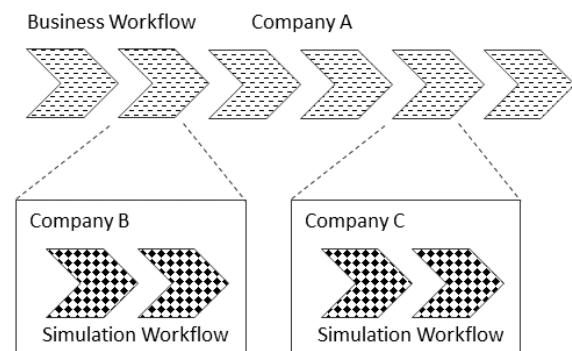


Figure 1: Relation of Business- and Simulation Workflows.

BPO results in decentralized business process flows: both inter-company flows, i.e. federated business process flows that cross the borders of companies, as well as intra-company workflows, distributed

business flows within the datacenters of one company. The business process modeling languages constructs offered by the current generation of workflow languages, e.g. Business Process Execution Language (BPEL) (OASIS, 2007) or Business Process Model and Notation (BPMN) ((OMG), 2011), are based highly on a centralized execution model and current BPMN engines offer no constructs for federation and decentralized execution.

The different parties involved in a federated workflow must be able to inspect the state of the external workflows, all the while hiding workflow implementation details. Business Process Modeling (BPM) languages must thus be extended to support modeling, monitoring and deployment of decentralized workflows in the specification.

The approach described in this paper is to extend BPMN using its extensibility such that the extensions are backwards compatible with the existing BPMN specification. The remainder of this paper is structured as follows. In Section 2, relevant related work is discussed. Sections 3 and 4 handle advanced correlation and monitoring extensions in BPMN respectively. Finally, our conclusions are drawn and future work is discussed in Section 5.

2 RELATED WORK

Previous research (Van Seghbroeck et al., 2007)(Stefanescu et al., 2014)(Barros, 2015)(Dumas and Kohlborn, 2015) has come to the same result with regard to the different steps in the development cycle of decentralized workflows or choreographies depicted in Figure 2. First, a top-down approach describes the service choreography, which can be validated. When the choreography is described, and after validation, all the different participants' stubs are extracted. As a result of the stub extraction and the different implementations, it is possible to use common process engines to execute a choreography.

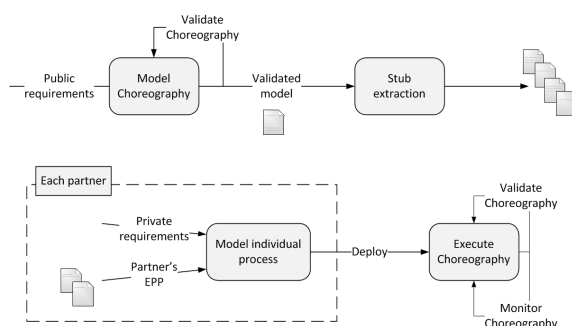


Figure 2: The complete development cycle for service choreographies.

The European Project CHOReOS (Large Scale Choreographies for the Future Internet) (Autili et al., 2014), which ended in 2013, resulted in a very interesting implementation of this development cycle. CHOReOS focusses only on BPMN, to describe its choreography, and to describe and execute the different participants parts of the choreography. CHOReOS, unfortunately, only monitors system level KPIs. In the monitoring model described in this paper, it is also possible to monitor application specific data.

The JBoss Savara¹ project can be used to create service choreographies. It uses Web Service Choreography Description Language (WS-CDL) (Kavantzas et al., 2005) to describe the choreography, but only some basic tools to monitor and validate the choreography are available. The Eclipse BPMN Modeler² can only be used to design choreographies, it does not have an execution environment incorporated, nor are there monitoring tools.

Correlating messages in choreographies comes with another level of complexity, not only do messages have to be correlated to each participant's workflow instance, but they also have to correlate these individual participants to the overall choreography instance. To the authors' knowledge, there is only one specification, WS-CDL, which has a very elaborate definition of and view on correlation. Since BPMN does not offer this, extensions to BPMN are needed.

Since there are no clear choreography execution environments yet, there are no ready-to-use monitoring environments to monitor choreography workflows. Monitoring, using the current tools and frameworks, would entail consolidating and aggregating the monitoring info from all the different choreography participants. It is clear that this is a fairly impossible task since every workflow engine has its own way to store the monitored data and has their own APIs to reference this information. The lack of a standardized monitoring API is a real problem here. Research has dealt with monitoring choreographies (Wetzstein et al., 2010; Lazovik et al., 2004; Roder et al., 2011), but very little research has been performed aimed specifically at BPMN. Another way to monitor a distributed environment is having a centralized system in place (e.g. Business Activity Monitoring) that gathers all necessary information from the individual partner's workflow engines, either via a pull or push method, or via monitoring agents. Another problem with regards to monitoring is that most of the current engines only provide monitoring information about system-wide aspects (e.g. number of re-

¹<http://savara.jboss.org>

²<http://eclipse.org/bpmn2-modeler>

quests) or about aspects related to particular processes or particular process instances (e.g. execution time, a log trace for the different activities of a process). Adding monitoring points as part of a specific process or even application is not possible yet. BPMN is the only standardized specification that already supports including monitoring injection points with its monitoring and auditing element. However, except for describing extensible placeholders, the specification claims details are out of scope and are left to the implementing BPMN engines.

3 CORRELATION EXTENSIONS

As mentioned in related work, BPMN doesn't provide enough functionality to correlate messages in choreographies. BPMN already has some notion of correlation. This is done in the *Collaboration* definition, more specifically in its *Conversations*. A *Conversation* is defined by an array of *CorrelationKeys*. The BPMN specification claims that the *CorrelationKeys* can be used to tie messages to a specific process instance. Correlating messages to process instances may be enough to support centralized workflows, but this is insufficient for decentralized workflows. In a decentralized context, it is key to also uniquely and formally define correlations between all process instances of all parties involved in the decentralized workflow. This requires thorough knowledge on how all instances are created and on how those instances can be correlated via the messages communicated between them. Such a complex model is possible in WS-CDL, by means of its *Identity* types, but not in BPMN. There is another big difference between correlation in BPMN and WS-CDL, WS-CDL defines correlation on *ChannelTypes*, well-defined communication endpoints of a *Participant*, whereas in BPMN it is all done on the *Conversations*, which define the communication potential between two or more *Participants*. There are two possible approaches to improve correlation in BPMN:

1. Extend the BPMN Correlation mechanism to hold similar information as the WS-CDL mechanism.
2. Aid the users by improving how correlation is perceived in BPMN, i.e. add semantical meaning to particular constructs and combinations of conversations.

3.1 Extending BPMN

The first option results in less conversations and most resembles the usage of *ChannelTypes* and *Identities* in

WS-CDL. A BPMN *Conversation* can be interpreted as a WS-CDL *ChannelType*. WS-CDL has four distinct correlation types: primary, alternate, derived and association. With these four identity types, it is possible to create complex correlation relations. BPMN's extension mechanism can be used to add an attribute to the *CorrelationKey* element. For example the attribute type, which can have the following values: primary, alternate, derived, or association. This way, we can mimic WS-CDL correlation types in BPMN.

3.2 Semantical Meaning

In the second option, the user interface will visualize correlation between messages or between conversations different. The user only has to be aware whether *CorrelationKeys* are used to correlate messages or conversations. *Message CorrelationKeys* in BPMN resemble the *Primary* and *Alternate* identities from WS-CDL. They are used to correlate messages belonging to the same *Conversation*. *Conversation CorrelationKeys* on the other hand are used to correlate conversations. *Derived* and *Association* identities take up this role in WS-CDL.

4 MONITORING EXTENSIONS

BPMN already has two placeholders available, auditing and monitoring, but as the specification mentions the actual definition of auditing/monitoring is out of scope of the specification. It is up to the implementers to specify how these elements are used and extended. Auditing and monitoring can be defined for all *FlowElements* (e.g. *Activities*, *Gateways*, *Events*, *Data Objects*, *Data Associations* and *Sequence Flows*), with the only limitation, that these *FlowElements* have to be part of a *Process* flow. Auditing and monitoring can also be set for a *Process* itself, which can be used (for instance) to define process-wide actions.

Setting the audit and monitoring tags for each *FlowElement* would be too time-consuming and prone to mistakes. Besides that, in many cases generic monitoring statements (e.g. *log servicetime between TaskA and TaskB*) can be used to define monitoring. Consequently, it would be more interesting to define the monitoring in a separate declarative model (as an extension on BPMN). References to elements in this new model can be used to connect the *FlowElements* and the *Process* to specific monitoring points. These monitoring points are described in declarative rules.

The following subsections describe in more details the declarative monitoring model, monitoring in-

formation, declaration, visualization and implementation of monitoring.

4.1 Monitoring Model

Several options are available for the Monitoring Model. Either a procedural model or a declarative model. Using a declarative model allows to describe the behaviour of the monitoring framework. Instead of describing in detail when to place a monitoring point, the model can describe under which conditions the framework should place a monitoring point. It is with a declarative model also possible to have simple rules that can result in multiple monitoring points. Some examples:

- after each Activity, log the name of the Activity
- servicetime between TaskB and TaskG
- delay between PartnerA and PartnerB

4.2 Monitoring Information

Metadata

The monitoring metadata describes which data can be captured from the *FlowElements*.

local_time

The timestamp as set by the participants system. In the assumption that all servers involved in the execution of one particular participant's process are in sync.

correlation_keys

An array of all the instances of the correlation keys as defined by the BPMN model.

process_id

The unique identifier for this process as set by the participant's system.

task_id

The unique identifier for this task as defined in the BPMN model.

data_context [optional]

An optional array of all data currently used in this process' instance. This is used to monitor application specific data and is realised by using *Data Objects*.

4.3 Describing Monitoring Declarations

There are different types of monitoring points: single monitoring points, monitoring ranges and monitoring aggregates. Single monitoring points are associated with a specific or all *FlowElements*. This can be a

single *Activity* (e.g. *Task*, *Sub-Process*, *Call Activity*), *Events*, etc. Monitoring ranges are used to monitor data that involves multiple *FlowElements* (e.g. monitoring the service time between two *FlowElements*). A monitoring range implies that multiple single monitoring points are set. Monitoring aggregates are similar to monitoring ranges, but they aggregate data within a certain scope (e.g. over all process instances).

Single Monitoring Points

Single monitoring points can be set before, after or during either all *FlowElements* or a specific *FlowElement*. A *FlowElement* can be specified by a BPMN attribute (e.g. *id*). The second function parameter ω determines where the monitoring point should be set. Setting monitoring points results in capturing all data specified by the metadata as specified in section 4.2. In order to additionally monitor application specific data, *Data Objects* can be associated with monitoring points.

$$\log(\text{FlowElement}[:id], \omega) \equiv \{\text{local_time}, \dots, \text{data_context}\} \quad (1)$$

Monitoring points can be placed depending on the outcome of a function. For example, function e can be $<$, $>$, $=$, *and*, *or*, *not*, etc. x and y can be *FlowElements*.

$$e(x, y, \omega) \rightarrow \log(\text{FlowElement}, \omega) \quad (2)$$

Monitoring Ranges

Monitoring ranges are associated with multiple *FlowElements*. Therefore an ordered list of *FlowElements* is passed to function f .

$$f(\text{FlowElement}, \dots) \quad (3)$$

An example is monitoring the servicetime between two *FlowElements*. The function *servicetime* implies that a single monitoring point must be placed after *FlowElementA* and before *FlowElementB*.

$$\begin{aligned} &\text{servicetime}(\text{FlowElementA}, \text{FlowElementB}) \\ &\rightarrow \log(\text{FlowElementA}, \text{after}) \text{ as } x \\ &\wedge \log(\text{FlowElementB}, \text{before}) \text{ as } y \\ &\quad \wedge y.\text{localtime} - x.\text{localtime} \end{aligned}$$

Monitoring Aggregates

Monitoring aggregates are similar to ranges, but they aggregate all monitoring data within a scope (e.g. average, sum, count, etc.). Parameter α can be function (1), (2) or (3).

$$g(\alpha, scope) \quad (4)$$

The monitoring scope can be:

- process instance, e.g. count of all tasks within a specific process instance.
- process: range of process instances, e.g. count of all tasks over a range of process instances.
- system: range of processes, e.g. when you want to know the service time of a particular participant over different processes.

4.4 Visualization

In order to allow non-developers (business users) to specify where to place monitoring points, a more visual representation of monitoring points is discussed in the next subsections.

Single Monitoring Points

Figure 3 corresponds to the first rule (1). A monitoring point can contain a *Data Object* for application data to be logged.

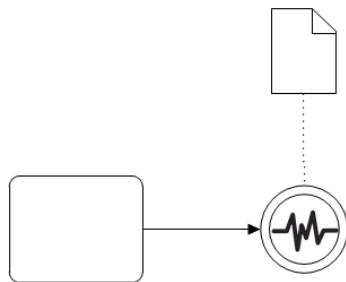


Figure 3: Placement of a single monitoring point.

Figure 4 corresponds to the second rule (2). For example a monitoring point will be placed if a *Data Object* and a particular *Message Event* have occurred.

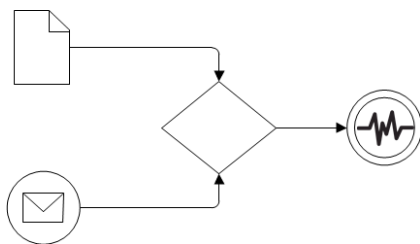


Figure 4: Placement of a conditional single monitoring point.

Monitoring Ranges

Figure 5 corresponds to the third rule (3).

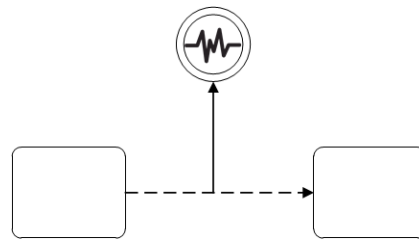


Figure 5: Placement of a monitoring range.

4.5 Implementation

Since BPMN is persisted in XML, the declarative rules should also have an XML counterpart, e.g. RuleML (a markup language for rules). Implementation of monitoring in BPMN can then be achieved by intercepting messages between tasks or by adding a wrapper around tasks. Automatically adding a wrapper around the monitoring tasks allows for more advanced monitoring capabilities. Figure 6 shows this wrapper.

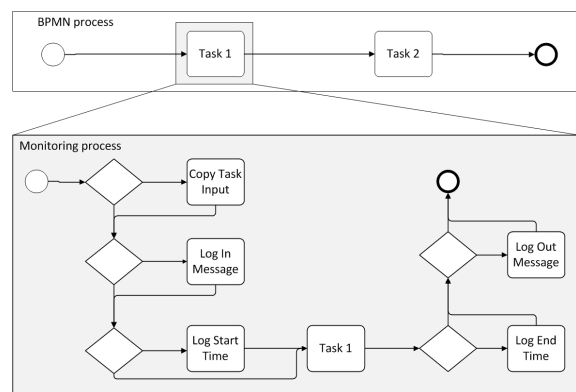


Figure 6: Wrapper around tasks to enable monitoring.

All of the supported monitoring operations must be included in the process. Whether or not they are active depends on the decisions in the process.

5 CONCLUSIONS

In this paper a number of backwards compatible extensions to the BPMN business process modeling language are presented to support correlation and monitoring in decentralized workflows. Correlation in decentralized workflows is achieved by using BPMN's extensibility. The BPMN correlation mechanism can then hold similar information as the WS-CDL mechanism. Another option is to aid the user by improving how correlation is perceived in BPMN. BPMN already has two placeholders available, auditing and

monitoring, but it is up to the implementers of the BPMN engine to specify how these elements are used and extended. Therefore, a declarative monitoring model as an extension on BPMN is described. This model results in declarative rules. Since BPMN is persisted in XML, the declarative rules also have an XML counterpart, e.g. RuleML. These rules have also a notation model. Finally, implementation of the monitoring model can be achieved by adding a wrapper around tasks.

Further research will show which correlation extension option is the more favorable. In future work tooling will be developed allowing non-developers (business users) to design and deploy decentralized business processes. A monitoring API will be designed to monitor these decentralized business processes.

ACKNOWLEDGEMENTS

The iMinds D-BASE project is co funded by iMinds (Interdisciplinary Institute for Technology), a research institute founded by the Flemish Government with project support of the IWT.

REFERENCES

- Autili, M., Inverardi, P., and Tivoli, M. (2014). Choreos: Large scale choreographies for the future internet. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 391–394.
- Barros, A. (2015). Process choreography modelling. In vom Brocke, J. and Rosemann, M., editors, *Handbook on Business Process Management 1*, International Handbooks on Information Systems, pages 279–300. Springer Berlin Heidelberg.
- Dumas, M. and Kohlborn, T. (2015). From business process models to service interfaces. In vom Brocke, J. and Rosemann, M., editors, *Handbook on Business Process Management 1*, International Handbooks on Information Systems, pages 557–578. Springer Berlin Heidelberg.
- Kavantzaz, N., Fletcher, T., Burdett, D., Lafon, Y., Barreto, C., and Ritzinger, G. (2005). Web services choreography description language version 1.0. Candidate recommendation, W3C. <http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/>.
- Lazovik, A., Aiello, M., and Papazoglou, M. (2004). Associating assertions with business processes and monitoring their execution. In *Proceedings of the 2Nd International Conference on Service Oriented Computing*, ICSOC '04, pages 94–104, New York, NY, USA. ACM.
- OASIS (2007). OASIS Web Services Business Process Execution Language. <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>.
- (OMG), O. M. G. (2011). Business process model and notation (bpmn) version 2.0. Technical report.
- Roder, A., Lehmann, M., and Kabitzsch, K. (2011). Monitoring service choreographies. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pages 142–147.
- Stefanescu, A., Wiczorek, S., and Schur, M. (2014). Message choreography modeling. *Software & Systems Modeling*, 13(1):9–33.
- Van Seghbroeck, G., De Turck, F., Dhoedt, B., and De-meester, P. (2007). Web service choreography conformance verification in m2m systems through the pix-model. In *Pervasive Services, IEEE International Conference on*, pages 385–390.
- Wetzstein, B., Karastoyanova, D., Kopp, O., Leymann, F., and Zwink, D. (2010). Cross-organizational process monitoring based on service choreographies. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 2485–2490, New York, NY, USA. ACM.

A Smart Decisional Cognitive System based on Self-adaptability of Web Services to the Context

Faïçal Felhi¹, Marwa Ayadi^{2,3} and Jalel Akaichi¹

¹*BESTMOD Laboratory, High Institute of Management, Tunis University, Tunis, Tunisia*

²*InterVPNC Laboratory, FSJEGJ Jandouba, Janboubia University, Jandouba, Tunisia*

³*UEVE University, IBISC Laboratory, Paris, France*

{felhi_fayssal, marwaayadi04}@yahoo.fr, jalel.akaichi@isg.rnu.tn

Keywords: Cognitive Stimulation, Pervasive Smart System, Web Services, Workflow, Context Awareness, Self-Adaptability.

Abstract: Memory loss or cognitive stimulation application for handicapped people is the subject of a recent field of studies in a information systems. In this way, Web services are a solution for the integration of distributed information systems, autonomous, heterogeneous and auto adaptable to the context. In this paper, we are interested in defining a new solution for a smart and decisional cognitive system based on self-adaptability of Web services to the context and showing this solution by a case study.

1 INTRODUCTION

Cognitive stimulation (Emilie et al., 2007) key many parts in a person who suffers from a loss of autonomy such as Psycho-Social, Cognitive and Functional. It strengthens motivation and verbal and nonverbal communication. It also keeps the residual cognitive resources and optimal autonomy and optimize cognitive functioning (memory, language, attention,...) and social (motivation, sociability) preserved by exploiting the capabilities of patients.

System information must meet some specific constraints surrounding context adaptation in the case of ubiquitous computing (Weiser, 1993). Computing applications now operate in a variety of new settings; for example, embedded in cars or wearable devices. They use information about their context to respond and adapt to changes in the computing environment. They are, in short, increasingly context aware. Considerable approaches related to adaptability with different modes of implementation such as: Aspect Oriented Programming (Kiczales et al., 1997). This aspect used by various platforms on the goal to adapt the Web service (WS, 2004) to the context dynamic changes of environment. Web services, like any other middleware technologies, aim to provide mechanisms to bridge heterogeneous platforms, allowing data to flow across various programs. The Web services technology looks very similar to what

most middleware technologies looks like. The emergence of Web services as a model for integrating heterogeneous Web information has opened up new possibilities of interaction and adaptability to context when offered more potential for interoperability. However, from a set of requirements on SOA (Service Oriented Architecture) (Curbera et al., 2008), and to provide self adaptation to the context of Web services, we need to integrate more generic connector that takes into account all ambient or distant events. The SOA offer great flexibility that is a great ability to functional and technical changes. Moreover, this type of architecture is most often used as Web services support, which provide the flexibility and interoperability expected, that is the ability to communicate between heterogeneous systems. The application in such information systems that incorporate SOA need to communicate across the exchange software (middleware or platforms). These middleware are the source of our work. It is on them that will think the same expectations in terms of flexibility, interoperability and adaptability

Be advised that papers in a technically unsuitable form will be returned for retyping. After returned the manuscript must be appropriately modified.

The rest of this paper is organized as follows: In Section 2, we present our solution for a smart decisional cognitive system. In section 3 we present our approach for a context meta-model for a self

adaptability of SOA. In Section 4, we review previous research on context awareness and adaptability of Web services. Finally, we summarize our work and discuss future research in Section 5.

2 SMART CONITIVE SYSTEM

2.1 Architecture

Our smart cognitive system helps doctors and memory handicapped person workers to evaluate state of patient and use a new event related to patient and help them to refresh her cognitive memory in a short time.

In Figure 1, we presented our architecture general solution for pervasive decisional and smart cognitive system. This architecture represents the different tools and components necessary that helps a doctor to evaluate and treat the condition of a patient has memory loss.



Figure 1: Smart cognitive system.

Our system is based on a workflow; this workflow can test a request from a doctor by a rules engine that will transform the requests in the form of rules. Our system can also give and automatically generate Web services that represent different functionality of the test used by a doctor to evaluate memory state of patient and host under a registry “Cognitive Services”. The personal information for each patient is provided by the middleware stored in a “Data Base” for subsequent needs state.

Under WComp we have integrated a rule engine that can provide management rules that deal with business logic. The rules engine can communicate with a workflow engine, which helps optimize and evolution of these assemblies separating the events produced by the components defined in an application WComp.

2.2 Modelling

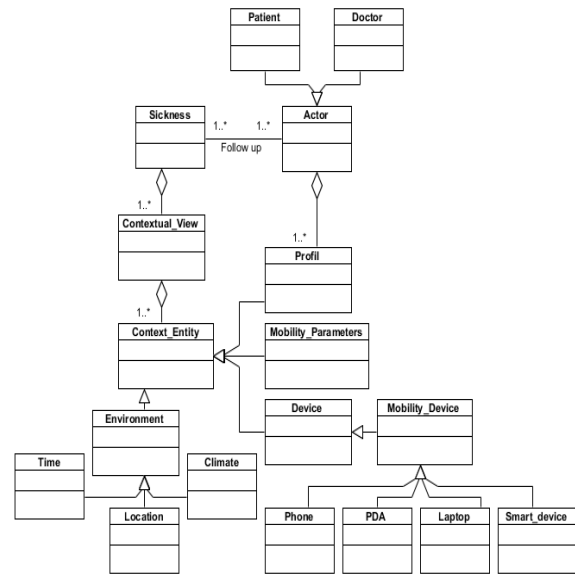


Figure 2: System modelling.

By using our meta model of context, Figure 2 represent the model of our solution to help doctors to invoke web services related a new event related to personal information of patient. This model represents many equipment and resources used in ambient space.

2.3 Implementation

We chose to implement a decision support for patients who have memory loss. Our smart system is a set of ordered tests and uses personal information for patients, such as privacy in its ambient space, these contacts, these family, and every time we introduce events that can refresh his memory. Thereafter, and end testing stages, the doctor can see the score as a percentage of correct answers.

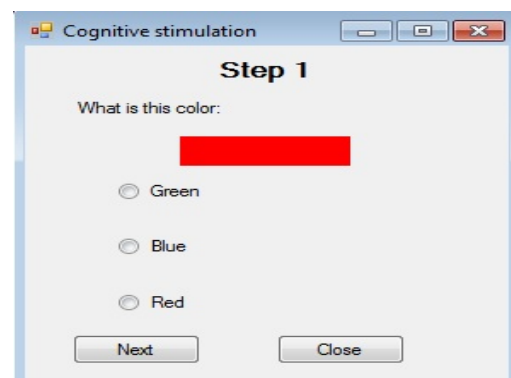


Figure 3: Color recognition.

This result helps him to the decision on the patient's condition and assesses his memory.

In Figure 3, we present a first step of test. This step is color recognition, when the patient must know the color presented.

In Figure 4, we present a Final step of test. This step is face familial recognition, when the patient must know the person in her family or her friend.

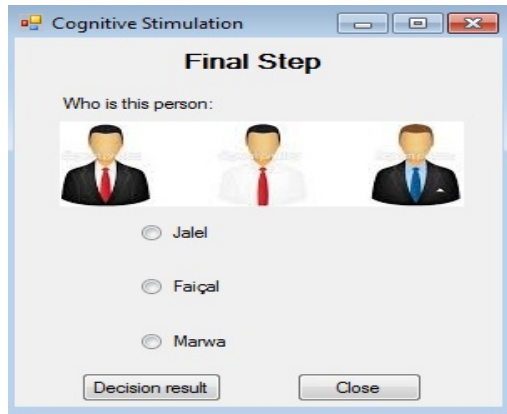


Figure 4: Face familial recognition.

In Figure 5, we show the end result percentage test, accurate answers. This result gives an idea of the patient's memory status and helps the doctor decide whether to continue treatment taken by the patient.

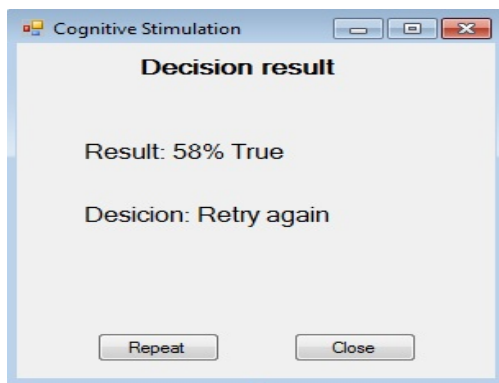


Figure 5: Decision result.

3 SELF-ADAPTABILITY OF WEB SERVICES TO THE CONTEXT

3.1 Architecture

In Figure 6, we presented our research results based on the needs in terms of self adaptability of service oriented architecture to the context. Our architecture

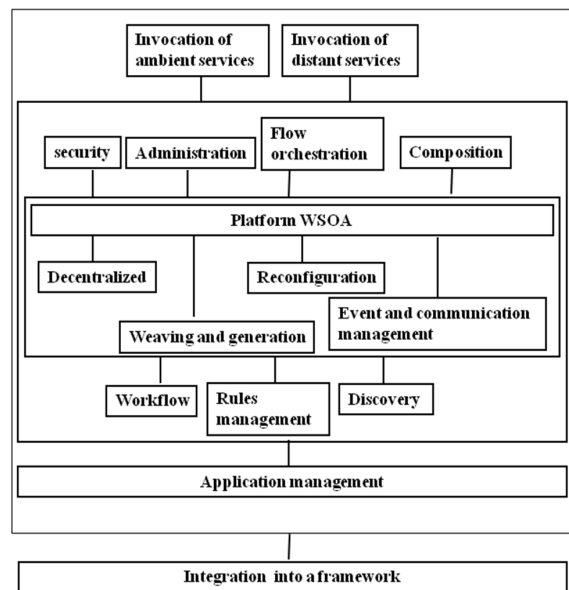


Figure 6: System architecture.

is based on objects or components to make the dynamic reconfiguration of components using more advanced mechanisms. It qualifies the distribution of applications across multiple servers and not the increase in service levels. There is a distributed architecture whose purpose is to deliver services to their audience and they will be accessible from any types of clients. Security and administration are offered by this system in treating the business logic from the workflow and rules.

Contextual resource discovery is the use of context data to discover other resources within the same context. The invocation of distant and ambient services is also permitted by this architecture using technologies dedicated to each type of invocation.

3.2 Context Awareness Modelling

This model (Figure 7) shows the different entities involved in consideration of context. Contextual view consists of several entities of contexts such as the environment (time, location, climate, etc...), mobility profile of the actor (all information that can specify the actor, age, gender these studies, the leisure ...) computing entity at large (especially with mobile device such as a laptop, PDA, phone, etc...) etc...

All the information related to the three dimensions can also be shared by other mobile applications. Our meta model identifies and adds the most relevant and generic contextual entities that will be held in account in modeling any mobile and context aware application. This context metamodel

consists of six generic contextual entities and four deduced entities specific to a category of mobile applications. The class “ContextView” groups all contextual entities involved in a given application. It is identified by name attribute and has two types of relation: the aggregation “involves” and the association “belongsTo”. The first relation expresses that a given “ContextView” is composed of many “ContextEntity” that are involved in a context-aware application. The second relation “belongsTo” expresses the use of historical context information. A given context entity may have participated in different context views. This information can be helpful in the design of future context views. The second generic entity of the meta model is the “ContextEntity”. As we see on the figure bellow, it is specialized in three generic entities: Actor, Computational Entity and Environnement. Actor may be a person or another object that has a state and profile. It evolves in an environment and uses computational devices to invoke services. With the Computational entity, the computational device is used by the actor to access the services and to capture contextual information from the environment.

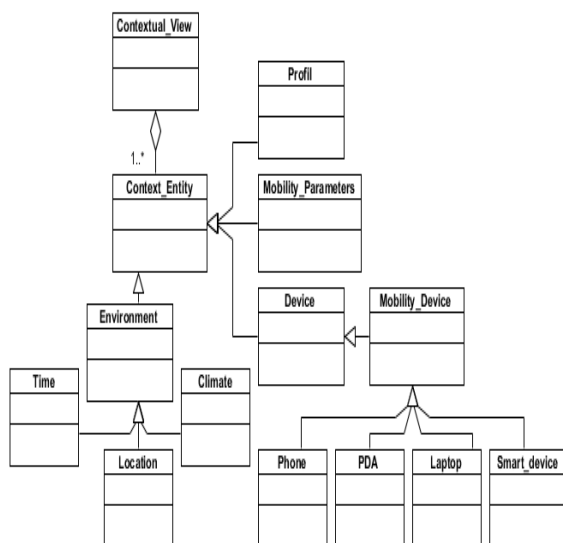


Figure 7: Context awareness meta-model.

Usually, a mobile device is used in context aware mobile applications, and can obtain information concerning the type of device it is (PDA, laptop, cellular phone...), the application, the network, etc. The environment is constituted of all the information surrounding the actor and its computational device that can be relevant for the application. It includes different categories of information as :(i) Spatial context information can be location, city, building,

(ii) Temporal context information comprises time, date, season, (iii) Climate can be temperature, type of weather.... The last entity is a profile. We are convinced this entity is important in any user centered context aware application. In fact, profile is strongly attached to the actor and contains the information that describes it. An actor can have a dynamic and/or a static profile.

The static profile gathers information relevant for any mobile context-aware application. It can be the “date of birth”, “name” or “sex”. On the opposite, dynamic profile includes customized information depending on the specific type of application and/or the actor. It can be goals, preferences, intentions, desires, constraints, etc.

4 STATE OF THE ART

Cognitive stimulation techniques are represented in the form of applications that offer exercises and activities to improve and develop cognition of a person with a loss of autonomy, which needs a refresh and stimulate his memory.

Several studies show the cognitive stimulation in several technical and several forms. The Creasoft group (CREASOFT, 2014) gave us several applications. PRESCO is a program that focuses on memory, attention, language, visual spatial and executive functions. Tvneurones and Words and head travel are an applications in games form at different levels. They can stimulate the evocation to work the lack of the word, vocabulary and memory strategies proposed by various. MonAgenda Memory is a personal book; proposes adapted agendas and cognitive stimulation games. Allows the elderly and / or disoriented to keep his schedule and play regularly in cognitive stimulation activities.

The context awareness (Monfort & Felhi, 2010a, 2010b; Monfort et al., 2010) of such applications is the subject of a recent field of studies in pervasive computing called: context-aware systems. In (Monfort & Hammoudi, 2010, 2009; Vale & Hammoudi, 2008), authors define context-awareness as the ability of a program or device to sense or capture various states of its environment and itself. Referring to these latter definitions a context-aware application must have the ability to capture the necessary contextual entities from its environment, use them to adapt its behavior (run time environment) and finally present available services to the user. In (Gu et al., 2005), the authors introduce another definition in which they insist on the use of context and the relevance of context

information. The authors consider a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevance depends on the user's task. In (Emanuele & Koetter, 2007), the authors considered context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. The authors give a general definition that can be used in a wide range of context-aware applications. In (Winograd, 2001) the author approves this definition and claims that it covers all proposed works in context. However he considers it as a general definition that does not limit a context. Thus he proposes his own definition in which he limits a context in a set of information, which is structured and shared. It evolves and is used for interpretation. We stress that the notion of hierarchy (structure) of context introduced by (Winograd, 2001) is important. The definition proposed in (Chen & Kotz, 2000) also presents the context as hierarchically organized. In this work the authors differentiate between environmental information that determines the behavior of mobile applications and that which is relevant to the application. They thus define the context as the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user.

Web service is the best fitted technology for implementing Service Oriented Architectures (SOA) offering flexibility and interoperability. WSs provide a minimalist mechanism to interconnect different applications. But one fundamental point is the importance of the WSDL (Web Services Description Language) (WSDL, 2007) being the exact interface of the system. WSDL is responsible for the message payload, itself described with the equally famous protocol SOAP (Object Access Protocol) (SOAP, 2007), while data structures are explained by XML (eXtended Markup Language) (XML, 2012). Very often, WS are stored in UDDI (Universal Description Discovery and Integration) (UDDI, 2004) registry.

Many approaches treat the adaptability of SOA in joining with Web services, to context. Charfi and al. approach (Charfi & Mezini, 2004) propose a framework that provides support for middleware BPEL (Business Process Execution Language) (BPEL, 2003) engines. The authors apply the concepts of deployment descriptor and container for the Web service composition. Ferraz Tomaz and al.

approach (Ferraz et al., 2006) proposed a tool for weaving aspects for a simple adaptability of the Web services, implementing aspects of the services as loosely coupled, where aspects are woven dynamically. In this approach, aspects are themselves Web services, thus they are independent of languages and platforms. Mehdi Ben Hmida approach (Ben Hmida et al., 2006) extended the solution proposed by (Ferraz et al., 2006) to specify BPEL processes adaptable, that is to say, the adaptability of complex services. Hence the need to extend the semantic aspects and Web services, which resulted in the ASW (Aspect Service Weaver). Aspects are themselves loosely coupled Web services, they are independent of languages and platforms, but, this approach has limitations.

Adaptation to context is not taken into account, that is to say, if an event occurred during a search on a Web service, this approach does not take into account this event. In the other approaches we find those based on context adaptation (Garlan et al., 2002; Biegel and Cahill, 2004; Anastasopoulos et al., 2006; Roman and Islam, 2004). The ambient computing encourages the proliferation of associated devices. We cited WComp approach (Tigli et al., 2009a, 2009b, 2009c) which represents the implementation of experimental models for lightweight components for service composition SLCA (Service Lightweight Component Architecture) which enables the design of ambient computing applications by assembling software components, orchestrating access to services through infrastructure devices from ambient. WComp supports protocols such as UPnP (Universal Plug and Play) (UPnP, 2012) and Web services, allowing components through the proxy to interact with them. To promote adaptation to context WComp uses Aspect Assembly paradigm. Aspect Assemblies can either be selected by a user or fired by a context adaptation process. It uses a weaver that allows adding and or suppressing components. With this architecture WComp allows: i) managing devices heterogeneity and dynamic discovering by using UPnP, ii) events driven interactions with devices, iii) managing dynamic devices connection and disconnection (dynamic re configuration on run time) in infrastructure.

In our research work (Felhi & Akaichi, 2012a, 2012b, 2013a, 2013b), we presented a proposal to a self-adaptable SOA to the context based on workflow (Workflow, 2006) by presenting the functional and technical architecture of our approach. In this architecture we have given different features in terms of the needs of self-

adaptability offered by the integration of workflow, which allows the management rules (Rules, 2010) and a kind of security and administration of Web services. This solution which can offer management rules that deal with business logic. Business logic can help in the development and optimization of these assemblies separating the events produced by the components of Web services.

4 CONCLUSIONS AND FUTURE WORKS

In this paper we have shown the interest of self adaptability of SOA based on workflow since it often involves multiple heterogeneous systems, and in particular for cognitive decisional and smart system. We proposed our solution helps doctors and memory handicapped person workers to evaluate state of patient and use a new event related to patient and help them to refresh her cognitive memory in a short time.

We hope in our future work enhance our approaches another application domain..

ACKNOWLEDGEMENTS

We thank everyone.

REFERENCES

- Emilie W., Inge C. K., Jocelyne D. R., Pia G., Florence M., F riel B., Aurore R., Martha D. S., & Anne S. R. (2007). Cognitive stimulation intervention for elders with mild cognitive impairment compared with normal aged subjects: preliminary results. *Aging Clinical and Experimental Research*. Volume 19, Issue 4, pp 316-322.
- Weiser M. (1993). Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*. Volume 36, no. 7, pp. 75-84.
- Kiczales G., Lamping J., Maeda C., & Lopes C. (1997). Aspect-oriented programming. *Proceedings European Conference on Object-Oriented Programming (ECOOP'97)*. volume 1241, pp 220-242. Springer-Verlag, Berlin, Heidelberg, and New York.
- WS Retrieved (2004), from <http://www.w3.org/TR/ws-arch/>.
- Curbera F., Khalaf R., & Mukhi N. Quality of Service in SOA Environments. An Overview and Research Agenda (Quality of Service in SOA-Umgebungen). *it - Information Technology*. 50(2): 99-107, 2008.
- CREASOFT Retrieved (2014), from: <http://www.editions-creasoft.com/>
- Monfort, V., & Felhi, F. (2010). Context Aware Management Platform to Invoke remote or local e Learning Services Application to Navigation and Fishing Simulator. *International Symposium on Ambient Intelligence, ISAMI'10 Publisher*. Special Volume in Advances in Intelligent and Soft Computing (Springer), Guimar es, Portugal.
- Monfort, V., & Felhi, F. (2010). A contextual approach to invoke intelligent house Services: an application to help physically handicapped persons. *1rst International Workshop on Recent Trends in SOA Based Information Systems in conjunction with ICEI*. Funchal Madeira, Portugal.
- Monfort, V., Khemaja, M., Ammari, N., & Felhi, F. (2010). Using SaaS and Cloud computing For "On Demand" E Learning Services: Application to Navigation and Fishing Simulator. *10th IEEE International Conference on Advanced Learning Technologies*, Sousse, Tunisia.
- Vale, S., & Hammoudi, S. (2008). Context-aware Model Driven Development by Parameterized Transformation, *Proceedings of MDISIS*.
- Monfort, V., & Hammoudi, S. (2010). When Parameterized MDD Supports Aspect Based SOA. *IEBR International Journal of E-Business Research*.
- Monfort, V., & Hammoudi, S. (2009). Towards Adaptable SOA: Model Driven Development, Context and Aspect. *The 7th International Conference on Service Oriented Computing*, Stockholm, Sweden.
- Gu, T., Pung, H., & Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services, *Journal of Network and Computer Applications*, 28 1-18.
- Emanuele, J., & Koetter, L. (2007). Workflow opportunities and challenges in healthcare. *BPM & Workflow Handbook*.
- Winograd, T. (2001). Architectures for context. *Human-Computer Interaction (HCI) 16(2-4)*, 401-419.
- Chen, G., & Kotz, D. (2000). A survey of context-aware mobile computing research. *Technical Report (ACM)*, Dept. of Computer Science, Dartmouth College.
- WSDL. Retrieved (2007), from <http://www.w3.org/TR/wsdl20/>.
- SOAP. Retrieved (2007), from <http://www.w3.org/TR/SOAP>.
- XML. Retrieved (2012), from <http://www.w3.org/XML/>.
- UDDI. Retrieved (2004), from http://www.uddi.org/pubs/uddi_v3.htm.
- Charfi, A., & Mezini, M. (2004). Aspect-Oriented Web Service Composition with AO4BPPEL. *2nd European Conference on Web Services (ECOWS) Publisher*. Volume 3250 of LNCS, Springer, pp. 168-182.
- BPPEL. Retrieved (2003), from <http://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>.
- Ferraz Tomaz, R., Ben Hmida, M., M., & Monfort, V. (2006). Concrete Solutions for Web Services Adaptability Using Policies and Aspects. *JDIM - Journal of Digital Information Management*. Publisher.

- Ben Hmida, M., M., Ferraz Tomaz, R., F., & Monfort, V. (2006). Applying AOP concepts to increase Web services flexibility. *Journal of Digital Information Management (JDIM)* Publisher.
- Garlan, D., Siewiorek, D. P., Smailagic, A., & Steenkiste, P. (2002). Aura: Toward distraction free pervasive computing. *IEEE Pervasive Computing*. Publisher.
- Biegel, G., & Cahill, V. (2004). A framework for developing mobile, context-aware applications. *2nd IEEE Conference on Pervasive Computing and Communication*. pp.361–365.
- Anastasopoulos, M., Klus, H., Koch, J., Niebuhr, D., & Werkman, E. (2006). DoAml – a middleware platform facilitating re-configuration in ubiquitous systems. *System Support for Ubiquitous Computing Workshop, At the 8th Annual Conference on Ubiquitous Computing (UbiComp)* Publisher.
- Roman, M., & Islam, N. (2004). Dynamically programmable and reconfigurable middleware services. *Middleware, Springer Publisher*. Volume 3231 in LNCS, pp. 372–396.
- Tigli, J. Y., Lavirotte, S., Rey, G., Hourdin, V., & Riveill, M. (2009). Lightweight Service Oriented Architecture for Pervasive Computing. *IJCSI International Journal of Computer Science Issues*. Volume 4, No. 1, ISSN (Online): 1694-0784, ISSN (Print): 1694-0814.
- Tigli, J. Y., Lavirotte, S., Rey, G., Hourdin, V., & Riveill, M. (2009). Context-aware Authorization in Highly Dynamic Environments. *IJCSI International Journal of Computer Science Issues*. Volume 4, No. 1, ISSN (Online): 1694-0784, ISSN (Print): 1694-0814.
- Tigli, J. Y., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo D., Callegari, E., & Riveill, M. (2009). WComp Middleware for Ubiquitous Computing: Aspects and Composite Event-based Web Services. *Annals of Telecommunications*. Volume 64, n° 3-4, pp 197. ISSN 0003-4347.
- UPNP. Retrieved (2012), from <http://www.upnp.org/>.
- Felhi, F., & Akaichi, J. (2012). Adaptation of Web services to the context based on workflow: Approach for self-adaptation of service-oriented architectures to the context. *International Journal of Web & Semantic Technology (IJWesT)*. Volume3, No.4, Publisher.
- Felhi, F., & Akaichi, J. (2012). Towards the self-adaptability of Service-Oriented Architectures to the context based on workflow. *International Journal of Advanced Computer Science and Applications (IJACSA)*. Volume3, No.12, Publisher.
- Felhi, F., & Akaichi, J. (2013). Self-adaptability of SOA to the context based on workflow in a e-Healthcare monitoring system. *International Conference on Web and Information Technologies (ICWIT'13)*. Hammamet, Tunisia.
- Felhi, F., & Akaichi, J. (2013). Pervasive e-healthcare system based on self-adaptability of SOA to the context. *IEEE International Conference on Information Technology & e-Services (ICITeS' 2013)*. Sousse, Tunisia.
- Workflow Retrieved (2006), from, <http://www.bpmbuletin.com/2006/06/21/difference-entre-workflow-et-moteur-de-regle/>.
- Rules Retrieved (2010), from, <http://www.vdocsoftware.com/vdoc/easysite/InVDOC2010/news/innovation/agi-lite-regles-metiers>.

CLOUD COMPUTING PLATFORMS AND APPLICATIONS

FULL PAPERS

Secure Evidence Collection and Storage for Cloud Accountability Audits

Thomas Ruebsamen¹, Tobias Pulls² and Christoph Reich¹

¹*Cloud Research Lab, Furtwangen University, Furtwangen, Germany*

²*Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden*
{thomas.ruebsamen, christoph.reich}@hs-furtwangen.de, tobias.pulls@kau.se

Keywords: Cloud Computing, Security, Accountability, Digital Evidence.

Abstract: Cloud accountability audits can be used to strengthen trust of cloud service customers in cloud computing by providing reassurance regarding the correct processing of personal or confidential data in the cloud. However, such audits require various information to be collected. The types of information range from authentication and data access logging to location information, information on security controls and incident detection. Correct data processing has to be proven, which immediately shows the need for secure evidence record storage that also scales with the huge number of data sources as well as cloud customers. In this paper, we introduce Insynd as a suitable cryptographic mechanism for storing evidence for accountability audits in our previously proposed cloud accountability audits architecture. We present our reasoning for choosing Insynd by showing a comparison of Insynd properties with requirements imposed by accountability evidence collection as well as an analysis how security threats are being mitigated by Insynd. Additionally, we describe an agent-based evidence collection process with a special focus on security and privacy protection.

1 INTRODUCTION

Cloud Computing is known for its on demand computing resource provisioning and has now become mainstream. Many businesses as well as private individuals are using cloud services on a daily basis. The nature of these services varies heavily in terms of what kind of information is being out-sourced to the cloud provider. More often than not that data is sensitive, for instance when Personal Identifiable Information (PII) is being shared by an individual. Also, businesses that move (parts of) their processes to the cloud, for instance by using a Customer Relationship Management Software as a Service provider, are actively participating in a major paradigm shift from having all data on-premise to moving data to the cloud.

New challenges come along with this trend. Two of the most important issues are customer trust and compliance (Jansen and Grance, 2011; Pearson, 2011). These issues are closely tied to the loss of control over data. When moving to the cloud, direct control over i) where data is stored, ii) who has access to it and iii) how it is shared and processed is given up.

Because of this loss of control, cloud customers have to trust cloud providers that they treat their data in an appropriate and responsible way. This in-

cludes providing information about data locality, isolation, privacy controls and data processing in general. One way to enable that trust is by strengthening transparency and accountability (Haeberlen, 2009; Weitzner et al., 2008) of the cloud provider and services.

To regain information on the kind of data processing, cloud audits can be used to check how it has been done. An important part of cloud audits is evidence collection. Depending on the data processing policies in place, various sources of evidence need to be considered. Logs are a very important source of evidence, when it comes to auditing the cloud operation (e.g., access logs and error logs). However, other sources of information are also important, such as files or events registered in the cloud management system. To capture evidence from this variety of sources, centralized logging mechanisms are not enough. We therefore propose a system for accountability evidence collection and audit. With this system, cloud providers are enabled to demonstrate their compliance with data handling policies to their customer's and third-party auditors in an automated way.

In our previous work, we introduced a system (Ruebsamen and Reich, 2013) for cloud accountability audits, that enables automated collection of evidential data in the cloud ecosystem with the goal of

performing accountability audits. A key mechanism of this system is the secure and privacy-friendly collection and storage of evidence. In our previous work we also explored the use of a somewhat homomorphic encryption scheme to secure evidence collected in the evidence store (Lopez et al., 2014). In this paper, we present a more practical alternative that imposes less restrictions on evidence collection. The contributions of this paper are:

- An architecture for automated evidence collection for the purpose of cloud accountability audits
- A process for secure and privacy-protecting evidence collection and storage

The remainder of this follow-up paper is structured as follows: in Section 2 we present related work in the area of secure evidence collection and cloud auditing. The core principles of Insynd are introduced in Section 3. Following that, we present in Section 4 a mapping of typical characteristics of digital evidence and secure evidence collection in the cloud to how these are addressed by integrating Insynd in our audit agent system. In Section 5 we describe the architectural details of the Insynd integration. We present a scenario-based informal evaluation of our system in Section 6 and conclude this paper in Section 7.

2 RELATED WORK

Redfield and Date propose a system called Gringotts (Redfield and Date, 2014) that enables secure evidence collection, where evidence data is signed at the system that produces it, before it is sent to a central server for archival using the Evidence Record Syntax. It is similar to our system with respect to the automatic collection of evidential data from multiple sources. However, their focus is on the archival of evidence, whereas we propose a system that also enables automated evidence processing for audits. Additionally, our system also addresses privacy concerns of evidence collection in a multi-tenant environment such as the cloud by introducing evidence encryption, whereas Redfield and Date focus on archival and preservation of evidence integrity.

Zhang et al. (Zhang et al., 2013) identify potential problems when storing massive amounts of evidential data. They specifically address possible information leaks. To solve these issues, they propose an efficient encrypted database model that is supposed to minimize potential data leaks as well as data redundancy. However, they focus solely on the storage backend

and do not provide a workflow that addresses secure evidence collection as a whole.

Gupta (Gupta, 2013) identifies privacy issues in the digital forensics process, when it comes to data storage devices that typically do not only contain investigation related data, but may also hold sensitive information that may breach privacy. He also identifies a lack of automation in the digital investigation process. To address these issues, Gupta proposes the Privacy Preserving Efficient Digital Forensic Investigation (PPEDFI) framework. PPEDFI automates the investigation process by including knowledge about previous investigation cases, and which kinds of files were relevant then. With that additional information, evidence search on data storage devices is faster. However, while Gupta acknowledges privacy issues, the PPEDFI framework is focused on classic digital forensics and may not be applicable to a cloud ecosystem, where there is typically no way of mapping specific data objects to storage devices, in full.

The Security Audit as a Service (SAaaS) system proposed by Doelitzscher et al. (Doelitzscher et al., 2012; Doelitzscher et al., 2013) is used to monitor cloud environments and to detect security incidents. SAaaS is specifically designed to detect incidents in the cloud and thereby consider the dynamic nature of such ecosystems, where resources are rapidly provisioned and removed. However, the main focus of SAaaS is not to provide auditors with a comprehensive way of auditing the cloud provider's compliance with accountability policies, which requires additional security and privacy measures to be considered in the data collection process.

3 INSYND

Insynd is a cryptographic scheme where a forward-secure *author* sends messages intended for *clients* through an untrusted *server* (Pulls and Peeters, 2015b; Pulls and Peeters, 2015a; Pulls et al., 2013). The author is forward-secure in the sense that the author is initially trusted but assumed to turn into an active adversary at some point in time (Bellare and Yee, 2003). Insynd protects messages sent prior to author compromise. The server is completely untrusted, which is possible thanks to the use of Balloon, a forward-secure append-only persistent authenticated data structure (Pulls and Peeters, 2015a). This means that the server storing all messages can safely be outsourced, e.g., to traditional cloud services. Clients are assumed trusted to read messages sent to them by authors. Insynd contains support for clients to also be in the forward-security model, by discarding key-

material as messages are read.

Insynd provides the following properties:

Forward Integrity and Deletion Detection. Nobody can modify or delete messages sent prior to author compromise, as defined by Pulls et al. (Pulls et al., 2013). This property holds independently for Balloon (the data structure) and the Insynd scheme. For Balloon, anyone can verify the consistency of the data structure, i.e., it is publicly verifiable (Pulls and Peeters, 2015a).

Secrecy. Insynd provides public-key authenticated encryption (An, 2001) thanks to the use of NaCl (Bernstein et al., 2012).

Forward Unlinkability of Events. For each run by the author of the protocol to send new messages, all the events sent in that run are unlinkable. This implies that, e.g., an attacker (or the server) cannot tell which events belong to which client (Pulls and Peeters, 2015b). When clients receive their events by querying the server, if they take appropriate actions including but not limited to accessing the server over an anonymity network like Tor (Dingledine et al., 2004), their events remain unlinkable.

Publicly Verifiable Proofs. Both the author and client receiving a message can create publicly verifiable proofs of the message sender (the author), the receiving client (by registered identity), and the time the message was sent relative to e.g. a time-stamping authority (Pulls and Peeters, 2015b). The proof-of-concept implementation of Insynd uses Bitcoin transactions (Nakamoto, 2008) as a distributed time-stamping server.

Distributed Settings. Insynd supports distributed authors, where one author can enable other authors to send messages to clients it knows of without requiring any interaction with clients. Client identifiers (public keys) are blinded in the protocol, ensuring forward-unlinkable client identifiers between different authors (Pulls and Peeters, 2015b).

Pulls and Peters show that Insynd provides forward integrity and deletion detection, secrecy, publicly verifiable proofs, and forward-unlinkability of client identifiers in the standard model under the assumptions of the decisional Diffie-Hellman (DDH) assumption on Curve25519, an unforgeable signature algorithm, an unforgeable MAC, a collision and pre-image resistant hash function, and the security of the time-stamping mechanism (in our case, the Bitcoin block-chain) (Pulls and Peeters, 2015b). Forward unlinkability of events is provided in the random oracle model under the DDH assumption

on Curve25519 (Pulls and Peeters, 2015b). The prototype implementation of Insynd shows performance comparable to state-of-the-art secure logging schemes, like PillarBox (Bowers et al., 2014), securing syslog-sized messages (max 1KiB) in the order of hundreds of microseconds on average on a commodity laptop. We stress that Insynd is subject to its own review and evaluation; in this paper, we use Insynd as a building block to facilitate secure evidence collection and storage for cloud accountability audits.

4 AUDIT EVIDENCE STORAGE REQUIREMENTS

In this Section, we present a comparison of general evidence attributes, how they apply in the context of evidence collection for cloud accountability audits and how the integration of Insynd solves key issues in evidence storage.

4.1 Requirements of Digital Evidence

In (Mohay et al., 2003) the core principles of any evidence are described as:

Admissibility. Evidence must conform to certain legal rules, before it can be put before a jury.

Authenticity. Evidence must be tieable to the incident and may not be manipulated.

Completeness. Evidence must be viewpoint agnostic and tell the whole story.

Reliability. There cannot be any doubts about the evidence collection process and its correctness.

Believability. Evidence must be understandable by a jury.

These principles apply to common evidence as well as digital evidence. Therefore, the evidence collection process for audits has to consider special requirements, which help in addressing these attributes and ensure best possible validity in audits and applicability in court.

In Table 1 we present a mapping of the previously described evidence attributes and how they are supported by the integration of Insynd as a means of storing evidence records. We thereby focus on the key properties of Insynd as described in Section 3.

Admissibility of digital evidence is influenced by the transparency of the collection process and data protection regulation. Digital evidence can be any kind of data (e.g., e-mail messages, social network messages, files, logs etc.). Insynd does not have any

direct influence on the admissibility of the evidence stored in it.

Authenticity of digital evidence before court is closely related to the integrity requirement put on evidence records. Evidence may not be manipulated in any way and must be protected against any kind of tampering (willingly and accidentally). Insynd ensures that data cannot be tampered with once it is stored.

Completeness is not directly ensured by Insynd, but rather needs to be ensured by the evidence collection process as a whole. Especially important are the definition of which evidence sources provide relevant evidence that need to be considered during the collection phase. Insynd can complement the evidence collection process by providing assurance of that all data stored in the evidence store are made available as evidence, and not cherry-picked.

Reliability is indirectly supported by integrating necessary mechanisms into the evidence collection process, such as Insynd.

Believability of the collected evidence is not influenced by implemented mechanisms, but rather by the interpretation and presentation by an expert in court. This is due to judges and juries usually being non-technical, which requires an abstracted presentation of evidence. Insynd does not influence the believability in that sense.

Table 1: Mapping the Impact of Insynd Properties to Evidence Attributes.

		Insynd	
		Forward Integrity and Deletion Detection	Publicly Verifiable Proofs
ES	Admissibility		
	Authenticity	✓	✓
	Completeness	✓	✓
	Reliability	✓	✓
	Believability		

4.2 Privacy Requirements

Not all requirements that a secure evidence storage has to fulfill can be captured by analyzing the attributes of digital evidence. Other aspects have to be taken into account to address privacy concerns. Protecting privacy in the process of evidence collection is utmost importance, since the collected data is likely to contain personal data. For cloud computing, one limiting factor may be whether or not the cloud provider

is willing to provide deep insight into its infrastructure. Table 2 presents a mapping of privacy principles and properties of our evidence process.

Below we summarise some key privacy principles:

Confidentiality. of data evolves around mechanisms for the protection from unwanted and unauthorized access. Typically, cryptographic concepts, such as encryption, are used to ensure confidentiality of data.

Data Minimization. states that the collection of personal data should be minimized and limited to only what is strictly necessary.

Purpose Binding. of personal data entails that personal data should only be used for the purposes it was collected for.

Retention Time. is concerned with how long personal data may be stored and used, before it needs to be deleted. These periods are usually defined by legal and business requirements.

Insynd and our evidence process provides various mechanisms that support these privacy principles.

Confidentiality A central property of Insynd is that it is always encrypting data using public-key cryptography. By encrypting the evidence store, compromising the privacy of cloud customer data that has been collected in the evidence collection processes becomes almost impossible by attacking the evidence store directly. This goes as far as being able to safely outsource the evidence store to an untrusted third-party, a key property of Insynd (Pulls and Peeters, 2015b).

Data Minimisation Furthermore, Insynd provides forward unlinkability of events and client identifiers, as described in Section 3, which helps prevent several types of information leaks related to storing and accessing data. Collection agents are always configured for a specific audit task, which is very limited in scope of what needs to be collected. Agents are never configured to arbitrarily collect data, but are always limited to a specific source (e.g., a server log) and data objects (e.g., a type of log events).

Purpose Binding Neither Insynd nor our evidence process can directly influence the purpose for which collected data is used. Indirectly, the use of an evidence process like ours, incorporating secure evidence collection and storage, may serve to differentiate data collected for auditing purposes with other data collected e.g., for marketing purposes.

Retention time poses a real challenge. In cloud computing, the precise location of a data object is usually not directly available, i.e., the actual storage medium used to store a particular block is unknown, making data deletion hard. However, if data has been

encrypted before storage, a reasonably safe way to ensure “deletion” is to discarding the key material required for decryption. Insynd supports forward-secure clients, where key material to decrypt messages are discarded as messages are read.

Table 2: Mapping of Insynd properties to Evidence Collection Requirements.

		Insynd		
		Secrecy	Forward Unlinkability of Events	Forward Unlinkability of Recipients
ES	Confidentiality	✓	✓	✓
	Data Minimisation		✓	✓
	Purpose Binding			
	Data Retention	✓		

In Section 6, we also describe the threat model for the system described in this paper and present an evaluation of how Insynd is used to mitigate these threats.

5 SECURE EVIDENCE STORAGE ARCHITECTURE

In this Section, we provide an architectural overview of the integration of Insynd into a secure evidence collection and storage process. We describe the overall architecture and its components, how the components of Insynd are mapped into the audit agent system and which setup process is required to use Insynd for securing evidence collection and storage.

5.1 Architecture

In this Section we discuss the architectural integration of Insynd as an evidence store in our audit system. There are basically three different components required to perform secure evidence collection. Figure 1 shows an overview of these components - *Evidence Source*, *Evidence Store* and *Evidence Processing* - as well as the flow of data between them. From the various sources of evidence in the cloud, evidence records are collected that will be stored in the evidence store on a per-tenant basis. The evidence store

is thereby located on a separate server. As previously mentioned, the server may be an untrusted third-party cloud storage provider. This is important to ensure so that this approach scales well with a growing number of tenants, evidence sources and evidence records.

Our architecture is built around using software agents for evidence collection, evidence evaluation and controlling the overall system. Agent technology helps with extensibility by allowing us to easily introduce new evidence sources and processors by building new agents. On top of that, it allows the audit system to address rapid infrastructure changes, which are very common in cloud infrastructures by easily deploying and destroying agents when needed. We base our system on the Java Agent DEvelopment Framework (JADE, 2015). This effectively means that anywhere, where a Java runtime environment is available, a collection agent can be deployed.

5.1.1 Evidence Collection

There are various evidence sources to be considered, such as logs, cryptographic proofs, documentation and many more. For each, there needs to be a suitable collection mechanism. For instance, a log parser for logs, a tool for cryptographic proofs or a file retriever for documentation. This is done by a software agent called *Evidence Collection Agent* that is specifically developed for the data collection from the corresponding evidence source. The collection agent acts as an *Insynd Author* meaning it uses the *Sender API* to store evidence into the Evidence Store. The encryption happens in the Sender API. Typically, this agent incorporates or interfaces with a tool to collect evidential data, for instance forensic tools, such as file carvers, log parsers or simple search tools. Another type of collection agent have client APIs implemented to interface with more complex tools, such as Cloud Management Systems (CMS). Generally, these agents receive or collect information as input and translate that information into an evidence record, before storing it in the Evidence Store.

5.1.2 Evidence Storage

From the Evidence Collection Agent, evidence records are sent to the Evidence Store. The Evidence Store is implemented by the *Insynd Server*. Since Insynd functions as a key-value store for storing evidence records (encrypted messages identified by a key) NoSQL or RDBMS-based backend for persisting evidence records can be used. All data contained in the Evidence Store is encrypted. Each record is addressed to a specific receiver (e.g., an Evidence Processing Agent). The receiver’s public key is used in

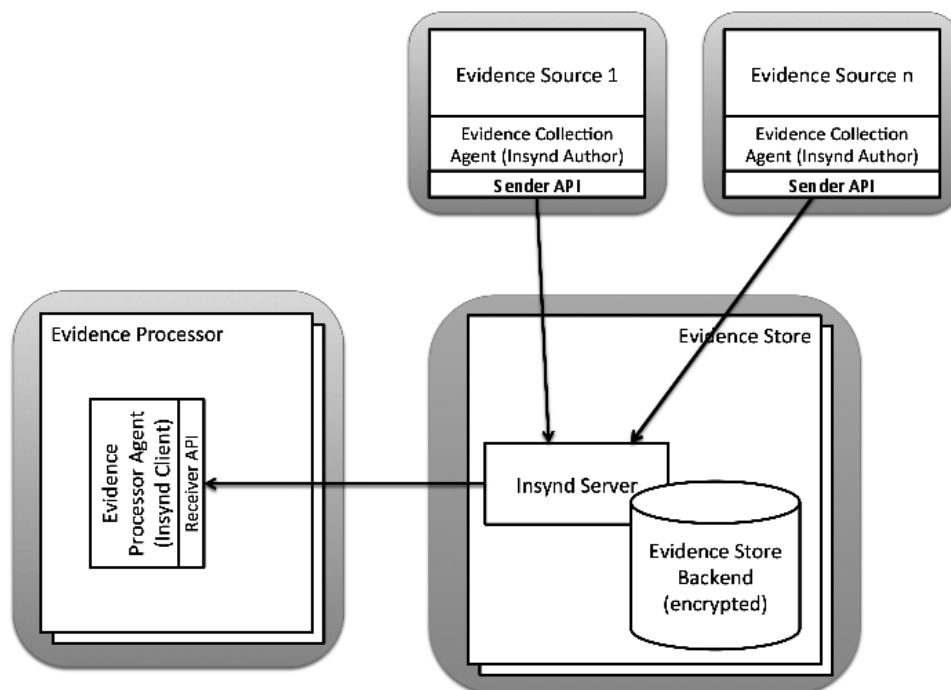


Figure 1: Evidence Collection, Storage and Processing Workflow.

the Sender API to encrypt the record on the Evidence Store. This means that only the receiver is able to access the evidence data from the Evidence Store. Isolation between tenants in a single Evidence Store is achieved by providing one container for each tenant where his evidence records are stored. However, even stronger isolation is also possible by providing a separate Evidence Store hosted on a separate VM. Additionally, Evidence records require a unique identifier in the Evidence Store to enable selective retrieval of records. In our implementation, we use a combination of a policy identifier and a rule identifier (where a rule is part of a policy) to enable the receiver to reduce the amount of records to receive to a manageable size.

5.1.3 Evidence Processing

Evidence Processing components are located at the receiving end of this workflow. The Receiver API is used by the processing agent (Insynd Client) to retrieve evidence records from the Evidence Store. The receiver can request multiple records from a period of time at once. The Client is also in possession of the corresponding private key to decrypt evidence records, which means records can only be decrypted at the Client.

5.2 Identity Management and Key Distribution

Since asymmetric encryption is such an important part of our system, we describe the encryption key distribution sequence next. In this software agent-based system, the automated setup of key material and registration with Insynd is particularly important. Figure 2 depicts the initialization sequence of collection and processing agents with a focus on key distribution.

In Figure 2 we introduce an additional component beyond those already described in the general architecture: the *Controller*. The Controller serves as an entry point that controls the agent setup and distribution process in the audit system. It is an important part of the lifecycle management of the system's agents (e.g., creating and destroying of agents or migration between platforms).

In Figure 2 we describe the initialization sequence for a simple scenario, where a particular tenant wishes to audit compliance with a policy and one rule included in that policy in particular. The following steps have to be performed to setup the evidence collection and storage process for that particular rule:

1. In the first step, a Processing Agent is created and configured according to the input policy and rule respectively for the tenant.

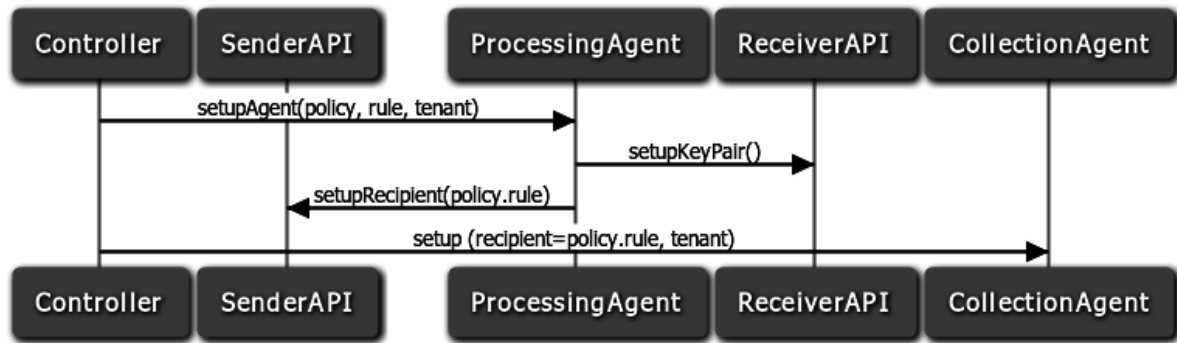


Figure 2: Evidence Collection Setup Sequence.

2. During the setup phase, the Processing Agent sets up a keypair at the Receiver API. The Receiver API is a RESTful service that holds private key material and is therefore located at the same servers hosting the Processing Agents (i.e., a trusted environment).
3. After the key material has been generated, the Processing Agent registers itself as a recipient at the Sender API. For this, it uses a unique identifier generated from the policy ID and the rule ID (i.e., *policyID.ruleID*).
4. In the last step, the Controller sets up the required Collection Agents and connects them with the corresponding Processing Agents by using the unique recipient identifier.

Now, it is possible for the Collection Agents to send evidence records to their corresponding Processing Agents. The messages will be encrypted at the Sender API service before storage, using the provided recipient's public key. The Processing Agent then pulls the evidence records from the Evidence Store using the Receiver API the records are decrypted using the receiver's private key.

6 EVALUATION

In this Section we present an informal security evaluation of the system we have implemented for secure evidence collection. We describe the evidence collection work flow using a fictitious scenario. By applying the evidence collection and storage process to the setting described in this scenario, we demonstrate how the requirements stated in Section 4 are addressed. Additionally, we provide a model that states threats and adversaries to the process as well as the mitigation functions introduced by Insynd.

In this scenario, the CCOMP company is a customer of the Infrastructure as a Service provider

CloudIA. In particular, we analyze the security properties of the evidence collection process by looking at the data at rest as well as the data in transit protection at any time during the flow from the evidence source to its processor. We thereby assume that CloudIA is using OpenStack (OpenStack, 2015) as its Cloud Management System (CMS), since this a widely popular open source CMS, which we use for developing our audit agent system. However, any other CMS could be used as well as long as it provides the needed monitoring interfaces.

6.1 Scenario

CloudIA is specialized in providing its customers with virtualized resources in the form of virtual machines, networks and storage. CCOMP has outsourced most of its IT services to CloudIA. Among them is a service that processes data of CCOMP's customers. For that data, CCOMP has to guarantee data retention. CCOMP has identified snapshots to be one major problem with respect to the data retention policy, since the virtual machine's storage is duplicated in the process. This means for CCOMP that in order to be compliant with the data retention policy, a snapshot of that virtual machine may have a maximum lifetime of one day, which limits its usefulness to e.g., backing up before patching. Now, we assume a trustworthy but sloppy administrator at CCOMP who creates a snapshot before patching software on the virtual machine, but then omits deleting the snapshot after he is done. However, an automated daily audit of its cloud resources was put in place by CCOMP to detect such compliance violations.

6.2 Implementation

The collection agent required for the above scenario communicates with our OpenStack CMS to gather evidence of the CMS behavior regarding virtual ma-

chine snapshots. The processing agent contains the logic for detecting snapshot violations (i.e., base virtual machine and a maximum age of the snapshot derived from the retention policy). The collection agent is deployed at the CMS controller node and has access to OpenStack's RESTful API. The processing agent is located on the same trusted host as the controller agent (see Figure 1 for reference). The evidence store is located on a separate, untrusted virtual machine. Now, the following steps are performed:

1. The collection agent opens a connection to the OpenStack RESTful API on the same host and requests a history of snapshot events for CCOMP's virtual machine. Despite there being no communication over the network, HTTPS is used to secure the communication between the collection agent and the CMS. Since the policy only requires information about snapshots to be collected, the CMS agent limits evidence record generation to exactly that information, nothing more.
2. The collection agent sets up the receiver of the evidence according to the process depicted in Figure 2 and sends the collected records to the evidence store (Insynd). The communication channel is encrypted using HTTPS and the payload (evidence records) is encrypted with the receiving agent's public key.
3. The processing agent pulls records from the evidence store in regular intervals (e.g., every 24 hours), analyses them and triggers a notification of a detected violation. The communication between the processing agent and the evidence store is secured using HTTPS.
4. In the last step, evidence records are deleted because their retention limit has been reached. This is done by discarding the keys required for decryption.

6.3 Threat Model

To demonstrate which security threats exist for the evidence collection process and Insynd is used to mitigate them, we describe the threat model for this system categorized according to the STRIDE (Microsoft Developer Network, 2015) threat categorization:

- Spoofing Identity
- Tampering with Data
- Repudiation
- Information disclosure
- Denial of Service
- Elevation of Privilege

We have identified the following major threats to the evidence collection and storage process:

- *Unauthorized access to evidence (S,I)*: the protection of evidence from being accessed by unauthorized persons. Possible adversaries are a malicious third-party evidence storage provider (cloud service provider), another tenant (isolation failure) or an external attacker. Using Insynd for evidence collection and storage addresses this threat since recipients of messages are authenticated using appropriate mechanisms such as user credentials for API authentication and public keys for encryption.
- *Data leakage (S,I)*: the protection from unintentional data leakage. This could be caused by misconfiguration (e.g., unencrypted evidence being publicly available). Using Insynd for evidence collection and storage addresses this threat by encrypting data by default.
- *Eavesdropping, (T,I)*: the protection of evidence during the collection phase, especially in transit. Possibly adversaries are another tenant (isolation failure) or external attackers in case evidence is transported to an external storage provider or auditor. Using Insynd for evidence collection and storage addresses this threat by using transport layer as well as message encryption.
- *Denial of Service (D)*: the protection of the evidence collection and storage process from being attacked directly with the goal of disabling or shutting it down completely (e.g., to cover-up simultaneous attacks on another service). Possible adversaries are external attackers. This is a very generic threat that cannot be addressed by a single tool or control but rather requires a set of measures (on the network and application layer) to enhance denial of service resilience.
- *Evidence manipulation (T,R,I)*: the protection of evidence from intentional manipulation (e.g., deletion of records, changing of contents, manipulation of timestamps). Possible adversaries are malicious insiders and external attackers. Using Insynd for evidence collection and storage addresses this threat, since Insynd provides tampering and deletion detection.

Some of these threats can be mitigated by implementing appropriate security controls (i.e., using Insynd for evidence transport and storage). It provides effective protection by employing security techniques described in Section 3.

6.4 Requirements Evaluation

In this section, we evaluate the integration of Insynd against the requirements described in Section 4. In step 1 of the fictitious scenario, the data minimization principle is being followed because the specialized agent only collects evidence on the existence of snapshots.

This workflow is secure as soon as the collection agent inserts data into the evidence store in step 2. More precisely, evidence records are tamper-evident and encrypted. This is true, even though the evidence is actually stored on an untrusted virtual machine. The only way to compromise evidence now, is to attack the availability of the server hosting the Insynd server.

When the processing agent in step 3 retrieves records for evaluation, it can be assured of the authenticity of the data and that it has been provably collected by a collection agent. Since evidence records may be subject to maximum data retention regulation, records that are not needed anymore are deleted.

As previously mentioned in Section 5 we use JADE as an agent runtime. To secure our system against non-authorized agents, we use the TrustedAgents add-on for the JADE platform. This ensures that only validated agents are able to join our runtime environment. This effectively prevents agent injection attacks, where malicious agents could be inserted at either the collection or processing side to compromise our system.

As can be seen, the evidence records are protected all the way from the evidence source to the processing agent using only encrypted communication channels and having an additional layer of security (message encryption) provided by Insynd. Additionally, while the evidence is being stored, it remains encrypted.

6.5 Scalability

Obviously, since there is a vast amount of evidence sources and therefore a potentially equal number of collection agents, ensuring the scalability of the process and the implementation is very important. This has been considered very early in the design process by choosing an software agent-based approach for the system architecture. Software agents are inherently distributable and allow for complex message flow modeling in an infrastructure. Therefore, the core components evidence collection, storage and processing become distributable as well. In our future work, we'll focus on the scalability aspects. We will follow a methodology where we focus on the following technical key scalability indicators:

- Data transfer volume: amount of evidence data being transferred over the network
- Message volume: amount of evidence message transmissions over the network
- Storage volume: amount of storage required for evidence
- Encryption overhead: performance impact introduced by encryption and decryption

Based on the identified performance impact of each of these indicators, in the second step, we model different message flow optimization strategies to alleviate their impact and ensure scalability.

7 CONCLUSIONS

In this paper, we presented our system design and implementation for secure evidence collection in cloud computing. The evidence provides the general basis for performing cloud accountability audits. Accountability audits take a large variety of evidence sources and data processing requirements into account.

We showed what the requirements for a secure evidence collection process are and demonstrated how these issues are addressed by incorporating Insynd into our system. We described how the core principles of digital evidence are addressed by our system. Additionally, we considered data protection principles for the evidence collection process, how they influence our approach and how they are addressed in our system by integrating Insynd. For this, we presented the relevant architectural parts of our prototype.

In our future work, we will focus on the scalability of our audit system in general and the scalability of the components involved in evidence collection in particular. For that reason, we will focus on the distribution of the audit system and evidence collection not only in the same domain (i.e., in the same infrastructure), but also taking into account outsourcing and multi-provider collection scenarios.

ACKNOWLEDGEMENTS

This work has been partly funded from the European Commissions Seventh Framework Programme (FP7/2007-2013), grant agreement 317550, Cloud Accountability Project - <http://www.a4cloud.eu/> - (A4CLOUD).

REFERENCES

- An, J. H. (2001). Authenticated encryption in the public-key setting: Security notions and analyses. *IACR Cryptology ePrint Archive*, 2001:79.
- Bellare, M. and Yee, B. (2003). Forward-security in private-key cryptography. In *Topics in Cryptology—CT-RSA 2003*, pages 1–18. Springer.
- Bernstein, D. J., Lange, T., and Schwabe, P. (2012). The security impact of a new cryptographic library. In Hevia, A. and Neven, G., editors, *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, volume 7533 of *Lecture Notes in Computer Science*, pages 159–176. Springer.
- Bowers, K. D., Hart, C., Juels, A., and Triandopoulos, N. (2014). PillarBox: Combating Next-Generation Malware with Fast Forward-Secure Logging. In *Research in Attacks, Intrusions and Defenses Symposium*, volume 8688, pages 46–67. Springer.
- Dingledine, R., Mathewson, N., and Syverson, P. F. (2004). Tor: The second-generation onion router. In Blaze, M., editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320. USENIX.
- Doelitzscher, F., Reich, C., Knahl, M., Passfall, A., and Clarke, N. (2012). An Agent Based Business Aware Incident Detection System for Cloud Environments. *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1):9.
- Doelitzscher, F., Ruebsamen, T., Karbe, T., Reich, C., and Clarke, N. (2013). Sun behind clouds - on automatic cloud security audits and a cloud audit policy language. *International Journal On Advances in Networks and Services*, 6(1 & 2).
- Gupta, A. (2013). Privacy preserving efficient digital forensic investigation framework. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 387–392.
- Haeberlen, A. (2009). A case for the accountable cloud. In *Proceedings of the 3rd ACM SIGOPS International Workshop on Large-Scale Distributed Systems and Middleware (LADIS'09)*.
- JADE (2015). Java Agent DEvelopment framework. <http://jade.tilab.com>.
- Jansen, W. and Grance, T. (2011). Sp 800-144. guidelines on security and privacy in public cloud computing. Technical report, Gaithersburg, MD, United States.
- Lopez, J., Ruebsamen, T., and Westhoff, D. (2014). Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption. In *Innovations for Community Services (I4CS), 2014 14th International Conference on*, pages 95–103.
- Microsoft Developer Network (2015). The Stride Threat Model. [https://msdn.microsoft.com/en-US/library/ee823878\(v=cs.20\).aspx](https://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx).
- Mohay, G. M., Anderson, A. M., Collie, B., de Vel, O., and McKemmish, R. D. (2003). *Computer and Intrusion Forensics*. Artech House, Boston, MA, USA. For more information about this book please refer to the publisher's website (see link) or contact the authors.
- Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28.
- OpenStack (2015). Openstack. <http://www.openstack.org/>.
- Pearson, S. (2011). Toward accountability in the cloud. *Internet Computing, IEEE*, 15(4):64–69.
- Pulls, T. and Peeters, R. (2015a). Balloon: A forward-secure append-only persistent authenticated data structure. *Cryptology ePrint Archive*, Report 2015/007.
- Pulls, T. and Peeters, R. (2015b). Insynd: Secure one-way messaging through Balloons. *Cryptology ePrint Archive*, Report 2015/150.
- Pulls, T., Peeters, R., and Wouters, K. (2013). Distributed privacy-preserving transparency logging. In Sadeghi, A.-R. and Foresti, S., editors, *WPES*, pages 83–94. ACM.
- Redfield, C. M. and Date, H. (2014). Gringotts: Securing data for digital evidence. In *Security and Privacy Workshops (SPW), 2014 IEEE*, pages 10–17.
- Ruebsamen, T. and Reich, C. (2013). Supporting cloud accountability by collecting evidence using audit agents. In *Cloud Computing Technology and Science (Cloud-Com), 2013 IEEE 5th International Conference on*, volume 1, pages 185–190.
- Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., and Sussman, G. J. (2008). Information accountability. *Commun. ACM*, 51(6):82–87.
- Zhang, R., Li, Z., Yang, Y., and Li, Z. (2013). An efficient massive evidence storage and retrieval scheme in encrypted database. In *Information and Network Security (ICINS 2013), 2013 International Conference on*, pages 1–6.

Supporting Multiple Persistence Models for PaaS Applications using MDE

Issues on Cloud Portability

Elias Adriano Nogueira da Silva¹, Daniel Lucrédio², Ana Moreira³ and Renata Fortes¹

¹ICMC-USP, São Carlos, Brazil

²DC-UFSCar, São Carlos, Brazil

³NOVA-LINCS, FCT - UNL, Lisboa, Portugal

{eliasnog,renata}@icmc.usp.br; daniel@dc.ufscar.br; amm@fct.unl.pt

Keywords: Cloud Computing, Model-driven Engineering, Platform-as-a-Service, Portability, Persistence.

Abstract: In cloud computing, lock-in refers to the difficulty of porting an application from one platform to another. An example of such difficulty can be witnessed when porting an application from Platform-as-a-Service Google App Engine to Microsoft Azure. Differences in their implementations are substantial, yielding non-portable applications. Standardization could address this problem, but existing initiatives are still to be accepted. This paper addresses lock-in by proposing a model-driven engineering design approach that decouples platform specific code from the application logic. The resulting platform-independent models, as well as corresponding model transformations, can be reused to generate distinct platform-specific implementations, hence reducing the programming effort spent coding repetitive tasks. Such transformations can be made available for reuse on a repository for cloud providers. We have implemented transformations to handle persistence for Google App Engine and Azure, and discuss how model-driven engineering can reconcile the differences between features of the persistence models of GAE and Azure.

1 INTRODUCTION

In a recent report, the *IEEE Computer Society*¹ highlights 22 technologies with potential to change the scenario of computer science and its role in industry until 2022 (Alkhatib et al., 2014). Apart from foreseeing relevant research roadmaps, it also discusses a vision for each of those technologies. One such technology is cloud computing.

Cloud computing is not a new technological model, but the integration of technologies from the past (Chen et al., 2011). What is new though, is the different ways in which it is used to provide computing power as-a-service through the Internet. According to Armbrust et al., cloud computing permits acquiring computing resources on demand, enables payment according to the utilization volume, and allows a company to ignore the sources of the resources (Armbrust et al., 2009).

Several technological requirements are needed for the cloud model to operate properly. The most common are virtualization technologies, standards and in-

terfaces that allow shared access, facilitated instantiation and management of virtual servers (IaaS – Infrastructure-as-a-Service). Moreover, there are different kinds of resources in the cloud. Load balancing, data persistence and analytics are just some of the many options available for application developers. Given this variety, and depending on its field of expertise, each cloud provider offers a different set of computational resources. Some even provide a complete development platform (PaaS – Platform-as-a-Service) that puts together many different resources under control of the cloud provider.

Both the heterogeneity and diversity of cloud services result in increased complexity as well as reduced reuse and portability of the applications (da Silva et al., 2013). In practice, some applications need to be highly specialized with respect to a particular type of resource (e.g. hardware, platform and/or set of services), yielding the *lock-in* problem (Armbrust et al., 2009). For example, when choosing a particular PaaS provider, the application developer usually has to follow a specific data management system and programming style. This typically reduces porta-

¹<http://www.computer.org/>

bility, resulting in applications “locked-in” to that particular environment.

Some strategies based on standardization have been proposed to address that portability issue (Armbrust et al., 2009; Petcu and Vasilakos, 2014). However, standards take time to define and approve, and require time to be accepted by a large part of development community. Currently, there are so many different standards being proposed (Petcu and Vasilakos, 2014) that even choosing one may be a difficult task. Moreover, cloud providers may wish to use specific technologies to create solutions that are aligned with their own business requirements, hence choosing not to follow the standards. Thus, until standardization becomes a fact, the portability problem, and in particular lock-in, remains.

We have been exploring how Model-Driven Engineering (MDE) (see Section 2) can be used to address portability (da Silva et al., 2013). Approaches based on MDE (France and Rumpe, 2007) have been investigated in several other contexts and may constitute an interesting alternative to address the problem. Our long-term goal is to build a repository of MDE transformations and use code generation to reduce the development effort for each platform, consequently reducing repetitive programming tasks, increasing portability and minimizing the lock-in effects.

The present paper takes Google App Engine (GAE) and Microsoft Azure, two well-known platforms available in the market, and shows how MDE conciliates the differences between their cloud persistence models. We show how these differences can be hidden behind a single conceptual model and discuss a set of MDE artifacts to support this idea. This can be seen as an abstraction layer that allows specifying entities and a set of code generators that use these entities to build similar persistence models even if the storage mechanisms are different.

The two central points of the idea are (i) using a DSL for modeling entities and (ii) building a set of transformations that can generate code for different targets from the same set of source models. Such code-generation approach allows developers focusing on platform-independent models, thus achieving portability by reducing the lock-in effects. Both the models created using DLS and their respective transformations for various different platforms can be made available in a repository for reuse.

In a previous paper (da Silva et al., 2013) we discuss the general approach, but we do not show how persistence is dealt with, which is one of the most interesting parts of our work. Here we extend that work, presenting the differences between the persistence models of GAE and Azure. We also show

details of how these differences can be conciliated through an MDE process, resulting in applications that can be more easily ported between these two cloud providers.

As our approach is generic, transformations considering standards may also be added to the repository later. Although the typical claimed MDE-benefits are expected (e.g., facilitated maintenance and increased productivity), an analysis of the economic viability of creating and maintaining a repository of transformations is out of the scope of this paper².

The rest of this paper is organized as follows. Section 2 presents some conceptual background, including a more detailed definition of the lock-in problem, the different types of cloud portability, concepts of MDE and an overview of the previously proposed MDE approach. Section 3 discusses the two platforms that were the subject of this study (GAE and Azure), focusing on the differences in their persistence models. Section 4 presents our proposed solution using MDE and Section 5 discusses some points about the performed evaluation of the proposal. Section 6 presents related work and, finally, Section 7 concludes with some final remarks and future work.

2 BACKGROUND

This section starts with a discussion of lock-in and known types of portability. It then introduces model-driven engineering, and finishes with a summary of our vision on the use of MDE to support PaaS portability.

2.1 The Lock-in Problem

Lock-in is the difficulty faced to move data and programs from one cloud platform to run on another one (Armbrust et al., 2009). This is a major issue in the PaaS scenario: in order to take advantage of a very flexible cloud architecture, the applications are developed conforming to the specificity of the chosen platform. For example, to offer great elasticity, the GAE PaaS provider imposes a specific programming style and specific data management policy. Thus, an application developed for it may not be easily ported to a different PaaS provider, nor can its data. Even if the developer wants to host an application in his own private cloud later, considerable effort may be necessary to rebuild the code, redeploy it, and migrate all

²Mohagheghi and Dehlen presented a review of experiences from applying MDE in industry (Mohagheghi and Dehlen, 2008).

the data. This lack of portability causes the lock-in effect.

The possibility of becoming “locked in” on a particular platform, not being able to choose a different one later (customer lock-in), leaves developers in a difficult position. They mostly fear being charged abusive fees later, or having their applications unavailable due to lack of service quality (Armbrust et al., 2009).

2.2 Types of Portability

Prior to deciding on the adoption of a cloud model, an organization should take into account the viability of the one that better fits its business. It must carefully analyze the constraints related to cloud platforms, both technical and organizational (da Silva et al., 2013), as well as its business requirements (Khajeh-Hosseini et al., 2011).

Portability is a key attribute for the improvement and dissemination of the cloud model. The existing literature discusses four main types of portability in the cloud scenario: (i) portability of virtual machines between cloud providers, (ii) portability of applications in the context of IaaS, (iii) portability of PaaS applications and (iv) data portability between cloud providers (Bozman, 2010; Petcu et al., 2013; Ranabahu and Sheth, 2010; Shirazi et al., 2012).

This paper focuses on portability of PaaS applications. However, the code generators and repository proposed here can be used in other contexts. Petcu et al. present a list of initiatives to handle portability (Petcu and Vasilakos, 2014). They also discuss the reasons, scenarios, taxonomies, measurements, and requirements for portability. Several other authors are also looking at the problem and proposing alternatives to address it (see Section 6). One such alternative is the use of model-driven engineering (MDE).

2.3 Model-Driven Engineering (MDE)

Despite the advancements of the software development techniques, concerns about reuse, productivity, maintenance, documentation, validation, optimization, portability and interoperability are still under discussion.

Model-Driven Engineering (MDE) aims at solving some of those issues (Kleppe et al., 2003), shifting the focus of modern development methodologies from implementation to conceptual modeling. Thus, models are now first-class citizens, and transformation mechanisms are used to generate code from them, reducing developers’ effort (Kleppe et al., 2003) and increasing portability and productivity (da Silva et al.,

2013). The vision is that MDE will reduce the accidental complexity by increasing the level of abstraction used to develop software.

According to Schmidt, MDE technologies “offer a promising approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively” (Schmidt, 2006). That is exactly our goal: use MDE to abstract away platform-specific details, building conceptual domain models that express the essence and logic of the domain. From these models, applications can then be generated through automatic transformations, thus reducing the development effort for implementations on different platforms.

Such models are abstract descriptions or specifications of the system and are usually represented as a combination of graphical (Domain-Specific Modeling Languages – DSML) and textual elements (Domain-Specific Languages – DSL) (Brambilla et al., 2012). DSLs are small languages focused on a particular problem/domain, and are normally declarative (Deursen et al., 2000). The language definition usually requires a metamodel which is capable of capturing the common and variable points of a specific domain (Brambilla et al., 2012; Deursen et al., 2000).

2.4 A Model-driven Approach for Cloud PaaS Portability

In a previous study (da Silva et al., 2013) we discussed a vision for using MDE to increase cloud PaaS portability, and discussed how to build a DSL and a set of code transformations, based on the Model-View-Controller (MVC) architecture, to reduce the effort of developing cloud applications. We also presented a DSL metamodel, samples of code transformations, the grammar of the DSL and a quasi-experiment (Wohlin et al., 2000; Juristo and Moreno, 2010) showing that MDE helps both reducing the development effort and achieving portability. Fig. 1 summarizes the methodology followed.

Adopting a typical MDE life-cycle, this methodology obeyed to the following strategy:

1. Case studies were developed to identify the main concepts of PaaS. These studies involved a careful analysis of the different providers’ documentation, as well as the development of sample applications for different platforms.
2. Next, these concepts were used to prototype a specification language. This language serves to support the creation of platform-independent models that developers will use to specify the applications’ structure and logic. This step involves

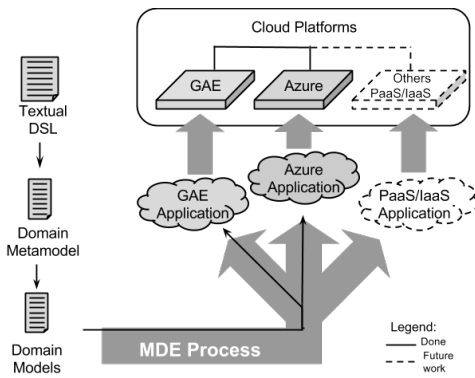


Figure 1: A MDE approach for cloud PaaS portability.

the development of a domain metamodel and a concrete syntax.

3. Based both on the case studies and on the specification language, transformations were defined to automatically generate code for cloud platforms.
4. Tests were performed to verify the conformance between the generated code and the platforms' requirements.

3 PERSISTENCE IN PaaS

The PaaS model leverages the flexibility of the cloud model, by providing a complete platform for software development. A cloud platform hides many of the complexities of developing cloud software, therefore increasing scalability and elasticity. In the PaaS model, the development platform is provided as-a-service. Applications that are developed for this particular platform can benefit from a specific programming model that can be fully, fine grained, managed by the platform provider.

Among the existing platforms, we selected two well-known ones for developing our prototype: the Google App Engine³ (GAE) and the Windows Azure. However, as the developed DSL is platform independent (although domain dependent), it can be used to generate code for any other platform. One of the services managed by the provider is data persistence. By defining its own way to store data, a provider may incorporate services such as load balancing, automatic data distribution and optimized querying. This is so for the two selected platforms for our study. Both GAE and Azure offer PaaS solutions incorporating NoSQL storage. This service is provided to applications through simple configuration steps.

³<https://cloud.google.com/appengine>

Actually, Azure offers two types of cloud services: IaaS and PaaS. It is not hard porting an IaaS application because this offer is based on virtual machines (VM). Just migrating a VM from one provider to another causes little impact on the systems being virtualized, as all that is needed for them to run is a copy of the virtual disk. The Open Virtualization Format (OVF⁴) makes this task even easier, by providing a standard format so that there is little effort to port a virtual machine from one provider to another, as long as both support this format. The main issue in this case is to choose a different provider that accepts the same VM format⁵.

However, in terms of PaaS, both Azure and GAE offer solutions based on Java servlets and JSP with NoSQL storage. Even if they allow the same set of technologies (Java based), applications implemented for them are not portable. This happens mainly because of the differences between their persistence models. (Section 3 discusses these models in detail.) Indeed, Gorton in a post⁶ at Software Engineering Institute's blog and Armbrust et al. (Armbrust et al., 2010) highlight that the differences in data management technologies make applications less reusable by different providers.

The next subsections present specific details of each platform persistence model, and finish with a discussion of the main issues found.

3.1 Google App Engine

Google App Engine DataStore is typically one of the first choices for big data applications. The DataStore is GAE's native API and its scalability is managed by the platform itself, which means that the user does not need to worry about the actual storage details.

GAE's DataStore offers two mechanisms to specify persistent entities: Java Data Objects⁷ (JDO) and Java Persistence API⁸ (JPA). The JDO and JPA interfaces are implemented using the Datanucleus⁹ platform, which is an open-source implementation of JDO and JPA. With JDO/JPA, GAE allows the definition of simple entity relationships. As a result, even without direct relational support from the actual

⁴OVF: <http://www.dmtf.org/standards/ovf>

⁵Paasify may be an interesting solution to select compatible PaaS: <http://www.paasify.it/vendors>.

⁶<http://blog.sei.cmu.edu/post.cfm/importance-software-architecture-big-data-systems-013>

⁷<http://www.oracle.com/technetwork/java/index-jsp-135919.html>

⁸<http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>

⁹<http://www.datanucleus.org/>

database system, applications can use GAE's DataStore to manage related entities.

For simplicity reasons, we chose JDO for this study. To persist an entity in GAE's DataStore, all that is necessary is to annotate a class according to the JDO specification. Related entities (one-to-one and one-to-many) are also managed by GAE automatically through proper annotations.

Let us consider a simple example: a clinical laboratory system must maintain a record of customers, doctors and examinations; each customer has one doctor, and each doctor may choose among a set of examinations to be performed.

The first step is to annotate the classes that represent persistent entities according to the JDO specification. After this, calls to JDO's CRUD¹⁰ methods can be used directly. In summary, all that is needed to make an entity persistent are some annotations. The actual storage of the entity and its related entities is performed by the platform.

It is important to stress that even with the possibility to define simple relationships through annotations, the GAE DataStore service is a NoSQL solution. If a relational solution is needed, a fully-fledged SQL solution, such as the MySQL-based service offered by GAE (Google Cloud SQL), is recommended. A trade-off between scalability and robustness is necessary in these cases.

3.2 Windows Azure

Windows Azure is the Microsoft's cloud platform, which offers different services, such as virtualization, storage and web hosting. Similarly to GAE, Azure's PaaS solution supports regular web-based applications (pages, controllers and other classes/libraries), but with a wider choice of languages (.Net, Node.js, PHP, Java, etc.).

Azure offers persistence through four main storage options:

- **Table Storage:** this is Azure's NoSQL persistence solution. It is a simple persistence model that allows applications to store basic data types (e.g., integer, string, boolean). It is a highly scalable solution, but with three major restrictions. First, Azure's Table Storage structures do not directly support relationships between entities. Second, there is a limit of 255 properties per entity, and every entity must define at least two properties for identification, which leaves 253 properties for general use. Third, data in a single entity cannot exceed one MByte.

¹⁰CRUD: Create, Retrieve, Update and Delete.

- **SQL Database:** formerly known as SQL Azure, this is a fully managed relational database service. Being a relational database, it is not as scalable as the NoSQL service.
- **SQL Server in Windows Azure VM:** if the developer wants more control over the DBMS, he may opt to deploy his own instance of SQL Server in a virtual machine. This renders more control, but also requires more effort to setup, manage the database server, and the virtual machine.
- **Blob Storage:** this service supports the storage of large, non-structured data. It has great scalability, but it is focused on files like audio and video.

As Azure also allows NoSQL services, which offer a good combination between storage and scalability for big data applications, we also chose this model in Azure for our study. However, unlike GAE JDO/JPA-based implementation, Azure does not have an official support for JPA/JDO. As a result, the developer has to deal with relationships manually. Additionally, there are many restrictions in Azure, for example: persistence is defined through inheritance, and not annotations as in GAE; the identification field (primary key) has to be manually managed.

In Azure Table Storage, entities are stored in table structures called partitions. One partition can store multiple entities, which may be of different types. An entity has a unique identification field. Partition names and identification fields are both strings, and are inherited by the entity classes.

To perform CRUD operations, the Table Storage API has some predefined methods. Listing 1 shows an example of how an entity can be persisted. The method "saveOrUpdate" (line 1) is used to either create or update an entity. First, a table client object is obtained ("tableClient"), based on some predefined connection string (line 2). Next, a table operation is created, in this case, to insert or replace an entity (line 3). Then, the table (partition) is created, if it does not exist already (line 5). Finally, the operation is executed (line 6). In this example, for simplicity, the name of the partition and of the entity class will be the same.

Listing 1: Persisting an entity in Azure.

```
1 public void saveOrUpdate(
    TableServiceEntity tse) {
2     CloudTableClient tableClient =
        CloudStorageAccount.parse(
            storageConnectionString).
            createCloudTableClient();
3     TableOperation tableOperation =
        TableOperation.
            insertOrReplace(tse);
4     try {
```

```

5      tableClient.getTableReference
      (tse.getClass().
        getSimpleName()).
        createIfNotExist();
6      tableClient.execute(tse.
        getClass().getSimpleName
        (), tableOperation);
7  } catch (StorageException e) {
      ... }
8  }

```

Dealing with relationships requires manual management of the id fields. For one-to-one relationships, it is possible to simply store the id field of the related (dependent) entity as a property in the container entity. For example, in the customer-has-a-doctor one-to-one relationship, to obtain the doctor for a given customer, first we obtain its id field, and then we perform a query in the Doctor table.

For one-to-many or many-to-many relationships, the strategy is to maintain a separate entity for relationships. Listing 2 illustrates the idea. In this example, “Relationship” (line 1) is a persistent entity that merely stores two string values: the “end1” and the “end2” (lines 2 and 3), each representing an end of the relationship. This entity will be stored in a partition of its own, called “Relationship” (line 5).

Listing 2: Persistent relationship in Azure.

```

1 public class Relationship extends
  TableServiceEntity {
2     private String end1;
3     private String end2;
4     public Relationship() {
5         this.partitionKey = "
          Relationship";
6     }
7     ... setters and getters ...
8 }

```

The “Relationship” entity from Listing 2 can be used to establish a relationship between any two entities. A Method “saveRelationship” realises a relationship between two entities, instantiating the “Relationship” entity and persisting it using calls to Azure’s API.

Once the relationship is established, retrieving all related entities can be done by searching through the “Relationship” partition. The example of Listing 3 shows a way to implement this strategy. The method “getAllRelatedEntities” (line 1) gets all entities related to a given entity. The id of the containing entity and the class of the related entity are provided as arguments. First, all relationships are retrieved (line 2) through the “getAll” method, which is not shown here but should be trivial to imagine. Then, the resulting list is iterated in search for instances that have

a matching “end1” property (lines 4-5). For those matching relationships, the instance corresponding to the “end2” is retrieved and added to the result (line 6). A method “retrieve”, which is not shown here, looks into the partition of the corresponding entity class and returns the instance itself.

Listing 3: Retrieving related entities in Azure.

```

1 public List getAllRelatedEntities(
  String end1Id, Class end2Class) {
2     List<Relationship> temp = getAll(
        Relationship.class);
3     List result = new ArrayList();
4     for (Relationship r : temp) {
5         if (r.getEnd1().equals(end1Id
        )) {
6             result.add(retrieve(
                end2Class, c.getEnd2
                ());
7         }
8     }
9     return result;
10 }

```

The implementation of Listing 3 is not very efficient, as it examines all relationships every time. However, it is not difficult to optimize this code with more refined structures such as trees or hash functions.

3.3 Difficulties in Conciliating Both Persistence Models

Although both GAE and Azure offer NoSQL services, GAE adds a layer that facilitates the management of relationships between persistent entities, while Azure demands some additional effort to be able to deliver similar functionality. The problem, however, is not the extra effort required by Azure. In fact, the jpa4azure¹¹ third-party API, adds an object-relational mapping layer to Azure, similar to what is natively available in GAE. (At the time we started our research, this API was not stable, at least according to our tests; so we decided to implement our own layer.) The problem, really, is that even allowing the use of the same set of technologies, the differences between the platforms impose specific programming styles on developing for each one. For this reason, the effort spent on specific programming tasks cannot be reused. Even considering the existence of a common API, the problem remains, due to the differences between the implementations and storage philosophies. Standardization could be an alternative, but as we discussed before, it is not the path followed in this work.

¹¹<https://jpa4azure.codeplex.com/>

Hence, despite the apparent similarities of the platforms (which use the same set of technologies: Java back-end, web-based front-end, and NoSQL persistence), the resulting applications have considerable differences. If for a small application like the one presented here the differences are so substantial, in a real case, managing thousands of persistent entities, the effort of developing such a system can increase very fast. If we consider other platforms, supporting different technologies such as Redis¹² or memcachedB¹³, the problem becomes even worse.

We argue that MDE can solve the portability problem in a more fundamental way, reaching flexibility levels that no API or standard can provide. The next section describes our proposal, based on a single platform-independent development model that hides the details of the platforms. This proposal also helps to reduce the extra effort needed by Azure, or any other platform that uses different technology.

4 SUPPORTING MULTIPLE PAAS PERSISTENCE MODELS USING MDE

This section presents a model implemented using the previously developed DSL, discusses the specific details of the generated code for GAE and Azure, gives a synthesis of the whole generation process, and offers some highlights on the work done.

Listing 4 presents the model for the clinical laboratory system. This example uses the language presented in a previous work (da Silva et al., 2013), which is summarized next. First, the model defines some basic configuration properties, such as the application name (line 1), visual theme (line 2), version (line 3), title (line 4), and a set of tabs to be displayed in the main interface (lines 5-10). Next are the entities and their relationships. The syntax is straightforward. Some points to highlight are the definition of the primary keys (lines 14, 27 and 34), which are inspired by JDO's annotations, and the possibility to define custom labels to be displayed in the main interface (line 36).

Listing 4: Model of the clinical laboratory system.

```
1 application weblab {
2   theme = "default"
3   version 1
4   title "WebLab - Exam Requests"
5   tab tab1 {
```

¹²<http://redis.io/>

¹³<http://memcachedb.org/>

```
6     title "Requests"
7     contains : Customer
8     contains : Examination
9     contains : Doctor
10  }
11 }
12
13 entity Customer {
14   pk { id:Key(strategy=IDENTITY)
15       readOnly=true }
16   property name : String
17   property address : String
18   property email : String
19   property phone1 : String
20   property phone2 : String
21   property birth : Date
22   property doctor : Doctor
23   property gender : String
24   property examinations : Examination
25   []
26 }
27
28 entity Examination {
29   pk { id:Key(strategy=IDENTITY)
30       readOnly = true }
31   property name : String
32   property material : String
33   property price : Double
34 }
35
36 entity Doctor {
37   pk { id:Key(strategy=IDENTITY)
38       readOnly= true }
39   property name : String
40   property nr : String title = "
41       License Number"
42 }
43 }
```

Listing 4 also shows the relationships established for this system. One customer has one doctor (line 21) and many examinations (line 23 - the [] suffix indicates that a property may have multiple instances). These appear in the model as properties mapped to other entities.

We developed two sets of transformations, one for GAE and another for Azure. A more generic view of this process can be seen in previous work (da Silva et al., 2013). Here we extend that description by detailing how persistence can be handled. The resulting transformations are to be collected to populate our repository.

4.1 Generating Persistence Code for GAE

Since GAE has JDO support, the transformations are not too difficult to define. One JDO-annotated Java class is generated for each persistent entity, including its properties and relationships. There is a single,

generic, non-generated data-access object (DAO) that performs basic CRUD operations. The invocations of the CRUD operations for each entity are generated in specific controller classes. One controller class is generated for each entity.

Listing 5 shows part of the generated controller class for the “Customer” entity in GAE. The method “saveCustomer” (line 3) persists a customer, given its properties and the doctor’s id. Among other actions, such as obtaining parameters from the HTTP request and dealing with errors and page re-directions, this controller method retrieves the corresponding doctor (line 5), associates it with the customer being persisted (line 6), and saves the instance (line 7). Please, note the calls to the generic DAO in lines 5 and 7.

Listing 5 also shows how one-to-many relationships are persisted. The method “addExaminationToCustomer” (line 11) first obtains the related entities, in this case, customer (line 13) and examination (line 14), then it adds the examination to the customer’s list of examinations (line 15), and finally it asks DAO to persist the customer and its examinations (line 16). Please note the calls to the generic DAO in lines 13, 14 and 16.

Listing 5: Generated Controller for GAE.

```

1 public class CustomerController {
2     ... // other controller actions
3     public void saveCustomer(Customer
4         c, int doctorId) {
5         ... // other actions
6         Doctor d = (Doctor)
7             GenericDAOJDO.INSTANCE.
8             retrieve(Doctor.class,
9             doctorId);
10        c.setDoctor(d);
11        GenericDAOJDO.INSTANCE.save(c
12            );
13        ... // other actions
14    }
15    public void
16        addExaminationToCustomer(int
17            customerId, int examinationId
18        ) {
19        ... // other actions
20        Customer c = (Customer)
21            GenericDAOJDO.INSTANCE.
22            retrieve(Customer.class,
23            customerId);
24        Examination e = (Examination)
25            GenericDAOJDO.INSTANCE.
26            retrieve(Examination.
27            class, examinationId);
28        c.addExamination(e);
29        GenericDAOJDO.INSTANCE.save(c
30            );
31        ... // other actions
32    }

```

19 }

4.2 Generating Persistence Code for Azure

For Azure, one class per persistent entity is generated. For basic CRUD operations, as well as for dealing with relationships manually, there is a single, generic, non-generated data-access object (DAO). Listings 1, 2 and 3 illustrate the idea of how this generic DAO works. Finally, invocations to the CRUD operations are generated in controller classes, similarly to GAE. Listing 6 shows part of the generated controller class for the “Customer” entity in Azure. It is similar to the GAE controller, with the following three differences:

- the relationship between customer and doctor (one-to-one) is based exclusively on the doctor’s id (line 5);
- ids need to be manually managed. In this case, and for simplicity, a random unique id is generated whenever a new entity is persisted (line 6);
- the relationship between customer and examination (one-to-many) is established by persisting a new relationship entity (line 13). This “saveRelationship” method is related to relationship shown in Listing 2.

Listing 6: Generated Controller for Azure.

```

1 public class CustomerController {
2     ... // other controller actions
3     public void saveCustomer(Customer
4         c, String doctorId) {
5         ... // other actions
6         c.setDoctor(doctorId);
7         c.setId(UUID.randomUUID().
8             toString());
9         TableStorage.INSTANCE.save(c)
10            ;
11        ... // other actions
12    }
13    public void
14        addExaminationToCustomer(
15            String customerId, String
16            examinationId) {
17        ... // other actions
18        TableStorage.INSTANCE.
19            saveRelationship(
20            customerId, examinationId
21        )
22        ... // other actions
23    }
24 }

```

4.3 The Code Generation Process

The architecture of the generated applications is similar for GAE and Azure. Two JSP pages (for editing and listing entities) are generated for each entity, as well as one persistent class and one controller. Figure 2 illustrates this architecture.

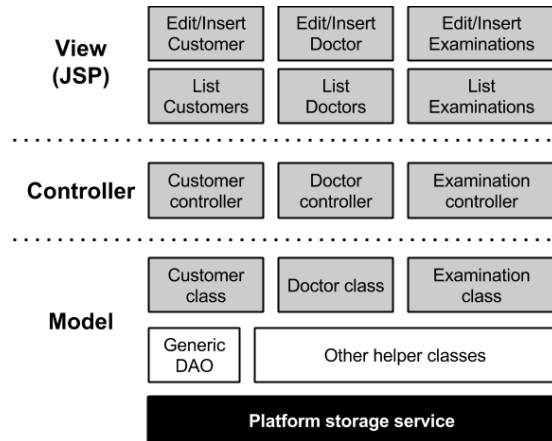


Figure 2: Architecture of the generated applications (GAE and Azure). Shaded elements are generated. White elements are non-generated. Black elements represent platform services.

However, there are significant differences between these two platforms. In GAE, there is no need to manually deal with relationships or identification fields. Hence, the generated application is simpler, with annotated entities and basic controller classes being generated straightforwardly. A simple, generic DAO was enough for basic CRUD operations.

For Azure, the generated applications are less simple. The extra layer to deal with relationships resulted in a more complex generic DAO. There is also the need to deal with identification fields.

5 FINAL DISCUSSION

The strategy presented here is similar to Object-relational mapping (ORM) frameworks like Hibernate. However, while Hibernate uses annotated java classes or a relational-entity model to generate SQL commands and other elements of the software such as classes, views, controllers and database structure, we use a DSL for modeling entities and generate MVC applications including annotated classes, based on specific details of each PaaS embedded in MDE transformations. Both strategies use code generation, but our approach can be considered as a level above and could even include ORM frameworks. Once the trans-

formations are defined, the developer no longer needs to worry about platform-specific details. As long as the transformations are correct, s/he only needs to work on the platform-independent model. In the end, applications developed with our approach can be made as portable as necessary, by including new transformations to support other platforms or technologies.

Adjustments and adaptations in the code generation process, if necessary, become less frequent over time, and the investment made through this extra effort eventually pays off. In this research, the initial infrastructure described here was built by a single developer in a period of 3.2 months, including the time to study the related technologies (Xtext¹⁴ and Xtend¹⁵).

From an evidence-based point of view (see for example (Tichy, 1998; Juristo and Moreno, 2010; Wohlin et al., 2000)), the case study discussed constitutes some evidence that it is possible to use our approach to deal with different persistence models at an higher level of abstraction and to port applications between different cloud providers. But to reinforce such evidence, we performed a more careful evaluation.

We defined a set of test cases, which 10 users executed on the same application generated for the two platforms (GAE and Azure). After executing the tests, the users perceived no difference in terms of functionality, what indicates an evidence that portability can be achieved by means of our approach. We also observed considerable gains in productivity, due to the automation power of MDE transformations.

From that evaluation, we concluded that it was possible to port an application between cloud platforms in such a way that the final users do not perceive the differences when using the two versions. This is particularly interesting if we consider that the underlying data management mechanisms are different, as discussed in Section 3. This promotes MDE as a possible alternative to port application between different cloud providers.

6 RELATED WORK

There are several different proposals for developing portable cloud applications, being standardization and open source software the more popular in the industry. In academia, many authors also attempt to use MDE to solve to lock-in problem.

Sharma and Sood (Sharma R., 2011) present a model-driven approach for interoperability in SaaS

¹⁴<https://eclipse.org/Xtext/>

¹⁵<http://eclipse.org/xtend/>

(Software-as-a-Service). They define the models at different abstraction levels, based on the separation of concerns between CIM (Computation Independent Model), PIM (Platform Independent Model) and PSM (Platform Specific Model), hence building on MDA¹⁶. Each level can be composed by one or more models to specify the structural, functional and behavioral aspects of a system. For PIMs a formal definition of the operations offered by the service is used, which can be accessed through an interface that must later be composed with other services to build a complete system. Business rules are specified through the declaration of restrictions, pre-conditions and post-conditions and invariants in OCL (Object Constraint Language). Transformations convert the PIM into a Web-Service Description Language (WSDL) PSM. The final step is the transformation of the WSDL PSM into WSDL specifications. The main difference from our work is that they use MDE to generate WSDL. Their approach is for SOA while ours is specific for cloud PaaS.

Miranda et al. present their vision on how MDE can support the development of adaptive multi-cloud applications, thus integrating MDE and Software Adaptation techniques (Miranda et al., 2013). Developers are requested to tag the components indicating in which cloud they will be deployed. MDE techniques are then applied to generate an XML-based cloud deployment plan. The source code and the XML deployment plan are processed to generate cloud compliant artifacts to access the underlying cloud services. This work aims at generating the deployment plan while our targets the design and development time.

MODAClouds¹⁷ (ModelDriven Approach for the design and execution of applications on multiple Clouds) aims at supporting system developers and operators in exploiting multiple clouds and in migrating their applications from cloud to cloud as needed (Ardagna et al., 2012). Its main objective is to provide methods, a decision support system, an open source IDE and runtime environment for the high-level design, early prototyping, semi-automatic code generation, and automatic deployment of applications on multiple clouds. It also helps administrators to monitor the services and measure their quality. While the project is developing a post-fact adoption standard (Petcu, 2011) with CloudML, a domain-specific modeling language and runtime environment that facilitates the specification of cloud application provisioning, deployment, and adaptation, we argue that each enterprise can build its own language or generation

strategy more aligned with their business.

The REMICS project proposes an approach for migrating legacy systems to the cloud (Mohagheghi and Sæ andther, 2011; Mohagheghi and Dehlen, 2008). Formed by a consortium of several research institutions, consulting and cloud users, the REMICS has a robust design. Its main purpose is to specify, develop and evaluate a tool for migrating services using MDE. The proposed migration process consists of understanding the legacy system in terms of its architecture and functionality, and designing a new Service-Oriented Architecture (SOA) application that provides the same or better functionality. This project is more related to reengineering and migration strategies for legacy applications while ours is for new ones.

All these approaches use MDE to protect the developer from platform details, which is one of the intended uses of MDE. Our approach focuses on PaaS portability, with special emphasis on persistence. Our results are similar to what is seen in the literature, combining the portability of MDE with its inherent productivity benefits, we expect that our efforts support the leveraging of this new computation model.

A strategy to solve the portability without MDE is described in (Giove et al., 2013). Giove et al. propose a library called CPIM (Cloud Provider Independent Model), that encapsulates PaaS-level services such as message queues, noSQL, and caching. Instead of relying on the providers following a standard, they add a mediation layer that hides the details of the underlying PaaS provider and exposes a common API that allows platform-independent code to be developed on top of it. The result is that applications can be more easily ported between providers, as long as both sides of the implementation (application and supporting library) comply with the mediation layer. Their currently supported platforms are GAE and Azure, but new platforms can be added by providing a proper library to the layer.

Both our approach and the CPIM library attempt to deal with the differences between PaaS services. Both agree that standardization may not be the only solution. And both allow platform-independent applications to be specified. Our proposal has the advantage of allowing developers to work on a higher abstraction level. Therefore, we can collect additional benefits in terms of productivity and maintenance. On the other hand, CPIM requires no effort to setup a modeling and code generation environment, resulting in less upfront investment and being easier to adopt. In fact, an hybrid solution, combining MDE and a mediation layer, could bring benefits from both approaches.

¹⁶<http://www.omg.org/mda/>

¹⁷<http://www.modacLOUDS.eu/>

More research issues and approaches related to the development of systems to the cloud model can be found in Armbrust et al. (Armbrust et al., 2009) and our previous work (da Silva et al., 2013). Cloud computing is still evolving, and research opportunities are still being identified. The presented approaches are still being investigated and are far from being mature. More research and evaluations are still necessary.

7 CONCLUDING REMARKS AND FUTURE WORK

This paper shows how the differences in cloud persistence models can make an application difficult to reuse and/or be ported to a different provider. It extends our previous work (da Silva et al., 2013) on exploring the use of MDE to overcome portability in cloud computing, and shows how that previous approach can be used to solve the persistence related lock-in issue.

The main contribution of our work is to show that there is an alternative path to the standardization of cloud technologies. MDE can increase the portability of the applications, but it can also lead to additional benefits inherently associated with it, consequently, reducing the impacts of lock-in.

Our approach is focused on persistence, and therefore it has good support for CRUD operations.

A limitation of our approach, that is inherent to most MDE approaches, is that if the generated code needs to be adapted or modified, the MDE life-cycle can be broken. Changes in the generated code have to be replicated, either in the models or in the transformations, which is not a trivial task. This is why it is often recommended to leave generated code unmodified¹⁸.

In the near future we plan to include more platforms to implement the repository of models and transformations, and to perform some more evaluations, which includes applying our approach to other case studies.

ACKNOWLEDGEMENTS

We would like to thank FAPESP (processes 2012/24487-3 and 2012/04549-4), Coordination of Superior Level Staff Improvement - CAPES and

Brazil-Europe Erasmus Mundus project (process BM13DM0002) for partially funding this research.

REFERENCES

- Alkhatib, H., Faraboschi, P., Frachtenberg, E., Kasahara, H., Lange, D., Laplante, P., Merchant, A., Milojevic, D., and Schwan, K. (2014). IEEE CS 2022 Report.
- Antkiewicz, M. and Czarnecki, K. (2006). Framework-specific modeling languages with round-trip engineering. *Model Driven Engineering Languages and Systems*, pages 692–706.
- Ardagna, D., Di Nitto, E., Mohagheghi, P., Mosser, S., Ballagny, C., D’Andria, F., Casale, G., Matthews, P., Nechifor, C.-S., Petcu, D., and Others (2012). Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds. In *Modeling in Software Engineering (MISE), 2012 ICSE Workshop on*, pages 50–56. IEEE.
- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 28*.
- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.
- Bozman, J. (2010). Cloud Computing: The Need for Portability and Interoperability. *IDC Analyze the Future, Sponsored by Red Hat, Inc.*
- Brambilla, M., Cabot, J., and Wimmer, M. (2012). Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering*, 1(1):1–182.
- Chen, Y., Li, X., and Chen, F. (2011). Overview and analysis of cloud computing research and application. In *E-Business and E-Government (ICEE), 2011 International Conference on*, pages 1–4.
- da Silva, E. A. N., Fortes, R. P. M., and Lucredio, D. (2013). A Model-Driven Approach for Promoting Cloud PaaS Portability. In *Annual International Conference on Software Engineering-CASCON*.
- Deursen, V., Klint, A., and Paul and Visser, J. (2000). Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):26–36.
- France, R. and Rumpe, B. (2007). Model-driven Development of Complex Software: A Research Roadmap. In *2007 Future of Software Engineering, FOSE ’07*, pages 37–54, Washington, DC, USA. IEEE Computer Society.
- Giove, F., Longoni, D., Yancheshmeh, M. S., Ardagna, D., and Di Nitto, E. (2013). An approach for the development of portable applications on paas clouds. *Proceedings of CLOSER*, pages 591–601.
- Hettel, T., Lawley, M., and Raymond, K. (2008). Model synchronisation: Definitions for round-trip engineering. *Theory and Practice of Model Transformations*, pages 31–45.

¹⁸There are some efforts to solve the inconsistencies between changes made manually in generated code (Antkiewicz and Czarnecki, 2006; Hettel et al., 2008). Such research area is often called round-trip engineering.

- Juristo, N. and Moreno, A. M. (2010). *Basics of software engineering experimentation*. Springer Publishing Company, Incorporated.
- Khajeh-Hosseini, A., Sommerville, I., Bogaerts, J., and Teregowda, P. (2011). Decision Support Tools for Cloud Migration in the Enterprise. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 541–548.
- Kleppe, A., Jos, W., and Wim, B. (2003). *MDA Explained, The Model-Driven Architecture: Practice and Promise*. Addison-Wesley.
- Miranda, J., Guillén, J., Murillo, J. M., and Canal, C. (2013). Development of Adaptive Multi-cloud Applications - A Model-Driven Approach. In *Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development*, pages 321–330. SciTePress - Science and Technology Publications.
- Mohagheghi, P. and Dehlen, V. (2008). Where is the proof? - A review of experiences from applying MDE in industry. In *Model Driven Architecture—Foundations and Applications*, pages 432–443.
- Mohagheghi, P. and Sæ andther, T. (2011). Software Engineering Challenges for Migration to the Service Cloud Paradigm: Ongoing Work in the REMICS Project. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 507–514.
- Petcu, D. (2011). Portability and interoperability between clouds: challenges and case study. In *Towards a Service-Based Internet*, pages 62–74. Springer.
- Petcu, D., Macariu, G., Panica, S., and Crăciun, C. (2013). Portable cloud applications—from theory to practice. *Future Generation Computer Systems*, 29(6):1417 – 1430. Including Special sections: High Performance Computing in the Cloud & Resource Discovery Mechanisms for {P2P} Systems.
- Petcu, D. and Vasilakos, A. V. (2014). Portability in clouds: approaches and research opportunities. *Scalable Computing: Practice and Experience*, 15(3).
- Ranabahu, A. and Sheth, A. (2010). Semantics Centric Solutions for Application and Data Portability in Cloud Computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 234–241.
- Schmidt, D. C. (2006). Model-driven engineering. *Computer-IEEE computer society-*, 39(2):25.
- Sharma R., S. M. S. D. (2011). Modeling cloud SaaS with SOA and MDA. *Communications in Computer and Information Science*, 190 CCIS(PART 1):511–518.
- Shirazi, M. N., Kuan, H. C., and Dolatabadi, H. (2012). Design Patterns to Enable Data Portability between Clouds’ Databases. In *Computational Science and Its Applications (ICCSA), 2012 12th International Conference on*, pages 117–120.
- Tichy, W. F. (1998). Should computer scientists experiment more? *Computer*, 31(5):32–40.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA.

A Lightweight Tool for Anomaly Detection in Cloud Data Centres

Sakil Barbhuiya, Zafeirios Papazachos, Peter Kilpatrick and Dimitrios S. Nikolopoulos

*School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN, Belfast, U.K.
{sbarbhuiya03, z.papazachos, p.kilpatrick, d.nikolopoulos}@qub.ac.uk*

Keywords: Anomaly Detection, Cloud Computing, Data Centres, Monitoring, Correlation.

Abstract: Cloud data centres are critical business infrastructures and the fastest growing service providers. Detecting anomalies in Cloud data centre operation is vital. Given the vast complexity of the data centre system software stack, applications and workloads, anomaly detection is a challenging endeavour. Current tools for detecting anomalies often use machine learning techniques, application instance behaviours or system metrics distribution, which are complex to implement in Cloud computing environments as they require training, access to application-level data and complex processing. This paper presents LADT, a lightweight anomaly detection tool for Cloud data centres that uses rigorous correlation of system metrics, implemented by an efficient correlation algorithm without need for training or complex infrastructure set up. LADT is based on the hypothesis that, in an anomaly-free system, metrics from data centre host nodes and virtual machines (VMs) are strongly correlated. An anomaly is detected whenever correlation drops below a threshold value. We demonstrate and evaluate LADT using a Cloud environment, where it shows that the hosting node I/O operations per second (IOPS) are strongly correlated with the aggregated virtual machine IOPS, but this correlation vanishes when an application stresses the disk, indicating a node-level anomaly.

1 INTRODUCTION

Data centres that host Cloud computing services are catalysts for the economy and for society. Despite the success and proliferation of the Cloud computing paradigm, hosting data centres face immense challenges in terms of managing the scale and complexity of their hardware infrastructure and their system software stack. The hosted workloads also become increasingly complex and diverse. The complexity of the data centre ecosystem gives rise to frequent disruption of service, which manifests as lack of responsiveness, lower than expected performance, or complete disruption of service. Anomaly detection in data centres is a recently emerged field of research which explores methods to automate the detection and diagnosis of service disruption in data centres via the monitoring of system generated logs and metrics.

Log analytic tools (Lou et al., 2010; Xu et al., 2009) extract features from logs and use statistical learning techniques to automatically build models that detect and diagnose system anomalies in data centres. These models are not easy to understand by human operators as they may detect problems in a high dimensional feature space without providing meaningful explanations for the detected problems. Besides, learning-based tools require a log parser for mining

the console logs in order to create the features of the model. Log parsers require the source code of hosted applications in order to recover the inherited syntax of logs, and source code may not always be accessible for legacy applications. As an alternative to using console logs, a number of anomaly detection tools use system metrics (Tan et al., 2012; Wang, 2009; Ward and Barker, 2013; Kang et al., 2010; Jiang et al., 2009). Such metrics can be collected with minimum overhead and without requiring access to the source code of hosted applications.

Some tools (Wang, 2009; Kang et al., 2010) use system metrics for detecting anomalies in Cloud data centres and are developed to take the elasticity of Cloud computing environments into consideration. These tools provide higher accuracy and effectiveness of detection. However, implementing them in a large-scale Cloud system is complex as the tools are configured in multiple layers of data monitoring and analysis. EbAT (Wang, 2009) implements metrics distribution, entropy time series construction, and entropy time series processing across multiple monitoring layers, before deploying an online tool for detecting anomalies. Peerwatch (Kang et al., 2010) extracts correlated behaviours between multiple instances of individual applications and then applies those extracted correlations to identify anomalies. The use of

application-level metrics in anomaly detection algorithms may raise enormously the volume of data to process in a large-scale system.

This paper introduces LADT (Lightweight Anomaly Detection Tool), a lightweight tool which monitors system-level and virtual machine (VM)-level metrics in Cloud data centres to detect node-level anomalies using simple metrics and correlation analysis. LADT addresses the complexity of implementing efficient monitoring and analysis tools in large-scale Cloud data centres, by collecting and storing the metrics generated by nodes and VMs using Apache Chukwa (Rabkin and Katz, 2010). The LADT algorithm performs correlation analysis on the collected data using Apache Pig (Olston et al., 2008) and MapReduce jobs in order to complete the analysis in a timely manner.

As a use case, we deploy LADT to add a new dimension in anomaly detection for Cloud data centres, namely correlation analysis between node-level disk I/O metrics and VM-level disk I/O metrics. LADT implements a simple algorithm, which uses Pearson's correlation coefficient. When the average correlation coefficient value for a number of consecutive time intervals drops below a threshold, an alarm is raised to indicate an anomaly. The metrics data required in this use case are limited to disk I/O activity data and the algorithm to detect the anomalies is lightweight. The major contribution of LADT over other tools in this context is the ability to efficiently monitor and detect anomalies in a Cloud data centre without requiring complex algorithms, application source code availability, or complex infrastructure set up.

We evaluate LADT in a lab-based Cloud ecosystem, where it continuously collects and stores system metrics from all nodes and VMs. The tool shows that the disk I/O metrics in each hosted node are strongly correlated with the aggregate VM disk I/O metrics, but this correlation vanishes when a disk-stressing application is introduced in a node, indicating a node-level anomaly.

The remainder of the paper is organised as follows. Section 2 presents background and related work in anomaly detection. Section 3 and Section 4 provide detail of the LADT architecture and algorithm, respectively. Experimental results are presented and discussed in Section 5. Section 6 concludes the paper and discusses future work.

2 BACKGROUND AND RELATED WORK

In this section we describe the challenges of detecting

anomalies in Cloud data centres. A number of different tools and methods are considered and placed into separate categories based on the input they analyse. We consider two cases for monitoring and detecting anomalies: console log based anomaly detection and anomaly detection based on system metrics.

2.1 Anomaly Detection Challenges

Cloud data centres are implemented as large-scale clusters with demanding requirements for service performance, availability and cost of operation. As a result of scale and complexity, data centres exhibit large numbers of system anomalies caused by operator error (Oppenheimer et al., 2003), resource over/under provisioning (Kumar et al., 2007), and hardware or software failures (Pertet and Narasimhan, 2005). These anomalies are inherently difficult to identify and resolve promptly via human inspection (Kephart and Chess, 2003). Thus, automatic system monitoring that captures system state, behaviour and performance becomes vital. Computer system logs are the main source of information for anomaly detection. Logs can be of two types: structured or unstructured. Unstructured logs are free-form text strings, such as console logs, which record events or states of interest and capture the intent of service developers (Lou et al., 2010), whereas structured logs are numerical logs, such as logs of system metrics, which capture workload and system performance attributes, such as CPU utilisation, memory usage, network traffic and I/O.

2.2 Console Log based Anomaly Detection

Analytic tools for anomaly detection based on console logs, such as SEC (Rouillard, 2004), Logsurfer (The and Prewett, 2003) and Swatch (Hansen and Atkins, 1993) check logs against a set of rules which define normal system behaviour. These rules are manually set by developers based on their knowledge of system design and implementation. However, rule-based log analysis is complex and expensive because it requires significant effort from system developers to manually set and tune the rules. Moreover, modern systems consisting of multiple components developed by different vendors and the frequent upgrades of those components make it difficult for a single expert to have complete knowledge of the total system and to set the rules effectively. This complexity has given rise to statistical learning based log analytic tools such as the works of Lou et al. (Lou et al., 2010) and Xu et al. (Xu et al., 2009), which extract features from

console logs and then use statistical techniques to automatically build models for system anomaly identification.

Lou et al. (Lou et al., 2010) propose a statistical learning technique which consists of a learning process and a detection process. The learning process groups the log message parameters and then discovers the invariants among the different parameters within the groups. For new input logs, the detection process matches their invariants among the parameters with learned invariants from the learning process. Each mismatch in the invariants is considered to be anomalous. Xu et al. (Xu et al., 2009) propose a new methodology to mine console logs to automatically detect system problems. This first creates feature vectors from the logs and then applies the PCA (Principal Component Analysis) algorithm on the feature vectors to detect anomalies. However, the learning based tools require a custom log parser for mining the console logs in order to create the features for the learned model. The log parsers require source code of the hosted applications to recover the inherent structure of the logs.

2.3 System Metric based Anomaly Detection

A number of anomaly detection tools use system metrics (Tan et al., 2012; Wang, 2009; Ward and Barker, 2013; Kang et al., 2010; Jiang et al., 2009), which can be collected with minimum overhead and without requiring any access to the source code of hosted applications. Using system metrics for detecting anomalies has advantages over traditional log-based anomaly detection tools due to consideration of elasticity and workload evolution in Cloud computing, but also due to provisioning, scaling, and termination of services in short periods of time. Some of these tools are based on feature selection and machine learning outlier detection to flag anomalies (Azmandian et al., 2011).

EbAT (Wang, 2009) is a tool that uses entropy based anomaly detection. EbAT analyses metric distributions and measures the dispersal or concentration of the distributions. The metrics are aggregated by entropy distributions across the Cloud stack in order to form entropy time-series. EbAT uses online tools like spike detection, signal processing and subspace methods to detect anomalies in the entropy time-series. The tool incurs the complexity of analysing the metric distributions and also requires third party tools to detect anomalies.

PeerWatch (Kang et al., 2010) uses canonical correlation analysis (CCA) to extract the correlations between multiple application instances, where attributes

of the instances are system resource metrics such as CPU utilisation, memory utilisation, network traffic etc. PeerWatch raises an alarm for an anomaly whenever some correlations drop significantly. As a result of analysing the application instance behaviours and correlating them, this tool is capable of detecting application-level or VM-level anomalies. However, this approach requires statistical metrics analysis and knowledge of the hosted applications, which is a limitation in large-scale Clouds, where hundreds of different types of applications run on the VMs.

Varanus (Ward and Barker, 2013) uses a gossip protocol, which is layered into Clouds, groups and VMs in order to collect system metrics from the VMs and analyse them for anomalies. This approach allows in-situ collection and analysis of metrics data without requiring any dedicated monitoring servers to store the data. However, setting up a dedicated gossip protocol across thousands of VMs in a large-scale Cloud environment and maintaining the gossip based overlay network over each of the VMs is a challenging task.

The metric-based black box detection technique presented in (Tan et al., 2012) uses the LFD (Light-Weight Failure Detection) algorithm to detect system anomalies. LFD raises an alarm when there is a lack of correlation between two specific system metrics. The anomaly indicates a system problem and each such problem is associated with a specific system metrics pair. LFD is a lightweight algorithm with lower complexity than EbAT, PeerWatch and Varanus. Furthermore, LFD does not require any training or source code and understanding of hosted applications. The LFD follows a decentralised detection approach, where each node analyses its own system metrics in order to achieve higher scalability. However, this may also limit LFD in large-scale Cloud data centres, where it may not be feasible to implement LFD on each node individually, due to overhead.

In this paper we address the limitations of existing system anomaly detection tools by introducing LADT. LADT uses Apache Chukwa (Rabkin and Katz, 2010) for collecting metrics data from all nodes and VMs in a data centre, and HBase (Vora, 2011) for storing the data in servers to allow centralised monitoring of Cloud systems. LADT implements a new correlation algorithm to perform the correlation analysis on the centrally stored metrics data. The LADT algorithm correlates node-level and VM-level metrics, which is a new approach to correlation analysis in detecting Cloud system anomalies. Furthermore, the LADT algorithm deals with the synchronisation problem between the node and VM generated metrics timestamp. This problem arises due to

latency in storing the VM-level metrics in the monitoring server and results in poor correlation analysis. LADT is lightweight as it uses a simplified infrastructure set-up for metrics data monitoring and the LADT algorithm uses the simple Pearson correlation coefficient for analysing the metrics data. We program the algorithm using Apache Pig (Olston et al., 2008) to leverage MapReduce jobs in order to achieve higher throughput. We use disk I/O metrics from both nodes and VMs in an actual Cloud set-up to detect I/O performance anomalies.

3 LADT ARCHITECTURE

The following sub-sections describe the architecture of the LADT tool and its functionality.

3.1 Metrics Data Monitoring

LADT utilises an agent-based monitoring architecture to retrieve system metrics from the hosting nodes and VMs in a Cloud data centre. The monitoring agent extracts system metrics from the nodes and VMs at regular time intervals. The collector gathers the data extracted by the agents. LADT uses the agents and collectors provided by Apache Chukwa's (Rabkin and Katz, 2010) runtime monitoring. The Chukwa agent collects CPU, memory, disk, and network information from the hosting nodes using sigar (Sigar, 2014) and from the VMs using Virt-Top (Virt-Top, 2014). The Chukwa collector then collects the output generated by the agents. The collector processes the data and registers the input to HBase, which is installed in the monitoring node.

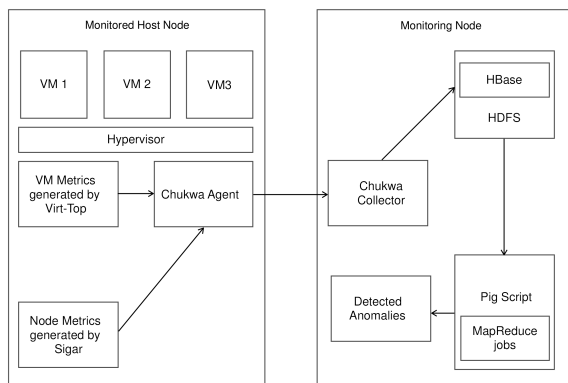


Figure 1: LADT Architecture.

Figure 1 illustrates the architectural overview of LADT. The tool installs one Chukwa agent for collecting both node-level and VM-level metrics on each monitored node in the data centre. LADT uses

Chukwa collectors running on data analysis servers, for collecting node-level and VM-level metrics into HBase. The correlation analysis on the stored metrics is performed with Pig Script. Each Chukwa agent consists of adaptors which are dynamically loadable modules that run inside the agent process. LADT sends the metrics from the agents to the Chukwa collectors via HTTP. The primary task of the collector is to parse the collected data from the agent and store the extracted information in an HBase database. HBase runs on top of the Hadoop Distributed File System (HDFS) (Shvachko et al., 2010).

3.2 Metrics Data Analysis

LADT runs the metrics correlation analysis on the stored metrics using Apache Pig (Olston et al., 2008), which in turn executes MapReduce jobs. A Pig Script written in Pig Latin implements the LADT algorithm to explore the correlation between the node-level and VM-level metrics and to detect anomalies. The Pig Script program first loads both the node-level and VM-level system metrics from HBase. It then takes the mean values of the metric samples in 15-second windows. The program groups the mean values into 5-minute windows and calculates the Pearson Correlation Coefficient between the node-level and VM-level metrics in each group. Finally, the program compares the correlation coefficient value with the threshold value and generates an anomaly alarm if it finds the coefficient value is below an adjustable threshold level.

4 LADT Hypothesis and Algorithm

The underlying approach of LADT is based on the premise of the LFD technique, which identifies two metrics that are correlated during normal operation but diverge in the presence of an anomaly.

4.1 LFD: The Baseline Method

LFD is a lightweight technique for anomaly detection proposed by Tan et al. (Tan et al., 2012). The hypothesis of LFD is that in an anomaly-free system, whenever an application requests service, the processing alternates between two phases: the communication phase and the compute phase. In the communication phase, the application responds to requests received from the user, reads data from the disk, or writes data to the disk. In the compute phase, the application operates on the received inputs or requests. At the operational level, the compute phase is characterised by

user-space CPU activity, whereas the communication phase is characterised by the behaviour of one or more types of system-level resource consumption, including kernel-space CPU time, disk I/O or network I/O. Behavioural change between the two phases results in a correlation between user-space CPU utilisation and system resource consumption. The LFD also hypothesises that in the case of an anomaly, there will be a significant change in the relationship between the compute phase and the communication phase. Hence, there will be lack of correlation between user-space CPU utilisation and system resource consumption, based on which anomalies can be detected in the system.

4.2 LADT Hypothesis

LADT formulates a new hypothesis, according to which there is strong correlation between the node-level and VM-level metrics in a Cloud system. Also, that this correlation will drop significantly in the event of any performance anomaly at the node-level and a continuous drop in the correlation can indicate the presence of a true anomaly in the node.

4.3 LADT Algorithm

We propose a new algorithm based on LFD, to correlate node-level and VM-level metrics. We use disk I/O metrics as a running example. The algorithm correlates disk I/O between the hosting node and the VMs in order to detect anomalies in IaaS Cloud environments. LADT computes the Pearson correlation coefficient (ρ) between the hosting node disk IOPS and the aggregated VM disk IOPS. Pearson's correlation coefficient is the ratio of the covariance between the two metrics to the product of their standard deviations as described in Equation 1 and ranges between -1.0 and 1.0.

$$\rho_{N,V} = \frac{\text{covariance}(N,V)}{\sigma_N \sigma_V} \quad (1)$$

where N = time-series of node disk IOPS
 V = time-series of VM disk IOPS

Similar to the LFD algorithm, there are five tunable parameters in the LADT algorithm, which are summarised in Table 1. LADT collects raw disk I/O metrics with a 3 second period from both the nodes and VMs. The metrics collection period is set to more than a second in order to mitigate the timestamp synchronisation problem between the node-level and VM-level metrics, which arises as a result of latency in storing VM-level metrics in the LADT monitoring server. Before calculating the correlation coefficient

between the metrics, their mean values are taken in small windows ($LW = 5$) in order to reduce noise. An anomaly alarm is raised when the average of D consecutive values of the correlation coefficient drops below the threshold T in order to detect the true anomalies by considering a larger range of system behaviour.

We keep the same parameter values as the LFD algorithm for four parameters ($LW = 5, LS = 5, W = 60, D = 10$) (Tan et al., 2012), which we experimentally found to achieve best performance. However, the correlation window slide, S is changed from 12 to 60 as this amortises better the overhead of a Pig (Hadoop) program, which is used to execute the LADT algorithm. Therefore, each correlation coefficient value considers the last $LW \times W = 300$ seconds (5 minutes) of system behaviour.

5 EXPERIMENTAL EVALUATION

This section describes the workload and the experimental set-up used to evaluate LADT. The section also analyses our experimental results and the functionality of the LADT tool.

5.1 Experimental Set-up

For evaluating LADT we have established a Cloud testbed with three compute nodes: Host1, Host2, and Host3 on three Dell PowerEdge R420 servers. Each node is running CentOS 6.5 with the 2.6.32 Linux kernel and has 6 cores, 2-way hyper-threaded, clocked at 2.20 GHz with 12GB DRAM clocked at 1600 MHz. Each node includes a 7.2K RPM hard drive with 1TB of SATA and an onBoard Broadcom 5720 Dual Port 1GBE network card. We run three VMs on each compute node using KVM. Each VM runs the disk I/O intensive Data Serving benchmark from CloudSuite (Ferdman et al., 2012). Disk read/write metrics are generated every three seconds in the nodes using sigar (Sigar, 2014) and in the VMs using Virt-Top (Virt-Top, 2014).

The experiment runs over a time period of 60 minutes, where we inject an anomaly in Host1 at the end of the first 30 minutes. We use a disk stressing application which increases the disk read/write operations and runs for two minutes with a two minute interval between runs for the remaining 30 minutes. This anomaly reflects a Blue Pill or a Virtual Machine Based Rootkit (VMBR) attack on a Cloud system, where the attacker introduces fake VMs via a hidden hypervisor on the victim hosting node to get access to the hardware resources such as CPU, memory, network or disk (Dahbur et al., 2011). This anomaly may

Table 1: Tunable parameters in LADT Algorithm.

LW , low-pass window width	raw samples to take mean
LS , low-pass window slide	raw samples to slide
W , correlation window width	samples to correlate
S , correlation window slide	samples to slide detection window
D , diagnosis window	correlation coefficients to consider

also represent a distributed denial-of-service (DDoS) attack, which forces the cloud service to consume system resources such as processor power, memory, disk space, or network bandwidth to an intolerable level (Rajasekar and Imafidon, 2010).

Earlier research (Antunes et al., 2008) has already used system resource-level anomaly analysis to deal with such attacks. The work of Antunes et al. (Antunes et al., 2008) analyses system resource utilisation to explore the normal system behaviour and builds a model, based on which it detects the abnormal behaviours in the system, and subsequently, the attacks. However, this approach to detecting attacks requires a large amount of historical data and use of machine learning techniques. In contrast, LADT can detect the attacks using correlation analysis between the node-level and the VM-level metrics.

LADT is implemented in the testbed, which uses Chukwa agents in each of the hosting nodes to collect both the hosting node and VM disk read/write metrics. The tool stores the collected metrics in the HBase running in the monitoring node, which is an Intel Xeon E5-2650 server. Finally, in the monitoring node, LADT analyses the stored metrics by running the algorithm programmed in Pig Latin, which calculates the correlation between the individual hosting node disk I/O operations and the aggregated disk I/O operations of all the VMs in that node, to detect the anomaly injected in Host1.

5.2 Results and Discussion

We provide experimental results for each host in time-series of total disk I/O operations per second (IOPS) and correlation coefficient values between the node disk IOPS and aggregated disk IOPS of all the VMs in the node. We present the normalised values of the IOPS, which are the mean values in small windows of 15 seconds, including 5 samples of metrics data (the frequency of metrics collection is 3 seconds). The correlation coefficient values are calculated in correlation windows of 5 minutes, covering 5 minutes of metrics data. Hence, there are 12 correlation intervals in the 60 minutes of experiment.

The results presented in Figure 2 shows that Host1's disk IOPS remains around 400 for the first 30 minutes and starts fluctuating in the next 30 min-

utes as a result of the injected anomaly, whereas the aggregated disk IOPS of the VMs remains below 200 throughout the experiment. The reason for the lower values of the aggregated disk IOPS of the VMs is the use of the extra software layer of the hypervisor, which is interposed between guest operating systems and hardware (Li et al., 2013). The hypervisor multiplexes I/O devices by requiring guest operating systems to access the real hardware indirectly and hence induces an overhead in the I/O context.

The correlation analysis for Host1 in Figure 2 clearly shows that there is a strong correlation between the hosting node IOPS and the aggregated IOPS of the VMs for the first half of the experiment, where the correlation coefficient values are often above 0.5 with an average value of 0.6. However the correlation value drops below 0.0 suddenly at the sixth interval when the disk stress starts. The coefficient value remains very low throughout the second half with an average value of -0.02. This is a clear reflection of the injected anomaly in Host1, which distorts the correlation between the hosting node IOPS and aggregated IOPS of the VMs.

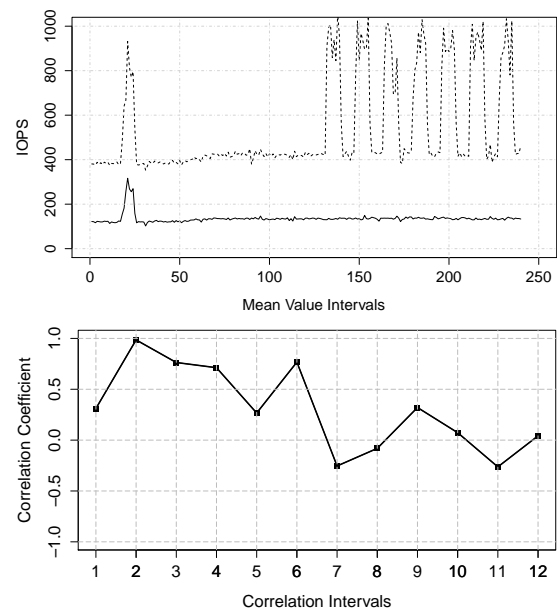


Figure 2: (Top) Time-series of node disk IOPS (dashed) and aggregated VM disk IOPS (solid) in Host1. (Bottom) Correlation coefficients between the two time-series measurements.

Figure 3 shows that for Host2, the node IOPS and aggregated IOPS of the VMs are steady and their correlation coefficient average value is 0.4. The low average is due to the temporary drops, which happen because of the fluctuation in the overhead of I/O operations in the VMs (Li et al., 2013). Similar behaviour is exhibited by Host3 (Figure 4), where in fact there are more correlation coefficient values above 0.5 and a better average correlation coefficient value of 0.5.

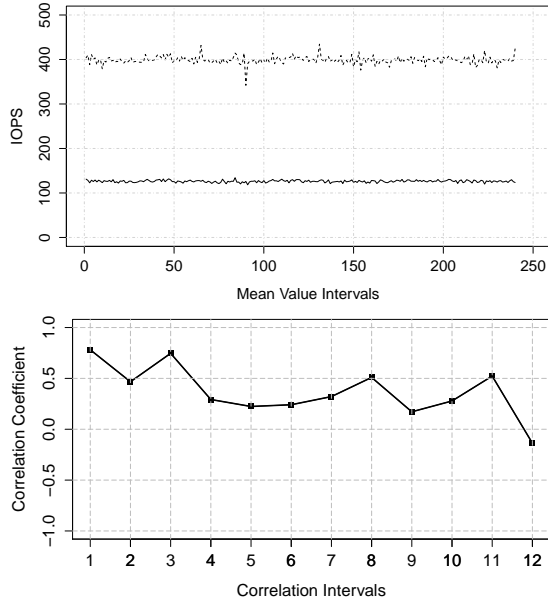


Figure 3: (Top) Time-series of node disk IOPS (dashed) and aggregated VM disk IOPS (solid) in Host2. (Bottom) Correlation coefficients between the two time-series measurements.

Experimental results also reveal that when using LADT for disk I/O metric data collection and correlation analysis, a latency of 10 minutes is observed, where a 5 minute latency is required to hold a historical metrics data set of a 5 minute window and 5 additional minutes are used to analyse the metrics data. This latency is comparable to the latency of on-line data centre service troubleshooting tools (Wang, 2009) and, as yet, benefits from no optimisation of the analysis workflow at the server, which we are exploring in ongoing work.

5.3 LADT Overhead

Our next experiment assesses LADT in terms of the overhead that it introduces on hosted application VMs, because of the LADT agents that collect metrics simultaneously with the execution of application workloads. To investigate this we executed a test where a single VM runs a data serving benchmark for

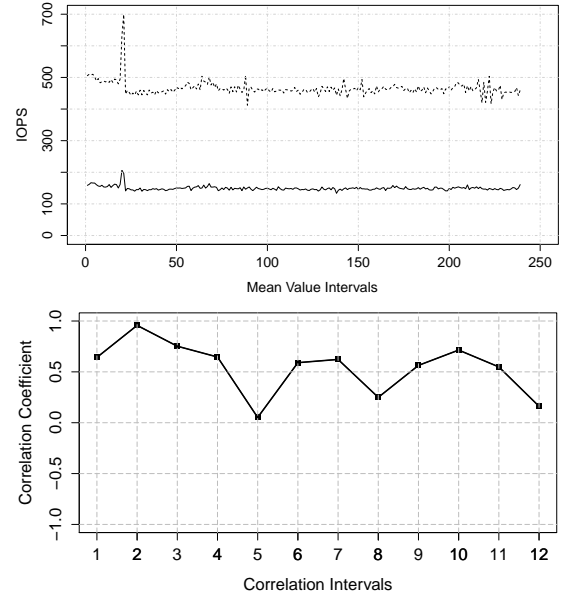


Figure 4: (Top) Time-series of node disk IOPS (dashed) and aggregated VM disk IOPS (solid) in Host3. (Bottom) Correlation coefficients between the two time-series measurements.

the duration of 10000 data operations with a rate of 200 operations/sec. With an agent running concurrently with the data serving benchmark, and during an execution interval of 50 seconds, the average update and read latencies of the benchmark were 0.21 ms and 8.28 ms, respectively. With no agent running and during the same 50 second execution interval, the average update and read latencies were 0.21 ms and 6.97 ms, respectively. In both cases we observed the expected response time from the benchmark and any differences observed in the average latencies of the update and read operations were justified by the variability introduced by the storage medium.

5.4 Further Analysis

We observe that during normal operation there is a strong correlation between the node disk IOPS and aggregated disk IOPS of the VMs in all three hosting nodes. However, the correlation becomes weaker during some intervals for short periods. This happens because in these intervals the overhead on the VM I/O operations resulting from accessing the disk indirectly via the hypervisor (Li et al., 2013) rises unpredictably and degrades the VM IOPS with respect to the corresponding node IOPS. Although the correlation coefficient value drops in some intervals even when the hosts are in a stable expected state, this drop is not as significant as it is in the case of an anomaly. Moreover, the correlation coefficient average drops below

0 when an anomaly occurs, whereas the average coefficient value ranges between 0.4 and 0.6 when the hosts are anomaly-free. We also observe that, even though all three hosts are running identical workloads in their VMs, both the IOPS and the correlation coefficient averages vary across the hosts. This again happens because of the varying overhead in the IOPS due to the hypervisor layer between the hosts and the VMs.

We conclude that the correlation coefficient values require normalisation in order to avoid false alarms for anomaly, which could arise because of a fluctuation in the overhead on VM IOPS. We detect the true anomalies by considering a larger period of system behaviour and this is done by taking the average of D consecutive coefficient values and checking if it is below the threshold value, T . The values of T and D depend on how often the user wishes to get an alarm for the anomalies. From the results, we observe the anomaly as the security attack in Host1 when the correlation coefficient value drops significantly and stays low for a longer period of time.

6 CONCLUSION AND FUTURE WORK

We presented LADT, a lightweight anomaly detection tool for Cloud data centres. LADT is based on the hypothesis that, in an anomaly-free Cloud data centre, there is a strong correlation between the node-level and VM-level performance metrics and that this correlation diminishes significantly in the case of abnormal behaviour at the node-level. The LADT algorithm raises an anomaly alarm when the correlation coefficient value between the node-level and VM-level metrics drops below a threshold level. We have demonstrated a lightweight distributed implementation of LADT using Chukwa and also demonstrated that the tool can detect node-level disk performance anomalies by correlating the hosting node IOPS with the aggregated hosted VM IOPS. Such anomalies may arise as a result of security attacks such as distributed denial-of-service (DDoS). We also demonstrated that LADT introduces acceptably low overhead, while recognizing that the implementation is amenable to optimisation along the entire path of metrics collection, data aggregation and analysis.

We intend to conduct a detailed analysis of possible attack models of the system. LADT can also detect CPU/memory/network related performance anomalies, due to the performance implications of virtualisation and resource management software stacks. We wish to explore these anomalies

in more detail, using both controlled and uncontrolled set-ups (i.e. production-level set-ups with unseen anomalies) in our Cloud testbed. We plan to conduct a more thorough analysis of LADT performance, scalability and intrusion minimisation with respect to the hosted VMs. We are particularly interested in co-executing VMs with diverse characteristics (e.g. CPU-intensive, I/O-intensive), and latency sensitivity. Our aim is to understand whether adapting parameters such as the number of agent adaptors in the hosts, the frequency of data collection per VM in the hosts and the number of data aggregation tasks and cores used by collectors is necessary to keep the monitoring overhead low.

ACKNOWLEDGEMENTS

This work has been supported by the European Commission and the Seventh Framework Programme under grant agreement FP7-610711 (CACTOS).

REFERENCES

- Antunes, J., Neves, N., and Verissimo, P. (2008). Detection and prediction of resource-exhaustion vulnerabilities. In *Software Reliability Engineering, 2008. IS-SRE 2008. 19th International Symposium on*, pages 87–96.
- Azmandian, F., Moffie, M., Alshawabkeh, M., Dy, J., Aslam, J., and Kaeli, D. (2011). Virtual machine monitor-based lightweight intrusion detection. *ACM SIGOPS Operating Systems Review*, 45(2):38–53.
- Dahbur, K., Mohammad, B., and Tarakji, A. B. (2011). A survey of risks, threats and vulnerabilities in cloud computing. In *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications, ISWSA '11*, pages 12:1–12:6, New York, NY, USA. ACM.
- Ferdman, M., Adileh, A., Kocerber, O., Volos, S., Alisafae, M., Jevdjic, D., Kaynak, C., Popescu, A. D., Ailamaki, A., and Falsafi, B. (2012). Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '12*, pages 37–48, New York, NY, USA. ACM.
- Hansen, S. E. and Atkins, E. T. (1993). Automated system monitoring and notification with swatch. In *Proceedings of the 7th USENIX Conference on System Administration, LISA '93*, pages 145–152, Berkeley, CA, USA. USENIX Association.
- Jiang, M., Munawar, M. A., Reidemeister, T., and Ward, P. A. (2009). System monitoring with metric-correlation models: Problems and solutions. In *Pro-*

- ceedings of the 6th International Conference on Autonomic Computing, ICAC '09*, pages 13–22, New York, NY, USA. ACM.
- Kang, H., Chen, H., and Jiang, G. (2010). Peerwatch: A fault detection and diagnosis tool for virtualized consolidation systems. In *Proceedings of the 7th International Conference on Autonomic Computing, ICAC '10*, pages 119–128, New York, NY, USA. ACM.
- Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.
- Kumar, V., Cooper, B. F., Eisenhauer, G., and Schwan, K. (2007). imanage: Policy-driven self-management for enterprise-scale systems. In *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, Middleware '07, pages 287–307, New York, NY, USA. Springer-Verlag New York, Inc.
- Li, D., Jin, H., Liao, X., Zhang, Y., and Zhou, B. (2013). Improving disk i/o performance in a virtualized system. *J. Comput. Syst. Sci.*, 79(2):187–200.
- Lou, J.-G., Fu, Q., Yang, S., Xu, Y., and Li, J. (2010). Mining invariants from console logs for system problem detection. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10*, pages 24–24, Berkeley, CA, USA. USENIX Association.
- Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. (2008). Pig latin: A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, pages 1099–1110, New York, NY, USA. ACM.
- Oppenheimer, D., Ganapathi, A., and Patterson, D. A. (2003). Why do internet services fail, and what can be done about it? In *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems - Volume 4, USITS'03*, pages 1–1, Berkeley, CA, USA. USENIX Association.
- Pertet, S. and Narasimhan, P. (2005). Causes of failure in web applications. Technical report, CMU-PDL-05-109.
- Rabkin, A. and Katz, R. (2010). Chukwa: A system for reliable large-scale log collection. In *Proceedings of the 24th International Conference on Large Installation System Administration, LISA'10*, pages 1–15, Berkeley, CA, USA. USENIX Association.
- Rajasekar, N. C. and Imafidon, C. (2010). Exploitation of vulnerabilities in cloud storage. In *Proceedings of the First International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 122–127.
- Rouillard, J. P. (2004). Refereed papers: Real-time log file analysis using the simple event correlator (sec). In *Proceedings of the 18th USENIX Conference on System Administration, LISA '04*, pages 133–150, Berkeley, CA, USA. USENIX Association.
- Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA. IEEE Computer Society.
- Sigar (2014). <https://support.hyperic.com/display/sigar/home>.
- Tan, J., Kavulya, S., Gandhi, R., and Narasimhan, P. (2012). Light-weight black-box failure detection for distributed systems. In *Proceedings of the 2012 Workshop on Management of Big Data Systems, MBDS '12*, pages 13–18, New York, NY, USA. ACM.
- The, J. P. and Prewett, J. E. (2003). Analyzing cluster log files using logsurfer. In *Proceedings of the 4th Annual Conference on Linux Clusters*.
- Virt-Top (2014). <http://virt-tools.org/about/>.
- Vora, M. (2011). Hadoop-hbase for large-scale data. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 1, pages 601–605.
- Wang, C. (2009). Ebat: Online methods for detecting utility cloud anomalies. In *Proceedings of the 6th Middleware Doctoral Symposium, MDS '09*, pages 4:1–4:6, New York, NY, USA. ACM.
- Ward, J. S. and Barker, A. (2013). Varanus: In situ monitoring for large scale cloud systems. In *Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science - Volume 02, CLOUDCOM '13*, pages 341–344, Washington, DC, USA. IEEE Computer Society.
- Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles, SOSP '09*, pages 117–132, New York, NY, USA. ACM.

Performance and Cost Evaluation for the Migration of a Scientific Workflow Infrastructure to the Cloud

Santiago Gómez Sáez, Vasilios Andrikopoulos, Michael Hahn, Dimka Karastoyanova,
Frank Leymann, Marianna Skouradaki and Karolina Vukojevic-Haupt
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany
{gomez-saez, andrikopoulos, hahn, karastoyanova, leymann, skouradaki, vukojevic}@iaas.uni-stuttgart.de

Keywords: Workflow Simulation, eScience, IaaS, Performance Evaluation, Cost Evaluation, Cloud Migration.

Abstract: The success of the Cloud computing paradigm, together with the increase of Cloud providers and optimized Infrastructure-as-a-Service (IaaS) offerings have contributed to a raise in the number of research and industry communities that are strong supporters of migrating and running their applications in the Cloud. Focusing on eScience simulation-based applications, scientific workflows have been widely adopted in the last years, and the scientific workflow management systems have become strong candidates for being migrated to the Cloud. In this research work we aim at empirically evaluating multiple Cloud providers and their corresponding optimized and non-optimized IaaS offerings with respect to their offered performance, and its impact on the incurred monetary costs when migrating and executing a workflow-based simulation environment. The experiments show significant performance improvements and reduced monetary costs when executing the simulation environment in off-premise Clouds.

1 INTRODUCTION

In the last years the workflow technology has been widely adopted in several domains, e.g. business or eScience, which often have different domain-specific requirements in terms of supported functionalities and expected behavior of the underlying infrastructure. Focusing on eScience applications, simulation workflows are a well-known research area, as they provide scientists with the means to model, provision, and execute automated and flexible long running simulation-based experiments (Sonntag and Karastoyanova, 2010). Such simulation-based experiments typically comprise large amounts of data processing and transfer and consume multiple distributed simulation services for long periods of time. Due to the access and resource consumption nature of such simulation environments, previous works have targeted the migration and adaptations of such environments to be deployed, provisioned, and executed in Cloud infrastructures (Juve et al., 2009; ?; Vukojevic-Haupt et al., 2013; Zhao et al., 2014).

The Cloud computing paradigm has led in the last years to an increase in the number of applications which are partially or completely running in different *Everything-as-a-Service* Cloud offerings. The increase of available and optimized Cloud services has intro-

duced further efficient alternatives for hosting application components with special resources consumption patterns, e.g. computationally or memory intensive ones. However, such a wide landscape of possibilities has become a challenge for deciding among the different Cloud providers and their corresponding offerings. Previous works targeted such a challenge by assisting application developers in the tasks related to selecting, configuring, and adapting the distribution of their application among multiple services (de Oliveira et al., 2011; Gómez Sáez et al., 2014a). There are multiple decision points that can influence the distribution of an application, e.g. cost, performance, security concerns, etc. The focus of this research work is to provide an overview, evaluate, and analyze the trade-off between the performance and cost when migrating a simulation environment to different Cloud providers and their corresponding Infrastructure-as-a-Service (IaaS) offerings. The contributions of this work can therefore be summarized as follows:

- the selection of a set of viable and optimized IaaS offerings for migrating a previously developed simulation environment,
- an empirical evaluation focusing on the performance and the incurred monetary costs, and,
- an analysis of the performance and cost trade-off

when scaling the simulation environment workload.

The remaining of this paper is structured as follows: Section 2 motivates this work and depicts the problems that aim to be achieved. The simulation environment used for evaluation purposes in this work is introduced in Section 3. Section 4 presents the experiments on evaluating the performance and incurred costs when migrating the simulation environment to different IaaS offerings, and discusses our findings. Finally, Section 5 summarizes related work and Section 6 concludes with some future work.

2 MOTIVATION & PROBLEM STATEMENT

Simulation workflows, a well-known topic in the field of eScience, describe the automated and flexible execution of simulation-based experiments. Common characteristics of such simulation workflows are that they are long-running as well as being executed in an irregular manner. However, during their execution a wide amount of resources are typically provisioned, consumed, and released. Considering these characteristics, previous works focused on migrating and executing simulation environments in the Cloud, as Cloud infrastructures significantly reduce infrastructure costs while coping with an irregular but heavy demand of resources for running such experiments (Vukojevic-Haupt et al., 2013).

Nowadays there exists a vast amount of configurable Cloud offerings among multiple Cloud providers. However, such a wide landscape has become a challenge for deciding among (i) the different Cloud providers and (ii) the multiple Cloud offering configurations offered by such providers. We focus in this work on IaaS solutions, as there exists a lack of Platform-as-a-service (PaaS) offerings that enable the deployment and execution of scientific workflows in the Cloud. IaaS offerings describe the amount and type of allocated resources, e.g. CPUs, memory, or storage, and define different VM instance types within different categories. For example, the Amazon EC2¹ service does not only offer VM instances of different size, but also provides different VM categories which are optimized for different use cases, e.g. computation intensive, memory intensive, or I/O intensive. Similar offerings are available also by other providers,

¹Amazon EC2: <http://aws.amazon.com/ec2/instance-types/>

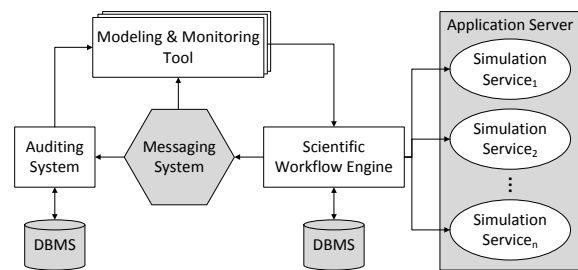


Figure 1: System Overview of the SimTech Scientific Workflow Management System (SWfMS).

e.g. Windows Azure² or Rackspace³. The offered performance and incurred cost significantly vary among the different Cloud services, and depend on the simulation environment resource usage requirements and workload. In this work, we aim to analyze the performance and cost trade-off when migrating to different Cloud offerings a simulation environment developed and used as case study, as discussed in the following section.

3 THE OPAL SIMULATION ENVIRONMENT

A Scientific Workflow Management System (SimTech SWfMS) is being developed by the Cluster of Excellence in Simulation Technology (SimTech⁴), enabling scientists to model and execute their simulation experiments using workflows (Sonntag and Karastoyanova, 2010; Sonntag et al., 2012). The SimTech SWfMS is based on conventional workflow technology which offers several non-functional requirements like robustness, scalability, reusability, and sophisticated fault and exception handling (Görlach et al., 2011). The system has been adapted and extended to the special needs of the scientists in the eScience domain (Sonntag et al., 2012). During the execution of a workflow instance the system supports the modification of the corresponding workflow model, which is then propagated to the running instances. This allows running simulation experiments in a trial-and-error manner.

The main components of the SimTech SWfMS shown in Fig. 1 are a modeling and monitoring tool, a workflow engine, a messaging system, several databases, an auditing system, and an application server running simulation services. The workflow engine provides an execution environment for the work-

²Windows Azure: <http://azure.microsoft.com/en-us/>

³Rackspace: <http://www.rackspace.com/>

⁴SimTech: <http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/>

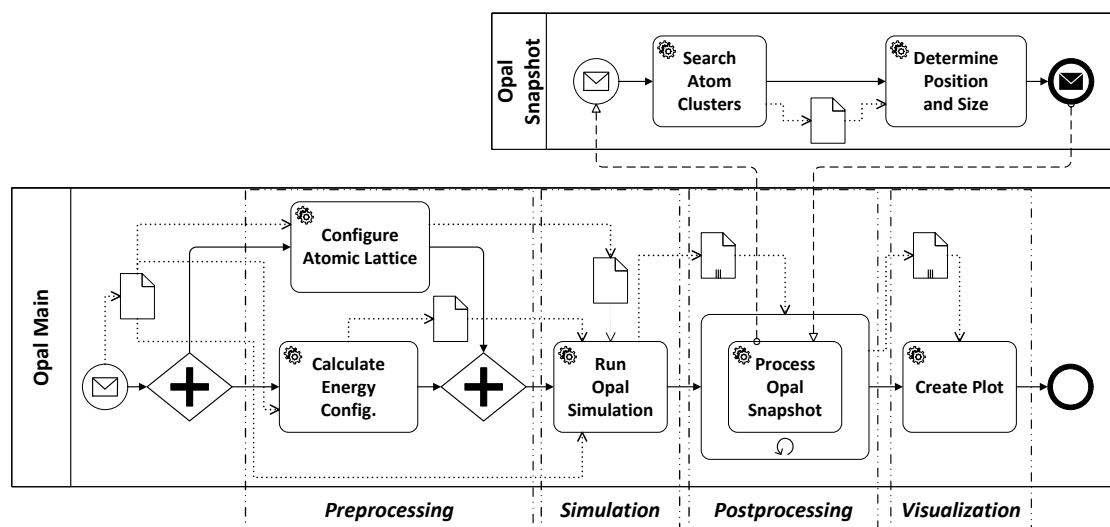


Figure 2: Simplified Simulation Workflows Constituting the OPAL Simulation Environment (Sonntag and Karastoyanova, 2013).

flows. The messaging system serves as communication layer between the modeling- and monitoring tool, the workflow engine, and the auditing system. The auditing system stores data related to the workflow execution for analytical and provenance purposes.

The SimTech SWfMS has been successfully applied in different scenarios in the eScience domain; one example is the automation of a Kinetic Monte-Carlo (KMC) simulation of solid bodies by orchestrating several Web services being implemented by modules of the OPAL application (Sonntag et al., 2011a). The OPAL Simulation Environment is constituted by a set of services which are controlled and orchestrated through a main OPAL workflow (the *Opal Main* process depicted in Figure 2). The simulation services are implemented as Web services and divided into two main categories: (i) resource management, e.g. distributing the workload among the different servers, and (ii) wrapped simulation packages depicted in (Binkele and Schmauder, 2003; Molnar et al., 2010). The main workflow can be divided in four phases as shown in Fig. 2: preprocessing, simulation, postprocessing, and visualization. During the preprocessing phase all data needed for the simulation is prepared. In the simulation phase the workflow starts the Opal simulation by invoking the corresponding Web service. In regular intervals, the Opal simulation creates intermediate results (snapshots). For each of these snapshots the main workflow initiates the postprocessing which is realized as a separate workflow (*Opal Snapshot* process in Figure 2). When the simulation is finished and all intermediate results are postprocessed, the results of the simulation are visualized.

4 EXPERIMENTS

4.1 Methodology

As shown in Fig. 2, the OPAL Simulation Environment is comprised of multiple services and workflows that compose the simulation and resource management services. The environment can be concurrently used by multiple users, as the simulation data isolation is guaranteed through the creation of independent instances (workflows, services, and temporal storage units) for each user's simulation request. The experiments must therefore consider and emulate the usage of the environment by multiple users concurrently.

The migration of the simulation environment to the Cloud opens a wide set of viable possibilities for selecting and configuring different Cloud services for the different components of the OPAL environment. However, in this first set of experiments we restrict the distribution of the simulation environment components by hosting the complete simulation application stack in one VM, which is made accessible to multiple users. Future investigations plan to distribute such environment using different Cloud offerings, e.g. Database-as-a-Service (DBaaS) for hosting the auditing databases. We therefore focus this work on *driving a performance and cost analysis when executing the OPAL Simulation Environment in on- and off-premise infrastructures, and using different IaaS offerings and optimized configurations.*

Table 1 shows the different VM categories, based on their characteristics and offered prices by three ma-

Table 1: IaaS Ubuntu Linux On-demand Instances Categories per Provider (in January 2015).

Instance Category	Cloud Provider	Instance Type	vCPU	Memory (GB)	Region	Price (US\$/h)
Micro	on-premise	micro	1	1	EU (Germany)	0.13
	AWS EC2	t2.micro	1	1	EU (Ireland)	0.014
	Windows Azure	A1	1	1.75	EU (Ireland)	0.06
	Rackspace	General 1	1	1	USA	0.06
General Purpose	on-premise	large	2	4	EU (Germany)	0.26
	AWS EC2	m3.large	2	7.5	EU (Ireland)	0.154
	Windows Azure	A2	2	3.5	EU (Ireland)	0.12
	Rackspace	General 2	2	2	USA	0.074
Compute Optimized	on-premise	compute3.large	4	4	EU (Germany)	0.52
	AWS EC2	c3.large	2	3.75	EU (Ireland)	0.120
	Windows Azure	D2	2	7	EU (Ireland)	0.23
	Rackspace	Compute 1-3.75	2	3.75	USA	0.1332
Memory Optimized	on-premise	memory4.large	2	15	EU (Germany)	0.26
	AWS EC2	r3.large	2	15.25	EU (Ireland)	0.195
	Windows Azure	D3	4	14	EU (Ireland)	0.46
	Rackspace	Memory 1-15	2	15	USA	0.2522

for Cloud providers: Amazon AWS, Windows Azure, and Rackspace. In addition to the off-premise VM instances types, multiple on-premise VM instances types were created in our virtualized environment, configured in a similar manner to the ones evaluated in the off-premise scenarios, and included in such categories. The on-premise VM instances configurations are based on the closest equivalent to the off-premise VM configurations within each instance category. The encountered providers and offerings showed two levels of VM categories, i.e. based on the optimization for custom use cases (*Micro*, *General Use*, *Compute Optimized*, and *Memory optimized*), and based on a quantitative assignment of virtualized resources. This fact must be taken into consideration in our evaluation due to the variation in the performance, and its impact on the final incurred costs for running simulations in different Cloud offerings. The pricing model for the on-premise scenarios was adopted from (Walker, 2009) as discussed in the following section, while for the off-premise scenarios the publicly available information from the providers was used (Andrikopoulos et al., 2013), taking into account on-demand pricing models only.

4.2 Setup

The scientific workflow simulation environment is constituted by two main systems: the SimTech SWfMS (Sonntag and Karastoyanova, 2010; Sonntag et al., 2012), and a set of Web services bundling resource management and the KMC simulation tasks depicted in (Binkele and Schmauder, 2003; Molnar et al., 2010). The former comprises the following

middleware stack:

- an Apache Orchestration Director Engine (ODE) 1.3.5 (Axis2 distribution) deployed on
- an Apache Tomcat 7.0.54 server with Axis2 support.
- The scientific workflow engine (Apache ODE) utilizes a MySQL server 5.5 for workflow administration, management, and reliability purposes , and
- provides monitoring and auditing information through an Apache ActiveMQ 5.3.2 messaging server.

The resource management and KMC simulation services are deployed as Axis2 services in an Apache Tomcat 7.0.54 server. The underlying on- and off-premise infrastructure configurations selected for the experiments are shown in Table 1. The on-premise infrastructure aggregates an IBM System x3755 M3 server⁵ with an AMD Opteron Processor 6134 exposing 16 CPU of speed 2.30 GHz and 65GB RAM. In all scenarios the previously depicted middleware components are deployed on an Ubuntu server 14.04 LTS with 60% of the total OS memory dedicated to the SWfMS.

For all evaluation scenarios a system's load of 10 concurrent users sequentially sending 10 random and uniformly distributed simulation requests/user was created using Apache JMeter 2.9 as the load driver. Such a load aims at emulating a shared utilization of the simulation infrastructure. Due to the asynchronous

⁵IBM System x3755 M3: <http://www-03.ibm.com/systems/xbc/cog/x3755m3.7164/x3755m3.7164aag.html>

nature of the OPAL simulation workflow, a custom plugin in JMeter was realized towards receiving and correlating the asynchronous simulation responses. The perceived by the user latency for each simulation was measured in milliseconds (ms). Towards minimizing the network latency, in all scenarios the load driver was deployed in the same region as the simulation environment.

The incurred monetary costs for hosting the simulation environment on-premise are calculated considering firstly the purchase, maintenance, and depreciation of the server cluster, and secondly by calculating the price of each CPU time. (Walker, 2009) proposes pricing models for analyzing the cost of purchasing vs. leasing CPU time on-premise and off-premise, respectively. The real cost of a CPU/hour when purchasing a server cluster, can be derived using the following equations:

$$\frac{(1 - 1/\sqrt{2}) \times \sum_{T=0}^{Y-1} \frac{C_T}{(1+k)^T}}{(1 - (1/\sqrt{2})^Y) \times TC} \quad (1)$$

where C_T is the acquisition (C_0) and maintenance ($C_{1..N}$) costs over the Y years of the server cluster, k is the cost of the invested capital, and

$$TC = TCPU \times H \times \mu \quad (2)$$

where $TCPU$ depicts the total number of CPU cores in the server cluster, H is the expected number of operational hours, and μ describes the expected utilization. The utilized on-premise infrastructure total cost breaks down into an initial cost (C_0) of approximately 8500\$ in July 2012 and an annual maintenance cost ($C_{1..N}$) of 7500\$, including personnel costs, power and cooling consumption, etc. The utilization rate of such cluster is of approximately 80%, and offers a reliability of 99%. Moreover, the server cluster runs six days per week, as one day is dedicated for maintenance operations. Such a configuration provides 960K CPU hours annually. As discussed in (Walker, 2009), we also assumed in this work a cost of 5% on the invested capital. The cost for the off-premise scenarios was gathered from the different Cloud provider's Web sites.

Table 1 depicts the hourly cost for the CPUs consumed in the different on-premise VM configurations. In order to get a better sense of the scope of the accrued costs, the total cost calculation performed as part of the experiments consisted of predicting the necessary time to run 1K concurrent experiments. Such estimation was then used to calculate the incurred costs of hosting the simulation environment in the previously evaluated on- and off-premise scenarios. The monetary cost calculation was performed by linearly extrapolating the obtained results for the 100 requests to a total of 1K requests. The scientific library Numpy of Python

2.7.5 was used for performing the prediction of 1K simulation requests. The results of this calculation, as well as the observed performance measurements are discussed in the following.

4.3 Evaluation Results

4.3.1 Performance Evaluation

Figure 3 shows the average observed latency for the different VM categories depicted in Table 1 for the different Cloud providers. The latency perceived in the scenarios comprising the selection of *Micro* instances have been excluded from the comparison due to the impossibility to finalize the execution of the experiments. More specifically, the on-premise micro-instance was capable of stably running approximately 80 requests (see Figure 4(a)), while in the off-premise scenarios the load saturated the system with 10 requests approximately in the AWS EC2 and Windows Azure scenarios (see Figures 4(b) and 4(c), respectively). For the scenario utilizing Rackspace, the VM micro instance was saturated immediately after sending the first set of 10 concurrent simulation requests.

With respect to the remaining instance categories (*General Purpose*, *Compute Optimized*, and *Memory Optimized*), the following performance variation behaviors can be observed:

1. the on-premise scenario shows in average a latency of 320K ms. over all categories, a 40% higher average than the perceived latency in the off-premise scenarios.
2. However, the performance is not constantly improved when migrating the simulation environment off-premise. For example, the *General Purpose* Windows Azure VM instance shows a degraded performance of 11%, while the Windows Azure *Compute Optimize* VM instance shows only

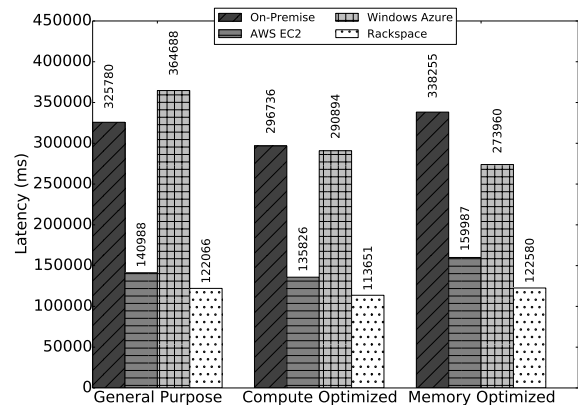


Figure 3: Average Simulation Latency per Provider & VM Category.

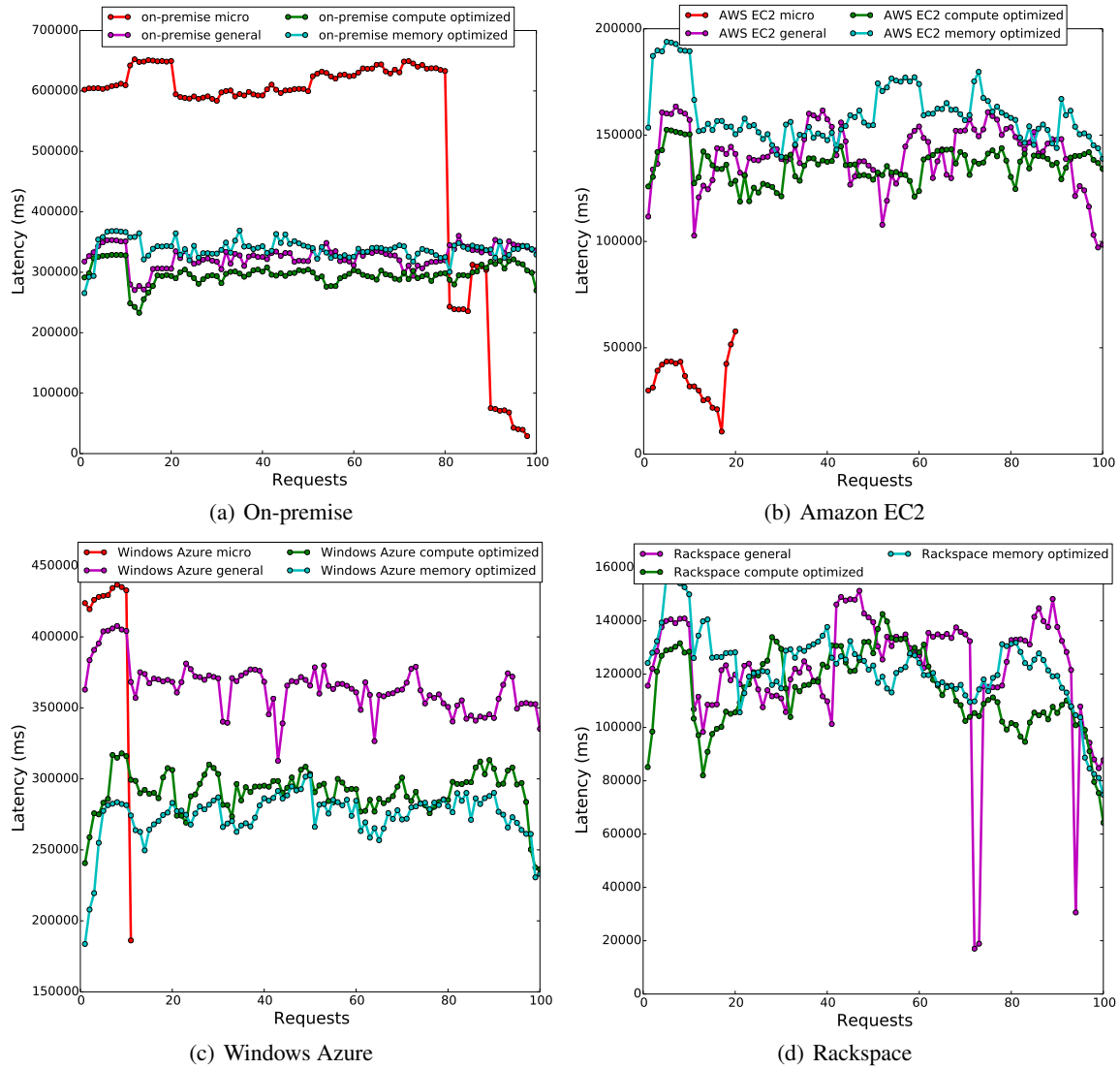


Figure 4: Performance Analysis per Provider & VM Category.

a slightly performance improvement of 2%, when compared with the on-premise scenario.

3. The performance when migrating the simulation environment to the Cloud improves by approximately 56% and 62% for the AWS EC2 and Rackspace *General Purpose* VM instances, respectively,
4. 54%, 2%, and 61% for the AWS EC2, Windows Azure, and Rackspace *Compute Optimized* VM instances, respectively, and
5. 52%, 19%, and 63% for the AWS EC2, Windows Azure, and Rackspace *Memory Optimized* VM instances, respectively.

When comparing the average performance improve-

ment among the different optimized VM instances, the *Compute Optimized* and *Memory Optimized* instances enhance the performance by 12% and 6%, respectively.

Figure 4 shows the perceived requests' latency individually. It can be observed when executing the simulation environment in the Rackspace infrastructure that the performance highly varies when increasing the number of requests (see Figure 4(d)). Such performance variation decreases in the on-premise, AWS EC2, and Windows Azure infrastructures (see Figures 4(a), 4(b), and 4(c), respectively). In all scenarios, the network latency does not have an impact in the performance due to the nature of our experimental setup described in the previous section.

When comparing the performance improvement

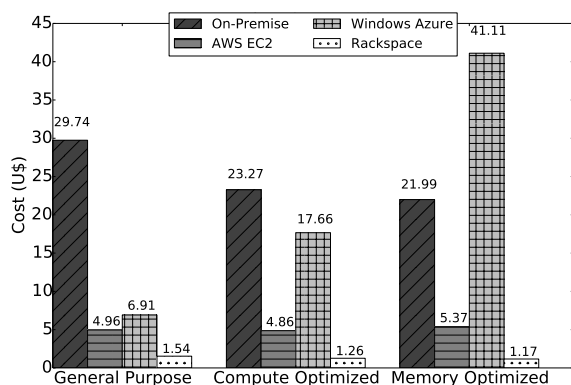


Figure 5: Cost Comparison extrapolated to 1K Simulation Requests (in January 2015 Prices).

among the different VM instances categories, the Windows Azure infrastructure shows the greater when selecting a *Compute Optimized* or *Memory Optimized* VM instance over a *General Purpose* VM instance (see Figure 4(c)).

4.3.2 Cost Comparison

Figure 5 presents an overview of the expected costs for running 1K experiments among 10 users. The following pricing variations can be observed:

1. The incurred costs of hosting the simulation environment on-premise is 25\$ in average.
2. When migrating the simulation infrastructure off-premise, the cost descends in average 80%, 12%, and 94% when utilizing the AWS EC2, Windows Azure, and Rackspace IaaS services.
3. When comparing the incurred costs among the different VM categories, the *Memory Optimized* categories are in average 61% and 47% more expensive when compared to the *Compute Optimized* and *General Purpose* VM categories, respectively.
4. Among the different off-premise providers, Windows Azure is in average 900% more expensive for running the simulation environment.

4.4 Discussion

The experiments driven as part of this work have contributed to derive and report a bi-dimensional analysis focusing on the selection among multiple IaaS offerings to deploy and run the OPAL Simulation Environment. With respect to performance, it can be concluded that:

1. The migration of the simulation environment to off-premise Cloud services has an impact on the

system's performance, which is beneficial or detrimental depending on the VM provider and category.

2. The selection of *Micro* VM instances did not offer an adequate availability to the simulation environment in the off-premise scenarios. Such a negative impact was produced by the non-automatic allocation of swap space for the system's virtual memory.
3. When individually observing the performance within each VM category, the majority of the selected off-premise IaaS services improved the performance of the simulation environment. However, the *General Purpose* Windows Azure VM instances showed a degradation of the performance when compared to the other IaaS services in the same category.
4. The perceived by the user latency was in average reduced when utilizing *Compute Optimized* VM instances. Such an improvement is in line with the compute intensity requirements of the simulation environment.

The cost analysis derived the following conclusions:

1. There exists a significant monetary cost reduction when migrating the simulation environment to off-premise IaaS Cloud services.
2. Despite of the improved performance observed when running the simulation environment in the *Compute Optimized* and *Memory Optimized* VM instances, scaling the experiments to 1K simulation requests incurred in an average increase of 9% and 61% with respect to the *General Purpose* VM instances cost, respectively.
3. The incurred monetary costs due to the usage of Windows Azure services tend to increase when using optimized VM instances, i.e. *Compute Optimized* and *Memory Optimized*. Such behavior is reversed for the remaining off-premise and on-premise scenarios.
4. Due to the low costs demanded for the usage of Rackspace IaaS services (nearly 40% less in average), the final price for running 1K simulations is considerably lower than the other off-premise providers and hosting the environment on-premise.

The previous observations showed that the IaaS services provided by Rackspace are the most suitable for migrating our OPAL Simulation Environment. However, additional requirements may conflict with the migration decision of further simulation environments, e.g. related to data privacy and transfer between EU and USA regions, as Rackspace offers a limited set of optimized VMs in their European region.

5 RELATED WORKS

We consider our work related to the following major research areas: performance evaluation of workflow engines, workflow execution in the Cloud, and migration and execution of scientific workflows in the Cloud.

When it comes to evaluating the performance of common or scientific workflow engines, a standardized benchmark is not yet available. A first step towards this direction is discussed in (Skouradaki et al., 2015), but propose approach is premature and could not be used as the basis for this work. Beyond this work, performance evaluations are usually custom to specific project needs. Specifically for BPEL engines not much work is currently available. For example (Röck et al., 2014) summarize nine approaches that evaluate the performance of BPEL engines. In most of the cases, workflow engines are benchmarked with load tests with a workload consisting of 1-4 workflows. Throughput and latency are the metrics most frequently used.

There are only few Cloud providers supporting the deployment and execution of workflows in a Platform-as-a-Service (PaaS) solution. The WSO2 Stratos Business Process Server (Pathirage et al., 2011) and Business Processes on the Cloud is offered by IBM Business Process Manager⁶ offer the necessary tools and abstraction levels for developing, deploying and monitoring workflows in the Cloud. However, such services are optimized for business tasks, rather than for supporting simulation operations.

Scientific Workflow Management Systems are exploiting business workflows concepts and technologies for supporting scientists towards the use of scientific applications (Sonntag et al., 2011b; Sonntag and Karastoyanova, 2010). Zhao et al. (Zhao et al., 2014) develop a service framework for integrating Scientific Workflow Management Systems in the Cloud to leverage from the scalability and on-demand resource allocation capabilities. The evaluation of their approach mostly focuses on examining the efficiency of their proposed PaaS based framework.

Simulation experiments are driven in the scope of different works (Binkele and Schmauder, 2003; Molnar et al., 2010). Later research efforts focused on the migration of simulations to the Cloud. Due to the diverse benefits of Cloud environments the approaches evaluate the migration with respect to different scopes. The approaches that study the impact of migration to the performance and incurred monetary costs is considered more relevant to our work. In (de Oliveira

et al., 2011) the authors examine the performance of X-Ray Crystallography workflows executed on the Sci-Cumulus middleware deployed in Amazon EC2. Such workflows are CPU-intensive and requires the execution of high parallel techniques. Likewise, in (Juve et al., 2009) the authors compare the performance of scientific workflows migrated from Amazon EC2 to a typical High Performance Computing system (NCSA's Abe). In both approaches the authors conclude that migration to the Cloud can be viable but not equally efficient to High Performance Computing environments. However, Cloud environments allow the provisioning of specific resources configurations irregularly during the execution of simulation experiments (Strauch et al., 2013). Moreover, the performance improvement observed in Cloud services provide the necessary flexibility for reserving and releasing resources on-demand while reducing the capital expenditures (Ostermann et al., 2010). Research towards this direction is a fertile field. Juve et al. (Juve et al., 2013) execute nontrivial scientific workflow applications on grid, public, and private Cloud infrastructures to evaluate the deployments of workflows in the Cloud in terms of setup, usability, cost, resource availability, and performance. This work can be considered complementary to our approach, although we focused on investigating more on public Cloud providers and took into account the different VM optimization categories.

6 CONCLUSION AND FUTURE WORK

Simulation workflows have been widely used in the eScience domain due to their easiness to model, and flexible and automated runtime properties. The characteristics of such workflows together with the usage patterns of simulation environments have made these type of systems suitable to profit from the advantages brought by the Cloud computing paradigm. The existence of a vast amount of Cloud services together with the complexity introduced by the different pricing models have become a challenge to efficiently select which Cloud service to host the simulation environment. The main goal of this investigation is to report the performance and monetary cost findings when migrating the previously realized OPAL simulation environment to different IaaS solutions.

A first step in this experimental work consisted of selecting a set of potential IaaS offerings suitable for our simulation environment. The result of such selection covered four major deployment scenarios: (i) in our on-premise infrastructure, and in (ii) three off-premise infrastructures (AWS EC2, Windows Azure,

⁶<http://www-03.ibm.com/software/products/en/business-process-manager-cloud>

and Rackspace). The selection of the IaaS offerings consisted of evaluating the different providers and their corresponding optimized VM instances (*Micro*, *General Purpose*, *Compute Optimized*, and *Memory Optimized*). The simulation environment was migrated and its performance evaluated using an artificial workload. A second step in our analysis consisted on extrapolating the obtained results towards estimating the incurred costs for running the simulation environment on- and off-premise. The analyses showed a beneficial impact in the performance and a significant reduction of monetary costs when migrating the simulation environment to the majority of off-premise Cloud offerings.

Despite our efforts towards analyzing and finding the most efficient IaaS Cloud service to deploy and run our simulation environment, our experiments solely focused on IaaS offerings. Future works focus on analyzing further service models, i.e. Platform-as-a-Service (PaaS) or Database-as-a-Service (DBaaS), as well as evaluating the distribution of the different components constituting the simulation environment among multiple Cloud offerings. Investigating different autoscaling techniques and resources configuration possibilities is also part of future work, e.g. feeding the application distribution system proposed in (Gómez Sáez et al., 2014b) with such empirical observations.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the FP7 EU project ALLOW Ensembles (600792), the German Research Foundation (DFG) within the Cluster of Excellence in Simulation Technology (EXC310), and the German DFG project BenchFlow (DACH Grant Nr. 200021E-145062/1).

REFERENCES

- Andrikopoulos, V., Song, Z., and Leymann, F. (2013). Supporting the migration of applications to the cloud through a decision support system. In *Cloud Computing (CLOUD)*, 2013 IEEE Sixth International Conference on, pages 565–572. IEEE.
- Binkele, P. and Schmauder, S. (2003). An atomistic Monte Carlo Simulation of Precipitation in a Binary System. *Zeitschrift für Metallkunde*, 94(8):858–863.
- de Oliveira, D., Ocaña, K. A. C. S., Ogasawara, E. S., Dias, J., Baião, F. A., and Mattoso, M. (2011). A Performance Evaluation of X-Ray Crystallography Scientific Workflow Using SciCumulus. In Liu, L. and Parashar, M., editors, *IEEE CLOUD*, pages 708–715. IEEE.
- Gómez Sáez, S., Andrikopoulos, V., Leymann, F., and Strauch, S. (2014a). Design Support for Performance Aware Dynamic Application (Re-)Distribution in the Cloud. *IEEE Transactions on Services Computing (to appear)*.
- Gómez Sáez, S., Andrikopoulos, V., Wessling, F., and Marquezan, C. C. (2014b). Cloud Adaptation & Application (Re-)Distribution: Bridging the two Perspectives. In *Proceedings EnCASE'14*, pages 1–10. IEEE Computer Society Press.
- Görlach, K., Sonntag, M., Karastoyanova, D., Leymann, F., and Reiter, M. (2011). *Conventional Workflow Technology for Scientific Simulation*, pages 323–352. Guide to e-Science. Springer-Verlag.
- Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., and Vahi, K. (2013). Characterizing and Profiling Scientific Workflows. *Future Gener. Comput. Syst.*, 29(3):682–692.
- Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B., and Maechling, P. (2009). Scientific Workflow Applications on Amazon EC2. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 59–66.
- Molnar, D., Binkele, P., Hocker, S., and Schmauder, S. (2010). Multiscale Modelling of Nano Tensile Tests for different Cu-precipitation States in α -Fe. In *Proc. of the 5th Int. Conf. on Multiscale Materials Modelling*, pages 235–239.
- Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D. (2010). A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing. In *Cloud Computing*, pages 115–131. Springer.
- Pathirage, M., Perera, S., Kumara, I., and Weerawarana, S. (2011). A Multi-tenant Architecture for Business Process Executions. In *Proceedings of the 2011 IEEE International Conference on Web Services, ICWS '11*, pages 121–128, Washington, DC, USA. IEEE Computer Society.
- Röck, C., Harrer, S., and Wirtz, G. (2014). Performance Benchmarking of BPEL Engines: A Comparison Framework, Status Quo Evaluation and Challenges. In *26th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 31–34, Vancouver, Canada.
- Skouradaki, M., Roller, D. H., Frank, L., Ferme, V., and Pautasso, C. (2015). On the Road to Benchmarking BPMN 2.0 Workflow Engines. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering ICPE 2015*, pages 1–4. ACM.
- Sonntag, M., Hahn, M., and Karastoyanova, D. (2012). Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In *Proceedings of the Demo Track of the 10th International Conference on Business Process Management (BPM 2012), CEUR Workshop Proceedings, 2012*, pages 1–5. CEUR Workshop Proceedings.
- Sonntag, M., Hotta, S., Karastoyanova, D., Molnar, D., and Schmauder, S. (2011a). Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications. In Abramowicz, W., Llorente, I., Surridge, M., Zisman, A., and Vayssi re, J., editors, *Towards a Service-Based Internet, Proceed-*

- ings of the 4th European Conference ServiceWave 2011, Poznan, Poland, 2011*, pages 1–12. Springer-Verlag.
- Sonntag, M., Hotta, S., Karastoyanova, D., Molnar, D., and Schmauder, S. (2011b). Workflow-based Distributed Environment for Legacy Simulation Applications. In *ICSOF (1)*, pages 91–94.
- Sonntag, M. and Karastoyanova, D. (2010). Next Generation Interactive Scientific Experimenting Based On The Workflow Technology. In Alhajj, R., Leung, V., Saif, M., and Thring, R., editors, *Proceedings of the 21st IASTED International Conference on Modelling and Simulation (MS 2010)*, 2010. ACTA Press.
- Sonntag, M. and Karastoyanova, D. (2013). Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. *Journal of Grid Computing*, 11(3):553–583.
- Strauch, S., Andrikopoulos, V., Bachmann, T., Karastoyanova, D., Passow, S., and Vukojevic-Haupt, K. (2013). Decision Support for the Migration of the Application Database Layer to the Cloud. In *Cloud Computing Technology and Science (CloudCom)*, 2013 IEEE 5th International Conference on, volume 1, pages 639–646. IEEE.
- Vukojevic-Haupt, K., Karastoyanova, D., and Leymann, F. (2013). On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows. In *Service-Oriented Computing and Applications (SOCA)*, 2013 IEEE 6th International Conference on, pages 91–98. IEEE.
- Walker, E. (2009). The Real Cost of a CPU Hour. *IEEE Computer*, 42:35–41.
- Zhao, Y., Li, Y., Raicu, I., Lu, S., Lin, C., Zhang, Y., Tian, W., and Xue, R. (2014). A Service Framework for Scientific Workflow Management in the Cloud. *Services Computing, IEEE Transactions on*, PP(99):1–1.

SHORT PAPERS

An Approach in the Design of Common Authentication Solution for a Multi-Platform Cloud Environment

Primož Cigoj^{1,2}, Borka Jerman Blažič² and Tomaž Klobučar²

¹*Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia*

²*Jozef Stefan Institute, Laboratory for Open Systems and Networks, Jamova cesta 39, 1000 Ljubljana, Slovenia
{primoz, borka, tomaz}@e5.ijs.si*

Keywords: Single Sign-on, Identity Management, Identity Federation, Cloud Computing, Security, Cloud Management, Cloud Provisioning, Infrastructure-as-a-Service, IaaS, Multi-platform Cloud, Access Control, Authentication, Authorization, Cloud Service Provider, Privacy, Software Platform, Centralized Systems, OpenStack, VMware.

Abstract: The security provision within multi-platform cloud computing environment is still considered not to be properly solved due to different problems with technical and human-based origin. This paper presents an attempt to provide an authentication and authorization solution based on the single sign-on (SSO) approach for cloud service users and administrators in a multi-platform environment. The problem of authentication in cloud services is briefly introduced and the approach implemented for cloud environment with two different proprietary (VMware) and open source (OpenStack) platforms is described.

1 INTRODUCTION

Cloud computing has revolutionized the provision of computing services, transferring them from local to locally unspecified remote environments, which are controlled by third party service providers. The main cloud computing challenge for vendors, providers and users of clouds remains protection of the cloud technology, services and users with adequate security measures. According to the survey conducted by Microsoft and the National Institute of Standards and Technology (NIST), security in the cloud computing model was the ICT executives' main concern (Jansen and Grance, 2011; Microsoft, 2010). Consequently, many business entities are still not very keen on adopting cloud computing.

In the cloud users alone have no control over their data and their cloud identity. Installation procedures are often complex and there is no adequate and complete security solution that ensures the safe deployment of the underlying infrastructure and safe use of the services. Especially in the multi-platform clouds and inter-clouds, where multiple cloud systems can be accessed, users and administrators are faced with different authentication and authorisation systems, different login dialogs, and each login dialog is matched with different credentials. In addition to being difficult implementing strong authentication at

the user level (Tripathi and Mishra, 2011), it is also complex to manage and create authentication mechanisms for several services and several platforms (Fernandes, Soares, Gomes, Freire and Inácio, 2014). Identity federation and single sign-on (SSO) techniques address these issues by allowing exchange of authentication and authorization information between two parties, such as an Identity Provider (IdP) and a Service Provider (SP).

While the identity federation and single sign-on concepts have been well covered in the literature in the past years, in general as well as in the cloud context e.g. (Pérez-Méndez, Pereniguez-Garcia, Marin-Lopez, López-Millán and Howlett, 2014), (Cruz Zapata, Fernández-Alemán and Toval, 2014), there are still practical issues that prevent their straightforward and simple application in the multi-platform cloud environments. Our attempt described in this paper was to design and develop a solution that would provide trustworthy and secure authentication and authorization service in such environment. The proposed solution is based on the SSO principle and was implemented on two different platforms (OpenStack and VMware). It was tested and evaluated within a large National Competence Centre project on cloud-assisted services for different fields of application.

The paper discusses the problem and related work first. Then it introduces the selected SSO- based

solution, and describes the development approach and the implementation course. The presentation ends with discussion and plans for future work.

2 AUTHENTICATION AND IDENTITY MANAGEMENT SERVICES IN MULTI-PLATFORM CLOUD INFRASTRUCTURE

According to the Cloud Security Alliance (CSA) (Simmonds, Rezek and Reed, 2011), the leader in the field of cloud security, the largest identified cloud security problem is related to the shared technology issues. Infrastructure resource sharing can potentially allow one consumer to peek into another consumer's data if the system does not provide strong system for authorization and authentication. Here, the problems of account hijacking or user credential theft are also relevant. The traditional identity management (IdM) approach is more centralized compared to the current solution used in cloud computing, and usually is based on user personal data, such as real name, user name, e-mail address, identification number, access permissions, etc. The use of a separate IdM system within an organisation, and its connection with the cloud is quite complicated, and there is no simple way to extend its use to the cloud (Lonea, Tianfield and Popescu, 2013). In order this to become part of the cloud system the following actions and the corresponding implementations are required (Cantor, Kemp, Philpott and Maler, 2005):

- Registration of identities: Verification of a user account is needed before proceeding with the registration in accordance with the relevant security standards. Organizations that transfer their user accounts to the cloud must make sure their user account management system is up-to-date and safe.
- Authentication: Management and implementation of the user authentication must be performed in a trustworthy way. The IdM systems should allow configuration of the authentication systems. Another important property of the system should be the cloud-providers' identification disclosure to third party providers and the use of authentication by both parties.
- Federation of identities: Federation of identities allows users to use the same set of credentials to obtain access to different resources. User's electronic identity and

attributes are securely shared across multiple IdM systems. Federation of identities can be achieved in several ways; e.g., based on SAML (Cantor, Kemp, Philpott and Maler, 2005) or the OpenID solution (Ferg, Fitzpatrick and Howells, 2007). One of the important properties is the required compatibility of cloud providers' IdM systems.

- Authorization: Authorization specifies the rights of individual user accounts. It is important for the cloud the account management procedures to be set up and for the rights verification from the highest system authority. Furthermore, the granted right should be consistent with the policy.
- Access control: Access control requirements vary widely according to the type of the end-user (an individual or an organization). In order to implement an access control system, an access control policy must be set, and its implementation should allow the performed actions to be traceable.

Apart from these recommendations it becomes obvious that an optimal system for ensuring authentication and authorization in a multi-platform cloud system would also benefit from the single sign-on approach principles. The Open Group defines SSO as "a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where that user has access permission, without the need to enter multiple passwords" (Group TO, 2014). The most important security property that the SSO principle brings is the common secure infrastructure, which can be carefully managed and protected (Andronache and Nisipasiu, 2011). At the same time, when using the SSO approach, the cloud services do not have to manage each user account. Account management can be carried out by a central authentication system.

Although the concepts of identity federation and single sign-on are well known in the cloud research literature there are still some practical design and implementation issues worth discussing, especially in the multi-platform cloud environments, cloud federations, and inter clouds. General proposed solutions for a multi-platform environment often do not take into account the current implementation status, functionalities and openness of the available cloud platforms, as well as not take advantage of their existing services. The papers also do not give enough details of the proposed solutions and their integration into existing platforms.

Panarello et al., for example, analyse requirements for IaaS cloud federation (Panarello,

Celesti, Fazio, Villari and Puliafito, 2014). While they envisage SAML, OpenID and Shibboleth as an authentication solution, only a standard and general identity federation model is described without details and without taking into account the platform implementation specifics, such as current missing SAML support in OpenStack and Hyper-V. They also highlight only one type (OpenQRM) of cloud platforms.

Different initiatives, projects and libraries on regulation technologies for the Inter-Cloud environment are covered in (Grozev and Buyya, 2014), (Toosi, Calheiros and Buyya, 2014). Libraries, for example JCloud (java library), Apache LibCloud (python library), or Apache DeltaCloud (ruby library), and projects such as InterCloud, OPTIMIS, mOSAIC, STRATOS, Contrail, have been designed to abstract the programmers from the differences in the management APIs of clouds and to provide control over resource provisioning. The EU Contrail project, for example, provides support for SAML and OAuth in their latest version. However, this is still not a practical solution for various cloud platforms, as they are closed and don't have built-in support for SAML yet (e.g. Hyper-V).

In (Abdo, Demerjian, Chaouchi, Barbar and Pujolle, 2013) Abdo et al. discuss the cross-cloud federation manager and propose a centralised broker-based approach. A new entity named "broker" is similar to the centralised solution described in this paper, but has other goals (change of discovery and match making).

3 THE APPROACH TAKEN IN THE SOLUTION DESIGN

The multi-platform cloud infrastructure consisted of the OpenStack and VMware cloud platforms, two of the most known and widely used platforms for provision of a cloud infrastructure as a service. OpenStack is open source software for the construction of private and public clouds. It uses a role-based access control (RBAC) mechanism to manage accesses to its resources (Ferraiolo et al., 2001). The identity service in OpenStack is named Keystone. The service authenticates users and provides them with authorization tokens that can be used for accessing the OpenStack services. The current version of Keystone is centralized, and all its users need to be registered in the Keystone database.

The other platform, VMware is considered as one of the most feature-completed platforms in the field.

VMware vCloud Director is a cloud platform software solution that enables enterprises to build secure, multi-tenant private clouds by pooling infrastructural resources into the virtual data centres, and exposing them to users through the web-based API and REST interfaces as fully automated, catalogue-based services. The vCloud Director also supports RBAC. The authentication method used in vCenter that helps automate VMware vCloud Director and other virtualization management system processes is called "Share a unique session". vCenter uses a single set of credentials to enable connection to the vCloud. The latest release 5.5 of the vCenter application has introduced a new SSO approach capability, which allows users to log in just once, and obtain the valued authentication for all vCloud's components. The vCloud API uses basic HTTP authentication, which enables clients to obtain authentication tokens.

By considering these characteristic the solution for enabling a central SSO facility for both platforms was designed with an aim to provide flexible and secure authentication and authorization service for both platforms. The solution is briefly described in the next section. It was named Common Authentication Solution for multi-platform cloud or shortly CAS.

4 COMMON AUTHENTICATION SOLUTION FOR MULTI-PLATFORM CLOUD SERVICE

4.1 Functional Description

The main objective followed during the development of the CAS solution was to enable the cloud infrastructure administrators and the other users to access heterogeneous cloud's infrastructure services using just a single credential. The solution takes into account cloud platforms' specifics, as well as the services and APIs already offered by those platforms, and extends its usage to authorization and user access rights management. Other objectives that were followed during the development were to enable:

- better user experience; users should be able to move between services securely and uninterrupted,
- one dashboard for interacting with different cloud SPs,
- reduction of the processing costs, obtained by reducing the number of calls,

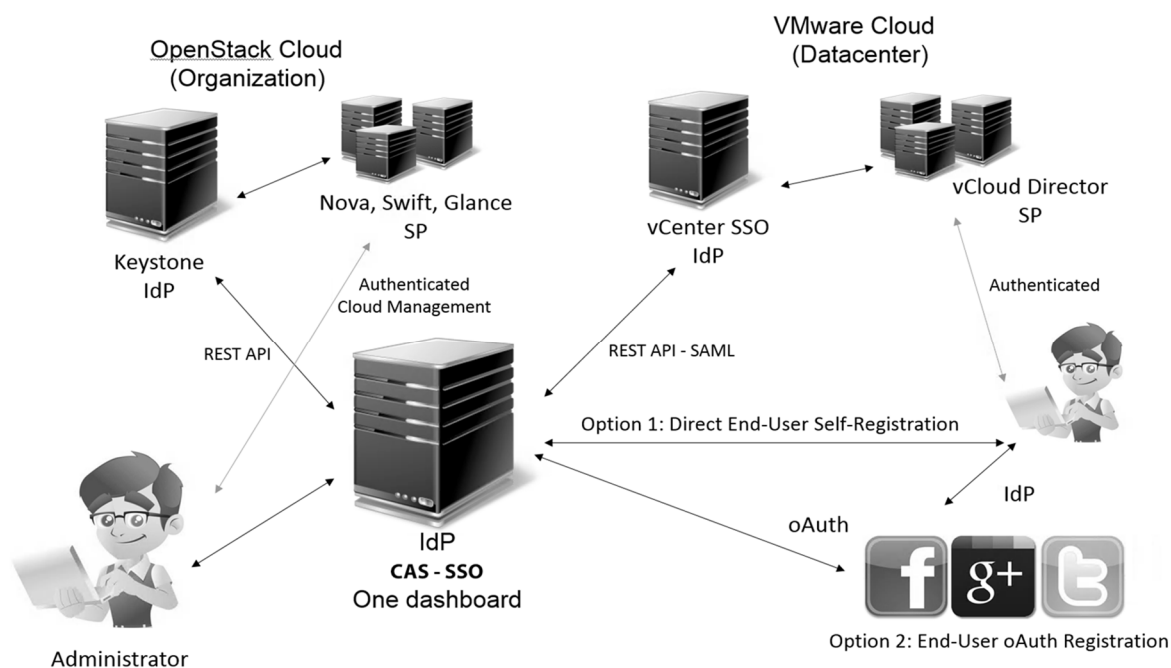


Figure 1: The CAS solution scheme.

- significantly shorter time required for management of multiple accounts,
- higher level of the information security,
- provision of an audit log of the user operations and actions.

By following these objectives CAS becomes a favourable solution that enables administrators, employees, consumers, and customers to access clouds and their services with a single credential. In order to illustrate how CAS works, we are using here the example of an organisation (e.g. corporation or university) with several different cloud infrastructures on site or at a dislocated facility. The infrastructure consists of different platforms for each of the cloud, e.g. OpenStack and VMware. CAS is located on the organisation site, and allows the site administrator to add multiple cloud platforms, either inside or outside the organisation. The CAS application contains a database with the end users of the organisation enabling the site administrator to manage the allocation of resources and, the end users applications in the different system platforms. The end users need only one login. A user logs into the CAS system and he finds there virtual machines assigned to different cloud platforms. Therefore, the federated approach and SSO in the CAS allows the successfully authenticated end users to access additional cloud platforms and services without re-authenticating, since the relationship of trust had already been established and was provided by the CAS system.

The basic operational principles implemented in the system are shown in Fig. 1. The system is split in two parts – front end and back end. The first part of the system is a web interface, built with the help of the programming languages PHP, HTML, and JavaScript. This interface enables end users to sign into a central system with a browser and gain access to all assigned cloud platforms. An administrator assigns permissions to end-users, and manages the access to different cloud platforms through the web browser. The back end of CAS is responsible for mapping, synchronizing and removing end users from the remote cloud platforms. It uses REST access calls (REST over HTTP protocol) for communication with a remote terminal. The format of the exchanged data depends on the end cloud platform accessed (JSON and XML).

CAS provides two ways of end-user registration; the first option is self-registration, and the second one is registration through a social network account (OAuth2). The OAuth2 solution, which has become a popular open standard for authorization supported by Google, Facebook, Twitter, and Yahoo was implemented to enable another simple access to the end users. It was designed for web applications with servers that store confidential information, maintain, state, and provide a secure way for an application like CAS without requiring usernames and passwords (Oracle, 2013). In addition to the OAuth and self-registration, CAS supports also LDAP or AD integration.

Before the CAS administrator can manage end-users, an instance of the remote cloud platform must be created in the central system. Figure 2 shows the CAS administrator dashboard, enabling the administrator to easily add any cloud platform that becomes part of the CAS. First, the administrator selects the type of the platform and triggers the generation of the extra required data fields for the selected platform type. These fields are needed for creation of the local instance of the remote platform. In the case of OpenStack, the fields that belong to the remote administrator, such as password, token and tenant, are required to be filled with relevant data. Once the cloud platform is added, the administrator can import users from remote cloud platforms or add them to the remote cloud platform on the CAS administrator dashboard.

The back end part of the system automatically adds a new end-user with the help of the REST calls. In case of OpenStack, the REST calls are used to insert end-users into the Keystone application. A more detailed explanation of the end-user mapping by the central system in the cloud platform can be found in (Cigoj, 2014).

An end user logs into CAS by providing his CAS login credentials. After a successful login, the user is presented with the dashboard of remote platforms that are accessible to him with a single click on the login button without a username or a password. The end

user can then create, run and power off virtual instances.

4.2 Implementation

The implementation of the CAS was based on the available APIs of the cloud platforms, known applications, such as CURL, REST, HTTPS, and the Python programming languages.

For the OpenStack platform PHP scripts with the support of CURL (a command line tool for transferring data) were used to ensure the transfer of end users from CAS into the remote OpenStack and to automate common end user and administrative tasks in combination with the CAS. The OpenStack authentication service provided with the Keystone’s API interface which is a component that allows the administrators to manage and map end users into the remote Keystone database was used for enabling the end-user authentication process between the CAS and the remote OpenStack platform. The Keystone Identity Service enables clients to obtain tokens for access to the OpenStack cloud services. The Keystone API is using the RESTful web service interface where all authentication and operation requests directed to the Keystone API are performed with the SSL protocol over the HTTP (HTTPS). Since Keystone code can be executed by use of the HTTP sessions, the external authentication methods were

sso class Admin ▾

Welcome, admin ▾

Clouds / New cloud

Cloud settings

Type:
select type
✓ OpenStack
VMware

Name:

Endpoint:

Dashboard:

Save

Additional info for OpenStack

Admin user:

Admin pass:

Admin token:

User tenant:

Active: ☒

Figure 2: CAS administrative dashboard.

applied. The Keystone SQL identity back end facility was used together within the CAS for mapping the end users credentials only once and for login them into the OpenStack dashboard. Their credentials are supplied to CAS only once. The Keystone API supports both JSON and XML data serialization formats, the response format uses JSON by default. This feature was used in the CAS and the X-Auth-Token is accompanied with the URLs of the other services in the cloud.

The vCloud connection with the CAS is similar to the solution applied in the OpenStack platform. The vCloud APIs provide rich functionalities for the management part of the vCloud platform, the vCloud Director. Two methods were possible to be used for the interaction of the vCloud Director cells with the CAS: through a web browser UI or through the vCloud API. The vCloud APIs are RESTful- based, they are highly scalable, and use the HTTP or HTTPS protocol for communication. As the vCloud directory contains a set of APIs for vCloud's provisioning and management controls the programming of the necessary extensions was relatively easy. The returned objects from the API follow the XML scheme where the properties are represented as elements, and the object values as element attributes.

The prototype code that works across the OpenStack and vCloud platforms is offered as an open source code (<https://github.com/primozc/ss0>).

4.3 User Migration and Federation

During the development of CAS a special attention was given to the migration of the users from the CAS to the cloud platform. An end user who is migrated to the cloud and authenticated at CAS is able to access the cloud platform assigned to him by an administrator. As all users are registered and authenticated in the CAS, the management of the end users in remote identity service is not necessary anymore.

API operations are performed to map the end users in the local CAS database to the OpenStack Keystone and the vCloud user database. These operations provided by CAS enable administrators to obtain and validate access tokens, manage users, tenants, roles, and service endpoints. The administrative API calls against services require authentication; the calls to discover services are the only ones without authentication.

An administration token (token for API calls) is used for various administrative operations, such as the integration of the OpenStack with the CAS or user mapping. The Keystone Identity API verifies the

issued administration token and defines the administration role. Administration tokens are stored in the local CAS database system in the process of adding a new cloud platform. A set of identity attributes, such as email, username and tenant, are additionally inserted into the OpenStack Keystone API when the migration of the users into a cloud platform is performed

The connection between the vCloud component of vCloud platform and CAS is similar to the one described above. CAS connects a user to vCloud on the user's request by an API call where user credentials are passed as parameters. Since vCloud supports SAML, the open source SimpleSAMLphp library, which implements the SAML 2.0 standard, was used for exchange of user messages between CAS and the remote vCloud platform. In CAS an IdP was defined according to the vCloud Organization Federation Settings. End users, user groups' data and their roles in vCloud are required to be mapped from the organization's database and from the organization's SAML provider. This restriction required additional functionality in CAS to be developed for provision of the data mapping stored in the vCloud Director database. As SimpleSAMLphp application does not support dynamic generation of metadata in an SP's remote configuration file, an additional upgrading code was developed. The generated XML metadata file from the CAS IdP contains several certificates and information (e.g., SingleLogoutService or AssertionConsumerService) which are necessary by the vCloud Director to be able to communicate with the CAS IdP and to validate if it is sufficiently trustworthy. The generated XML metadata file from the CAS IdP needs to be uploaded into the vCloud federation metadata XML form (VMware, 2012). Since SAML users and groups cannot be found by use of a search function, user's data have to be mapped into vCloud with an automated API call. Adding this feature to the SSO the vCloud Director integration became complete.

5 DISCUSSION AND CONCLUDING REMARKS

The security threats in cloud computing can be removed to a big extent with the system protection capable to resist the attacks. When business entity networks are migrated to the cloud, their data and systems are no longer isolated, they share resources with many other organizations, and this new status is becoming much more attractive for malicious

attackers. In the cloud computing core technologies identified vulnerabilities are mainly related to the virtual machine escape, poor password protection, poor authentication and authorization systems, session hijacking, and insecure cryptographic algorithms. The presented work in this paper can be considered as an attempt to remove some of these vulnerabilities that are related to the user authentication and authorization.

Authentication and authorization of the OpenStack and VMware platforms were carefully studied before a solution enabling a federated IdM approach to be applied in cloud computing environment emerged. The developed solution, the Common Authentication Solution connects both platforms and enables secure authentication service and friendly remote user management. CAS has been implemented for integrating the user authentication of the addressed platforms, but it is sufficiently general to be used for other environment as well. The Microsoft cloud solution already offers some options. A SOAP library (Ruby library), for example, can be used to use the functionality in Windows Remote Management (WinRM) to call native object in Windows. This includes, but is not limited to, running batch scripts, powershell scripts and fetching WMI variables. This way, we can communicate and map users between CAS and the Hyper-V cloud. Another popular open source cloud platform OpenNebula contains a patch in the authentication system, and two standard SimpleSAMLphp modules that can be used to establish connection between CAS and OpenNebula. Furthermore, Eucalyptus and CloudStack are still missing the SAML support in their authentication system, but their aim is to integrate the SSO SAML support. Despite the lack of SAML support there can be a patch developed to support this feature. It is necessary to reiterate at this point that our aim was to provide a unified interface for many other well-known cloud providers and provide simple integration of our platform with other IaaS platforms.

Acting as a kind of a broker, CAS introduces only a slight overhead (login to CAS) to the multi-platform cloud operation from the user point of view when only one platform is accessed. On the other hand CAS relieves the user from frustration of having to remember multiple passwords and enables him easier access to multiple cloud platforms. CAS functionality improves administration performance by providing one interface to manage multiple cloud platforms. The amount of time spent for logging on to different cloud platforms is reduced and it provides faster access to the resources.

Cloud computing still needs much more development and deployment for provision of secure and trustworthy services. The future development is planned to be oriented towards provision of a unified access point for many other well-known cloud providers such as Amazon, DigitalOcean, Slicehost, or Rackspace. For this reason, our future work is oriented to the extension of the functionality of the CAS system in order to support other features that are common to different cloud providers and platforms, such as management of a cloud network, virtual machine, image and storage.

REFERENCES

- Abdo, J. B., Demerjian, J., Chaouchi, H., Barbar, K., & Pujolle, G. (2013). Broker-Based Cross-Cloud Federation Manager. In *Internet Technology and Secured Transactions (ICITST)*, 2013 8th International Conference for (pp. 244-251). IEEE.
- Andronache I., Nisipasiu C., 2011. Web single sign-on implementation using the simpleSAMLphp application. *Journal of Mobile, Embedded and Distributed Systems*. 3(1):21-9.
- Cantor S., Kemp I.J., Philpott N.R., Maler E., 2005. Assertions and protocols for the oasis security assertion markup language. *OASIS Standard*.
- Cigoj P., 2014. Cloud computing security and identity management in the OpenStack platform. Ljubljana: Jožef Stefan International Postgraduate School.
- Cruz Zapata, B., Fernández-Alemán, J.L., & Toval, A. (2014). Security in Cloud Computing: a Mapping Study. *Computer Science and Information Systems* 12(1):161–184.
- Ferg B., Fitzpatrick B., Howells C., Recordon D., Hardt D., Reed D., et al. 2007. OpenID authentication 2.0.
- Fernandes, D.A.B., Soares, L.F.B, Gomes, J.V., Freire, M.M., & Inácio, P.R.M., 2014. Security issues in cloud environments: a survey. *International Journal of Information Security*, vol. 13, iss. 2, pp. 113-170.
- Ferraiolo D.F., Sandhu R., Gavrila S., Kuhn D.R., Chandramouli R., 2001. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*. 4(3):224-74.
- Group TO, 2014. Single Sign On. Available from: <http://www.opengroup.org/security/sso/>.
- Grozev, N., & Buyya, R. (2014). Inter-Cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3), 369-390.
- Jansen, W., Grance, T., 2011. Guidelines on security and privacy in public cloud computing. *NIST special publication*. 800:144.
- Lonea A.M., Tianfield H., Popescu D.E., 2003. Identity management for cloud computing. *New Concepts and Applications in Soft Computing*: Springer. 175-99.

- Microsoft. Microsoft Urges Government and Industry to Work Together to Build Confidence in the Cloud 2010. Available from: <http://www.microsoft.com/en-us/news/press/2010/jan10/1-20brookingspr.aspx>.
- Oracle, 2013. Oracle Access Management OAuth Service 2013. Available from: <http://www.oracle.com/tech/network/middleware/id-mgmt/overview/oauthservice/white-paper-2110557.pdf>.
- Panarello, A., Celesti, A., Fazio, M., Villari, M., & Puliafito, A. (2014). A Requirements Analysis for IaaS Cloud Federation. In 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain.
- Pérez-Méndez, A., Pereniguez-Garcia, F., Marin-Lopez, R., López-Millán, G., & Howlett, J. (2014). Identity Federations Beyond the Web: A survey. *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 4.
- Simmonds, P., Rezek, C., Reed, A., 2011. Security guidance for critical areas of focus in cloud computing v3.0. Cloud Security Alliance. 176 pages.
- Tripathi, A., Mishra, A. (2011). Cloud computing security considerations. In: *IEEE International Conference on Signal Processing, Communications and Computing*, pp. 1–5.
- Toosi, A. N., Calheiros, R. N., & Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)*, 47(1), 7.
- VMware, 2012. vCloud director user's guide, 2012. Available from: http://pubs.vmware.com/vcd-51/topic/com.vmware.ICbase/PDF/vcd_51_users_guide.pdf.

Executing Bag of Distributed Tasks on Virtually Unlimited Cloud Resources

Long Thai, Blesson Varghese and Adam Barker

*School of Computer Science, University of St Andrews, Fife, U.K.
{l1t2, varghese, adam.barker}@st-andrews.ac.uk*

Keywords: Cloud Computing, Bag of Distributed Tasks, Cost vs Performance Trade-off, Decentralised Execution.

Abstract: Bag-of-Distributed-Tasks (BoDT) application is the collection of identical and independent tasks each of which requires a piece of input data located around the world. As a result, Cloud computing offers an effective way to execute BoT application as it not only consists of multiple geographically distributed data centres but also allows a user to pay for what is actually used. In this paper, BoDT on the Cloud using virtually unlimited cloud resources is investigated. To this end, a heuristic algorithm is proposed to find an execution plan that takes budget constraints into account. Compared with other approaches, for the same given budget, the proposed algorithm is able to reduce the overall execution time up to 50%.

1 INTRODUCTION

Bag-of-Tasks (BoT) is the collection of identical and independent tasks executed by the same application in any order. Bag-of-Distributed-Tasks (BoDT) is a subset of BoT in which each task requires data from somewhere around the globe. The location where a task is executed is essential for keeping the execution time of the BoDT low, since data is transferred from a geographically distributed location. It is ideal to assign tasks to locations that would be in geographically close proximity to the data.

The centralised approach for executing BoDT, in which data from multiple locations are transferred and executed at a single location, tends to be ineffective since some data resides very far from the selected location and takes a long time to be downloaded. Another approach is to group the tasks of the BoDT in such a way that each group can be executed near the location of the data. However, this approach requires an infrastructure which is decentralised and globally distributed. Cloud computing is ideally suited for this since public cloud providers have multiple data centres which are globally distributed. Furthermore, since clouds are available on a pay-as-you-go basis, it is cost effective as a user only pays for Virtual Machines (VMs) that are required.

Cloud computing can facilitate the execution of BoDT, and at the same time introduce the challenge of assigning tasks to VMs by considering the location for processing each task, the user's budget con-

straint, as well as the desired performance, i.e. execution time, for executing the task. In an ideal case, it is expected that maximum performance is obtained while minimising the costs.

In our previous paper (Thai et al., 2014b), we approached this problem by assuming limited resources were available. However, as Cloud providers offer virtually unlimited resources, the limit should be determined based on the user's budget constraint. In this paper, we present our approach for executing BoDT on the Cloud with virtually unlimited resources and is only limited by a user specified budget constraint. Compared with other approaches, with the same given budget, our algorithm is able to reduce the overall execution time up to 50%.

The contributions of this paper are i) a mathematical model of executing a BoDT application on the Cloud with budget constraints, ii) a heuristic algorithm which assigns tasks to Cloud resources based on their geographical locations, and iii) an evaluation comparing our approach with centralised and round robin approaches.

The remainder of paper is structured as follow. Section II presents the mathematical model of the problem. Section III introduces the heuristic algorithms producing an execution plan based on the user's budget constraint. Section IV evaluates the approach. Section V presents the related work. Finally, this paper is concluded in section VI.

2 PROBLEM MODELLING

Let $L = \{l_1 \dots l_m\}$ be the list of Cloud locations, i.e. location of Cloud provider's data centres, and $VM = \{vm_1 \dots\}$ be the list of Cloud VMs. For $vm \in VM$, $l_{vm} \in L$ denotes the location in which vm is deployed. Let $VM_l \subset VM$ be the list of all VMs deployed at location $l \in L$. The number of items in VM is not fixed since a user can initiate as many VMs as possible.

Let $T = \{t_1 \dots t_n\}$ be the list of tasks, and $size_t$ denote the size of a task. The time (in seconds) taken to transfer data from a task's location to a Cloud location is denoted as $trans_{t,l}$. Similarly, $trans_{t,vm}$ for $vm \in VM$ is the cost of moving t to vm (or to a location on which vm is running; $trans_{t,vm} = trans_{t,l_{vm}}$). We assume that there is only one type of VM is used, hence, the cost of processing one unit of data is identical and is denoted as $comp$.

The time taken to execute task t at vm is:

$$exec_{t,vm} = exec_{t,l_{vm}} = (trans_{t,vm} + comp) \times size_t \quad (1)$$

Let $T_{vm} \subset T$ be the list of tasks executed in $vm \in VM$. All tasks must be executed and is represented as the following constraint:

$$\bigcup_{vm \in VM} T_{vm} = T \quad (2)$$

One task should not be executed in more than one location expressed as an additional constraint:

$$T_i \cap T_j = \emptyset \quad \text{for } i, j \in VM \text{ and } i \neq j \quad (3)$$

The execution time of all tasks on $vm \in VM$ is:

$$exec_{T_{vm}} = \sum_{t \in T_{vm}} exec_{t,vm} \quad (4)$$

As it takes some times to create a VM, the overhead associated with the start up of each VM denoted as $start_up$. The execution time of $vm \in VM$ to execute all tasks in T_{vm} is:

$$exec_{vm} = start_up + exec_{T_{vm}} \quad (5)$$

It should be noted that Equation 5 can only be applied if there are task(s) assign to a VM, i.e. $T_{vm} \neq \emptyset$. Otherwise, it is unnecessary to create a VM, thus its execution time is zero.

Assuming each VM is charged by hour, i.e. 3600 seconds, the number of charged time blocks is:

$$tb_{vm} = \lceil \frac{exec_{vm}}{3600} \rceil \quad (6)$$

Equation 6 contains the ceiling function, which means the execution time is rounded up to the nearest hour in order to calculate the number of used time blocks. In other words, a user has to pay for a full hour even if only a fraction of the hour is used.

Let $P = \{T_{vm_1} \dots T_{vm_p}\}$ be the execution plan, whose each item is a group of tasks assigned to one $vm \in VM$. Let VM_P denote the list of VMs used by execution plan P . Similarly, let L_P be the list of locations where all VMs of plan P are deployed. Moreover, P_l denotes the execution plan for location $l \in L$, which means $L_{P_l} = \{l\}$ and $VM_{P_l} = VM_l$.

As all VMs are running in parallel, the execution time of a plan is equal to slowest VM's:

$$exec_P = \max_{vm \in VM_P} exec_{vm} \quad (7)$$

The total number of time blocks used is the sum of the time blocks used by each VM, represented as:

$$tb_P = \sum_{vm \in VM_P} tb_{vm} \quad (8)$$

The budget constraint is the amount of money that a user is willing to pay for executing the BoDT. Even though Cloud providers charge users for using compute time on virtual machines and transferring data, only the renting cost is considered as the amount of downloaded is unchanged for any given problem, i.e. regardless the execution plan, the same amount of data is downloaded, thus the data transferring cost.

The budget constraint is mapped onto the number of allowed time blocks tb_b by dividing the budget to the cost of one time block (this is possible, because of the assumption that there is only one VM type). Hence, the problem of maximising the performance of executing a BoDT on the Cloud with a given budget constraint is to find an execution plan P in order to minimise $exec_P$ while keeping $tb_P = tb_b$ and satisfying constraints in Equations 2 and 3.

3 ALGORITHMS

As stated in the previous section, the optimal plan for executing BoDT on the Cloud with budget constraint can be found by solving the mathematical model. However, solving the mathematical model can take considerable amount of time since it involves considering multiple possibilities of assigning tasks to different VMs at multiple Cloud locations. In this section, we propose an alternative approach which is a heuristic algorithm for finding an executing plan for a BoDT based on a user's budget constraint.

3.1 Select Initial Number of VMs at Each Location

The main idea of the approach presented in this paper is to specify a set of VMs for each location, then

to reduce the number until the total number of VMs across all locations is tb_b .

In order to determine the initial number of VMs at each location, we make an assumption that it is possible to limit each VM to be executing in one time block, i.e. if a VM finishes its execution in more than one time block, its tasks can be split and scheduled onto two VMs. Then, the total number of time blocks is equal to the total number of VMs across all locations. Thus, the constraint tb_b also limits the total number of VMs, each of which uses no more than one time block. Hence, initially, the number of VMs at each location, i.e. \bar{VM}_l for $l \in L$, can be set to tb_b .

3.2 Find Execution Plan based on Budget Constraint

Let P_{nl} be the plan in which tasks are assigned to their nearest location, i.e. the location in which $exec_{t,l}$ is minimum. Each item in P_{nl} represents the list of tasks assigned to a location (not a VM).

Algorithm 1: Find Execution Plan based on Budget Constraint.

```

1: function FIND_PLAN( $tb_b, P_{nl}, VM$ )
2:    $P \leftarrow \emptyset$ 
3:   for  $l \in L_{P_{nl}}$  do
4:      $P_l \leftarrow ASSIGN(T_l, VM_l)$ 
5:     if  $tb_{P_l} > tb_b$  then
6:       FAIL
7:     end if
8:      $P \leftarrow P_l$ 
9:   end for
10:   $P \leftarrow REDUCE(P, \emptyset, TRUE)$ 
11:  if  $tb_P > tb_b$  then
12:     $P \leftarrow REDUCE(P, \emptyset, FALSE)$ 
13:  end if
14:  if  $tb_P > tb_b$  then
15:    FAIL
16:  end if
17:   $P \leftarrow BALANCE(P)$ 
18:  return  $P$ 
19: end function
    
```

Algorithm 1 finds a plan with minimum execution time based on the budget constraint tb_b . The nearest plan P_{nl} and the initial list of virtual machines VM are provided as input. The algorithm uses three functions, namely *ASSIGN*, *REDUCE* and *BALANCE*.

First of all, the algorithm assigns tasks to VMs deployed in their nearest locations (From Line 3 to 9). Line 5 checks if the number of used time block in a location is more than the budget constraint. If that

is the case, then it is impossible to find an execution plan satisfying the given budget constraint.

Secondly, some VMs are removed by moving its tasks to other ones until the budget constraint is satisfied (From Line 10 to 13). The reassignment can be performed between VMs in the same location or across multiple locations. If after reducing, the number of VMs is still higher than tb_b , it is impossible to satisfy the budget constraint (Lines 14 and 15).

Finally, as the execution times between VMs are different (for example, one VM can take longer to finish than the other ones) it is necessary to balance out the execution times between all VMs so that they can finish at the same time, thus reduce the overall execution time (Line 17).

3.3 Assign Tasks to VMs

Algorithm 2 aims to evenly distributed tasks from T' to the set of receiving VMs.

Algorithm 2: Assign Tasks to VMs.

```

1: function ASSIGN( $T', VM'$ )
2:    $T' \leftarrow T'$  sorted by  $-exec_{t,l}$  for  $t \in T'$ 
3:   for  $t \in T'$  do
4:      $VM_0 \leftarrow VM'$  filtered  $exec_{vm} + exec_{t,vm} \leq$ 
       3600
5:     if  $VM_0 = \emptyset$  then
6:       FAIL
7:     end if
8:      $VM_0 \leftarrow VM'$  sorted by  $(trans_{t,vm}, exec_{vm})$ 
       for  $vm \in VM'$ 
9:        $VM_0 \leftarrow \arg \min_{vm \in VM'} trans_{t,vm}$ 
10:       $vm \leftarrow VM_0[0]$ 
11:       $T_{vm} \leftarrow T_{vm} \cup \{t\}$ 
12:    end for
13:     $P_{nl} \leftarrow \{T_{vm} \text{ for } vm \in VM'\}$ 
14:  return  $P_{nl}$ 
15: end function
    
```

First of all, tasks are sorted in descending order based on their execution times (Line 2). Then, for each task, all the VMs which can execute it without requiring more than one time block is selected (Line 4). If there is no VM selected, i.e. it will take more than one time block if a task is assigned to any given VMs, the function fails (Lines 5 and 6).

All the selected VMs are sorted based on the distance between VM's location and the task's location, and by their current execution time (Lines 8). The task is assigned to the first VM in the sorted collection (Lines 10 and 11). In other words, Algorithm 2 tries to assign a task to the nearest VM with the lowest execution time.

3.4 Reduce the Number of VMs

Algorithm 3 is used to reduce the number of VMs by moving all tasks from one VM to others which are either in the same or on different locations. It is a recursive process which takes the current plan P_n , and the list of VMs which cannot be removed from the plan Ign , and the boolean value indicating if the reducing process is applied locally or globally is_local .

Algorithm 3: Reduce VMs.

```

1: function REDUCE( $P, Ign, is\_local$ )
2:    $vm \leftarrow \arg \min_{vm \in VM_P} exec_{vm}$ 
3:   if  $is\_local = TRUE$  then
4:      $VM' \leftarrow VM_{l_{vm}} - vm$ 
5:   else
6:      $VM' \leftarrow VM_P - vm$ 
7:   end if
8:    $P' \leftarrow ASSIGN(T_{vm}, VM')$ 
9:   if  $tb_{P'} < tb_P$  then
10:     $P \leftarrow P'$ 
11:  else
12:     $Ign \leftarrow Ign \cup \{vm\}$ 
13:  end if
14:  if  $tb_P = tb_b$  or  $\overline{Ign} = \overline{VM_P}$  then
15:    return  $VM_l$  for  $l \in L$ 
16:  else
17:    return  $LOCAL\_REDUCE(P_n, Ign)$ 
18:  end if
19: end function

```

First, a VM with lowest execution time is selected (Line 2). Then the remaining VMs, which can be either in the same (Line 4) or on different Cloud location (Line 6), are selected as receiving VMs.

After that, all tasks from selected VM are reassigned to other VMs (Line 8) by reusing the Algorithm 2. Notably, the receiving VMs are not empty but already contain some tasks.

If the reassignment reduces the number of VMs (Line 9), the current plan is updated (Line 10). Otherwise, the selected VM is added into the ignore list Ign (Line 12). If the total time block satisfies the given constraint or all VMs are ignored (Line 14), the process stops and returns the current plan (Line 14), otherwise it continues (Line 17).

3.5 Balance Tasks between VMs

After the budget constraint is satisfied, the execution times between VMs can be uneven, i.e. some VMs can have higher execution times than the others. As the execution time of the plan $exec_P$ is based on the

VM with highest execution time, it is necessary to balance out execution time between them.

Algorithm 4: Balancing Algorithm.

```

1: function BALANCE( $P$ )
2:    $vm \leftarrow \arg \min_{vm \in VM_P} exec_{vm}$ 
3:    $T'_{vm} \leftarrow T_{vm}$  sorted by  $-exec_{t,vm}$ 
4:   for  $t \in T'_{vm}$  do
5:      $VM_1 \leftarrow (VM_P - \{vm\})$  sorted by  $trans_{t,vm}$ 
6:      $vm_0 \leftarrow NULL$ 
7:     for  $vm_1 \in VM_1$  do
8:       if  $t$  is never in  $vm_1$  then AND  $rt_{c_1} +$ 
           $exec_{t,c_1} < rtc_0$ 
9:          $vm_0 \leftarrow vm_1$ 
10:      BREAK
11:    end if
12:  end for
13:  if  $vm_0 \neq NULL$  then
14:    BREAK
15:  end if
16: end for
17: if  $vm_0 \neq NULL$  then
18:    $T'_{vm} \leftarrow T_{vm} - t$ 
19:    $T'_{vm_0} \leftarrow T_{vm_0} \cup \{t\}$ 
20:    $P \leftarrow (P - \{T_{vm}, T_{vm_0}\}) \cup \{T'_{vm}, T'_{vm_0}\}$ 
21:   go to 2
22: end if
23: return  $P$ 
24: end function

```

Algorithm 4 is an iterative process which tries to move tasks from a VM with highest execution time (Line 2) to the nearest VM possible. There are two conditions for selecting a receiving VM: the selected task is never assigned to it and its execution time after receiving the task is not higher than the current execution time of the giving VM (Line 8).

3.6 Dynamic Scheduling To Avoid Idle VM

Even though Algorithm 1 aims to build the plan in which all VMs finish their execution nearly at the same time, due to the instability of the network and other unaccountable factors, e.g. service failure, it is not unusual for one VM to finish before others. As the cost of a full hour is already paid, it is necessary to utilise the remaining time of the finished VMs in order to reduce not only idle and unpaid time but also the execution time of other VMs.

Let rt_{vm} be the actual running time of a VM. Let e_{vm} and $T_{r_{vm}}$ be the estimated remaining execution time and remaining tasks of $vm \in VM$.

terminate_time denote the time it take for a VM to be shut down. Finally, thr_1 and thr_2 are two threshold values indicating the required remaining execution time and number of tasks. As unfinished VMs are still running when the reassignment is being performed, those thresholds aim to avoid reassigning tasks already executed by one VM to another. The idea of dynamic rescheduling is to move $T_{r_{vm}}$ of a VM to another finished one while satisfying thr_1 and thr_2 in order to reducing its e_{vm} .

In order to support dynamic scheduling, we add a feature which monitors the execution of VMs, keeps track of the remaining tasks and execution times, and detects a VM which has just finished its execution.

Algorithm 5: Dynamic Reassignment.

```

1: function REASSIGN(vm)
2:   if  $3600 - rt_{vm} < terminate\_time$  then
3:     FAIL
4:   end if
5:    $VM_1 \leftarrow \{VM_P - \{vm\}\}$  sorted by  $-e_{vm_1}$  for
      $vm_1 \in VM_1$ 
6:    $vm_0 \leftarrow NULL$ 
7:   for  $vm_1 \in VM_1$  do
8:     if  $e_{vm_1} \leq thr_1$  AND  $\overline{T_{r_{vm_1}}} \leq thr_2$  then
9:        $vm_0 \leftarrow vm_1$ 
10:      BREAK
11:    end if
12:  end for
13:  if  $vm_0 = NULL$  then
14:    FAIL
15:  end if
16:   $T'_r \leftarrow T_{r_{vm}}$  sorted by  $trans_{t,vm}$  for  $t \in T'_r$ 
17:   $T \leftarrow \emptyset$ 
18:   $el \leftarrow 3600 - rt_{vm} - terminate\_time$ 
19:  for  $t \in T'_r$  do
20:     $exec'_T \leftarrow exec_T + exec_{t,vm}$ 
21:    if  $exec'_T \geq \frac{e_{vm_0} - thr_1}{2}$  OR  $exec'_T > el$  then
22:      BREAK
23:    end if
24:     $T \leftarrow T \cup \{t\}$ 
25:     $T'_r \leftarrow T'_r - \{t\}$ 
26:  end for
27:   $T_{r_{vm}} \leftarrow T_{r_{vm}} - T'_r$ 
28:   $T_{vm} \leftarrow T$ 
29:  TIME_OUT(vm, el)
30: end function
    
```

Algorithm 5 is invoked every time a VM that has just finished its execution. First, it check whether there is enough time in a finished VM to execute some tasks (Line 2). This check ensures that the finished VM is able to be terminated before using another time block. Then, the VM which not only has the highest

remaining execution time but also satisfies thr_1 and thr_2 is selected (Lines 5 to 15).

After that, some of the tasks are moved from the selected VM to the finished one until some conditions are met: i) the execution time of the finishes VM is greater or equal half of the remaining execution time of the giving one, or, ii) the finished VM will take more than one time block to finish its execution if more tasks are added (from Lines 16 to 26).

Notably, Algorithm 5 is invoked only one at a time, i.e. if there are multiple VMs that have completed executing their tasks, only one of them is reassigned tasks while other VMs wait.

Finally, the timeout feature is added to prevent the finished VM, which is just assigned some more tasks, to use more than one time block. Basically, it takes the VM and the allowed execution time as arguments (Line 29), if the VM is still running when time out, it is automatically terminated and the remaining tasks are moved to another VM with lowest remaining execution time, i.e. the one that is likely to finish first.

4 EXPERIMENTAL EVALUATION

4.1 Set-up

In order to evaluate our proposed approach, we developed a word count application in which each task involved downloading and counting the number of words in a file from a remote server. Those files were located on PlanetLab (PL), a test-bed for distributed computing experiments (Chun et al., 2003). We had 5700 files across 38 PL nodes and the total amount of data for each experiment run was more than 12 gigabytes. The VMs were deployed on eight Amazon Web Service (AWS) regions.

Prior to the experiment, we ran the test with fewer tasks in order to collect the computational cost, i.e. *comp*, and communicational costs between all AWS regions and PlanetLab Nodes (i.e. *trans*).

Based on our algorithm, at least four VMs were required to execute all 5700 tasks. We then set $tb_b = \{4, 6, 8, 10, 12, 14, 16, 18, 20\}$, i.e. the number of time block (or VMs) that we wanted to use. For each value of tb_b , we ran the execution three times to find the mean and standard deviation.

For comparison, we implemented two simple approaches for executing BoDT on the Cloud:

- Centralised approach: one centralised location was selected as $l_c = \arg \min_{l \in L} (\sum_{t \in T} trans_{t,l} * size_t)$, i.e. the cost of moving all tasks to this location is a minimum

when compared to other locations. This approach was developed based on the centralised approach introduced in our previous paper (Thai et al., 2014b), however, instead of using only one VM at the selected location, in this paper, the number of VMs was equal to the one used by our proposed approach. In other words, this centralised approach enjoyed the same execution parallelism as the proposed one.

- Round Robin approach: for this approach, all Cloud locations was sorted in ascending order based on their costs of moving all tasks to them. This means the first Cloud location was the one selected by the centralised approach. After that, VMs were added to each location in circular order, e.g. the first VM was added to the first Cloud location in the sorted list.

For both approaches, Algorithm 2 was used to evenly distribute tasks to all VMs.

4.2 Dynamic Reassignment

Before going into the main experiment, it is necessary to demonstrate the need of using dynamic reassignment for VMs that finish executing their assigned tasks earlier than others. Figure 1 presents the result of running the same execution plan with $tb_b = 4$, i.e. there were four VMs. Each bar represents the execution time of a VM. Without reassignment, one VM took longer to finish its execution thus increasing the overall execution time. Dynamic reassignment helped to balance out the execution time between VMs so that all VMs could finish at about the same time, which in turn reduced the overall execution time. Dynamic reassignment is applied for the remaining experiments presented in this section.

4.3 Experimental Results

Figure 2 presents the execution times corresponding for each value of the number of VMs for all three approaches. The centralised approach had the highest execution times as even though it selected the location with lowest transfer cost for all tasks but some tasks were very far from the Cloud location which resulted in the high data transfer time. On the other hand, the round robin approach performed better as it deployed VMs at multiple Cloud locations, which means it was possible for tasks to be executed near their data sources. Finally, it is evident that for the same number of VMs (or budget) our approach always had the lowest execution time in comparison with other two.

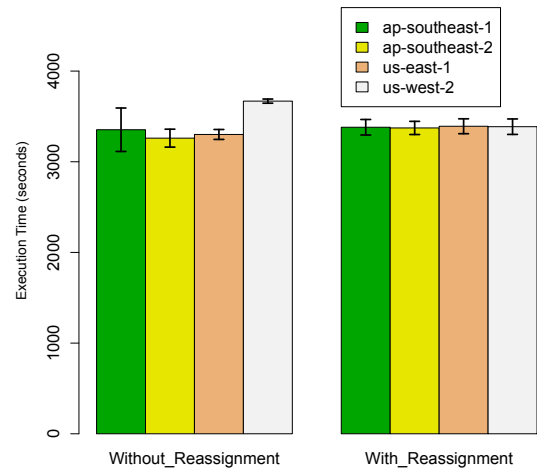


Figure 1: Compare execution without and with reassignment.

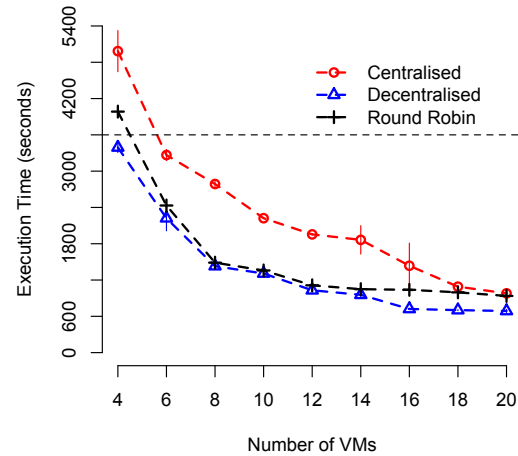


Figure 2: Execution Times.

A reason for the improvement is that our approach not only deployed VMs at multiple locations but also carefully selected those locations so that the majority of tasks could be executed near their data sources. The two simple approaches decided the location(s) of VMs based on **all** tasks, by assuming all tasks were assigned to one Cloud location. On the other hand, our approach took a more fine-grain method by assigning each task to its nearest location first and then reassigning them to others until the budget constraint was satisfied.

As the result, with the same given budget constraint, our approach was 30% to 50% faster than the centralised approach. In comparison to the round robin approach, ours was able to reduce the execution times up to 30%.

Figure 3 presents the number of actual time blocks, which can be mapped onto actual cost, con-

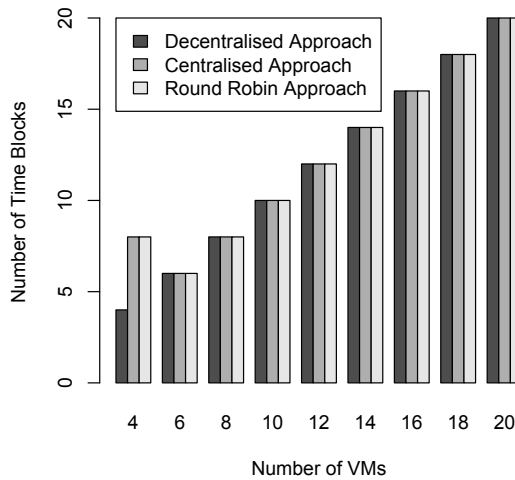


Figure 3: Actual Number of Used Time Blocks, i.e. cost.

sumed by three approaches. It shows that our approach was able to satisfy the budget constraint in all cases. Moreover, when there were four VMs, the centralised and round robin approaches were more expensive than the decentralised one. It was because each of their VMs required more than one hour to finish executing all the assigned tasks and the overall execution time was higher than 3600 seconds, as shown by Figure 2. Which means that the constraint $tb_b = 4$ could only be satisfied by the decentralised approach.

4.4 Trade-off between Cost and Performance

As presented in Figure 2, the higher the budget constraint is (i.e. more VMs), the better the performance is. In theory, it is possible to keep adding more VMs in order to achieve better performance. However, the performance gain for each additional VM also decreases as the total number of VMs increases.

Hence, it is up to the user to decide how much improvement in performance can be afforded. There are some simple criteria to consider such as a defined budget constraint, the desired execution time or defining a threshold in the performance gain (for example, stop adding more VM(s) if the performance gain is less than 60 seconds).

A user can also make the decision of how many VMs to use based on the trade-off between performance and cost, as mentioned in (Thai et al., 2014b).

5 RELATED WORK

In the Grid environment in which the resources are shared between multiple organisations, the overall

performance of a distributed framework by processing data in close proximity to where it resides is improved (Ranganathan and Foster, 2002). Similarly, a heuristic algorithm is proposed to improve the performance of executing independent but file-sharing tasks (Kaya and Aykanat, 2006). An auto-scaling algorithm is proposed to satisfy deadline and budget constraints when each task requires distributed data from multiple sources (Venugopal and Buyya, 2005).

However, the application of Grid computing research on Cloud computing is limited because: i) the Cloud resources are (virtually) unlimited, hence a user is free to add or remove VMs whenever she wants but ii) the monetary cost factor has to be considered as the resource is not available free-of-charge.

Recently, running application on the Cloud has received attention from many researchers. Statistical learning had been used to schedule the execution of BoT on the Cloud (Oprescu and Kielmann, 2010). The method for scaling resource based on given budget constraint and desired application performance was also investigated (Mao et al., 2010). Nevertheless, those papers do not consider the location of data.

Cloud computing is employed for improving the performance of data intensive application, such as Hadoop, whose data is globally located (Ryden et al., 2014). Research that takes geographical distance into account while executing workflows is also reported (Luckeneder and Barker, 2013; Thai et al., 2014a). However, recent researches on applying Cloud computing for applications with geographically distributed data only focus on improving the performance without considering the monetary cost.

Our previous work (Thai et al., 2014b) aimed to determine a plan for executing BoDT on the Cloud, however, it made an assumption that there was only one VM that could be deployed at each Cloud region.

Our paper differentiates itself from prior research by taking advantage of the decentralised infrastructure of Cloud computing in executing BoDT application. We tries to decide not only the amount of resources but also the locations where resources, i.e. VMs, must be located. Moreover, our research exploits of the virtually unlimited resources of Cloud computing by letting a user decides how much resources that she wants based on her budget. Finally, the trade-off between performance gain and additional cost is also presented.

6 CONCLUSION

Due to its decentralised infrastructure and virtually unlimited resources, Cloud computing is suitable to

execute BoDT, whose data is globally distributed all over the world. It is challenging to decide how to assign tasks to Cloud VMs based on a user's budget constraint while minimising the execution time.

The above problem was mathematically modelled in this paper. We also proposed a heuristic approach which assigned BoDT to Cloud VM(s) in order to maximise performance and to satisfy the allowed cost provided by a user.

Furthermore, we implemented a dynamic reassignment feature to utilise the idle time of a VM that completes execution ahead of others by assigning tasks from other VMs onto it. This feature reduces the overall execution time when a number of VMs take longer to finish their execution due to service failure or network instability.

Our approach was evaluated and able to provide execution plans which satisfied given budget constraints. Compared to the centralised and round robin approaches, our approach reduced the execution time on average by 27%. Our approach was also able to satisfy the low budget while the others did not.

In the future, we plan to further improve dynamic resource provisioning and tasks scheduling so that they can be performed during execution in order to handle expected events, e.g. network instability or machine failure. Moreover, the different types of Cloud instances, which have varying performance and cost will be taken into account.

ACKNOWLEDGEMENTS

This research is supported by the EPSRC grant 'Working Together: Constraint Programming and Cloud Computing' (EP/K015745/1), a Royal Society Industry Fellowship, an Impact Acceleration Account Grant (IAA) and an Amazon Web Services (AWS) Education Research Grant.

REFERENCES

- Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. (2003). Planet-lab: An overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12.
- Kaya, K. and Aykanat, C. (2006). Iterative-improvement-based heuristics for adaptive scheduling of tasks sharing files on heterogeneous master-slave environments. *Parallel and Distributed Systems, IEEE Transactions on*, 17(8):883–896.
- Luckeneder, M. and Barker, A. (2013). Location, location, location: Data-intensive distributed computing in the cloud. In *Proceedings of IEEE CloudCom 2013*, pages 647–653.
- Mao, M., Li, J., and Humphrey, M. (2010). Cloud auto-scaling with deadline and budget constraints. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 41–48.
- Oprescu, A. and Kielmann, T. (2010). Bag-of-tasks scheduling under budget constraints. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 351–359.
- Ranganathan, K. and Foster, I. (2002). Decoupling computation and data scheduling in distributed data-intensive applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, HPDC '02*, pages 352–, Washington, DC, USA. IEEE Computer Society.
- Ryden, M., Oh, K., Chandra, A., and Weissman, J. B. (2014). Nebula: Distributed edge cloud for data intensive computing.
- Thai, L., Barker, A., Varghese, B., Akgun, O., and Miguel, I. (2014a). Optimal deployment of geographically distributed workflow engines on the cloud. In *6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014)*.
- Thai, L., Varghese, B., and Barker, A. (2014b). Executing bag of distributed tasks on the cloud: Investigating the trade-offs between performance and cost. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 400–407.
- Venugopal, S. and Buyya, R. (2005). A deadline and budget constrained scheduling algorithm for escience applications on data grids. In *Proc. of 6th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP-2005)*, pages 60–72. Springer-Verlag.

Automatic Abstraction of Flow of Control in a System of Distributed Software Components

Nima Kaviani¹, Michael Maximilien², Ignacio Silva-Lepe² and Isabelle Rouvellou²

¹*University of British Columbia, Vancouver, Canada*

²*IBM Watson Research Center, New York, U.S.A.*

nkaviani@cs.ubc.ca, {maxim, isilval, rouvellou}@us.ibm.com

Keywords: Platform-as-a-Service (PaaS), CloudFoundry, Introspection.

Abstract: CloudFoundry (CF) provides an open source platform-as-a-service software for deploying scalable software systems to the cloud. The architecture for CF is distributed by design and consists of several components which interact with one another through a message-oriented middleware. This message-oriented distributed design delivers on the scalability and resiliency requirements of the platform. In such a complex distributed multi-component system, there is a steep learning curve for software developers to understand how components interact, what messages are exchanged between them, and how the message exchanges affect the behaviour of the system. In particular developers find it difficult to identify the execution flows, the authentication flows, interactions with the persistence layer, etc. We have developed a framework that allows interpreting the behaviour of the system by analyzing the exchanged messages between components, inspecting message contents, and extracting data and control flow across components. The paramount aim is to improve developers' understandability of the system and to examine software resiliency through approaches like bug injection and message alterations. An initial version of our framework was released to the CF community and we have collected feedback that indeed show that we are achieving some of our goals.

1 INTRODUCTION

Utilizing open source software (OSS) systems to manage infrastructure, platforms, or applications is increasingly popular in the domain of cloud computing (ope, a)(ope, b). With Openstack (ope, c) and CloudStack (clo, a) as examples of widely adopted open source Infrastructure-as-a-Service (IaaS) enablers, and CloudFoundry (clo, b) and OpenShift (ope, d) as examples of open source Platform-as-a-Service (PaaS) enablers, the anticipated role of OSS in the cloud becomes more apparent than ever before. As such, a lot of companies have started looking into understanding, deploying, and extending these open source platforms for their infrastructure. To name a few examples, IBM is partnered with Openstack (ibm, a) and CloudFoundry (ibm, b) to have their software deployed on its infrastructure; and Baidu (bai, b) has seven hundred developers working on CloudFoundry enabled deployments (bai, a).

With the rapid development cycles for these highly distributed open source cloud platforms, it has become increasingly more difficult for software developers to understand and assess the behaviour of an

existing open source cloud platform, track evolutions of software components across releases, or assess reliability of a new release. OpenStack has already gone through eight major revisions, CloudFoundry has moved from its first version to the second version, and Eucalyptus has already made six releases. In such a fast-evolving software ecosystem, developers and architects adopting these technologies need to understand the issues aforementioned to accurately answer the following questions: *i)* How do the components in the system interact with one another? What is the flow of control and data in the system? What message are exchanged and what are their types and contents? *ii)* How is the system evolved from one release to another? *iii)* How reliable is a new release with respect to changes or use cases required by a target client? and *iv)* How do we detect anomalies in the behavior of the system?

Many of these cloud platforms share a common architectural design, i.e., a distributed multi-component architecture in which component interactions happen through synchronous or asynchronous message exchanges. We developed an initial hypothesis that by capturing all message exchanges across

components in a cloud platform we should be able to address the above questions as follows: *i*) through message correlation and temporal analysis of message exchanges we should be able to derive message sequences and identify the patterns of communication across all messages in the system; *ii*) by analyzing and comparing message contents across different releases of a platform we should be able to track changes in message exchange patterns and project on evolutions at the level of system components; *iii*) by corrupting or interfering with the pattern of message exchanges we should be able to assess the resiliency of the platform from one release to another; and *iv*) by collecting a long enough history of message exchanges we should be able to detect anomalies and irregularities in the behaviour of the system by comparing the expected patterns of message exchange with the newly observed message exchange patterns.

In this paper we discuss how using an instrumentation technique we managed to extract sequences of message exchanges for CloudFoundry, analyze message context, and generate valuable information on the behavior of the system to be shared with the community of CloudFoundry developers. We also provide preliminary results of two releases of our framework to CF developers and users inside IBM as well as to the CF community at large. Finally, we discuss our plans to utilize the current technique to provide automated approaches for software testing and validation.

2 BACKGROUND

2.1 Instrumentation and Profiling

Analyzing system behaviour is done either through black-box profiling techniques or white box instrumentation strategies. In an instrumentation strategy, code snippets are injected into the original source code of the system under study in order to collect information on flow of control or data flow. In a profiling process however, the behaviour of the system is inferred through collecting footprints of system interactions with the underlying framework, the current platform, or the operating system which is used. The collected data then is analyzed or interpreted to form a view of the system's behavior (Beschastnikh et al., 2011). While data collected through black-box profiling is usually insufficient in effectively tracking and monitoring the behavior of a distributed system, instrumentation is also no panacea as it is typically hindered by limited accessibility and comprehension of system source code. Magpie (Barham et al., 2003), MANTICORE (Kaviani et al., 2012), and ARM instru-

mentation (arm,) are examples of systems that allow tracing of code and data through instrumentation. At the other end, Baset et al. (Baset et al., 2013), Aguilera et al. (Aguilera and et al., 2003), and Anandkumar et al. (Anandkumar et al., 2008) provide solutions on doing black-box tracking of software systems.

2.2 Aspect-oriented Programming

Aspect-Oriented Programming (AOP) (Kiczales et al., 2001) provides an abstraction of program execution with techniques that allow to change flow of control or data in order to separate crosscutting concerns spread across multiple abstraction layers in the system from the functional requirements at each abstraction layer. AOP is often conceptualized into the three concepts of *joinpoints*, *pointcuts*, and *advice*. A joinpoint is a metaprogram event identifying a distinguished point of interest in the program; a pointcut defines a query on selecting a certain set of joinpoints in the program; and an advice is a function associated with a pointcut to be executed at a matching joinpoint (Kiczales et al., 2001). AOP has been widely used to analyze and monitor the behavior of distributed systems by injecting monitoring and analysis code into components of a system. The works by Wohlstadter and Devanbu (Wohlstadter and Devanbu, 2006) and Whittle et al. (Al Abed and Kienzle, 2011) are examples of the efforts in utilizing AOP instrumentation in software development and modelling.

2.3 CloudFoundry Architecture

CloudFoundry v1.0 consists of the following major components: the *cloud controller* manages the overall behaviour of the system and instructs the internal components of CloudFoundry on their roles; The *health manager* monitors the well-being of the components; the *User Authorization and Authentication (UAA)* unit performs authorizations; the *stager* prepares deployments; the *Deployment Agent (DEA)* deploys the application and monitors its execution; and the *router* directs traffic from outside CloudFoundry into the deployed applications. Communication between CF components happens in two ways: *a) asynchronously* through messages sent to the pub/sub middleware called the NATS server (nat,) or *b) synchronously* by exchanging HTTP messages. A typical workflow in CF starts by a client interface sending a request to the CF controller through the router. The cloud controller captures the incoming message and initiates a series of message exchanges with other components in the system to deliver on the received


```

1 class Test
2   def _aspect_saved_Test_test_method
3     puts "Hello World!"
4   end
5   def test_method *args, &block_for_method
6     # advice chaining
7     puts "Pre-Aspect Execution."
8     _aspect_saved_Test_test_method
9     puts "Post-Aspect Execution."
10  end
11 end

```

Algorithm 2: The re-written Test class after applying the aspect from Algorithm 1.

```

1 Aspect.new
2   :around,
3   :calls_to => /(send|receive).data/,
4   :type_and_descendants =>
5     [/(NATS|EventMachine)::(.*)/,
6      /(NATS|EventMachine)::(.*)::(.*)/],
7   :method_options[:public] do |jpt, obj,
8     *args|
9     # analyzing captured NATS messages
10  end

```

Algorithm 3: The aquarium aspect to capture NATS messages in CloudFoundry.

and after the target method of the aspect. Algorithm 2 shows the modifications Aquarium makes to the body of the Test class in order to include the advice.

The NATS client used in CloudFoundry components is developed on top of the EventMachine (eve,) library that implements a reactive pattern for asynchronous communications with the NATS server. When exchanging messages with the NATS server, the client calls the send method from EventMachine which then calls an internal C-library to dispatch the message to the server. When receiving messages from the server, the NATS client extends the NATS template from EventMachine by implementing the receive method which can then extract and interpret the content of the message received from the NATS server. In order to capture NATS messages, we developed an aspect that would mine every CF component's code for the given methods and weave our profiling code into it. The code to capture NATS messages is shown in Algorithm 3. Similarly for the HTTP REST Client, mining its code revealed that each REST call is done through calling the request method in the library. This method receives the endpoint URL for the REST call as well as the parameters to be included, makes the invocation to the endpoint, and blocks until a response is received.

The process of instrumenting CF components involves having aspects added to the execution entry

point of every component in CF. Starting the component engages Aquarium which searches the component code to find the matching pointcuts and inject the advice from the aspect.

3.2 Analyzing CF Message Exchanges

Once the aspects are developed and added to every component, captured messages are collected and analyzed to extract their functional and temporal correlations. The advice code for all the aspects involves a short code snippet that dispatches collected message information to a centralized analysis server. Figure 2 shows the set of tasks done by the analysis server. The tasks can be categorized into two high level categories: *i) Message Pattern Analysis and Correlation* and *ii) Message Sequence Analysis*.

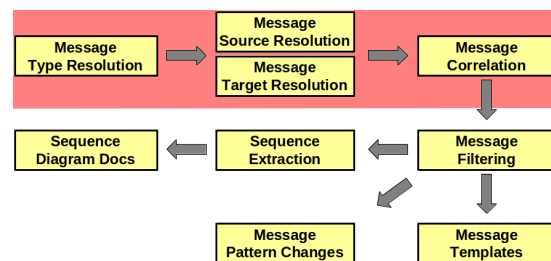


Figure 2: The overall architecture for the analysis server.

3.2.1 Message Patterns and Correlations

As shown at the top of Figure 2, message pattern analysis and correlation involves resolving message types as well as detecting the source and the target component for each message.

For NATS messages, this is done by analyzing subscriptions and publications to NATS channels for every CF component. Components in CloudFoundry announce their registrations to a channel by sending a subscription message through the NATS send method. The analysis server receives these subscription messages and stores a map of all the channels with their subscribers. At a later point in time, once a publish message is received by the analysis server, it searches through all the channels in its directory and correlates the component sending the message to the components previously subscribed to the channel.

HTTP communications are done by components targeting the REST API endpoints of other components. The analysis server maintains a list of the APIs it is aware of at any given time, which it extracts from the requests it receives as they come in^{1,2}. Upon an

¹Cloud controller's target API is <http://api.vcap.me>

²UAA's target API is <http://uaa.vcap.me>.

HTTP request call, the analysis server identifies the endpoint where the HTTP message is directed to and maps the endpoint to its corresponding component.

3.2.2 Message Sequences

As mentioned earlier, a workflow in CloudFoundry starts by a client interface sending a message to the cloud controller. In order to be able to capture message exchange sequences we employed a *snapshooting* technique as follows: we instrumented the command line interface (CLI) bundles embedded in CloudFoundry in such a way that it would notify the analysis server at the beginning and end of any command execution. When the message arrives at the server, the server marks the start of a new workflow execution and records all message exchanges and their temporal order to the point it receives a termination command from the CLI. Upon receiving a termination command all the captured messages are assigned to the latest executed CF command. Generating the message sequence however, requires two considerations:

1. Not all messages captured during the snapshotting process are dispatched in response to the executed command. To accurately capture message sequences, the analysis server employs two strategies to identify and dismiss irrelevant messages: *i)* CloudFoundry components may dispatch heartbeat messages or registering/unregistering messages to some pre-defined NATS channels irrespective of the command being executed³. The analysis server ignores messages published to these channels during an snapshotting process. *ii)* Another strategy in reducing noise comes as a consequence of a prolonged monitoring process of message exchanges. Upon collecting a long enough trace of exchanged messages, the analysis server goes through all message snapshots and assigns an occurrence frequency rate to each message in a snapshot. Messages whose occurrence frequencies fall below a given threshold can be eliminated from the generated sequence.

2. CloudFoundry allows for more than one CLI to dispatch messages to the cloud controller. However distinguishing messages dispatched by different CLIs requires detailed tracing of data flows which are not currently implemented into our profiling tool and analysis server. In order to avoid interference from several CLIs we run our CF deployment and the CLI in a completely controlled environment where only one instance of the CLI is allowed to dispatch messages to the CF deployment.

³e.g., `dea.heartbeat` is a channel used by DEA to notify the Health Manager of their well being.

Figure 3 shows an example of the message sequence captured by the analysis server. As shown in the figure, the sequence starts by the *vmc* CLI (the embedded CLI for CF v1.0) sending a message to the cloud controller which then triggers a sequence of message exchanges between CF components before returning a response to the CLI. The generated sequence diagram has the message types color coded, with the HTTP messages shown as blue (darker colour in grayscale) arrows and NATS messages shown as green (lighter colour in grayscale) arrows. For HTTP messages, labels above the arrows show the HTTP request method and the end point the message is directed to. For NATS messages the label shows the name of the channel to which the message is published.

We code-named the generated documentations as BlueDocs. The detailed list of all captured message sequences for all commands both in CF v1.0 and CF v2.0 can be found under our CloudFoundry BlueDocs GitHub repository (`cfb, a`).

4 EVALUATION

For the purpose of our evaluations, we took two strategies: *i)* tracking evolution from CF v1.0 to CF v2.0 by analyzing changes in message exchange patterns, and *ii)* sharing our results with the community of CF developers and surveying them to assess the benefits of our generated documentation.

4.1 Comparing CF v1.0 and v2.0

In our first evaluation, we provided comparison of message exchange patterns across different versions of CF. In Section 2, we mentioned that despite architectural changes from CF v1.0 to CF v2.0 the methods of synchronous and asynchronous communication stayed the same. For each version of CF, we generated documentation on message exchange templates including the communication channel names and the message contents. We converted the generated documents into sorted comparable strings and used the minimum edit distance algorithm (Atallah and Fox, 1998) to capture differences between the two message templates. We then compared the generated results with the message templates we captured through our prolonged tracing of message exchanges and updated the comparison results. For the NATS messages, we detected 24 different communication channels in CF v1.0. Out of these channels, two had their names changed from CF v1.0 to v2.0, one channel was removed, and five new channels were added. Also for

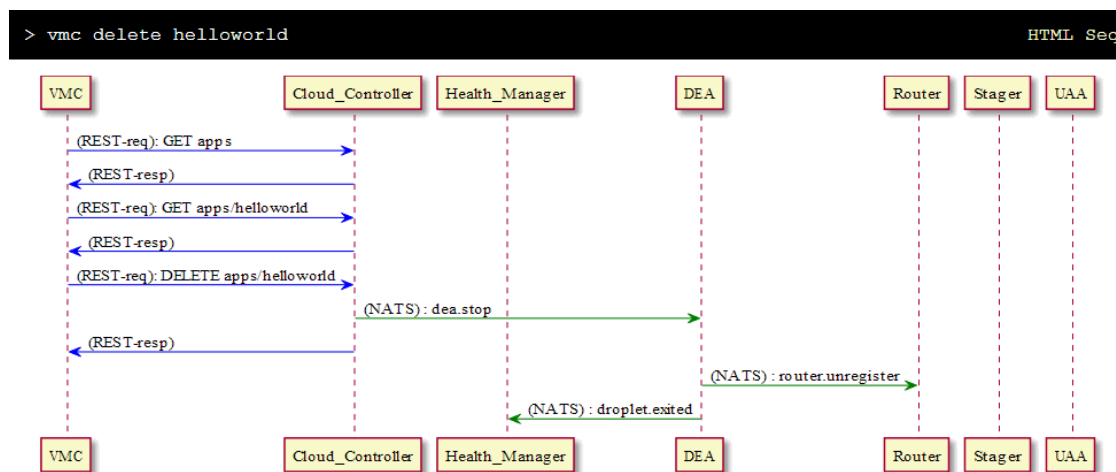


Figure 3: The sequence of exchanged messages for `vmc delete` with blue arrows showing HTTP messages and green arrows showing NATS messages.

all message templates captured, we discovered 222 key-value pairs in total out of which 28 keys were removed from v1.0 to v2.0, 12 were added, and 10 had their types changed. Details are available on the BlueDocs website (cfb, b).

4.2 Surveying the Developer Community

For the second evaluation, we presented the results of our instrumentation and analysis to the developer community for CloudFoundry. We asked the community to fill out a short survey with the following five questions:

1. Have you ever felt the need for documentation on internals of CloudFoundry? If yes, how do you find this documentation?
2. Do you think knowing details of CloudFoundry components, message types, and message sequences helps for the type of work you do with CloudFoundry?
3. Do you find the BlueDocs on message exchanges in CloudFoundry helpful?
4. What do you find useful in the auto-generated BlueDocs documentation for CloudFoundry?
5. What additions or modifications do you like to see in the BlueDocs CloudFoundry documentation?

We received 12 responses from the CloudFoundry developers, 6 from within IBM and 6 from the open source community. All respondents described themselves as developers or system architects working on the internals of CloudFoundry.

When asked about their needs to have documentation on the internals of CloudFoundry, all 12 respondents replied with a *yes*. Also, out of all who took the survey, all except for one thought that such documentation on the internals of CloudFoundry would be helpful for the type of work they were doing.

We then asked the CF developers to investigate the generated BlueDocs documentation and tell us if they find it useful. The survey showed that 9 out of the 12 participants found the generated documentation helpful. When asked about what they found interesting in the generated BlueDocs, the developers made interesting statements like the followings: “*it might allow for auto-generated “diffs” of the documentation between versions. I don’t trust that the APIs of CF will be stable - the core team doesn’t seem to have API stability in the heart & soul. So it will be important for us to identify the changes in the internal APIs.*”. We also received comments that pinpointed problems such as: “*I would rather the message content be formalized as classes. The interactions are somewhat interesting. It doesn’t guarantee that if someone is posting, that there is in fact a listener who cares*”.

The developers continued to make interesting insights and suggestions as a response to our last question. The following suggestions were made by our respondents: “*Correlate/integrate BlueDocs with existing documentation [on CloudFoundry].*” or “*I’m looking for flow diagrams, description of each function, and how each module idempotently operates for specific application lifecycle functions (e.g., push app, start app, delete app, create service, bind service, identify unresponsive app, etc)*”.

5 FUTURE WORK AND CONCLUSIONS

In this paper we discussed our work developing a framework that would allow for software analysis, documentation generation, testing, and debugging, particularly targeted towards the Ruby-based open source cloud platform: CloudFoundry. The aim of the work is to enable developers to better understand and analyze patterns of message exchange across components in CloudFoundry. Our early analysis of the results showed significant interest from the open source community in having this type of analysis in place. We are extending the framework to enable message tacking, data flow analysis, resiliency testing, and increased automation in order to improve the accuracy of collected data and make it more readily available to the open source developer community.

Throughout the development process of our analysis framework, we encountered several challenges that we had to resolve in order to make the framework functional. The first challenge is inherent to AOP. For our type of instrumentation, defined pointcuts were tightly coupled to the signature of the target methods of interest. This is restrictive in that our aspects code are only good for as long as the methods in the target libraries preserve their signature. Any change in the signature of the methods of interest would result in unmatched pointcuts. A more generic approach could search for all functions of a library establishing a network connection and then capture exchanged messages. A second challenge was with respect to injecting the profiling code into every CF component's code. Ruby, as a scripting interpreter-based language, does all the loading and linking at runtime. For the profiling code to capture and instrument the target methods in a Ruby program, it should be added to the component's code after the library of interest is loaded. We are developing a Domain Specific Language (DSL) in Ruby that could be utilized for automatic runtime injection of aspects to the code while verify if a given library is already loaded.

For the future work, we intend to focus on the following: (i) *Software Resiliency*: We believe our developed framework can help with software resiliency through interrupting, corrupting, or modifying message exchanges. In the current implementation, the analysis server makes no interferences to the content, order, or pattern of message exchanges. However, to test resiliency, the analysis server can have a more active role by allowing messages to be dropped, or by modifying message content, and monitoring how the change in the content or pattern of messages affects the overall behaviour of the system. (ii) *Testing & De-*

bugging: One major issue with debugging distributed systems is that often times the source of a problem is not in close proximity of where the failure is observed. When debugging, the long history of information for message exchanges allows to see for each component fan-in and fan-out of message exchanges to track a message back to the source of a discrepancy. Our strategy for testing and debugging relies on collecting a long enough history of messages exchanged and testing the newly arriving messages against the expected pattern of a given workflow.

REFERENCES

- OpenStack - Online: <http://www.openstack.org/>.
- CloudStack - Online: <http://cloudstack.apache.org/>.
- CloudFoundry - Online: <http://www.cloudfoundry.com/>.
- OpenShift - Online: <https://www.openshift.com>.
- Baidu Corp. - Online: <http://baidu.com>.
- ARM - Online: <http://www.opengroup.org/tech/management/arm/>.
- NATS library - Online: <https://github.com/derekkollison/nats/>.
- Aquarium: Aspect-Oriented Programming for Ruby - Online: <http://aquarium.rubyforge.org/>.
- EventMachine - Online: <https://github.com/eventmachine/eventmachine>.
- CloudFoundry BlueDocs - Online: <https://github.com/nkaviani/cloudfoundry-bluedocs/>.
- CloudFoundry BlueDocs version comparison - Online: <http://rawgit.com/nkaviani/cloudfoundry-bluedocs/master/cf-v2/docs/output.html>.
- (2012). In *Openness is Winning in the Cloud* - Online: <https://www.linux.com/news/featured-blogs/200-libby-clark/577866-marten-mickos-openness-is-winning-in-the-cloud>.
- (2012). IBM Announces Platinum Sponsorship of the New OpenStack Foundation - Online: <http://www.openstack.org/blog/2012/04/openstack-foundation-update/>.
- (2013). The role of open source in cloud infrastructure - Online: <http://www.informationweek.in/informationweek/news-analysis/176401/role-source-cloud-infrastructure>.
- (2013). IBM and Pivotal to Accelerate Open Cloud Innovation with Cloud Foundry - Online: <http://www-03.ibm.com/press/us/en/pressrelease/41569.wss>.
- (2013). Baidu report on CloudFoundry - Online: <http://www.slideshare.net/wattersjames/baidu-cloudfoundry-english-24626493>.
- Aguilera, M. K. and et al. (2003). Performance debugging for distributed systems of black boxes. *SIGOPS Oper. Syst. Rev.*, 37(5):74–89.
- Al Abed, W. and Kienzle, J. (2011). Aspect-Oriented Modelling for Distributed Systems. volume 6981, pages 123–137.

- Anandkumar, A., Bisdikian, C., and Agrawal, D. (2008). Tracking in a spaghetti bowl: monitoring transactions using footprints. In *Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '08, pages 133–144, New York, NY, USA. ACM.
- Atallah, M. J. and Fox, S., editors (1998). *Algorithms and Theory of Computation Handbook*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.
- Barham, P., Isaacs, R., Mortier, R., and Narayanan, D. (2003). Magpie: Online Modelling and Performance-aware Systems. In *HotOS*, pages 85–90.
- Baset, S. A., Tang, C., Tak, B., and Wang, L. (2013). Dissecting Open Source Cloud Evolution: An OpenStack Case Study. In *HotCloud*, pages 333–340. IEEE.
- Beschastnikh, I., Brun, Y., Schneider, S., Sloan, M., and Ernst, M. D. (2011). Leveraging existing instrumentation to automatically infer invariant-constrained models. In *SIGSOFT FSE*, pages 267–277. ACM.
- Kaviani, N., Wohlstadter, E., and Lea, R. (2012). MANTICORE: A framework for partitioning software services for hybrid cloud. In *CloudCom*, pages 333–340. IEEE.
- Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W. G. (2001). An Overview of AspectJ. In *Proceedings of the 15th European Conference on Object-Oriented Programming*, pages 327–353.
- Wohlstadter, E. and Devanbu, P. (2006). Transactions on Aspect-Oriented Software Development II. pages 69–100. Springer-Verlag, Berlin, Heidelberg.

Towards Cross-layer Monitoring of Cloud Workflows

Eric Kübler and Mirjam Minor

*Institute of Informatics, Goethe University, Robert-Mayer-Str.10, Frankfurt am Main, Germany
{ekuebler, minor}@informatik.uni-frankfurt.de*

Keywords: Cloud Applications Performance and Monitoring, Workflow, OpenShift.

Abstract: Prospective cloud management requires sophisticated monitoring capabilities. In this paper, we introduce a novel monitoring framework for cloud-based workflow systems called cWorkload. cWorkload integrates monitoring information from different layers of the cloud architecture. The paper puts its focus on the two-layer monitoring regarding the workflow layer and the PaaS layer. We present the layered monitoring architecture, an implementation of the two-layer cross-monitoring part, and an experimental evaluation with sample workflow data. Further, we discuss related work on cloud monitoring divided into one-layer, multi-layer, and cross-layer approaches. Our plans for future work on extending the implementation by further layers towards a cross-layer, prospective monitoring for prospective cloud management are described.

1 INTRODUCTION

Cloud management (CM) aims at an optimal resource and capacity planning. *Cloud monitoring* becomes essential to predict and keep track of the evolution of all the parameters involved in the process of assuring the *Quality of Service* (QoS) (Aceto et al., 2013). Monitoring capabilities facilitate cloud service providers to fulfill *service level agreements* (SLAs). Service consumers may use monitoring capabilities to audit whether SLAs have been violated. Today, monitoring services such as Amazon CloudWatch (Amazon, 2014b) provide data on the current state of particular cloud resources. However, they facilitate a rather reactive management of resources. The increasingly complex structure of cloud systems made of several layers requires more complex monitoring systems in future (Aceto et al., 2013).

We identified a research gap for prospective cloud monitoring capabilities. Such approaches that go beyond monitoring capabilities provided by cloud vendors might improve CM significantly. In this paper, we introduce the novel cross-layer monitoring framework cWorkload that integrates process monitoring and cloud monitoring capabilities. cWorkload is part of our process-oriented cloud management model whose fundamental ideas have been published in our previous work (Schulte-Zurhausen and Minor, 2014; Minor and Schulte-Zurhausen, 2014). The cloud management model has been inspired by the multi-tier model for cloud management introduced by Maurer et al. (Maurer et al., 2013), which we have

extended by the business process perspective. An integrated monitoring solution that is aware of the ongoing business processes contributes to a better estimation of approaching workloads. Cloud monitoring information from existing tools is integrated with process monitoring. As a consequence, interventions can be planned in advance and executed in a timely manner. This improvement of the cloud management yields benefits for both, a cloud service provider and a cloud user perspective. The novel monitoring approach might reduce the amount of overprovisioned resources extremely, which is required to maintain compliance with SLAs.

The Workflow Management Coalition (Workflow Management Coalition, 1999) defines a *workflow* as “the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”. A *task* also called activity is defined as “a description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources(s) to support process execution” (Workflow Management Coalition, 1999). Several instances of the same workflow can be executed at the same time. A *cloud workflow* is a workflow that is executed within a cloud environment (or within several cloud environments). For instance, a video surveillance process comprising of several analysis steps for recorded video sequences, such as identi-

ifying image changes that may correspond to humans moving within the observed area, provides a sample scenario for a cloud workflow. In this paper, we make use of the workflow notion to define workload as follows: A *workload* is a task (of a workflow instance) with its input data, its number of users, and a task type (CPU intensive, network intensive, memory intensive, storage intensive). While the workload of an ongoing task is well-specified the future workloads might be known only approximately, i.e. the workloads are still underspecified. Underspecified workloads are refined as soon as all properties of their according tasks are specified, such as input data and number of users. The overall workload of a cloud system at a particular execution time is then characterized by the set of current workloads (from different tasks being executed). The remainder of this paper is organized as follows. In Section 2, we present related work. Section 3 summarizes our cloud management and cross-layer monitoring approach. Section 4 contains our hypotheses, the experimental setup, and results of our experiments. We draw a conclusion in Section 5 and point out future work.

2 RELATED WORK

In order to compare cWorkload and other cloud monitoring platforms, we classify the related work into three types: single-layer monitoring, multi-layer monitoring, and cross-layer monitoring. A single-layer monitoring tool monitors only a single layer of the cloud architecture. This could be, for example, the application layer or the IaaS layer. A multi-layer monitoring platform is able to obtain and manage monitoring information from different layers at the same time. However, the collected information from each layer is not related to other layers. Thus, multi-layer platforms are per se not capable to observe workloads across layers in an integrated manner. A cross-layer monitoring platform receives monitoring information from several layers and is able to track the impact of a workload from layer to layer, for instance, to determine the resource utilization by a cloud workflow at lower layers. In the following, we present a selection of cloud monitoring platforms. The selection and grouping of work is an extension of the list of monitoring platforms discussed by Aceto et al. (Aceto et al., 2013).

Obviously, the related work on cross-layer monitoring forms the closest affinities to our approach. *AzureWatch* (Paraleap Technologies, 2014) in combination with other third-party monitoring services is capable of monitoring the application, PaaS and IaaS

layer and of aggregating the information. Monitoring values from leading indicators for scaling, such as queue depths or rate of change in demand, are combined with values from trailing indicators, such as CPU utilization, requests per second, or bandwidth. Thus, *AzureWatch* is a cross-layer platform. However, *AzureWatch* is restricted to technical solutions based on Windows Azure. The workflow paradigm is not addressed. *vRealize Hyperic* (vmware, 2014) is a cross-layer monitoring platform from VMWare. It monitors several cloud layers and is able to point out resource utilization peaks across the layers. This is called metric drill down. In contrast to our work, *vRealize Hyperic* does not consider a workflow layer.

Multi-layer monitoring approaches are also of interest for our work since they collect monitoring information from different layers. Some of them visualize the acquired monitoring values in common dashboards. *CloudWatch* (Amazon, 2014b) uses metrics from the Amazon Web Services (AWS), which are distributed on the PaaS and IaaS layer. *CloudWatch* is also capable of obtaining and evaluating monitoring information from other applications. Thus, *CloudWatch* is a multi-layer monitoring platform. However, due to the lack of aggregation of the pieces of information, *CloudWatch* is not a cross-layer monitoring platform. *Monitis* (monitis, 2014) is an application based on an agent paradigm. It is mainly for AWS and, similar to *CloudWatch*, is a multi-layer platform with a common dashboard for monitoring information from different layers. Further multi-layer approaches for PaaS and IaaS are, for instance, *Lattice* (Palmieri et al., 2012), *Gmone* (Montes et al., 2013) or the monitoring component of *Aneka* (Manjrasoft, 2014). *Nimsoft Monitoring Solution* (ca technologies, 2014) is a multi-layer monitoring platform primarily for the hardware and IaaS layer but it is also able to obtain pieces of information from other monitoring platforms. However, it does not track events across the layers. A similar approach for multi-layer IaaS and hardware monitoring is *GroundWork* (GroundWork, 2014) which uses the tool Nagios (Nagios, 2014) for the hardware layer. The *CLAMS* framework (Alhamazani et al., 2014) is a multi-layer approach with a special focus on integrating monitoring information from cloud environments of different vendors. Please note that the authors use an alternative notion of cross-layer monitoring regarding multi-clouds. Maurer et al. (Maurer et al., 2013) present an approach to manage a cloud configuration at different layers. They divide the resource allocation into three levels for scaling called escalation levels. Low-level metrics from the cloud infrastructure, such as *free_disk* and *packets_sent*, are mapped to high-level

monitoring parameters with the aim to fulfill the Service Level Agreements (SLAs) by automated scaling. We consider this a multi-layer approach since the information from a lower layer is propagated towards higher layers. A cross-layer monitoring would require receiving monitoring information from different layers.

Single-layer monitoring approaches are slightly related to our work. Monitoring one layer is a prerequisite for multi-layer and cross-layer monitoring. *OpenShift*, for instance, can be monitored in a single-layer manner by a monitoring component for each particular PaaS container called monitoring cartridges (Pousty and Miller, 2014). For sample work on monitoring at a single IaaS layer, we refer to the literature (OpenNebula, 2014; rackspace, 2014; LogicMonitor, 2014; Kung et al., 2011; Alcaraz Calero and Gutierrez Aguado, 2014).

3 MONITORING ARCHITECTURE

The cloud management problem could be mapped into the MAPE cycle (Monitoring - Analysis - Planning - Execution) (I.B.M. Corporation, 2006). The first step is to monitor the resources. The analysis step is to detect an event. This could be the threatening violation of an SLA, for instance. Another example for an event is the occurrence of a high workload that provides an indicator for scaling resources in order to decrease the required execution time. Step three is the search for a solution in the planning step. A solution is a reconfiguration of the cloud configuration. This might include changes in the distribution of workloads for a better fit of the resources to the requirements. This could be achieved by using configuration activities across the layers while activities at higher levels are preferable. In addition, different problem solving strategies could be considered, such as meta-heuristics (Beloglazov et al., 2012), genetic algorithms (Hu et al., 2010), or case based reasoning (Richter and Weber, 2013).

In the following, we will briefly summarize our cloud management approach that has been published in our previous work (Minor and Schulte-Zurhausen, 2014). Based on this, we will introduce the novel monitoring architecture. A multi-tier model separates physical from virtual resources in different layers. It allows cascading configuration activities from layer to layer. The model of our monitoring architecture is inspired by the cloud management model from Maurer et al. (Maurer et al., 2013) which consists of three layers for hierarchical configuration ac-

tivities. In our previous work (Minor and Schulte-Zurhausen, 2014), we have introduced a workflow tier on top of the three tiers from Maurer et al.’s model in order to achieve a process-oriented perspective (see left hand side of Figure 1). The physical machine tier at the bottom is manipulated by configuration activities like add and remove compute nodes. The virtual machine tier allows activities like to increase or decrease incoming and outgoing bandwidth of a virtual machine (VM), its memory, its CPU share, or to add or remove allocated storage by x%. Further, VMs can be migrated to a different physical machine or moved to and from other clouds in case of outsourcing/insourcing capabilities. The application tier is dedicated to management activities for individual applications. The same set of activities as for VMs can be applied but with an application-specific scope. This tier can also be a PaaS platform while the VM tier can be an IaaS platform. Obviously, the migration and insourcing/outsourcing activities refer to the placement of applications on VMs at the application tier. The workflow tier addresses the placement of workflow tasks on VMs. Tasks can be migrated to another VM or tailored, i.e. split into replicated tasks with a portion of the input data each.

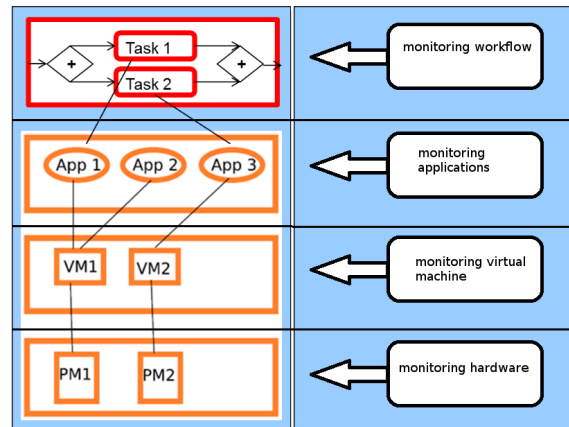


Figure 1: Process-oriented model for cloud management extending (Maurer et al., 2013).

cWorkload is a novel part of the cloud management architecture for cross-layer monitoring (see right hand side of Figure 1). In case of the workflow layer, it monitors the workflow state of instances and tasks. States reported frequently in the literature are for example “running”, “active” or “suspended”. We aim at using this information for a prediction of workloads as a core feature of cWorkloads. The monitoring results can be used to improve the management capabilities from a reactive towards a prospective management. To obtain the workload of an instance, we monitor the input and output data of the tasks. With

the information on the workloads we will be able to estimate the effort. This estimation will serve in our model as a means to forecast the need for scaling. We are planning to run simulations to collect information about the efforts of domain specific workloads and build an empiric knowledge repository about typical efforts.

At the PaaS layer, we aim to monitor the resources that are offered by the run-time environment. Generally, there are different tools and ways to accomplish this task depending on the used PaaS platform. In our implementation, we use OpenShift (Pousty and Miller, 2014) as a cloud environment. The cloud workflow is executed by a workflow engine running within OpenShift. Each workflow task is performed by one web service (Singh and Huhns, 2005). The workflow engine and the web services are operated on different containers. Monitoring information from all containers that are involved is gathered by the linux tool top (Unix Top, 2014).

Similar to the PaaS layer, the IaaS layer offers different tools to monitor the health of the VMs. In future work, we will use Eucalyptus (Eucalyptus Systems, 2014) or the Amazon web services (Amazon, 2014a), probably with CloudWatch (Amazon, 2014b) as a monitoring tool.

At the hardware layer, there are monitoring metrics available such as for the cpu utilization, the energy consumption and the temperature.

The long-term goal of cWorkload is to reduce the overprovisioning of resources. To achieve this goal, we will integrate the monitoring information by a smart handling of resources depending on workload forecasts. We will combine the information from different monitoring layers to obtain a better understanding of the state of our cloud.

In the remainder of this paper, we will present on the integration of the workflow and the PaaS layer. One essential part of this concept is the task placement. We define a *task placement* as the assignment of automated tasks to VMs and furthermore the assignment of VMs to physical machines (PMs). Figure 2 illustrates an example for a task placement. The placement includes two PMs and three VMs. The figure illustrates which VM is executed on which PM, for example “VM3” is running on “PM2”. This schema is repeated for the runtime environments of the PaaS layer and for the web services at the workflow layer. For instance, “s3” is executed on the “runtime3”, which is executed on “VM2”. On the right hand side, an example workflow is depicted that uses “s1” and “s5”.

The integration of the PaaS and the workflow layer is performed by cross-linking the metrics at the PaaS

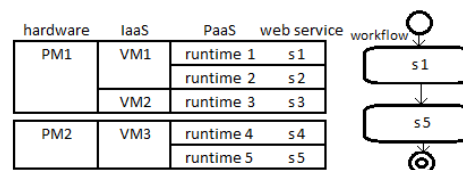


Figure 2: Sample task placement for a workflow.

layer with the monitored task states at the workflow layer. Each task in the state “running” is assigned to a measuring point for the cloud resource that has been assigned for execution. The measured values from both layers are aggregated into a multi-dimensional representation form based on time stamps from both, the workflow execution log and from the time of measuring recorded at the PaaS level. The aggregated information can be interpreted by an automated cloud management mechanism or visualized for a human being in a multi-dimensional presentation. For instance, in our preliminary implementation within OpenShift, each PaaS container comprises of exactly one web service that is assigned to a PaaS container via its URL. However, the same web service can be used by multiple workflow instances containing the same task that is performed by the web service. As a consequence, the measured metrics at the PaaS layer have to be partitioned in order to determine the portion of the measured values for each task. For example, if “s1” that is deployed on “runtime1” is used by three workflow instances at the same time and if the CPU utilization of “runtime1” is 100% we should not simply conclude that each instance uses 33% of the CPU because the CPU utilization for each instance depends on the input data. At the moment, we consider the measured values at the PaaS layer as an amalgam if multiple workflow task instances are running on the same PaaS container. A partition function is required to approximate the share of effort for each instance. We will address this in our future work in order to aggregate the values from the PaaS layer with the workflow layer more precisely than in our simplified experiments (see below).

4 EVALUATION

The cWorkload framework is implemented in a pilot system with a focus on the two-layer monitoring of the workflow and the PaaS layer. Integrating the monitoring capabilities of the workflow and the PaaS layer is an important step towards a cross-layer monitoring. A prerequisite for this is to check whether monitoring values from both layers can be aggregated in principle. The feasibility of the

two-layer approach has been investigated by running some experiments with sample workflows on the pilot system. Preliminary evaluation results have been achieved guided by the following three hypotheses:

H1: The Utilization of Cloud Resources Can Be Aligned with a Workflow Instance.

The core of the hypothesis is that it is possible to identify the PaaS resources used by a particular workflow task that has been executed. This includes recognizing the PaaS container that has been used, the utilization of the resources of the container and the duration of using the container. The hypothesis addresses a simple but critical point. Only with the capability to monitor the resources required by a task, a deep integration of workflow and cloud management can be achieved. Obviously, the PaaS resources can be monitored but for a workflow instance it is not straightforward to identify the resources (and shares of resources) that have been used by a particular workflow task during execution in a cloud environment.

H2: The More Workflow Instances Are Running on a PaaS the Higher Is the Demand for Resource Scaling.

Each workflow instance within a PaaS should be independent from each other. In order to avoid interferences between instances the resources might be scaled. The need for scaling might increase with the number of ongoing workflow instances. We assume that the sharing of resources actually occurs and that scaling is performed by a heuristically method, for instance by rules observing the number of open I/O connections to a PaaS container. The confirmation of the hypothesis is not obvious because if tasks are running on different physical machines they might not have any impact on each other.

H3: The Higher the Workloads from Ongoing Workflow Tasks on a PaaS the Higher Is the Demand for Scaling.

In addition to the number of running workflow instances, the type of workload that is processed within an instance has an impact on the demand for scaling. Again, we assume that resources are shared heuristically on the PaaS. If the workload is distributed advantageously it is possible to run several workflow instances without any interference between them. On the other hand, disadvantageous distributions of workload might lead to obstructions. To observe such effects, knowledge about the type of workloads is required.

We have designed two different, artificial work-

flows for the experiments. A sample modelled in BPMN (Grosskopf et al., 2009) is depicted in Figure 3. The trigger for starting a workflow instance is a message from the user as indicated by the envelope symbol. Two sample web services are included. The fibo web service (“WS: Fibo”) calculates the Fibonacci number for a given number. The prim web service (“WS: Prim”) calculates all prime numbers from 2 to n for a given n . The web services are executed in parallel as indicated by the plus symbol. The workflows have been modelled using the workflow designer *Intalio—BPM* (Intalio, 2014). Our work considers the task types of the workloads such as CPU intensive, storage intensive, memory intensive or network intensive tasks. The prim and fibo web services are clearly of the type CPU intensive since they mainly require CPU as a resource, do not require any storage, and use only a tiny quantity of memory and network capacities. The web services are deployed on the *OpenShift online* platform (Pousty and Miller, 2014) that provides a public cloud solution of a PaaS. Each web service is running on an OpenShift small gear, i.e. a small-size container of the runtime environment providing 512MB RAM, 1GB hard disc and a single CPU with 2.5GHz. The prim web service is deployed on gear no. 1 and the fibo web service on gear no. 2. We use a WildFly 8 application server (redhat, 2014b) as a cartridge, i.e. as a pre-configured application on top of a gear. The system *Intalio—BPM 6.5* serves as a workflow engine running on a 4-core CPU with 3.4GHz and 16GB RAM. The instances are started by hand. The CPU utilization for each gear is logged via the Linux tool *top* (Unix Top, 2014). Our cWorkload framework records the terminal output at regular intervals. In addition, Intalio—BPM logs the progress of the workflows. We run the simulation with several workloads and compare the results of the different runs. In case of the prim web service, the input data is a given number, for example 10. The result for this example is the set of prime numbers $\{2, 3, 5, 7\}$. The input data for the fibo web service is also a number. For example, the result is 55 for the number 10. The higher the input number the higher is the workload. Consequently, this requires a higher computational effort.

Within an initial test phase, we couldn’t observe any interference between the web services. Most probably, this was due to the fact that two gears on different Amazon EC2 instances had been selected automatically by the load balancer used by OpenShift online. Thus, we will discuss the results for the particular web services separately. In order to investigate hypothesis H1, cWorkload monitors the CPU utilization of the gears for new workflow instances that

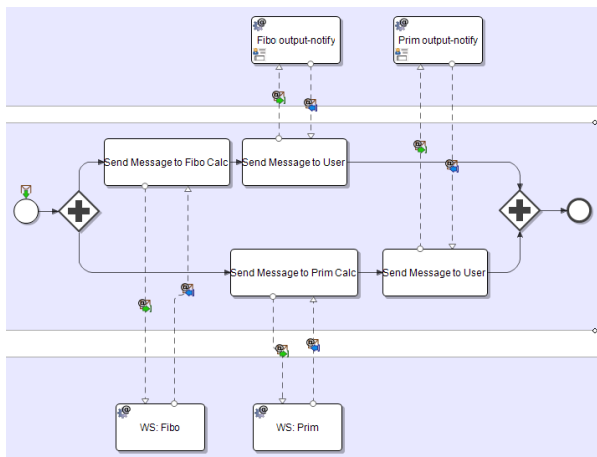


Figure 3: Workflow with a parallel call of web services.

have been started at irregular frequencies. Figure 4 and Figure 5 depict two sample sequences of values measured for the CPU utilization of gear 1 and gear 2. The sample input data was 45 for the fibo web service and 100,000 for the prim web service. There are some recognizable peaks within the time series. We have highlighted the areas where a web service is supposed to run according to the logs of the workflow engine. It can be observed that the log information matches the recognizable peaks. We conclude from the investigated samples that the utilization of cloud resources measured by the Top tool can be aligned with the CPU utilization of the gears consumed by the particular web service tasks. However, there are further single peaks in the time series. We assume that they are artifacts originating either from the Top tool, from the WildFly cartridges or from competing gears running on the EC2 instance. Hypothesis H1 has been confirmed by the measured samples.

For hypothesis H2, we have started a variable number of workflow instances simultaneously. The logs of the workflow engine have been analyzed to determine the throughput time of the web services, i.e. the duration of executing the workflow task. We have repeated this test for 5 samples with input 100,000 for the prim web service and 43 for the fibo web service. The average values are depicted in Figure 6 and Figure 7. The results show that with an increasing number of instances calling the same web service in parallel the throughput time increases significantly. At this point, our preliminary results are not yet a proof but a promising indicator that our hypothesis 2 is valid. In order to investigate hypothesis H3, we have logged the number of successfully executed requests for the fibo web services. A request is considered successful if the service has returned the result before receiving a time out. For instance, the Intalio workflow engine

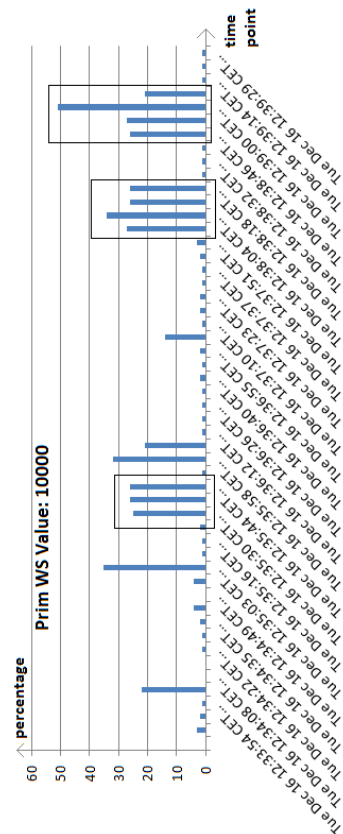


Figure 4: CPU utilization of prim.

throws such time outs after 60 seconds. We created different workloads from the same web service task by varying the input data. We have chosen an input value between 30 and 50. Each sample was repeated between 3 and 20 times. For input values of 43 and below, all executions have been successful. For input values of 48 and above, the execution was not successful. For values between, the success of the execution depends on the number of instances and on the system state. According to our experience, the duration of execution increases with the input value. Having executed 5 instances with input value 43 successfully makes it promising to run 5 instances with value 42 successfully as well. A systematic experimental proof of this experience as well as the development of a quantification function for workloads will be part of our future work. The preliminary results provide a first hint that the demand for scaling resources can be predicted approximately by means of the input values that should be calculated and, thus, by the workloads to be expected. Hypothesis H3 is not yet finally confirmed. Rather, the experiments play the role of a plausibility check. Based on the promising results, a partition function can be elaborated and hypothesis H1 can be investigated for multiple workflow in-

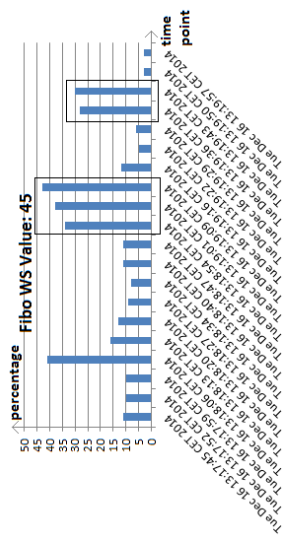


Figure 5: CPU utilization of fibo.

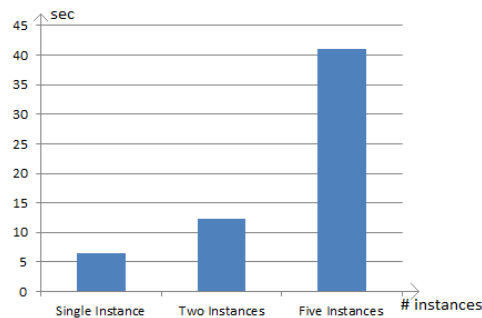


Figure 6: Average execution times for instances of fibo.

stances running in parallel in future.

5 CONCLUSION AND FUTURE WORK

In this paper, we have introduced cWorkload a novel monitoring framework for cross-layer monitoring of cloud workflows. The layer architecture of the cWorkload framework is designed in accordance with the layers of the cloud-based workflow system to be monitored. A workflow execution layer on top of the PaaS cloud system is monitored by means of the monitoring capabilities of the underlying PaaS layer. To achieve this two-layer integration, the resource utilization of the workflow tasks is determined by measuring values from the cloud resources and assigning the measured values according to the particular workloads of the workflow tasks. A preliminary evaluation of cWorkload is based on experiments with artificial workflow samples. The execution of CPU intensive

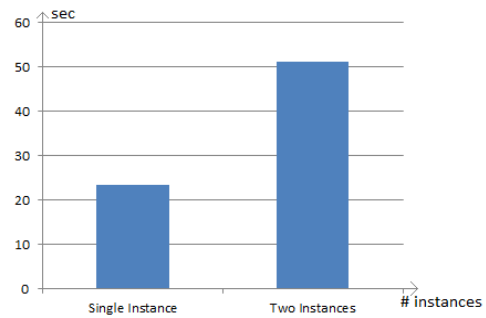


Figure 7: Average execution times for instances of prim.

workflow samples has provided that the CPU utilization of the cloud resources can be aligned with the workflow instances comprising of computationally intensive tasks. Further, we observed that the more workflow instances are running the higher are the utilization values of the according cloud resources. We conclude from the samples that the demand for scaling resources is increased by this. In addition, the higher the workloads from ongoing workflow tasks are the higher is the demand for scaling resources. The benefit of a cross-layer monitoring solution is that it provides deeper insights on the state of the system than single-layer monitoring approaches. In case of our two-layer approach that integrates the workflow layer with the PaaS layer, this results in a better understanding of the placement of workloads on PaaS containers such as OpenShift gears. Consequently, the assignment of workloads to containers can be improved, for instance by grouping tasks of different types on one gear or by selecting containers with an optimal size for a workload. Thus, the overprovisioning of resources might be reduced. The contribution of this paper and especially the preliminary experiments on the two-layer monitoring integrating the workflow and the PaaS layer provide quite promising results. The work is an important step towards the cross-layer monitoring of cloud workflows. However, it raises several novel research questions as follows.

In our future work, we will address mainly two issues. First, the experimental platform will be improved by a different technical environment. The workflow engine Intalio will be replaced by a workflow engine such as JBPM (redhat, 2014a) that facilitates the automated triggering of workflows and provides machine-readable logging information. Both will support us to conduct further experiments with a broader scope and with a deeper, automated analytics. The public cloud PaaS solution OpenShift Online will be substituted by a private installation of OpenShift in order to achieve better control of further layers below the PaaS layer. The two-layer integration will be ex-

tended to a cross-layer monitoring with multiple layers. Second, we will investigate how to predict future demands for scaling from expected workloads. The workflow notion will provide this information either based on the set of tasks to be triggered next or by a clocked simulation of further workflow execution. As a first step, a partition function for measured values on PaaS containers hosting multiple workflow task instances has to be implemented. A comparison study of the prospective solution with non-predictive cloud management approaches will highlight the importance of our approach.

REFERENCES

- Aceto, G., Botta, A., de Donato, W., and Pescap, A. (2013). Cloud monitoring: A survey. *Computer Networks*, 57(9):2093–2115.
- Alcaraz Calero, J. and Gutierrez Aguado, J. (2014). Mon-PaaS: An adaptive monitoring platform as a service for cloud computing infrastructures and services. *IEEE Trans. on Services Computing*, 8(1):65–78.
- Alhamazani, K., Ranjan, R., Mitra, K., Jayaraman, P., Huang, Z., Wang, L., and Rabhi, F. (2014). CLAMS: Cross-layer Multi-cloud Application Monitoring-as-a-Service Framework. In *2014 IEEE Int. Conference on Services Computing (SCC)*, pages 283–290.
- Amazon (2014a). Amazon web services. <http://aws.amazon.com/>, 12-19-14.
- Amazon (2014b). Cloudwatch. <http://aws.amazon.com/cloudwatch/>, 12-18-14.
- Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5).
- ca technologies (2014). Nimsoft. <http://www.ca.com/us/opscenter/ca-unified-infrastructure-management.aspx>, 12-11-14.
- Eucalyptus Systems (2014). <https://www.eucalyptus.com/>, 12-19-14.
- Grosskopf, A., Decker, G., and Weske, M. (2009). *The Process: Business Process Modeling Using BPMN*. Meghan Kiffer Pr.
- GroundWork (2014). <http://www.gwos.com/features/>, 12-19-14.
- Hu, J., Gu, J., Sun, G., and Zhao, T. (2010). A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *2010 Third Int. Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 89–96.
- I.B.M. Corporation (2006). An architectural blueprint for autonomic computing. http://www-03.ibm.com/autonomic/pdfs/AC_Blueprint_White_Paper_V7.pdf, 11-01-14.
- Intalio (2014). <http://www.intalio.com/>, 12-18-14.
- Kung, H. T., Lin, C.-K., and Vlah, D. (2011). CloudSense: continuous fine-grain cloud monitoring with compressive sensing. *USENIX Hot-Cloud*.
- LogicMonitor (2014). <http://www.logicmonitor.com/>, 12-19-14.
- Manjrasoft (2014). Aneka. <http://www.manjrasoft.com/>, 12-18-14.
- Maurer, M., Brandic, I., and Sakellariou, R. (2013). Adaptive resource configuration for cloud infrastructure management. *Future Generation Computer Systems*, 29(2):472–487.
- Minor, M. and Schulte-Zurhausen, E. (2014). Towards process-oriented cloud management with case-based reasoning. In *Proc. ICCBR 2014*, LNCS 8766, pages 303 – 312. Springer.
- monitis (2014). <http://www.monitis.com/>, 12-19-14.
- Montes, J., Sánchez, A., Memishi, B., Pérez, M. S., and Antoniu, G. (2013). Gmone: A complete approach to cloud monitoring. *Future Generation Computer Systems*, 29(8):2026 – 2040.
- Nagios (2014). <http://www.nagios.org/>, 12-19-14.
- OpenNebula (2014). <http://docs.opennebula.org/>, 12-19-14.
- Palmieri, R., di Sanzo, P., Quaglia, F., Romano, P., Peluso, S., and Didona, D. (2012). Integrated monitoring of infrastructures and applications in cloud environments. In *Euro-Par 2011*, pages 45–53. Springer.
- Paraleap Technologies (2014). Azurewatch. <http://www.paraleap.com/azurewatch>, 12-19-14.
- Pousty, S. and Miller, K. (2014). *Getting Started with OpenShift*. "O'Reilly Media, Inc."
- rackspace (2014). Cloudkick. <http://www.rackspace.com/cloud/monitoring/>, 12-15-14.
- redhat (2014a). jbpmp. <http://www.jbpmp.org/>, 11-26-14.
- redhat (2014b). WildFly. <http://www.wildfly.org/>, 12-18-14.
- Richter, M. M. and Weber, R. (2013). *Case-Based Reasoning: A Textbook*. Springer.
- Schulte-Zurhausen, E. and Minor, M. (2014). Task placement in a cloud with case based reasoning. In *CLOSER 2014*, pages 323 – 328, Barcelona, Spain. SciTePress.
- Singh, M. P. and Huhns, M. N. (2005). *Service-oriented computing - semantics, processes, agents*. Wiley.
- Unix Top (2014). <http://www.unixtop.org/>, 12-18-14.
- vmware (2014). <http://www.vmware.com/products/vrealize-hyperic/>, 12-19-14.
- Workflow Management Coalition (1999). Glossary & terminology. 5-23-14.

Automating Resources Discovery for Multiple Data Stores Cloud Applications

Rami Sellami, Michel Vadrine, Sami Bhiri and Bruno Defude

*Computer Science Departement, Institut Mines-Telecom,
Telecom SudParis, CNRS UMR Samovar, Evry, France
firstname.lastname@telecom-sudparis.eu*

Keywords: NoSQL Data Stores, Relational Data Stores, Polyglot Persistence, Manifest based Discovery, ODBAPI.

Abstract: The production of huge amount of data and the emergence of cloud computing have introduced new requirements for data management. Many applications need to interact with several heterogeneous data stores depending on the type of data they have to manage: traditional data types, documents, graph data from social networks, simple key-value data, etc. Interacting with heterogeneous data models via different APIs, multi-data store applications imposes challenging tasks to their developers. Indeed, programmers have to be familiar with different APIs. In addition, developers need to master and deal with the complex processes of cloud discovery, and application deployment and execution. Moreover, the execution of join queries over heterogeneous data models cannot, currently, be achieved in a declarative way as it is used to be with mono-data store application, and therefore requires extra implementation effort. In this paper we propose a declarative approach enabling to lighten the burden of the tedious and non-standard tasks of discovering relevant cloud environment and deploying applications on them while letting developers to simply focus on specifying their storage and computing requirements. A prototype of the proposed solution has been developed and is currently used to implement use cases from the OpenPaaS project.

1 INTRODUCTION

Cloud computing has recently emerged as a new computing paradigm enabling on-demand and scalable provision of resources, platforms and software as services. Cloud computing is often presented at three levels (Baun and et al., 2011): the Infrastructure as a Service (IaaS) giving access to abstracted view on the hardware, the Platform-as-a-Service (PaaS) provides to the developers programming and execution environments, and the Software as a Service (SaaS) enables end users to run cloud software applications.

Due to its elasticity property cloud computing provides interesting execution environments for several emerging applications such as big data management. According to the National Institute of Standards and Technology¹ (NIST), big data is *data which exceed the capacity or capability of current or conventional methods and systems*. It is based on the 3-Vs model where the three Vs refer to volume, velocity and variety properties (McAfee and Brynjolfsson, 2012). Volume means the process of large amounts of information. Velocity signifies the increasing rate at which

data flows. Finally, variety refers to the diversity of data sources. Against this background, the challenges of big data management result from the expansion of the 3Vs properties. In our work we focus mainly on the variety property and more precisely on multi-data store applications in the cloud.

In order to satisfy different storage requirements, cloud applications usually need to access and interact with different relational and NoSQL data stores having heterogeneous APIs. The heterogeneity of the data stores induces several problems when developing, deploying and migrating multi-data store applications. Below, we list the main 4 problems which we are tackling in this paper.

- Pb*₁. Heavy workload on the application developer: Nowadays data stores have different and heterogeneous APIs. Developers of multi-data store applications need to be familiar with all these APIs when coding their applications.
- Pb*₂. No declarative way for executing join queries: Due to the heterogeneity of the data models, there is currently no declarative way to define and execute complex queries over several data stores. That means developers have to cope

¹<http://www.nist.gov/>

themselves with the implementation of such complex queries.

- Pb*₃. Code adaptation: When migrating applications from one cloud environment to another, application developers have to re-adapt the application source code in order to interact with new data stores. Developers have potentially to learn and master new APIs.
- Pb*₄. Tedious and non-standard processes of discovery and deployment: Once an application is developed or migrated, developers have to deploy it into a cloud provider. Discovering the most suitable cloud environment providing the required data stores and deploying the application on it is a tedious and meticulous provider-specific process.

In our work, we are coping with these problems in order to support the developer during a multiple data stores based application life cycle. In a previous work, we proposed ODBAPI (OPEN-PaaS-DataBase API) a streamlined and a unified REST-based API (Sellami et al., 2014) for executing CRUD operations on relational and NoSQL data stores. The highlights of ODBAPI are twofold: (i) decoupling cloud applications from data stores in order to facilitate the migration process, and (ii) easing the developers task by lightening the burden of managing different APIs. In contrast, we present in this paper a declarative approach for discovering appropriate cloud environments and deploying applications on them while letting developers to simply focus on specifying their storage and computing requirements. A prototype of the proposed solution has been developed and is currently used to implement use cases from the OpenPaaS project.

The remainder of the paper is organized as follows. In section 2, we introduce the context and present a motivating example. In section 3, we give an overview of our approach and we detail in sections 4 the discovery step. In section 5, we present the implementation and validation of our solution. In Section 6, we discuss the related work. Section 7 concludes our paper and outlines directions of future work.

2 MOTIVATION

Our work is done in the context of the OpenPaaS project ² aiming at developing a PaaS technology dedicated to enterprise collaborative applications de-

²This work has been partly funded by the French FSN OpenPaaS grant (<https://research.linagora.com/display/openpaas/Open+PAAS+Overview>)

ployed on hybrid clouds (private/public). It is a platform that allows to design and deploy applications based on proven technologies provided by partners such as collaborative messaging system, integration and workflow technologies that will be extended in order to address Cloud Computing requirements. Target applications of OpenPaaS are applications that use multiple data stores that corresponds to what is popularly referred to as the polyglot persistence. For instance, an application can interact with three heterogeneous data stores at the same time: a *relational data store*, a document data store that is *CouchDB*, and a key value data store which is *Riak*. But, this example exposes some limits. Linking an application with multiple data stores is very complex due to the different APIs, data models, query languages and consistency models. If the application needs to query data coming from different data sources (e.g joining data, aggregating data, etc.), it can not do it declaratively unless some kinds of mediation have been done before. Finally, the different data stores may use different transaction and consistency models (for example classical ACID and eventual consistency). It is not easy for developers to understand these models and to maintain its properties while coding their application. Moreover, it is more complicated when it comes to execute complex queries and especially join queries. But first, the developer has to discover all data stores capabilities of the available cloud providers in order to elect the most suitable cloud provider to his application requirements to deploy it. However this in itself is a tedious and meticulous work. In this paper, we will tackle these problems and we will propose an end-to-end solution and we will focus on the step of the cloud data stores discovery.

3 OVERVIEW OF OUR APPROACH

The lion's share of the contribution of our work is dedicated to application developers. Indeed, we attempt to simplify the developer task during the life cycle of an application (i.e. development, discovery and deployment, and execution) using multiple and heterogeneous data stores in its source code. The developer has (i) to write the source code of his application using various APIs, (ii) to discover data stores of each cloud provider in order (iii) to deploy his application, and (iv) to execute queries.

Hence, for the purpose of simplifying the developers task and getting them rid of all their onerous responsibilities, we propose our solution that is based on three points: (i) using a unique API enabling the

coding and the execution of multiple data store based application, (ii) enabling join queries execution over heterogeneous data stores, and (iii) ensuring a neutral application deployment in a cloud provider that is already automatically discovered.

In Fig 1, we showcase an overview of our solution including the different steps of the application life cycle. Indeed, we depict the following four steps:

- *Development Step*: The first step is the application development. For this purpose, we propose to use a unique API enabling the interaction with relational and NoSQL data stores. In a previous work (Sellami et al., 2014), we defined a generic resources model defining the different concept used in each type of data store. These resources are managed by ODBAPI a streamlined and a unified REST API enabling to execute CRUD operations on different NoSQL and relational databases. ODBAPI decouples cloud applications from data stores alleviating therefore their migration. Moreover it relieves developers task by removing the burden of managing different APIs.
- *Discovery Step*: The second step consists in discovering data stores capabilities of each cloud provider. So, we can decide which cloud provider can support our application requirements. Doing so, the developer describe his requirements in the *Abstract application manifest* and sends it to the *discovery* module. The *discovery* module interacts with the cloud providers in order to discover the capabilities of each cloud provider in term of data store services. Then, it obtains as a result the *offer manifest*. Based on these two manifests, the *Matching module* elects the most appropriate cloud provider to the application in order to deploy it and edit its *deployment manifest*.
- *Deployment Step*: The third step represents the application deployment. An important part in this phase consists in generating the application executable and adding the *execution manifest* to it. This manifest contains the endpoint of the *ODBAPI server* or/and the data stores endpoints that the application requires. We propose to deploy an application by any deployment API (e.g. COAPS API (Sellami and et al., 2013), roboconf API³, etc.). In our work, we are building on the COAPS API that is proposed in our team and allows human and/or software agents to provision and manage PaaS applications. This API provides a unique layer to interact with any cloud provider based on manifests. We model the *deployment*

manifest based on the manifest of COAPS API and we enrich it with information about data stores to support ODBAPI-based application.

- *Execution Step*: The fourth step is the application execution. In this step, an application can execute two types of queries. On the one hand, it can execute CRUD queries; hence it interacts directly with the target data store based on its *execution manifest*. In this case, it uses a unique API like ODBAPI. On the other hand, it can execute join queries by interacting with virtual data stores. A virtual data store is created automatically for executing join queries. When a virtual data store receives a join query, it parses this query, rewrites it into sub-queries written in the ODBAPI syntax, and sends these sub-queries to the target data stores. Once a virtual data store receives the result of each sub-query, it forms the final result and returns it to the application. It is worth noting that when the user edits the *Abstract application manifest*, he has to precise that his application will manipulate complex queries. Hence, a virtual data store will be created and its endpoint will be added to the *execution manifest* also.

In the following section, we will focus only on the discovery of the data stores capabilities and the deployment of an ODBAPI-based application.

4 DISCOVERY OF DATA STORES OF CLOUD PROVIDERS

In this section, we present our logic to discover the capabilities of data stores of cloud providers meeting the application requirements (see Figure2). The developer coded an ODBAPI application and describes its requirements in the *abstract application manifest* in term of data stores and deployment. Then, he gives it as an input to the *matching algorithm*. This algorithm interacts with the *data stores directory* in order to obtain the data stores capabilities of each cloud provider stored in the *offer manifest*. This manifest represents the second input of this algorithm in order to obtain the *deployment manifest*. The *data stores directory* is automatically updated by interacting with the cloud providers using their APIs.

In the rest of this section, we introduce UML diagram classes illustrating the *Abstract application manifest* and *offer manifest* description. In addition, we introduce our *matching algorithm*.

³Roboconf home page: <http://roboconf.net/fr/index.html>

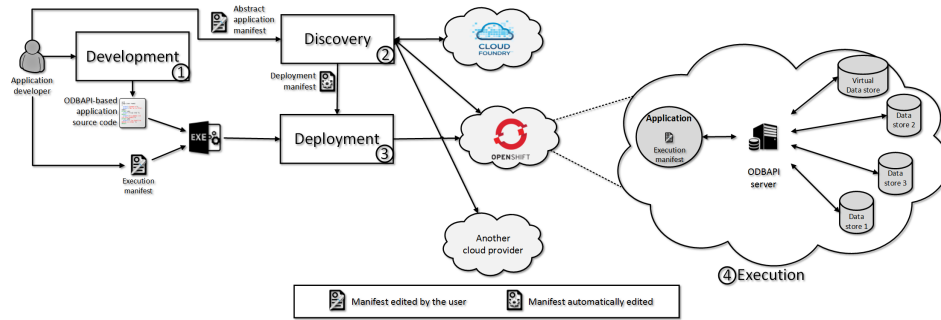


Figure 1: Overview of our solution.

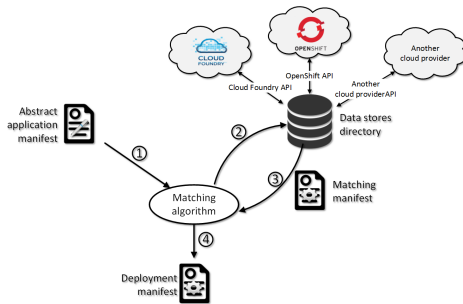


Figure 2: Zoom-in on the discovery module.

4.1 Abstract Application Manifest

This manifest contains two categories of requirements. First, we have requirements in term of data stores. The developer provides five information about the required data stores: its type, its name, its version, its size and the query type to execute. It is worth noting that when the developer fills this manifest, he has the freedom to specify one or multiple information. For each information, he gives a constraint expressed by a constant value, a joker (denoting any values) or some conditions (expressed as inequalities). Hence, we will ensure more flexibility in our model. For instance, one developer precises that he wants a data store having name MongoDB and type document and another developer precises that he wants a data store of type document without specifying its name (in this case any data store of type document fulfill the specification). Whereas the second category of the requirements is dedicated to the application deployment. Indeed, the developer precises the number of the virtual machines that he needs to run his application. In addition, he describes the executable of his application by giving its name, its type and its location.

We depict in Figure 3 an UML class diagram illustrating the *Abstract application manifest* model. The root class of our model is the *Abstract application manifest* class and it is identified by the attribute *name*. This class contains these following classes:

- The *User Information Class*: This class represents the required authentication information required to access the discovery module and consult the data stores repository. This class contains the *user login* class and the *user password* class that represents the developer identifiers.
- The *environment* class: This class represents the environment where an application will be deployed and it is instantiated from an environment *template* element that is characterized by a *name* attribute and a *memory* value. Each environment *template* contains at least one *node* class that represents resources in a cloud provider. A *node* class is identified by an *id* and a *content_type* attributes. This latter can be a *container* or a *database* to denote respectively engine resources to host and run services and storage resources. This class contains a *name* class and a *version* class. When the *content_type* attribute is equal to *database*, the *node* class contains also a *type* class, *query_type* class, and a *size* class to denote the type of data storage services, the query type (CRUD queries or join queries) and its size.
- The *application* class: This class represents the constraints that the user requires to deploy his application. It is characterized by a unique *name* attribute and the *environment* attribute where the application will be deployed. It has at least one *version* class that is identified by a *name* and a *label*. Each *version* class contains two types of classes: *deployable* class and *instance* class. The *deployable* class represents the application executable file. It is identified by a unique *id* attribute, a *content_type* attribute defining the executable file type, a *name* attribute denoting its name, a *location* attribute containing their URL, and a *multi-tenancy_level* attribute indicating the application tenancy degree. Whereas the *instance* class represents the running application instances required by the user. This class is identified by a unique *id* attribute, a *name* attribute, *initial_state* attribute

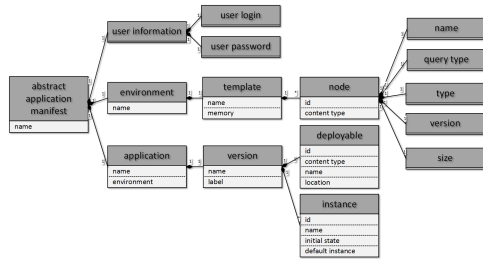


Figure 3: Abstract application manifest model.

defining the state of the application (e.g. running, stopped, etc) and *default_instance* attribute representing the running instances by default.

4.2 Offer Manifest

The offer manifest contains information about the capabilities of data stores of each discovered cloud provider. In Figure 4, we present a offer manifest modeling based on a class diagram. Indeed, the root class is *offer manifest* and it is identified by the *name* attribute. It contains one or multiple *cloud provider* class. This class represents a discovered cloud provider and it is identified by a unique *id* attribute. It contains the *name* class defining the cloud provider name and the *environment_response* class representing the capabilities that a cloud provider exposes according to an *abstract application manifest*. The *environment_response* class is composed by one or multiple *offer* class that contains one or multiple *node* class. This class is similar to the *node* element in the *abstract application manifest* modeling.

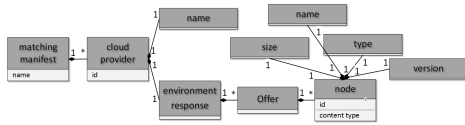


Figure 4: Offer manifest model.

4.3 Deployment Manifest

The *deployment manifest*'s structure is closest to the *abstract application manifest* and it is defined based on the COAPS API's manifest (Sellami and et al., 2013) (see Figure 5). Hence, in order to avoid the repetition, we do not describe this manifest structure in details. However, we will present we extend the COAPS API manifest structure. In fact, we add new attributes about data stores services in the *paas_node* class. These attributes are the *size* attribute, the *type* attribute and the *version* attribute.

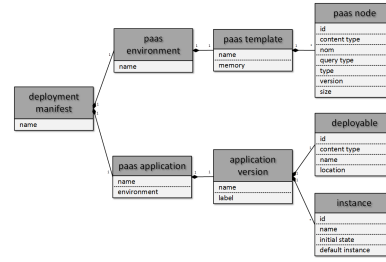


Figure 5: Deployment manifest model.

4.4 Matching Algorithm

In algorithm 1, we introduce the *matching algorithm* that elects the cloud provider that supports the best the application requirements in terms of data stores. We need a flexible matching allowing to elect a cloud provider which does not fulfill all the application's requirements but it is closed to them (that is the role of the computed distance). Furthermore, we impose that the result is a single cloud provider. First, the algorithm constructs the *offer manifest* using the operation *queryDataStoresRepository* by interacting with the data stores repository (line 4). We introduce this operation in Algorithm2. Second, it calculates for each cloud provider the number of differences between its capabilities and the user requirements described in the *abstract application manifest* (lines 6-18). Numbers of differences are stored in the data structure *distance*. These values are calculated as follows: for each property in the two manifest, if they are not corresponding then we update the distance by adding the appropriate penalty to the property. The two properties correspond if the actual value of the offer manifest property fulfill the requirement expressed by the abstract application manifest property (which is either a constant, a joker or a condition). These penalties are customized according to the property importance. By default, all penalties are fixed at 1; however the user can configure these penalties according to the importance that he gives to the properties. Once this step is achieved, we can now elect a cloud provider and construct the *deployment manifest* (lines 19-20). This is done through the operation *electCP* that takes as inputs the the data structure *distance* and the *threshold* and returns the identifier of the elected cloud provider if any. This identifier is the smallest value bounded between 0 and the *threshold*.

In algorithm 2, we present in more details the operation *queryDataStoresRepository* that returns a super-set of the result from the data stores repository. In fact, for each cloud provider it extracts all data stores corresponding to the data types of the *abstract application manifest*. If there are no corresponding data stores, the cloud provider is rejected (lines 10-

Algorithm 1: Matching algorithm.

```

1: input AAM: the abstract application manifest
2: input threshold: the threshold to limit the number of differences
3: output DM: the deployment manifest
4: OM  $\leftarrow$  queryDataStoresRepository(AAM) # see Algorithm 2#
5: i  $\leftarrow$  0
6: while (exist(Cloud Provider CP in OM)) do
7:   while (exist(Offer O in OM)) do
8:     distance[i]  $\leftarrow$  0
9:     for each node N in AAM do
10:      for each property prop in N do
11:        if (!valid(prop, OM.CP.O.node.prop)) then
12:          distance[i]  $\leftarrow$  distance[i] + updateDistance(prop,
            OM.CP.O.node.prop)
13:        end if
14:      end for
15:    end for
16:    i  $\leftarrow$  i + 1
17:  end while
18: end while
19: electedCP  $\leftarrow$  electCP(distance, threshold)
20: return createDM(AAM, OM, electedCP)

```

Algorithm 2: The queryDataStoresRepository algorithm.

```

1: input AAM: the abstract application manifest
2: output OM: the offer manifest
3: length  $\leftarrow$  0
4: for each node N in AAM do
5:   if (content-type == "database") then
6:     T[length]  $\leftarrow$  getType(N)
7:     length  $\leftarrow$  length + 1
8:   end if
9: end for
10: for each Cloud Provider CP in the data stores directory do
11:   for i=0 to length do
12:     if (CP contains T[i]) then
13:       Add all names of this type to the tree of the current CP
14:     else
15:       Reject the current CP
16:     end if
17:   end for
18: end for
19: return the resulted trees as the OM

```

18). For ease of presentation of this algorithm, we build ourselves on the Figure 6. Indeed, in the left side of the figure, we construct from the *abstract application manifest* a simple graph in which nodes represent the *type* elements in the *node* elements. This graph is a kind of a sample that will be used to construct the offer manifest. In the right side of Figure, we illustrate a data stores repository of a cloud provider having four types of data stores. Based on this repository and the graph based sample, we extract the list of the cloud provider's offers that we represent in the form of a graph. Once we check all cloud providers, we collect all the offers in the *offer manifest* (line 19).

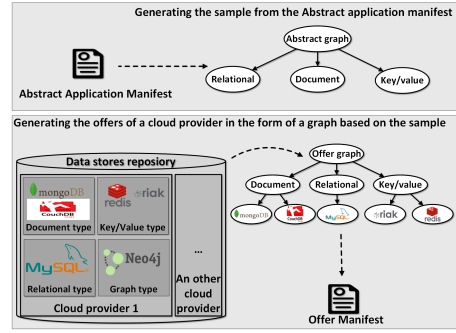


Figure 6: Generating the offer manifest.

5 IMPLEMENTATION AND VALIDATION

In this section, we present the tool implementing the *matching module*. Hence, in order to show the feasibility of this module, we propose to discover data stores of cloud foundry and open shift as cloud providers to deploy an ODBAPI-based application. This application is intended to handle the administration of relational and NoSQL data stores in a cloud provider. To do so, we propose to implement the manifests modeling with XML. We give in this section the example of the *abstract application manifest* (see Listing 1). Indeed, the developer provides *user1* as a login and *pswd* as a password. Then, he describes the environment that he requires in order to deploy his ODBAPI based application. Indeed, he chooses as name for the environment *ODBAPIEnv* and for the environment template *ODBAPIEnvTemp*. In this template, the user wants *tomcat* as an application container, a document data store without precising any name by filling the element *name* with the character * and MySQL as a relational data stores. Regarding the application configuration, the user names his application *ODBAPIApplication*. He precises that the application is a runnable file and he requires to have two application instances: one is running by default and the other is stopped. In this section we present only the XML based representation of the *abstract application manifest* for lack of space. We wish to emphasize that we provide our tools and videos demonstration at http://www-inf.int-evry.fr/~sellam_r/Tools/ODBAPI/index.html for more details.

```

1 <abstract-application-manifest name="AAM">
2   <user-information>
3     <user-login>user1</user-login>
4     <user-password>pswd</user-password>
5     <query-type>complex</query-type>
6   </user-information>
7   <environment name="ODBAPIEnv">
8     <template name="ODBAPIEnvTemp" memory="128">

```

```

9      <node id="1" content_type="container">
10        <name> tomcat </name>
11        <version> </version>
12      </node>
13      <node id="2" content_type="database">
14        <name>*</name>
15        <version> 1.0 </version>
16        <type> document </type>
17        <size> large </size>
18      </node>
19      <node id="3" content_type="database">
20        <name> mysql </name>
21        <version> * </version>
22        <type> relational </type>
23        <size> small </size>
24      </node>
25    </template>
26  </environment>
27  <application name="ODBAPIApplication" environnement=
28    "ODBAPIEnv">
29    <version name="version1.0" label="1.0">
30      <deployable id="1" content_type="artifact"
31        name="ODBAPIApplication.war" location="1444d7"
32        multitenancy_level="SharedInstance"/>
33      <instance id="1" name="Instance1"
34        initial_state="1" default_instance="true"/>
35      <instance id="2" name="Instance2"
36        initial_state="1" default_instance="false"/>
37    </version>
38  </application>
39 </abstract_application_manifest>

```

Listing 1: XML based representation of the abstract application manifest.

We programmed also a tool ensuring the discovery of cloud providers and the automatic deployment of an ODBAPI-based application. Indeed, the application programmer describes his requirements in the *abstract application manifest* and he uploads it through the interface that we illustrate in Fig. 7. Once this manifest is uploaded, this tool executes the *manifest based matching algorithm* to elect the appropriate cloud provider that supports the ODBAPI client requirements and returns the user the *deployment manifest*. Based on the *deployment manifest*, we deploy the ODBAPI client by the mean of COAPS API.

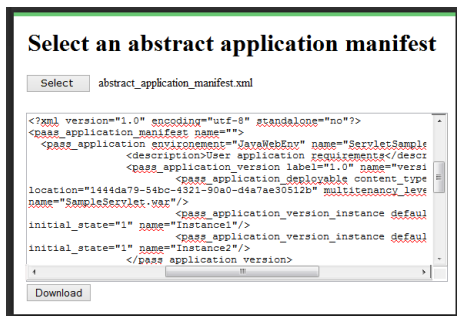


Figure 7: Screenshot of the interface allowing to select the user manifest in order to get the deployment manifest.

In our work, we deploy an ODBAPI-based client intended to handle the administration of relational and NoSQL data stores in a cloud provider. In Fig. 8, we show an overview of the databases created in each data store in the Cloud Foundry. Indeed, there is a MySQL database called *world* and it contains

id	countrycode	name	population
55	AND	Andorra la Vella	21189
56	AGO	Luanda	2022000
57	AGO	Huambo	163100
58	AGO	Lobito	130000
59	AGO	Benguela	128300
60	AGO	Namibe	118200
61	AIA	South Hill	961
62	AIA	The Valley	595

Figure 8: Screenshot of all databases overview.

three entity sets: *city*, *country*, and *countrylanguage*. Added to that, we have the MongoDB database that is named *person* and it is composed by two entity sets: *Student* and *Teacher*. We show also an overview of the entities of the *city* entity set.

6 RELATED WORK

In our previous work (Sellami and Defude, 2013), we focused on existing solutions of the state-of-the-art supporting multiple data stores based applications in the cloud environment. More precisely, (i) we described different scenarios related to the way applications use data stores, (ii) we defined the data requirements of applications in cloud environment, and (iii) we analyzed and classified existing works on cloud data management, focusing on multiple data stores requirements. As a result, we pointed out six requirements of using multiple data stores in a cloud environment. One of these requirements consists in choosing a data store based on a data requirements. We present three sub-requirements: defining application needs and requirements towards data, defining data store capabilities, and defining application needs and data stores capabilities matching.

Against this background, we find several worksq (Truong and et al., 2012), (Truong and et al., 2011), (Vu and et al., 2012), (Ghosh and Ghosh, 2012), (Ruiz-Alvarez and Humphrey, 2011), (Ruiz-Alvarez and Humphrey, 2012) enabling an application to negotiate its Data Management Contract (DMC), often referred to as data agreement or data license, with various clouds and to bind to the specific DBMSs according to its DMC. Truong et al. (Truong and et al., 2012), (Truong and et al., 2011), (Vu and et al., 2012) propose to model and specify data concerns in data contracts to support concern-aware data selection and utilization. For this purpose, they define an abstract model to specify a data contract and the main data contract terms. Moreover, they propose some algorithms and techniques in order to enforce the data contract usage. In fact, they present a data

contracts compatibility evaluation algorithm and they define how to construct, compose and exchange a data contract. In (Truong and et al., 2011), they introduce their model for exchanging data agreements in the Data-as-a-Service (DaaS) based on a new type of services which is called Data Agreement Exchange as a Service (DAES). This model is called DDescription Model for DaaS (DEMDS) (Vu and et al., 2012). However, Truong et al. propose this data contract for data and not to store data or to help the developer to choose the appropriate data stores for his application. In (Ghosh and Ghosh, 2012), Ghosh et al. identify non-trivial parameters of the Service Level Agreement (SLA) for Storage-as-a-Service cloud which are not offered by the present day cloud vendors. Moreover, they propose a novel SLA monitoring framework to facilitate compliance checking of Service Level Objectives by a trusted third part. Although Ghosh et al. try to enrich the SLA parameters to support the Storage-as-a-Service, this is still inadequate for our purpose in this paper. In (Ruiz-Alvarez and Humphrey, 2011), (Ruiz-Alvarez and Humphrey, 2012), Ruiz-Alvarez et al. propose an automated approach to selecting the PaaS storage service according an application requirements. For this purpose, they define a XML schema based on a machine readable description of the capabilities of each storage system. The goal of this XML schema is twofold: (i) expressing the storage needs of consumers using high-level concepts, and (ii) enabling the matching between consumers requirements and data storage systems offerings. Nevertheless, they consider in their work that an application may interact with only one data store and they did not invoke the polyglot persistence aspect.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a manifest-based solution for data stores discovery and automatic application deployment. Indeed, once the developer has completed the development of his application, we provided him the possibility to express his application requirements in terms of data stores in the *abstract application manifest*. Then, he sends it to the *discovery module*. This module interacts with the data stores directory to discover the capabilities of data stores of each cloud provider and constructs the *offer manifest*. Based on that, this module implements the *matching algorithm* in order to elect the adequate cloud provider to the application requirements. This algorithm takes as input the *abstract application manifest* and *offer manifest*, and returns the *deployment*

manifest of the application. Once it is done, we deploy the application using the COAPS API that takes as input the *deployment manifest*.

Currently, we are working on applying our solution to other qualitatively and quantitatively various scenarios in the OpenPaaS project. This allows us to identify possible discrepancies and make our work more reliable for real use. Our second perspective is to implement virtual data stores in order to execute join queries across NoSQL and relational data stores and to introduce more elaborate query processing optimization techniques.

REFERENCES

- Baun, C. and et al. (2011). *Cloud Computing - Web-Based Dynamic IT Services*. Springer.
- Ghosh, N. and Ghosh, S. K. (2012). An approach to identify and monitor sla parameters for storage-as-a-service cloud delivery model. In *Workshops Proceedings of the Global Communications Conference, GLOBECOM 2012, 3-7 December, Anaheim, California, USA*, pages 724–729.
- McAfee, A. and Brynjolfsson, E. (2012). Big data: The management revolution. (cover story). *Harvard Business Review*, 90(10):60–68.
- Ruiz-Alvarez, A. and Humphrey, M. (2011). An automated approach to cloud storage service selection. In *Proceedings of the 2Nd International Workshop on Scientific Cloud Computing, ScienceCloud '11*, pages 39–48.
- Ruiz-Alvarez, A. and Humphrey, M. (2012). A model and decision procedure for data storage in cloud computing. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, Ottawa, Canada, May 13-16*, pages 572–579.
- Sellami, M. and et al. (2013). Paas-independent provisioning and management of applications in the cloud. In *2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*, pages 693–700.
- Sellami, R., Bhiri, S., and Defude, B. (2014). ODBAPI: a unified REST API for relational and NoSQL data stores. In *The IEEE 3rd International Congress on Big Data (BigData'14), Anchorage, Alaska, USA, June 27 - July 2, 2014*.
- Sellami, R. and Defude, B. (2013). Using multiple data stores in the cloud: Challenges and solutions. In *Data Management in Cloud, Grid and P2P Systems - 6th International Conference, Globe 2013, Prague, Czech Republic, August 28-29, 2013. Proceedings*, pages 87–98.
- Truong, H. L. and et al. (2011). Exchanging data agreements in the daas model. In *2011 IEEE Asia-Pacific Services Computing Conference, APSCC 2011, Jeju, Korea (South), December 12-15*, pages 153–160.

- Truong, H. L. and et al. (2012). Data contracts for cloud-based data marketplaces. *IJCSE*, 7(4):280–295.
- Vu, Q. H. and et al. (2012). Demods: A description model for data-as-a-service. In *IEEE 26th International Conference on Advanced Information Networking and Applications, AINA, 2012*, Fukuoka, Japan, March 26-29, pages 605–612.

MusicBeetle

Intelligent Music Royalties Collection and Distribution System

Carlos Serrão, Hélder Carvalho and Nelson Carvalho
ISCTE-IUL/ISTAR-IUL, Ed. ISCTE, Av. das Forças Armadas, 1649-026 Lisboa, Portugal
carlos.serrao@iscte.pt, hel.carvalho@gmail.com, nelson.icebox@gmail.com

Keywords: Music, Related-Rights, Royalties, Distribution, Cloud, Recommendation.

Abstract: Music industry has been completely disrupted by a range of new online digital services and social networking systems that has forever changed the way users and businesses experience and use music. This had a tremendous impact on the established music business models that had guided a dozen year-old industry. On what concerns business music users, i.e. businesses that make use of music as part of their own business model, and on the business relation they establish with author societies or their representatives, they are required to pay royalties for the use of music. These royalties need to be distributed and authors will have the opportunity to see their work rewarded properly. The proper distribution of royalties is a non-transparent and complex process. In this paper, the authors present a system, called MusicBeetle that enables the identification, collection and distribution of music royalties through the usage of decentralised system and low cost hardware devices.

1 INTRODUCTION

The Internet and the digital technology has created serious challenges in terms of Intellectual Property protection and management of digital content assets, for both end-users, content authors, content distributors and rights collecting and distributing societies (Torres, Serrão, Dias and Delgado, 2008).

The Related Rights (RR) or Neighbouring Rights (NR) are terms in copyright law that represent the rights which are similar to the author rights but which are not connected with the actual author of the work (Frith and Marshall, 2004). Both the author rights and the related rights are copyrights. The RR/NR are independent of any authors' rights, which may also exist in the work (WIPO, 1961). The rights of performers, phonogram producers and broadcasting organisations are certainly covered, and are internationally protected by the RR/NR legislation (Correa, 2007). In the specific case of the music industry, and as an example, four different copyright-types rights will concurrently protect a CD recording of a song:

- The authors' rights of the composer of the music;
- The authors' rights of the lyricist;
- The performers' rights of a singer and the musicians;

- The producers' rights of the person or corporation, which made the recording.

Therefore one the most important activities of these Music Related/Neighbouring Rights Management Societies (MRNRMS) is the collection of neighbouring rights on behalf of producers and performers related to public performance of recorded music (Correa, 2007). Consequently the mission of a MRNRMS can be resumed in the following four major objectives:

- Raise public awareness to the reality of related/neighbouring rights and the need for its protection (a fact still relatively new and little known);
- Boosting the delivery of remuneration for distribution to the holders, be they producers or artists;
- Realize the collection of related/neighbouring rights to all places of public performance using recorded music for commercial purposes, as well as all the inspectors to use of recorded music, by any means;
- The community awareness in relation to associated rights will, in large part, be accomplished with the collaboration of public authorities with powers of supervision on Copyright and Related/Neighbouring Rights, as well as the users of

recorded music in various areas and industries that in compliance with the law, should ask for their license.

These MRNRMS are responsible for issuing licenses to businesses that use represented recording music as a mean to conduct their own business models. Moreover, they are also responsible for the effective collection and distribution of the associated fees to the music producers, performers and authors (Bustinza, Vendrell-Herrero, Parry, and Myrthianos, 2013). Most of these MRNRMS exist in nearly every civilised country in the World and they often operate their business based on manual and non-automatic processes, causing them to be less effective on their core business functions.

2 BUSINESS MUSIC USERS (BMU) AND ROYALTIES DISTRIBUTION

Businesses use in-store media entertainment content as a way to increase the perceived value of their core business while engaging more customers and creating the opportunities for them to stay longer and consume more (Teece, 2010). Music is an important part of their business model. They recognise its importance and therefore are willing to comply with the legal requirements that impose the payment of royalties to authors. Not only business music users (BMU) are required to use legal content (legally acquired music) but they also need a public execution license (Vaccaro and Cohn, 2004). Public execution licenses are a requirement for businesses that depend on the usage of music for public execution on their businesses - this includes, for instance, discos, bars, restaurants, stores, gyms, parking zones, hotels and many more.

The MRNRMS are responsible for collecting the rights royalties from the different BMU and distribute those royalties to the different beneficiaries of such royalties (artists, performers, and others). Usually this distribution method is performed through the sampling of the percentage of the number of times a given music is played on a given medium (Figure 1). Currently, some specific companies are hired to audit the music usage, using specific human auditors to listen to the different medium (radio stations, TV channels, and some other mass media mediums) for a given period of time and produce statistical data estimations that are used to extrapolate the real music usage ratings that are after used to perform royalties distribution. Also, some additional

criteria are used to charge BMU, like the business space, the number of days the business operates during the year, and other similar.

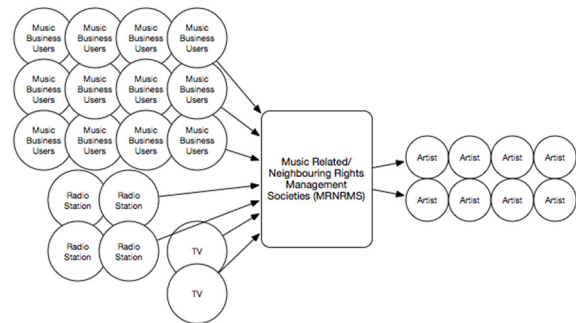


Figure 1: Related-rights distribution scenario.

The way, this all process is conducted, is completely error prone and not transparent. The process is not accurate, and leaves out from the royalties distribution chain some of the less well-known artists (Castro, Alves, Serrão and Caraway, 2010). Moreover, this system can only be used for larger music distribution channels, and are not adequate for the different BMU that use music as part of their business model - they represent a large amount of entities that are charged for a license that enable them to use music on their business and execute their business model.

These facts have created the need for a new type of system that allowed the MRNRMS to charge BMU in a fairer way and distribute owed royalties to artists in a more transparent manner.

The following sections of this paper present a system, called MusicBeetle, that is responsible for automatically auditing the music usage by BMU and by ensuring the appropriate royalties collection by the MRNRMS and distribution to the authors. The following section presents the MusicBeetle system that is divided into two different components - MusicBeetle.box and the MusicBeetle.cloud. The two components are further described and details about how they both operate to fulfil the automatic music auditing process and royalties distribution.

3 THE MUSICBEETLE SYSTEM

In order to improve the related-rights royalties collection and distribution process that is implemented manually by the MRNRMS, it was designed and implemented a system that automates the entire process.

The system, called MusicBeetle, was composed by two different components: (a) a critical client-side component that was capable of automatically identifying the music being used and create a report of all the music used on a given period of time by a specific BMU (MusicBeetle.box), and (b) a set of cloud-based services integrated with the MRNRMS information systems that registers all the different music business licensees, their music usage profile, and information about music identification and artists database (MusicBeetle.cloud).

In order for the entire system to work, the different BMU have to install specific hardware devices that are connected to the Internet (the MusicBeetle.box, that will be presented in the following section) and connected to the music sound system used by the BMU. On the other side, the MRNRMS has access to an large database of the entire music repertoire that they represent on a given country or has the means to access further information online.

The following sections provide an overview of these two different components of the system, how they interoperate and which are their major functionalities.

3.1 MusicBeetle.box

The MusicBeetle.box is one of the critical components on the system. This client-side hardware component allows the system to actively listen to the music being used at the BMU side, record the music candidate identification, and report back the music consumption to the MRNRMS.

Here are some of the most important requirements for the development of such critical component:

- The system should be able to connect to an external sound system;
- It should be able to listen to the music that is being played at the sound system;
- It should be connected to the Internet (or at least it should temporarily connected to the Internet - not requiring a permanent connection);
- The system should be able to create unique identifiers for the music that is listening (from time to time) based on audio fingerprinting technology (Cano, Batlle, Kalker, and Haitma, 2005);
- The system should be able to record at least a month time of audio identifiers;
- And finally, another important requirement is that the system should be inexpensive.

All of the above requirements were considered in the design and development of a solution.

3.1.1 MusicBeetle.box Hardware Architecture

Having into consideration all of the previous technical and financial requirements, the obvious choice was to select an inexpensive hardware solution, based on the “all-in-one” boards that existed on the market. After analysing some of the existing ones (Raspberry Pi, CubieBoard, PandaBoard, BeagleBoard, and CuBox), it was decided to select the Raspberry Pi (RPI). The RPi represents a cost effective solution that also presents the processing capabilities required by the solution to implement.

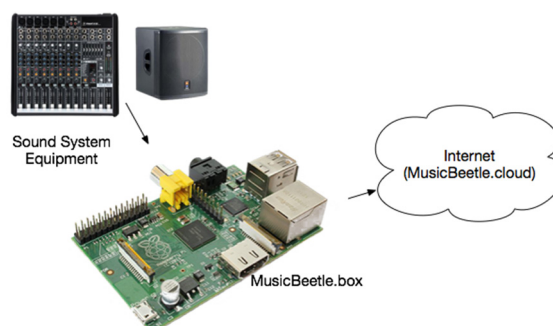


Figure 2: MusicBeetle.box (based on the Raspberry Pi hardware) and the integration with the client Sound System and the Internet.

Therefore, RPi was selected as the principal hardware component to implement the MusicBeetle.box prototype (Richardson and Wallace, 2012). Raspberry Pi is a credit card-sized single-board computer that was developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. Raspberry Pi is based on the Broadcom system on a chip (SoC), including an ARM processor, a GPU, and some amount of memory (originally 256 MB and later 512 MB). The system has Secure Digital (SD) or MicroSD sockets for boot media and persistent storage (Upton and Halfacree, 2012). Moreover, Raspberry Pi has also an HDMI port, several USB ports, an Ethernet port and an audio connector (Figure 2).

The Raspberry Pi board has become the natural and adequate solution to sustain the MusicBeetle.box system and to implement crucial requirements, like the capability to listen to music and connect to the Internet to report the audited music. Additionally, each of the RPi boards costs around 35 euros, making it inexpensive enough for mass distribution.

3.1.2 MusicBeetle.box Logical and Software Architecture

Apart from the hardware that was selected, the MusicBeetle.box is also composed by a set of software that was specifically developed to implement some of the requirements of the system. All the developed software runs on the Raspbian Linux-based (on the Debian Wheezy distribution) operating system. On top of the operating system, a set of specifically developed software is running to implement a set of operations that enable the correct operation of the MusicBeetle.Box.

The following schema (Figure 3) depicts how the MusicBeetle operates to conduct the music audits at the BMU side.

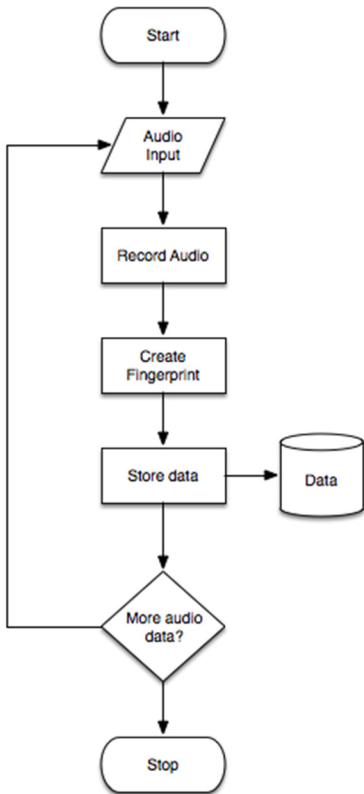


Figure 3: Schema that demonstrates how the MusicBeetle.box operates to audit music usage.

The implemented music auditing process is composed by the following operations:

1. The MusicBeetle.box has an active process that continuously listens for the existence of new music on the sound board;
2. Every time music/audio is detected, the process captures and records 15 seconds of audio data;

3. A candidate audio fingerprint is calculated for the audio sample from this 15 seconds of captured audio;
4. A timestamp of the date and time of the audio data capture and the candidate audio fingerprint are stored on a temporary MusicBeetle.box database;
5. After a waiting process of 30 seconds, the process verifies if there is more audio data available to capture. If there is, it repeats the entire workflow (returning to step 2).

In order to conduct this auditing process and being able to operate at the BMU side, the MusicBeetle.box implements the following software architecture (Figure 4). This software architecture lies on top of the specific Raspberry Pi Linux distribution (Raspbian), with some specific software developed for accomplishing the required tasks of the MusicBeetle.box.

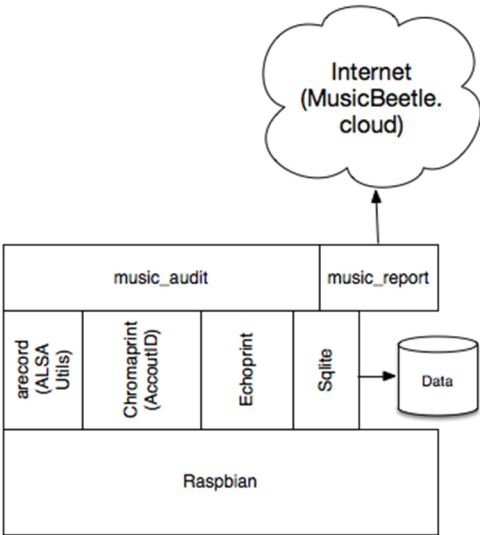


Figure 4: Schema that demonstrates how the MusicBeetle.box operates to audit music usage.

The MusicBeetle.box is composed by two different software processes that are running on the box. These two software processes are “music_audit” and “music_report”. Both of these processes are automatically activated immediately after the Raspberry Pi is turned on and the Raspbian operating system finishes booting up.

The “music_audit” is the process that is responsible for auditing and identifying the music that is used at the BMU side. This process conducts the following functions:

- Actively listens to the sound board for the existence of audio data. If that data exists, “mu-

sic_audit" records 15 seconds of the audio data, using "arecord" (an utility that is part of the ALSA utils package) and saves it to a temporary file;

- After this, "music_audit" uses two different audio fingerprint open-source tools (AccoustID and Echoprint) to create two candidate audio fingerprints from the audio samples that were previously captured and recorded, using "chromaprint" and "echoprint". These audio fingerprints are used to create a tentatively positive identification of the music being played, while performing a posterior comparison with a fingerprinting database in a matching process. The matching process takes place at the MusicBeetle.cloud;
- The following example describes how to use "echoprint" to create a audio fingerprint from an audio sample captured by the box: `./echoprint-codegen audio_sample_[01].mp3 0 15`.
- After creating the two candidate audio fingerprints, "music_audit" saves them, together with a timestamp of the date and time of the audio sample capture. This data is saved on an temporary SQLite database ("sqlite");
- "music_audit" continues to actively monitor the existence of more audio data on the audio board of the Raspberry Pi, repeating the recording, audio fingerprinting and saving processes.

The "music_report" is the process that is responsible for the communication of the candidate music identifications detected by the MusicBeetle.box to the MusicBeetle.cloud system. This process is responsible for the following functions:

- "music_report" is a process that is run by the "crond" Linux daemon. "crond" runs "music_report" on a daily basis (every 24 hours), at a given time (the frequency and time can be configured on the MusicBeetle.box);
- "music_report" connects to the SQLite database and extracts all the available records that correspond to the last period of audited music that has not yet been send to the MusicBeetle.cloud;
- The process builds a JSON data structure (Figure 5) that contains the necessary data to be sent to the MusicBeetle.cloud. This structure contains information about the unique identifier of the MusicBeetle.box on the system (UUID), a timestamp of the data and time of the transfer (TIMESTAMP), the two values of the samples fingerprints (FP1 and FP2, each one created with

a different audio fingerprinting algorithm) and the identifier of the audio sample (SID);

- After this, "music_report" establishes a secure connection, using SSL/TLS protocol, with the MusicBeetle.cloud service endpoint and posts the JSON data structure over the network. If no connection is available, MusicBeetle.box will retain the previous audit data and continues to store more data, until a connection becomes available and the transfer process concludes with success (after receiving a message from the MusicBeetle.cloud service confirming that the data reception was successful).

```
{
  "musicbeetlebox_uuid": "[UUID]",
  "transfer_timestamp":
    "[TIMESTAMP]",
  "audits": [
    {
      "sample_id": "[SID]",
      "sample_timestamp":
        "[TIMESTAMP]",
      "sample_fp": [
        "fp1": "[FP1]",
        "fp2": "[FP2]"
      ],
    },
    {
      "sample_id": "[SID]",
      "sample_timestamp":
        "[TIMESTAMP]",
      "sample_fp": [
        "fp1": "[FP1]",
        "fp2": "[FP2]"
      ],
    }
  ]
}
```

Figure 5: Sample JSON data structure containing the data to be sent from the MusicBeetle.box to the MusicBeetle.cloud. [UUID], [TIMESTAMP], [SID], [FP1] and [FP2] are simply placeholders that are replaced by the actual values.

Both the "music_audit" and "music_report" are two important processes in the way the MusicBeetle.box operates and conducts its major functionality: audit the BMU music real usage and reporting that information back to the MRNRMS. All the matching, accounting and management is performed on

the MusicBeetle.cloud, that will be detailed in the following section.

3.2 MusicBeetle.cloud and Services

MusicBeetle.cloud is a key element in the MusicBeetle ecosystem. It represents the services that are run on a cloud service by the MRNRMS that will enable the management of the music portfolio (and artists) whose rights are represented and managed by the MRNRMS. These services have to assure the following:

- The MRNRMS should have a system with data that represents the music and artist portfolio whose rights they represent. This system should contain not only information about the music, but also about the artists;
- MusicBeetle.cloud should also have information about the BMU that are registered at the MRNRMS, and information about the MusicBeetle.bboxes that are installed on their side;
- Also another important service that is required is the capability of being able to match the audio audits, sent from multiple MusicBeetle.bboxes, and then perform the identification of the music tracks, and account for the number of times a

given music track has been played by the BMU during a given period of time;

- The MusicBeetle.cloud service should also be capable of accounting the necessary amount to be distributed to a given artist, according to the music usage by the BMU ecosystem (that are legally licensed by the MRNRMS and possess a MusicBeetle.box) as well as to charge the specific rightful amount to the BMU according to the effective music usage.

The different services present at the MusicBeetle.cloud (Figure 6) cooperate with each other and with external services to ensure that MusicBeetle can meet the requirements that were previously identified. One of the most crucial operations of these services is the capability to receive reports from the different installed MusicBeetle.bboxes, and process such reports in terms of music identification, royalties accounting and fees to be charged to the BMU. This process is detailed on the next section.

3.2.1 Music Identification, Reporting and Royalties Management Process

One of the most vital operations that is executed at the MusicBeetle.cloud is the capability to receive the multiple reports from the MusicBeetle.bboxes, and process them in terms of music identification, royalties distribution and fees charging.

In order for these services to operate properly, the following assumptions need to be fulfilled:

1. It is necessary to build a repository that contains the music meta-information (related to the music track) and audio fingerprints that are unique to each of the music tracks. Each music track must also be associated with an artist (or multiple artists and/or recording label, if it is the case);
2. A repository with the information about the artists that are represented (registered) by the MRNRMS is necessary, and a relation with their music tracks;
3. It is also necessary to have all the information about the BMU that are licensed by the MRNRMS and the list of MusicBeetle.bboxes that are assigned to them.

After all of these pre-requisites are met, the process that is responsible for music identification, royalties distribution and fees charging is aligned with the following steps:

1. The process is initiated by the MusicBeetle.bboxes while sending report data to the “Music Identification and Matching Service”;
 - 1.1. The “Music Identification and Matching

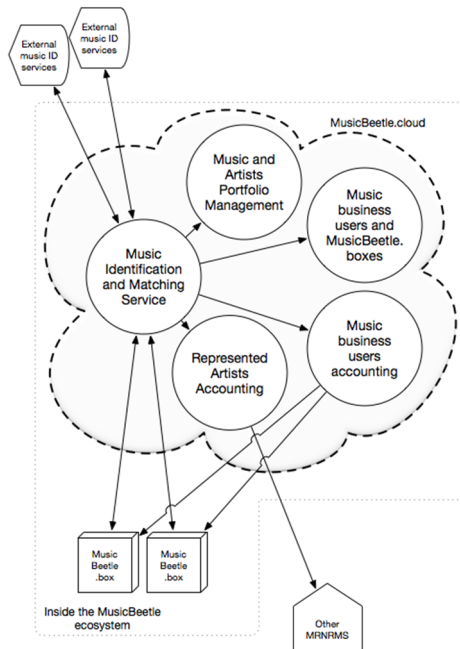


Figure 6: The schema represents the different services that are running on the MusicBeetle.cloud and how they interact with other systems (both internal to the MusicBeetle ecosystem or outside).

Service” receives the JSON data from the MusicBeetle.box;

1.2. The service verifies the field that contains “musicbeetlebox_uuid” and contacts the “Music business users and MusicBeetle.boxes service registry” to check if that specific MusicBeetle.box is registered and is valid within the MRNRMS context;

1.3. After validating the MusicBeetle.box, the service contacts the “Music Business users accounting service” to retrieve identification of the BMU that will be charged for the music usage.

2. After these initial steps, the “Music Identification and Matching Service” will try to identify the music tracks send on the JSON report.

2.1. The service looks into the “audits” field of the report and starts looking for existing music samples (“sample_id”) identification;

2.2. For each of the samples, the “sample_id”, “sample_timestamp” and “sample_fp” are retrieved;

2.3. Inside the “sample_fp” the values of both “fp1” and “fp2” are extracted;

2.4. “fp1” and “fp2” are used by the matching service to identify the music to which the samples refer to. The “Music and Artists Portfolio Management service” is used to perform this matching operation. If a match is found, the corresponding music meta-information can be retrieved and information about the music track and the corresponding artist can be used to establish the royalties distribution process;

2.5. If the sample fingerprints cannot be matched to any of the musics at the repository, it will be possible to pass that data to external matching and identification services to try a successful identification;

2.6. If it is not possible to identify a music sample, it is classified in a particular way, so that the MRNRMS can find a different way to distribute the royalties;

3. Finally, after the music track is identified, it is possible to direct the usage royalties to the appropriated artists and to charge the correct and fair fees to the BMU:

3.1. After the matching process concludes with success the service contacts the “Represented artists accounting service” to credit the artist for the corresponding value of its music usage;

3.2. The service also contacts the “Music

business users accounting service” to debit the BMU on the usage of that specific music track.

This process is repeated for each of the MusicBeetle.boxes that report music auditing information to the services on the MusicBeetle.cloud.

4 CONCLUSIONS

The music industry has changed across time. The well-established music business models that lasted for decades were completely shaken by a new emerging reality boosted by technology. Information and Communication Technologies (ICT) (Rosenblatt, Mooney and Trippe, 2001) has altered the relation between recording companies, artists, music and end-users (both individuals and businesses) (Vaccaro and Cohn, 2004). At the same time ICT has also created new music business models and raised opportunities for key actors in the music value-chain to become more efficient in its mission fulfillment (Handke and Towse, 2007).

One of these actors is the Music Related/Neighbouring Rights Management Societies (MRNRMS), entities that are responsible for the collection and distribution of royalties on the behalf of the artist (or other entities that represent them) (Kretschmer, Klimis and Wallis, 1999). These MRNRMS license BMU, charging them a fee for using commercial music as part of their core business model and distribute these fees to the represented artists (Towse, 1999). The problem is that collection and distribution process is not fair, accurate or transparent. Therefore there was the necessity for a system that could charge BMU according to their actual music usage, and distribute royalties to artists whose music’s have been played. The MusicBeetle system was developed to provide the necessary answer to these requirements.

The developed prototype was tested in the particular context of a Portuguese MRNRMS, where some of the properly licensed BMU were invited to participate in the system trials.

From the tests conducted it was possible to improve the way the license was charged to the MRNRMS customers, this more direct relation with the real music consumption, improved the way the royalties collection occurred (resulting from the direct music usage by the BMU) and also ensure more transparency on the way the royalties are distributed. Artists represented by the MRNRMS or by any of its associates received the royalties’ value according to its real music usage.

MusicBeetle contributed to the rethinking of an old paradigm in Related Rights (RR) or Neighbouring Rights (NR) royalties' collection and distribution, enabling a fairer rights charging and collection and a transparent distribution of royalties. Besides this, the system also provided the necessary mechanisms to audit the charging and distribution of such royalties.

Phonograms and Broadcasting Organizations. Retrieved February 06, 2015, from http://www.wipo.int/treaties/en/text.jsp?file_id=289757.

REFERENCES

- Bustinza, O. F., Vendrell-Herrero, F., Parry, G., and Myrthianos, V. (2013). Music business models and piracy. *Industrial Management and Data Systems*, 113(1), 4–22.
- Cano, P., Batlle, E., Kalker, T., and Haitsma, J. (2005). A review of audio fingerprinting. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 41(3), 271–284.
- Castro, H., Alves, A. P., Serrão, C., and Caraway, B. (2010). A new paradigm for content producers. *MultiMedia, IEEE*, 17(2), 90-93.
- Correa, C. (2007). Trade related aspects of intellectual property rights: a commentary on the TRIPS agreement. OUP Catalogue.
- Frith, S., and Marshall, L. (2004). *Music and copyright*. Edinburgh University Press.
- Handke, C., and Towse, R. (2007). Economics of copyright collecting societies. *International Review of Intellectual Property and Competition Law*, 38(8), 937-957.
- Kretschmer, M., Klimis, G. M., and Wallis, R. (1999). The changing location of intellectual property rights in music: A study of music publishers, collecting societies and media conglomerates. *Prometheus*, 17(2), 163-186.
- Richardson, M., and Wallace, S. (2012). *Getting Started with Raspberry Pi*. "O'Reilly Media, Inc."
- Rosenblatt, W., Mooney, S., and Trippe, W. (2001). *Digital rights management: business and technology*. John Wiley and Sons, Inc.
- Teece, D. J. (2010). Business models, business strategy and innovation. *Long range planning*, 43(2), 172-194.
- Torres, V., Serrão, C., Dias, M. S., and Delgado, J. (2008). Open DRM and the Future of Media. *MultiMedia, IEEE*, 15(2), 28-36.
- Towse, R. (1999). Copyright and economic incentives: an application to performers' rights in the music industry. *Kyklos*, 52(3), 369-390.
- Upton, E., and Halfacree, G. (2012). *Meet the Raspberry Pi*. John Wiley and Sons.
- Vaccaro, V. L., and Cohn, D. Y. (2004). The evolution of business models and marketing strategies in the music industry. *International Journal on Media Management*, 6(1-2), 46-58.
- WIPO. (1961). *WIPO-Administered Treaties: Rome Convention for the Protection of Performers, Producers of*

Context-aware MapReduce for Geo-distributed Big Data

Marco Cavallo, Giuseppe Di Modica, Carmelo Polito and Orazio Tomarchio

Department of Electrical, Electronic and Computer Engineering, University of Catania, Catania, Italy
{firstname.surname}@dieei.unict.it

Keywords: Big Data, MapReduce, Hierarchical Hadoop, Context Awareness, Partition Number.

Abstract: MapReduce is an effective distributed programming model used in cloud computing for large-scale data analysis applications. Hadoop, the most known and used open-source implementation of the MapReduce model, assumes that every node in a cluster has the same computing capacity and that data are local to tasks. However, in many real big data applications where data may be located in many datacenters distributed over the planet these assumptions do not hold any longer, thus affecting Hadoop performance. This paper addresses this point, by proposing a hierarchical MapReduce programming model where a toplevel scheduling system is aware of the underlying computing contexts heterogeneity. The main idea of the approach is to improve the job processing time by partitioning and redistributing the workload among geo-distributed workers: this is done by adequately monitoring the bottom-level computing and networking context.

1 INTRODUCTION

In the last few years, the pervasivity and the widespread diffusion of information technology services such as social computing applications and smart city services produced a significant increase of the amount of digital data, which in a single day may even reach a few petabytes (Facebook, 2012).

The new term “Big Data” has been created to indicate this phenomenon: it refers to collections of very large datasets, that require unconventional tools (e.g non-relational DBMS) to be managed and processed within a reasonable time (Zikopoulos, P. and Eaton, C., 2011). Big data analysis requires adequate infrastructure capable of processing so large amount of data: parallel and distributed computing techniques are commonly used to efficiently manipulate such data. MapReduce is probably the most known parallel programming paradigm that is nowadays used in the context of Big Data (Dean and Ghemawat, 2004). It is based on two functions, Map and Reduce: the first one generates data partitions based on a given user defined function, and the second one performs a sort of summary operation on Map outputs. Apache Hadoop is an open source implementation of the MapReduce approach (The Apache Software Foundation, 2011); in the last few years it has evolved by including many features and reaching a high level of adoption both in industry than in academic community. Hadoop has been designed mainly to work on clusters of homoge-

neous computing nodes belonging to the same local area network; data locality is one of the crucial factors affecting its performance. However, in many recent Big Data scenarios, it is not uncommon the need to deal with data which are geographically distributed. In fact, the design of geo-distributed cloud services is a widespread trend in cloud computing, through the distribution of large amounts of data among data centers located in different geographical locations. In these scenarios, the data required to perform a task is often non-local, which, as mentioned before, may severely affect the performance of Hadoop.

In this paper, we propose a novel job scheduling strategy that is aware of data location. The proposed approach takes into account the actual heterogeneity of nodes, network links and of data distribution. Our solution follows a hierarchical approach, where a top-level entity will take care of serving a submitted job: this job is split into a number of bottom-level, independent MapReduce sub-jobs that are scheduled to run on the sites where data reside. The remainder of the paper is organized as follows. Section 2 provides some motivation for the work and also discusses some related work. In Section 3 we introduce the system design and provide the details of the proposed strategy. Section 4 provides the details of the strategy adopted to partition data and distribute the workload. Finally, Section 5 concludes the work.

2 BACKGROUND AND RATIONALE

Well known implementations of MapReduce have been conceived to work on a single or on a few clusters of homogeneous computing nodes belonging to a local area network. Hadoop (The Apache Software Foundation, 2011), the most famous open source implementation of the MapReduce paradigm, performs very poorly if executed on data residing in geographically distributed datacenters which are interconnected to each other by means of links showing heterogeneous capacity (Heintz et al., 2014).

The main problem is that Hadoop is unaware of both nodes' and links' capacity, nor it is aware of the type of application that is going to crunch the data. This may yield a very bad performance in terms of job execution time, especially in the case a huge amount of data are distributed over many heterogeneous datacenters that are interconnected to each other's through disomogeneous network links. In the literature two main approaches are followed by researchers to efficiently process geo-distributed data: a) enhanced versions of the plain Hadoop implementation which account for the nodes and the network heterogeneity (*Geo-hadoop* approach) ; b) hierarchical frameworks which gather and merge results from many Hadoop instances locally run on distributed clusters (*Hierarchical* approach).

Geo-hadoop approaches (Kim et al., 2011; Matess et al., 2013; Heintz et al., 2014; Zhang et al., 2014) reconsider the phases of the job's execution flow (Push, Map, Shuffle, Reduce) in a perspective where data are distributed at a geographic scale, and the available resources (compute nodes and network bandwidth) are disomogeneous. In the aim of reducing the job's average *makespan*¹, phases and the relative timing must be adequately coordinated.

Hierarchical approaches (Luo et al., 2011; Jayalath et al., 2014; Yang et al., 2007) envision two (or sometimes more) computing levels: a bottom level, where several plain MapReduce computations occur on local data only, and a top level, where a central entity coordinates the gathering of local computations and the packaging of the final result. A clear advantage of this approach is that there is no need to modify the Hadoop algorithm, as its original version can be used to elaborate data on a local cluster. Still a strategy needs to be conceived to establish how to redistribute data among the available clusters in order to optimize the job's overall makespan.

¹The execution time of a job. It is measured from the time the job is submitted to the time results are gathered

The solution we propose belongs to the *hierarchical* category. We address the typical scenario of a big company which has many branches distributed all over the world producing huge amounts of business-sensitive data that need to be globally processed on demand. Examples of application domains that fall in this scenario are electronic commerce, content delivery networks, social networks, cloud service provisioning and many more. The Hadoop seems to offer the computing model that best suits this situation, because of its capability of providing parallel computation on multiple pieces of data. Unfortunately, company sites may be disomogeneous in terms of computing capabilities and the amount of stored raw data. Also, the inter-site network links have very limited and unbalanced bandwidth that is usually employed to support many types of inter-site communication. This makes the plain Hadoop unfit for the depicted scenario.

We believe a hierarchical computing model may help since it decouples the job/task scheduling from the actual computation. The approach we propose introduces a novel job scheduling algorithm which accounts for the discussed disomogeneity to optimize the job makespan. Basically, when a job is submitted, a top-level entity ("Orchestrator" in the remainder of the paper) will take care of serving the job. In particular, the job is split into a number of bottom-level, independent MapReduce sub-jobs that are scheduled to run on the sites where data reside. According to the original data localization, the computing capacity of involved sites and the available inter-site bandwidth, the Orchestrator may decide to migrate data (or pieces of them) from site to site before bottom-level MapReduce jobs are eventually started. Finally, the results of MapReduce sub-jobs are forwarded to a top-level Reducer that will package and deliver the final result. Unlike previous works, our job scheduling algorithm aims to exploit fresh information continuously sensed from the distributed computing context (available site's computing capacity and inter-site bandwidth) to guess each job's optimum execution flow.

3 DESIGN OVERVIEW

According to the MapReduce paradigm, a generic computation is called *job* (Dean and Ghemawat, 2004). A generic job is submitted to a *scheduling system* which is responsible for splitting the job in several *tasks* and mapping tasks to a set of available machines within a cluster. The performance of a job is measured by its completion time (some refers to it with the term

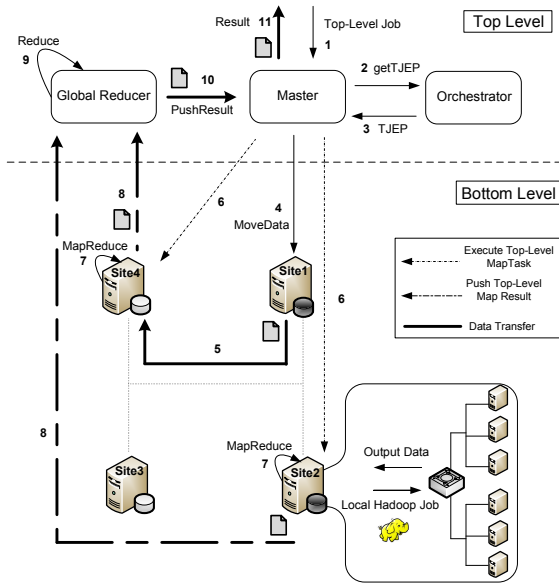


Figure 1: Overall architecture.

makespan), i.e., the time for a job to complete. That time heavily depends on the job's *execution flow* determined by the scheduling system and the computing power of the cluster machines where the tasks are actually executed.

In a scenario where computing machines belong to many geographically distributed clusters there is an additional parameter that may affect the job performance. Communication links among clusters (inter-cluster links) are often disomogeneous and have a much lower capacity than the communication links among machines within a cluster (intra-cluster links). Basically, if a scheduling system does not account for the unbalanced capacity of both machines and communication links, the overall job's performance may degrade dramatically.

The key point of our proposal for a hierarchical MapReduce programming model is the need of a top-level scheduling system which is aware of the underlying computing context's heterogeneity. We argue such awareness has to be created and augmented by periodically "sensing" the bottom-level computing context. Information retrieved from the computing context is then used to drive the generation of the particular job's execution flow which maximizes the job performance.

In Figure 1 the basic reference scenario addressed by our proposal is depicted. Sites (datacenters) populate the bottom level of the hierarchy. Each site stores a certain amount of data and is capable of running plain Hadoop jobs. Upon receiving a job, a site transparently performs the whole MapReduce process chain on the local cluster(s) and returns the result of

the elaboration. All the system business logic devoted to the management of the geo-distributed parallel computing resides in the top-level of the hierarchy. Upon the submission of a Hadoop job, the business logic schedules the set of sub-jobs to be disseminated in the distributed context, gathers the sub-job results and packages the overall computation result.

In particular, the system business logic is composed of the following entities:

- **Orchestrator.** It is responsible for the generation of a *Top-level Job Execution Plan* (TJEP). A TJEP contains the following information:
 - the Data Logistic Plan (DLP), which states how data targeted by the job have to be re-organized (i.e., shifted) among sites;
 - the Sub-job Scheduling Plan (SSP), which defines the set of Hadoop sub-jobs to be submitted to the sites holding the data.
- **Master.** It is the entity to which Hadoop Jobs are submitted. It calls on the Orchestrator for the generation of the TJEP, and is in charge of enforcing the TJEP according to the information contained in the DLP and the SSP.
- **Global Reducer.** It performs the top-level reduction of the results obtained from the execution of Hadoop sub-jobs.

At design time two important assumptions were made. First, at the moment only one Global Reducer is responsible for collecting and reducing the data elaborated by bottom-level sites. One may argue this choice impacts on the overall performance, nevertheless it does not invalidate the approach. Anyway, this assumption is going to be relaxed in future work. Second, being this approach a pure hierarchical approach, the top-level MapReduce job must be coded in such a way that the applied operations are "associative", i.e., may be performed recursively at each level of the hierarchy and the execution order of the operations does not affect the final result (Jayalath et al., 2014).

In the scenario of Figure 1 four geo-distributed sites are depicted that hold company's business data sets. The numbered arrows describe a typical execution flow triggered by the submission of a top-level job. This specific case envisioned a shift of data from one site to another one, and the run of local MapReduce sub-jobs on two sites. Here follows a step-by-step description of the actions taken by the system to serve the job:

1. A Job is submitted to the Master, along with the indication of the data set targeted by the Job.
2. The Master forwards the Job request to the Orchestrator, to get the TJEP;

3. The Orchestrator elaborates and delivers a TJEP. For the elaboration of the plan the Orchestrator makes use of information like the distribution of the data set among sites, the current computing capabilities of sites, the topology of the network and the current capacity of its links. A TJEP is broken down in two section: 1) the DLP containing data-shift directives and 2) the SSP containing data-elaboration directives;
4. The Master enforces the DLP. In particular, *Site1* is ordered to shift data to *Site4*;
5. The actual data shift from *Site1* to *Site4* takes place.
6. The Master enforces the SSP. In particular, top-level Map tasks are triggered to run on *Site2* and *Site4* respectively. We remind that a top-level Map task corresponds to a Hadoop sub-job;
7. *Site2* and *Site4* executes local Hadoop jobs on their respective data sets;
8. Results obtained from local execution are sent to the Global Reducer;
9. The Global Reducer performs the reduction of partial data;
10. Final result is pushed to the Master;
11. Final result is returned to the Job submitter.

One of the Orchestrator's tasks is to monitor the distributed context's resources, i.e., the sites' available computing capacity and the inter-site bandwidth capacity. In Figure 2 the context monitoring infrastructure is depicted.

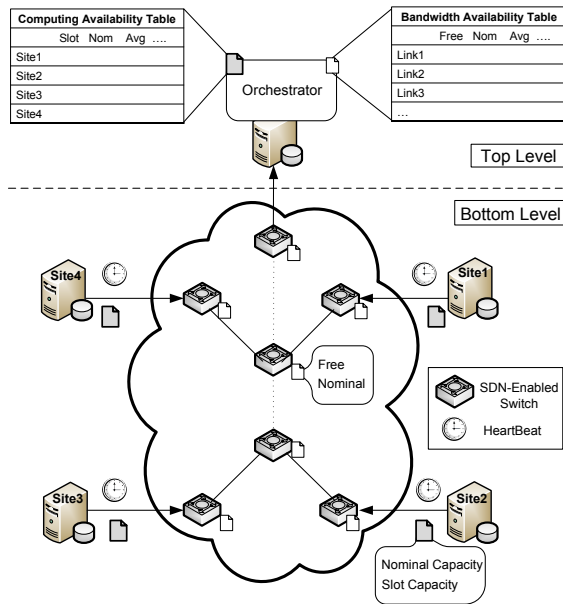


Figure 2: Context monitoring infrastructure.

As for the monitoring of the computing capacity, each site periodically advertises its capacity to the Orchestrator. Such capacity is expressed in teraFlops, and represents the overall computing capacity of the site for MapReduce purposes (overall nominal capacity). Further, we assume that sites enforce a computing capacity's allocation policy which reserves a given, fixed amount of capacity to any submitted MapReduce job. Since the amount of computing capacity potentially allocable to a single job (slot capacity) may differ from site to site, sites are also required to communicate that amount along with the overall nominal capacity. By using this information, the Orchestrator is able to build and maintain a *Computing Availability Table* that keeps track of every site's instant and future capacity, average capacity in time, and other useful historical statistics about the computing capacity. The available inter-site link capacity is instead "sensed" through a network infrastructure made of SDN-enabled (Open Networking Foundation, 2012) switches. Switches are capable of measuring the instant bandwidth occupied by incoming and outgoing data flows. The Orchestrator periodically enquires the switches to retrieve the bandwidth occupation that is then fed to a *Bandwidth Availability Table*, where statistics on the inter-site bandwidth occupation are reported.

Information contained in these two tables are extremely useful to the Orchestrator when it comes to elaborate an execution plan for a submitted job. The awareness of the underlying distributed computing context will guide the Orchestrator in defining a path which minimizes the overall job's makespan. The search for the path is committed to a scheduling system that is embedded in the Orchestrator. In the following section, details on the strategy implemented by the scheduling system are disclosed.

3.1 Job Scheduling System

Basically, the goal of the job scheduling system is to generate a number of possible execution paths, and to give each path a score. The path with the best score will eventually be chosen as the execution path to enforce. The calculation of the score for a given path consists in the estimation of the path's completion time; in the end, the path exhibiting the lowest completion time (best score) will be selected.

If it may appear clear that the sites' computing capacity and the inter-site bandwidth affect the overall path's completion time, some words have to be spent on the impact that the type of MapReduce application may have on that time. In (Heintz et al., 2014) authors introduce the expansion/compression factor α , that

represents the ratio of the output size of the Map phase to its input size. In our architecture focus is on the entire MapReduce process (not just the Map phase) that takes place in a site. Therefore we are interested in profiling applications as a whole. We then introduce a data **Compression factor** β_{app} , which represents the ratio of the output data size of an application to its input data size:

$$\beta_{app} = \frac{OutputData_{app}}{InputData_{app}} \quad (1)$$

The β_{app} parameter may be used to calculate the amount of data that is produced by a MapReduce job at a site, traverses the network and reaches the Global Reducer. Based on that amount, the data transfer phase may seriously impact on the overall top-level job performance. The exact value for β_{app} may not be a priori known (MapReduce is not aware of the application implementation). Section 3.2 will present an approximate function that provides a good estimate.

We adopt a graph model to represent the job's execution path. Basically, a graph node may represent either a data computing element (site) or a data transport element (network link). Arcs between nodes are used to represent the sequence of nodes in an execution path. A node is the place where a data flow arrives (input data) and another data flow is generated (output data). A node representing a computing element elaborates data, therefore it will produce an output data flow whose size is different than that of input; a node representing a data transport element just transports data, so input and output data coincide. Nodes are assigned an attribute that describes the **Throughput**, i.e., the rate at which node is capable of "processing" the input data. In the case of a *computing* node the throughput represents the speed at which the application's input data are actually processed, whereas in the case of a *transport* node the throughput coincides with the link capacity. Actually, the throughput of a *computing* node is the rate at which the node is capable of "processing" data when running that specific application. This parameter is strictly application bound, as it depends on how heavy is the type of computation requested by the application. Like for the β_{app} value, the exact Throughput value is not a priori known; Section 3.2 discusses a sample based procedure employed to derive the throughput of a *computing* node for a certain application.

Nodes are also characterized by the β_{node} attribute, which in the case of a *computing* node is an application-dependent parameter measuring the ratio between input data and output data (β_{app}), while in the case of a *transport* node it will assume the fixed

value 1 (in fact, a network link applies no data compression).

Arcs between nodes are labeled with a number representing the size of data leaving a node and reaching the next one. The label value of the arc connecting the j -th node with the $(j+1)$ -th node is given by:

$$DataSize_{j,j+1} = DataSize_{j-1,j} \times \beta_j \quad (2)$$

In Figure 3 an example of a branch made of two nodes and a connecting arc is depicted:

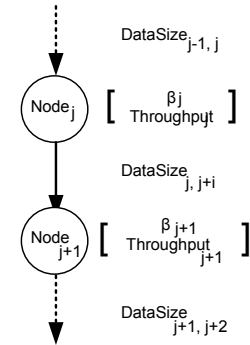


Figure 3: Nodes' data structure.

Next, for the generic node j we define the execution time as:

$$T_j = \frac{DataSize_{j-1,j}}{Throughput_j} \quad (3)$$

When a top-level job is submitted to the Master, the scheduling system is requested to search for the best execution path. The hard part of the scheduling system's work is the generation of all the potential execution paths, each of which is going to be modeled as a graph. The algorithm used to generate execution paths is discussed in Section 4. We now put the focus on how to calculate the execution time of a specific execution path.

Figure 4 depicts a scenario of seven distributed sites (S_1 through S_7) and a geographic network which interconnects the sites. One top-level job is requesting to run a MapReduce application on the data sets (5 GB sized each) located in the site S_5 and S_6 respectively. Let us assume that one of the execution-paths generated by the scheduling system involves the movement of data from S_6 site to S_3 , which will perform the bottom-level MapReduce sub-job. Data placed in site S_5 , instead, will be processed by the site itself; this case does not require any data transfer. The Global reduce of the partial results produced by local MapReduce sub-jobs will be executed in the node S_1 (so partial results will have to move to that site before the reducing occurs).

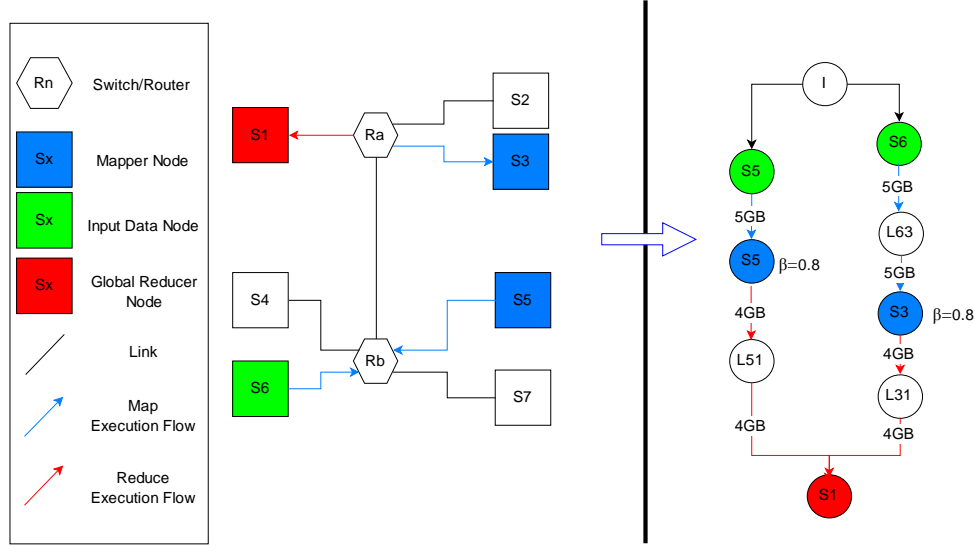


Figure 4: Example of graph modeling an execution path.

In the right part of the picture the graph that models the execution-path for the just discussed configuration is represented. Basically, a graph has as many branches as the number of bottom-level MapReduce. Branches are independent from each other's and execute in parallel. Every branch starts at the node I (initial node) and ends at the Global reducer's node. Next to node I is the node where the data set interested by the MapReduce computation initially resides. In the example, the graph is composed of two branches. The left branch models the elaboration of data initially residing in the node S_5 , that are map-reduced by node S_5 itself, and results are finally pushed to node S_1 (the Global reducer) through link L_{51} . Similarly, on the right branch data residing in node S_6 are moved to node S_3 through link L_{63} , are map-reduced by node S_3 and results are pushed to node S_1 through link L_{31} .

We define the execution time of a branch to be the sum of the execution times of the nodes belonging to the branch; note that the Global reducer node's execution time is left out of this sum. In particular, for the left and the right branches of Figure 4 the execution times will be respectively:

$$T_{left} = \frac{5}{Throughput_{S_5}} + \frac{4}{Throughput_{L_{51}}}$$

$$T_{right} = \frac{5}{Throughput_{L_{63}}} + \frac{4}{Throughput_{S_3}} + \frac{4}{Throughput_{L_{31}}}$$

So in general, the execution time of a branch is expressed as:

$$T_{branch} = \sum_{j=1}^{N-1} \frac{DataSize_{j,j+1}}{Throughput_{j+1}} \quad (4)$$

being N the number of nodes in the branch.

Next we calculate the execution time for the Global reducer node. The data pushed to that node is the sum of the data coming from the two branches. In the example, the execution time is given by:

$$T_{GR} = \frac{4 + 4}{Throughput_{S_1}}$$

So generalizing, the execution time of the Global reducer is given by the summation of the sizes of the data sets coming from all the branches over the node's estimated throughput. Let $DataSize(K)_{N-1,N}$ be the data size of the k -th branch reaching the Global reducer node. The execution time for the Global reducer will be:

$$T_{GR} = \frac{\sum_{K=1}^P DataSize(K)_{N-1,N}}{Throughput_{GR}} \quad (5)$$

being P the total number of branches in the graph. Finally, the overall execution time estimated for the specific execution path represented by the graph is defined as the sum of the Global reducer's execution time and the maximum among the branches' execution times:

$$T_{path} = \max_{1 \leq K \leq P} (T(K)_{branch}) + Throughput_{GR} \quad (6)$$

In this formula we are assuming that the global reduce phase will start as soon as the slowest (i.e., the one with the highest execution time) branch has finished its execution.

The scheduling system can generate many job's execution paths. For each, the execution time is calculated. In the end, the best to schedule will be, of course, the one showing the lowest execution time.

3.2 Application Profiling

As mentioned earlier, both the computing node's Throughput, and the Compression factor β_{app} are two parameters strictly dependent on the type of application requested by the top-level job. The estimate of these parameters is determined by an application profiling procedure executed prior to the run of the job on the requested data. The adopted approach, that recalls the one proposed in (Jayalath et al., 2014), is to request sites that hold the data sets to run the job's application on a sample of data. The results will provide an estimate of the parameters that will be used by the scheduling system to calculate the best execution path for the job.

The estimate is performed on a reference machine having a computing power of 1 Gflops. Regarding the Throughput, the objective is to evaluate the nominal capability of a 1 Gflops machine to process the data sample. So, the *nominal Throughput* is obtained by dividing the sample data size by the data processing time; the *nominal* β_{app} , as well, is given by the ratio between the output result size and the input sample data size.

The nominal values obtained from the sites are adequately averaged, and will constitute the official estimate parameters for that specific application. In particular, when it comes to calculate the Throughput of a certain computing node of the graph (representing a site), that value is calculated by multiplying the *nominal Throughput* times the number of Gflops advertised by the node. This estimate makes the assumption that the Throughput is a linear function of the computing power.

4 EXECUTION PATHS GENERATION

The scheduling system is in charge of generating a number of potential execution paths for each top-level job that is submitted. The variables that impact on the generation of paths are the number of sites devoted to the running of MapReduce and the amount of data each of those sites will be assigned. The number of potential paths may be very huge (and thus very hard to compute in an acceptable time) if you consider that data sets targeted by an application might be fragmented at any level of granularity, and each fragment

might potentially be moved to any of the available sites for bottom-level computation.

We now formulate the problem of data fragmentation and discuss the combinatorial approach we adopted to generate the execution paths. Let us assume that n , m and D be the number of nodes, the number of mappers and the Application data size respectively. In order to limit the number of potential paths, the basic assumption we make is that all data fragments must have the same size, and that the number of data fragments must be equal to the number of sites available for computation ($N_{frag} = n$). The resulting fragment size will then be:

$$Frag_{size} = \frac{D}{n} \quad (7)$$

A node may be assigned zero, one or more fragments to work on. Our algorithm will schedule which nodes have to be appointed top-level mappers and how many data fragments to assign each Mapper. In order to generate all possible combinations of mappers and the related assigned data fragments, we leverage on the combinatorial and on the partition number theory (Andrews, 1976).

By the notation $P(n, m)$ we refer to the number of partitions of the integer number n in the order m , where m is the number of addends in which n is to be partitioned. For instance, $P(5, 2)$ is the number of partitions of the number 5 in 2 addends. It is easy to understand that $P(5, 2) = 2$ (being the two combinations $1 + 4$ and $2 + 3$). If we had to partition the number 5 in 3 addends we would obtain $P(5, 3) = 2$ (combinations $1 + 2 + 2$ and $1 + 3 + 1$). We are going to use this technique to guess the number of possible ways the data of an application may be partitioned into a bunch of fragments. So, in the case that we have 5 data fragments to distribute over 2 sites, two configurations are possible: 1) 1 fragment on one site, 4 fragments on the other one; 2) 2 fragments on one site, 3 on the other one. Generalizing, the overall number of partitions of a number n in all the orders $m=1, 2, \dots, n$ is:

$$P(n) = \sum_{m=1}^n P(n, m) \quad (8)$$

Of course, the fragment configuration tell us just the ways to "group" fragments for distribution, but the distribution phase complicates the problem, as there are many possible ways to distribute group of fragments among sites. In the example concerning the $P(5, 2)$, 1 fragment may go to *mapper1* (in *site1*) and 4 fragments may go to *mapper2* (in *site2*), or viceversa. So for the distribution of fragments we have to call on the partial permutation theory. The number of possible ways of placing m mappers in n nodes is:

$$D_{n,m} = \frac{n!}{(n-m)!} \quad (9)$$

In the end, the calculus of the **number of all the execution paths** for a certain application has to consider both the fragment distribution configuration (eq. 8) and the partial permutation of mappers (eq. 9):

$$N_{exepath} = \sum_{m=1}^n P(n,m) \times \frac{n!}{(n-m)!} \quad (10)$$

For example, in the case of $n=7$ the number of generated paths will be around 18.000. For $n=8$ more than 150.000 configurations were obtained. Treating the problem of the generation of execution paths as an integer partitioning problem allowed us to apply well known algorithms working in constant amortized time that guarantee acceptable time also on off-the-shelf PCs (Zoghbi and Stojmenovic, 1994). For each configuration generated by the algorithm, a corresponding graph is built. On each graph's node, parameters (computing capacity, link capacity, β) are then assigned. Finally the graph's execution time is computed.

5 CONCLUSION

The increasing rate at which data grow have stimulated through the years the search for new strategies to overcome the limits showed by legacy tools that have been used so far to analyze data. MapReduce, and in particular its open implementation Hadoop, has attracted the interest of both private and academic research as the programming model that best fit the need for coping with big data. In this paper we address the peculiar need to handle big data which by their nature are distributed over many sites geographically distant from each other. Plain Hadoop was proved to be inefficient in that context. We propose a strategy which inspires to hierarchical approaches prior presented in other literature's works. The strategy leverages on the partition number and the combinatorial theory to partition big data into fragments and efficiently distributes the workload among datacenters. With respect to previous works, this exploits fresh context information like the available computing and the inter-site link capacity.

REFERENCES

Andrews, G. E. (1976). *The Theory of Partitions*, volume 2 of *Encyclopedia of Mathematics and its Applications*.

- Dean, J. and Ghemawat, S. (2004). MapReduce: simplified data processing on large clusters. In *OSDI04: Proceeding of the 6th Conference on Symposium on operating systems design and implementation*. USENIX Association.
- Facebook (2012). Under the Hood: Scheduling MapReduce jobs more efficiently with Corona. <https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona>.
- Heintz, B., Chandra, A., Sitaraman, R., and Weissman, J. (2014). End-to-end Optimization for Geo-Distributed MapReduce. *IEEE Transactions on Cloud Computing*, PP(99):1–1.
- Jayalath, C., Stephen, J., and Eugster, P. (2014). From the Cloud to the Atmosphere: Running MapReduce across Data Centers. *IEEE Transactions on Computers*, 63(1):74–87.
- Kim, S., Won, J., Han, H., Eom, H., and Yeom, H. Y. (2011). Improving Hadoop Performance in Intercloud Environments. *SIGMETRICS Perform. Eval. Rev.*, 39(3):107–109.
- Luo, Y., Guo, Z., Sun, Y., Plale, B., Qiu, J., and Li, W. W. (2011). A Hierarchical Framework for Cross-domain MapReduce Execution. In *Proceedings of the Second International Workshop on Emerging Computational Methods for the Life Sciences*, ECMLS '11, pages 15–22.
- Mattess, M., Calheiros, R. N., and Buyya, R. (2013). Scaling MapReduce Applications Across Hybrid Clouds to Meet Soft Deadlines. In *Proceedings of the 2013 IEEE 27th International Conference on Advanced Information Networking and Applications*, AINA '13, pages 629–636.
- Open Networking Foundation (2012). Software-Defined Networking: The New Norm for Networks. White paper, Open Networking Foundation.
- The Apache Software Foundation (2011). The Apache Hadoop project. <http://hadoop.apache.org/>.
- Yang, H., Dasdan, A., Hsiao, R., and Parker, D. S. (2007). Map-reduce-merge: Simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 1029–1040.
- Zhang, Q., Liu, L., Lee, K., Zhou, Y., Singh, A., Mandagere, N., Gopisetty, S., and Alatorre, G. (2014). Improving Hadoop Service Provisioning in a Geographically Distributed Cloud. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, pages 432–439.
- Zikopoulos, P. and Eaton, C. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw Hill.
- Zoghbi, A. and Stojmenovic, I. (1994). Fast algorithms for generating integer partitions. *International Journal of Computer Mathematics*, 80:319–332.

CLOUD COMPUTING ENABLING TECHNOLOGY

FULL PAPERS

Secure Keyword Search over Data Archives in the Cloud

Performance and Security Aspects of Searchable Encryption

Christian Neuhaus¹, Frank Feinbube¹, Daniel Janusz² and Andreas Polze¹

¹*Operating Systems and Middleware Group, Hasso Plattner Institut, Potsdam, Germany*

²*DBIS Group, Humboldt-Universität zu Berlin, Berlin, Germany*
firstname.lastname@hpi.de, janusz@informatik.hu-berlin.de

Keywords: Keyword Search, Searchable Encryption, Cloud Computing, Performance, Security, Data Confidentiality.

Abstract: Encryption can protect the confidentiality of data stored in the cloud, but also prevents search. To solve this problem, searchable encryption schemes have been proposed that allow keyword search over document collections. To investigate the practical value of such schemes and the tradeoff between security, functionality and performance, we integrate a prototypical implementation of a searchable encryption scheme into a document-oriented database. We give an overview of the performance benchmarking results of the approach and analyze the threats to data confidentiality and corresponding countermeasures.

1 INTRODUCTION

Data sharing is essential to companies and government services alike. A striking example is healthcare, where doctor's offices, hospitals, and administrative institutions rely on exchange of information to offer the best level of care and optimizing cost efficiency at the same time. For scenarios like these, moving to the cloud solves many problems: The scalability of the cloud makes resources simple to provision and extend and centralization of data improves the availability and helps to avoid information silos. Most importantly, cloud computing helps to reduce IT expenses – an effect most welcome in healthcare. However, concerns about data confidentiality still prevent the use of cloud in many domains. Traditional encryption is of little help: It effectively protects the privacy of data but also prevents important operations such as search. While efficient encryption schemes that enable generic operations on encrypted data are still elusive, searching over encrypted data is possible: searchable encryption schemes enable keyword search without disclosing these keywords to the cloud operator. The query performance of such schemes cannot match unencrypted operation, but may well be suitable for areas of application such as electronic health records, where data has to be retrieved from a cloud-hosted archive.

In this paper, we investigate the trade-off between performance and security when using searchable encryption schemes for data archives in the cloud. We

make the following contributions:

- 1) We report on an architecture for integrating Goh's Z-INDEX searchable encryption scheme (Goh et al., 2003) into a database and present a practical implementation by the example of MongoDB.
- 2) We discuss the overhead introduced by encrypted search and provide benchmark results on the performance of using Goh's scheme for encrypted search with MongoDB. These benchmarks give a meaningful account of the practical performance and usability of searchable encryption in databases.
- 3) We give a qualitative assessment of the security implications of using searchable encryption schemes for cloud data archives using attack-defense-tree models. This assessment is generic to searchable encryption and not limited to Goh's scheme. We also discuss mitigation strategies to manage threats by statistical inference attacks.

2 RELATED WORK

In this section, we review related work in the field of private database outsourcing and searchable encryption.

Private Database Outsourcing. Outsourcing private data to a remote database inherently bears the risk of exposure of confidential information – through eavesdropping, data theft or malfunctions. The key

challenge is to protect private data from being accessed by potentially untrusted cloud providers. In this paper, we focus on technologies that protect data within a database. While encryption is the basic mechanism to ensure data confidentiality, providing an efficient database-as-a-service that can run on encrypted data is a challenging task. Several recent approaches try to offer solutions for outsourcing private databases.

TrustedDB (Bajaj and Sion, 2011) and Cipherbase (Arasu et al., 2013) offer SQL database functionalities that support the full generality of a database system while providing high data confidentiality. Both systems use a secure co-processor for performing operations on the cloud server side. The drawbacks of such approaches are at least twofold: On one hand all clients have to trust the secure co-processor with their private data. On the other hand it is not clear how the co-processor scales up in the number of clients connected and the amount of data processed. In CryptDB (Popa et al., 2011), the authors apply an layered approach that makes use of several cryptographic schemes, where values are only decrypted to a level that is required to complete the query.

Another class of approaches aims at processing encrypted data directly without any decryption. To this day, there are no efficient encryption schemes that enable fully encrypted operation of a DBMS (database management system) without loss of functionality. An early approach for keyword search on encrypted data was published by (Song et al., 2000). An approach for securely processing exact match queries on database cells was proposed by (Yang et al., 2006). However, most DBMS rely on other common operations such as range and aggregation queries as well as updates, inserts and deletes. Existing approaches cannot efficiently process this type of queries on encrypted data. A common solution is to reduce data confidentiality to gain query efficiency, e.g., order preserving encryption (Agrawal et al., 2004) may reveal the underlying data order. Most methods can be attacked by statistical analysis of the encrypted data or the access patterns. Another solution is to lose some query efficiency in order to guarantee confidentiality. While (*fully*) *homomorphic encryption schemes* as proposed by Rivest et al. (Rivest et al., 1978) in fact allow the encrypted computation of any circuit (and therefore computer program), current constructions (see (Gentry, 2009; Van Dijk et al., 2010)) are yet too inefficient for practical application.

Traditional databases use indices for efficient record search. The existing methods have been adapted to work on encrypted data (Shmueli et al.,

2005). Private indexing (Hore et al., 2004) enable an untrusted server to evaluate obfuscated range queries with limited information leakage. Wang et al (Wang et al., 2011) propose a secure B^+ -Tree to efficiently process any type of database query. Encrypted index-based approaches do not rely on any trusted third parties or trusted hardware. This seems to be a practical and secure method to search in encrypted databases. The next section discusses searchable encryption.

Searchable Encryption. Searchable encryption schemes provide one or many cryptographic data structures called *search indices* that allow encrypted keyword search for exact keyword matches. A good overview of searchable encryption schemes is given in (Kamara and Lauter, 2010). In general, searchable encryption schemes do not replace symmetric encryption schemes but provide the search capability through additional data structures – the *index* (see figure 1). To provide keyword search on data, a list

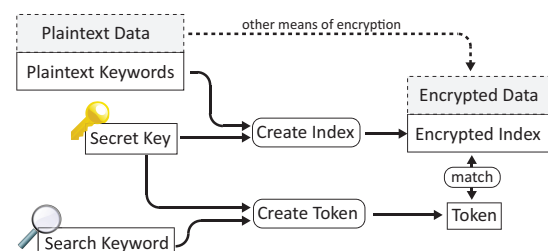


Figure 1: Searchable Encryption: Conceptual View.

of keywords is extracted from the plaintext. This keyword list is used to create a secure index using a dedicated secret key for the searchable encryption scheme. The data is encrypted separately (usually symmetric block ciphers such as AES) and uploaded stored alongside the encrypted index in a remote location. To search over the uploaded data in the remote location, a *search token* is generated for a search keyword using the secret key. This token is sent to the remote server. The remote server can now determine whether the token matches a search index without being able to learn the keyword.

Searchable encryption schemes can be distinguished between *Symmetric Searchable Encryption* (SSE) and *Asymmetric Searchable Encryption* (ASE) schemes. SSE schemes use the same secret key both for insertion and searching of data. In general, they are more efficient than ASE schemes and provide stronger security guarantees. They were first introduced by (Song et al., 2000), where the authors provide a linear search capability over ciphertext – one of the few schemes that does not make use of indices. To speed up search, the scheme of Goh (Goh et al.,

2003) uses indices that are created separately for every searchable data item, which enables efficient update. Improved search time can be achieved by using an inverted index (see e.g. (Curtmola et al., 2006)). A scheme that enables both efficient updates and optimal search time (linear in the number of documents that contain the keyword) is offered in a recent construction by Kamara et al. (Kamara et al., 2012).

In contrast, ASE schemes use different keys for insertion and searching of data, which provides greater flexibility. However, the constructions of ASE schemes are generally less efficient than those of SSE schemes and provide weaker security guarantees. The first construction was given by Boneh et al. (Boneh et al., 2004) and is based on elliptic curve cryptography. Improved constructions were introduced in (Abdalla et al., 2005). Unfortunately, ASE are generally susceptible to dictionary attacks against search tokens (see (Byun et al., 2006)). This limits the application of ASE schemes to use case where keywords are either hard to guess or the keyword attack is tolerable.

3 THE Z-IDX SCHEME

For our implementation, we chose the Z-IDX searchable encryption scheme by (Goh et al., 2003). As a symmetric scheme, it is not susceptible to dictionary attacks on search tokens like ASE schemes (see section 2). This scheme offers several desirable properties:

- **Maturity.** While the field of research in searchable encryption schemes is rather young, Goh's scheme was one of the earliest proposed. In contrast to more recent constructions, the scheme passed several years without the discovery of security flaws.
- **Per-document Indexing.** The Z-IDX scheme creates per-document indices. This property facilitates integration into existing DBMS.
- **Standard Cryptographic Primitives.** The cryptographic mechanisms used by Z-IDX are widely available in software libraries for most platforms.

In this section, we give an overview of *Bloom Filters* and how they are used to construct Gohs Z-IDX scheme.

3.1 Bloom Filters

The encrypted indices in Z-IDX make use of space-efficient probabilistic data structures called *bloom filters* (Bloom, 1970). For a set of elements $E =$

$\{e_1, \dots, e_n\}$, the set membership information is encoded in a bit array of length l . A number of r hash functions h_1, \dots, h_r is selected that map every element of E to a number $\in [1; l]$. To store the set membership of an element e_x in the filter, its hash value from every hash function h_1, \dots, h_r is calculated. These hash values $h_1(e_x), \dots, h_r(e_x)$ are used as index positions in the filter bit array. At every referenced index position, the bit in the array is set to 1. To test the set membership for an element e_y , the procedure is similar: All hash values $h_1(e_y), \dots, h_r(e_y)$ are calculated and used as index positions in the filter bit array. If all positions in the array pointed to by the hash function values are set to 1, the element is assumed to be in the set.

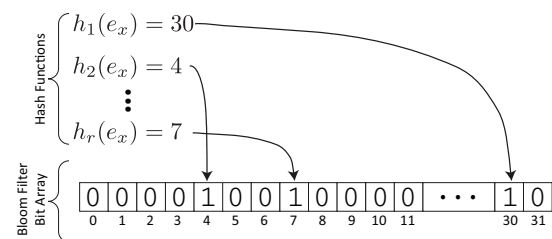


Figure 2: Example of a Bloom Filter with a 32-bit array.

This design of bloom filters can produce false positives: If all corresponding array positions of an element e_z were set to 1 by insertion of other elements, the bloom filter produces a false positive for e_z . On the other hand, false negatives do not occur. The false positive rate of a bloom filter can be influenced by adjusting the size of the bit array and the number of hash functions used.

3.2 Gohs Secure Indexes

Based on bloom filters, Goh constructs a secure index scheme called Z-IDX (Goh et al., 2003) that allows encrypted keyword search. Like similar schemes, it does not replace other means of encryption but provides additional data structures for its functionality (see figure 1). The scheme builds upon the abstraction of *documents*, which are the units of granularity for keyword search. Every document $d_i \in D$ can contain a number of keywords $w \in W$ and is identified by a unique ID $i \in I$. Authorized clients hold a secret key K_{priv} . The scheme is then defined by the following operations:

- **Keygen(s)** outputs a secret key K_{priv} , where s is a variable security parameter.
- **Trapdoor(K_{priv}, w)** outputs a trapdoor T_w for keyword w using the secret key K_{priv} .
- **BuildIndex(d, K_{priv})** outputs an encrypted index for document d using the secret key K_{priv} .

- $\text{SearchIndex}(T_w, d)$ takes a trapdoor for keyword w and tests for a match in the index of document d . If d contains w it outputs 1 and 0 otherwise.

Additionally, a pseudorandom function $f : \{0, 1\}^* \times \{0, 1\}^s \rightarrow \{0, 1\}^m$ is required. For a precise formal definition, e.g. with respect to bit string lengths, please see the original publication (Goh et al., 2003). We also omit the step of adding *blinding bits* to the filter. To set up the scheme, security parameter s , a number of hash functions r and a index size m are chosen (for choice of m and r , see section 5.1). Then, a secret key is generated by the Keygen operation, so that $K_{\text{priv}} = (k_1, \dots, k_r) \leftarrow \{0, 1\}^{sr}$.

To create a search index for a document d with a set of keywords $W_d = \{w_1, \dots, w_x\} \subset W$, BuildIndex operation first creates an empty bloom filter with a bit array of length m . First, a trapdoor T_w is calculated for every keyword w using the Trapdoor operation, so that $T_w = (t_{w_1}, \dots, t_{w_r}) = (f(w, k_1), \dots, f(w, k_r))$. This results in a set of trapdoors: Using the set of trapdoors T_{w_1}, \dots, T_{w_x} and the id of the document d , the set of codewords C_{w_1}, \dots, C_{w_x} is calculated. For every trapdoor T_w the codeword C_w is calculated so that $C_w = (c_{w_1}, \dots, c_{w_r}) = (f(id, t_{w_1}), \dots, f(id, t_{w_r}))$. Then, the filter of the document is populated by setting every bit position i to 1 that is referenced by the trapdoors: For every trapdoor C_w , the bits at positions c_{w_1}, \dots, c_{w_r} are set to 1 (see figure 2).

To query a collection of documents for a keyword w , the trapdoor T_w is calculated using the Trapdoor operation and sent to the server. To test whether a document contains the keyword, the server calculates the codeword C_w using the trapdoor T_w and the document id . Using the trapdoor C_w the server tests whether all bit at positions c_{w_1}, \dots, c_{w_r} are set to 1. If so, the document is sent back to the client as a match. This process is applied to all documents in the collection. In the Z-IDX scheme, a separate index data structure is created per document. This accounts for a search time that is linear over the number of documents, but facilitates the administration of secure indices, as they can be created stored alongside the documents. This makes the addition or removal of documents a simple operation.

From a more technical perspective, the above steps can be described and implemented using a keyed hash function such as HMAC-SHA1 (Krawczyk et al., 1997), which is also used in our implementation (see section 4). In a first step, a keyword w is hashed with all elements of the secret key k_1, \dots, k_r to obtain the trapdoor vector. The elements of the trapdoor vector are each hashed again together with the document identifier id to obtain the codeword vector. Each of

the codeword vector elements is used as an index position to set a bit in the bloom filter bit array to 1.

4 SEARCHABLE ENCRYPTION IN MongoDB

To evaluate the practical usability of searchable encryption, we integrated the Z-IDX scheme into the document oriented database *MongoDB*. In this section, we explain why we chose MongoDB, present the architecture of our prototype, introduce new commands for secure keyword search and present implementation details.

4.1 Selection of a Database System

While the searchable encryption scheme Z-IDX can be used standalone, its practical usability and performance under realistic workloads can only be evaluated if the scheme is used in conjunction with other means of encryption and data handling. To do this, we integrated Z-IDX into an existing DBMS. The choice of a DBMS has to correspond to the basic properties of the Z-IDX scheme – exact keyword matching as a search mechanism and the notion of *documents* as the basic units of granularity for searching.

To select a DBMS, we considered different database paradigms: The most widespread type of databases are **relational databases** – most of them supporting the *Structured Query Language* (SQL). This type of database has a long development history and offers features such as transactional security, clustering techniques and master-slave-configurations to ensure availability. The SQL language allows detailed queries, where specific data fields in the database can be selected and returned based on complex criteria based on structure or data field values and logical combinations thereof. The expressive power of the SQL language goes far beyond simple keyword search. It is therefore difficult to isolate queries that can make use of searchable encryption. Additionally, the fine-grained selection of data fields does not correspond well to the document-oriented approach of searchable encryption.

Besides relational databases, other database types have been developed under the umbrella term of NoSQL databases. A very minimalistic approach are *key-value stores* (e.g. *Redis*, *Dynamo*): They omit many of the features known from SQL databases in favor of simplicity and performance. However, the complexity of data structures is severely limited. This makes storing documents and associated indices difficult or impossible.

Document-oriented databases, however, are well-suited to implement searchable encryption. As the name suggests, data is organized in containers called *documents* as opposed to tables in relational databases. These documents are the units of granularity for search operations and can contain complex data structures without adhering to a schema definition. As this approach corresponds well to the properties of searchable encryption schemes, we chose to add searchable encryption features to the open-source document-oriented database *MongoDB*.

Floratou et al. (Floratou et al., 2012) compare MongoDB to Microsoft SQL Server. They show that relational databases may have better query performance. However, MongoDB is optimized for storing data records across multiple machines and offers efficient load balancing, which makes it more suitable for cloud-based applications. Furthermore, the increasing use of NoSQL databases in real world applications lead to an increasing demand for enhancing these databases with privacy technologies such as searchable encryption.

4.2 Extended MongoDB Commands

As MongoDB is a document-oriented database, a *document* is the primary unit of abstraction for organization of data. A document does not adhere to a fixed schema and can store data in a JSON-like fashion of field-value pairs. Like in JSON, documents support a number of primitive data types (e.g. *integer*, *String*) and a data structures like arrays. All of these data structures can be nested. In addition to standard JSON, MongoDB can also store binary data in fields.

Documents in MongoDB are stored in *collections*, these, in turn, are stored in a *database*. The prime commands for data handling in collections are `insert()` and `find()`. They accept a document as a parameter. To make searchable encryption explicitly available, we introduced two additional commands:

- The `insertSecure()` can be used to insert documents into a collection using searchable encryption. Using this command, every array of strings in the document is removed and its content used as keywords. The contained strings are inserted into a Z-IDX filter or encrypted search. Every other datatype remains untouched.
- The `findSecure()` command triggers encrypted search over all documents of a collection. As a parameter, it takes a keyword embedded in a document, e.g.: `findSecure({keyword: 'foo'})`

4.3 Architecture and Implementation

To integrate searchable encryption into MongoDB, we chose to add the extended functionality to the server and the command line client. An overview of the architecture of MongoDB server and client is given in figure 3. In theory, it is possible to add searchable encryption to MongoDB modifying only the client but not the server. However, this leads to a disproportionately high increase in communication overhead as per-document operations would have to be carried out on the client, each requiring the transmission of the documents Z-IDX data structures.

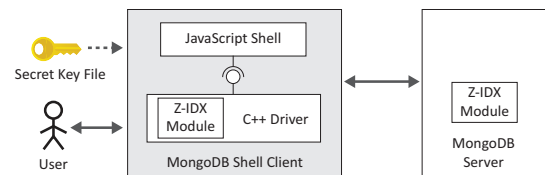


Figure 3: Architecture of MongoDB Server and Client.

The MongoDB command line client is comprised of a JavaScript shell that uses a core driver written in C++. The client connects to the server, which is also written in C++. To provide searchable encryption functionality, we implemented the Z-IDX scheme (see section 3) and additional helper functions in a separate module that is compiled both into the server and the C++ driver of the client (*Z-IDX Module*, see figure 3). As suggested by Goh, we apply data compression (zlib) to the index data structures before transmission over the network. As these data structures are very sparse, the compression works very effectively and the additional compute overhead is easily outweighed by reduced transmission times in most settings.

To integrate the functionality, we made the following modifications: The **JavaScript shell** is modified to read the secret key information from a file, which has to be passed as a parameter at startup. If a secure search or insert request is identified, the request is modified to include the secret key information. This information is stored in a dedicated `_zidx` field in the query. After this, the request is passed to the clients' C++ driver. The **C++ driver** is modified to recognize queries that contain Z-IDX key information injected by the JavaScript shell. For inserts, a Z-IDX filter is built and populated with the contained strings of every string array in the document. Subsequently, the string arrays and the key information are removed and the command is passed on to the server. For a search query, the C++ driver uses the key to compute trapdoors for every search keyword. The trapdoors are inserted, the key is removed and the query is passed

on to the server. The **MongoDB server** is modified to process the search queries. For the trapdoors of a search query, the server generates the corresponding codewords using the *document id*. These codewords are then checked against the bloom filters of a document to test for a match. This architecture and implementation makes searchable encryption available without affecting non-encrypted use of the database, as regular MongoDB commands are processed as expected.

5 PERFORMANCE EVALUATION

The use of encrypted search functionality introduces an overhead in computation, storage and data transmission. Since speed and throughput are critical factors for databases, we present performance measurements of our approach in this section. The figures allow to evaluate the practicability of searchable encryption in databases for real-life scenarios.

To assess the performance impact of our approach, we ran insert and search queries in encrypted and unencrypted settings under various parameters settings (dictionary size, false positive rate) and analyzed the memory footprint of the additional data structures of Z-IDX. To avoid synthetic test data, we chose the publicly available *Enron corpus* – a collection of emails which we use as documents. All benchmarks were run on a Intel Core i5-3470 machine with 8GB main memory, running Ubuntu 12.04 LTS.

5.1 Memory Footprint of Z-IDX Filters

As the encrypted filters are added to every document, they add overhead to communication and storage footprint. They are therefore a crucial factor that influences the performance of a database using this scheme.

The size of these data structures is determined by the desired false positive rate f_p and the number of unique keywords to be represented by the filters n . From the false positive rate f_p , the number of hash functions r is determined by calculating $r = -\log_2(f_p)$. From r , the number of bits m in the filter can be determined by calculating $m = nr/\ln 2$. In practice, these data structures can become quite large. This is especially unfavourable in settings with large numbers n of distinct keywords and small document sizes, as the filter sizes can easily exceed the size of the original documents.

To improve the efficiency of the scheme, data compression can be used on the filters (as suggested by Goh). While filter compression decreases storage

and communication overhead, it also introduces additional steps of computation on the client and server side: Upon document insertion, filters have to be compressed and decompressed for every search operation. This represents a tradeoff between data size and computational overhead.

To investigate this issue, we first tested the effectiveness of compression on indexes. In practice, these filters are bit array that contain mostly 0's and sparsely distributed 1's (depending on the number of contained keywords). To determine the achievable compression ratio, we used a set of 1000 documents from the Enron corpus containing 127.5 keywords on average. Assuming a set of 100000 distinct keywords and a false positive rate of 0.0001% leads to an uncompressed filter size of 252472 bytes. We implemented the compression of filters using the free *zlib*¹ compression library. Using the *zlib* standard compression strategy, the average compression ratio achieved is 0.02 with the given parameters. Using a run-length encoding strategy that exploits the sparse property of the filters, compression becomes even more effective with an average compression ratio of 0.0154. This means that using compression, filter sizes can be considerably reduced in size (here: to 1.54% of their original size, average size of compressed filters 3889 bytes).

Our benchmarking results show that using filter compression dramatically speeds up database operations even over fast network connections (100 Mbit/s speed). This means that the overhead for data compression is by far outweighed by the advantage in network transmission speed due to smaller filters. Therefore, we use RLE-based filter compression as a default in all subsequent measurements.

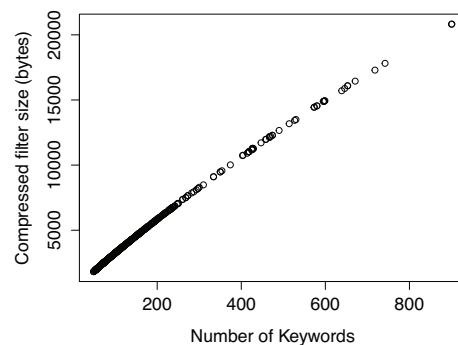


Figure 4: Relationship between number of document keywords and compressed filter size.

It can be observed that the size of compressed filters is closely correlated with the number of represented keywords (see figure 4): Documents with few

¹<http://zlib.net/>

keywords have small compressed filters while more keywords produce larger sizes. This means that a trade-off of the Z-IDX scheme is mitigated: To accommodate large sets of distinct keywords without false-positives, large filter sizes are required. These large filters take up of large amounts of memory – even for small documents with few or no keywords at all. However, using compression, filter sizes can be generously chosen as compressed filters remain compact, depending on the number of keywords in the document. In fact, using the settings above, compressed filter sizes are 3389 bytes on average. When increasing the number of unique keywords from 100000 to a million (tenfold), the average size is only 6648 bytes on average (only a twofold increase).

5.2 Query Performance

To assess the performance of the scheme, we evaluated insert and search performance of our Z-IDX implementation embedded in MongoDB. To obtain realistic results, we tested our setup under two different network profiles: The LAN profile corresponds to the typical properties of a wired local network (2ms ping, 100 Mbit/s), the WAN profile corresponds to the properties of a domestic internet connection in Germany (20ms ping, 10 Mbit/s). For reference, the same benchmarks were also conducted with a Localhost profile, where the network delays are essentially non-existent. The LAN and WAN profiles were generated by using network link conditioning on the machines' loopback network device, using Linux' `tc` command. All benchmarks were conducted using a false positive rate of 0.001 and a maximum dictionary size of 10000.

Insert Query Performance. To assess the performance of insert queries, we inserted a collection of 10000 documents from the Enron corpus in batches of 100. We ran every insert query 100 times and took the mean as our measurement value. The results for these queries in the Localhost, LAN and WAN profiles for encrypted and unencrypted operation are shown in figure 5. The longer duration of encrypted operation is explained by the additional steps required on the client: Before submission of a document, a Z-IDX filter has to be created using the document's keywords and the document content has to be encrypted. The Z-IDX filter introduces data which slightly increases the time of data transmission. On the server, no additional steps have to be executed on insert. Our experiments show that the performance penalty for encryption in insert queries is indeed moderate: In the Localhost- and LAN-settings, the insert time is about doubled

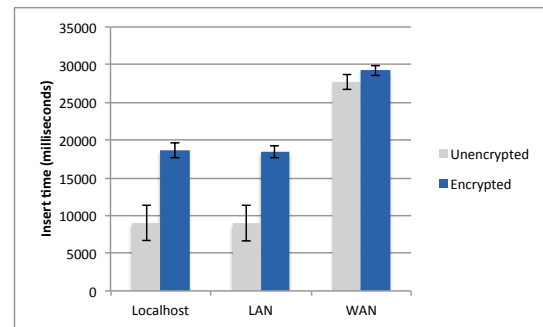


Figure 5: Benchmark: Insert of 10000 documents.

compared to the unencrypted setting. In the WAN setting, where network performance has a larger effect, the duration of encrypted and unencrypted insert queries are nearly the same.

Search Query Performance To determine the performance of search queries, we issued a search query with a randomly chosen keyword on the same document collection as used in the insert queries. We ran every search query 100 times and took the mean as our measurement value. The results for these queries in the Localhost, LAN and WAN profiles for encrypted and unencrypted operation are shown in figure 6. The duration of encrypted search queries

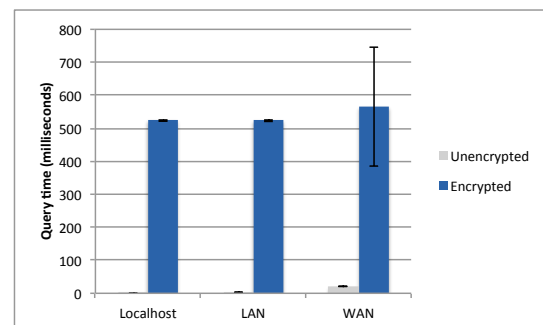


Figure 6: Benchmark: Query over 10000 documents.

is increased significantly compared to unencrypted operation, due to a fundamental difference between search implementation: Searching in an unencrypted database is usually carried out using an inverted index, where the matching documents for a given keyword can be looked up with linear complexity ($O(1)$). In encrypted operation using the Z-IDX-scheme, search complexity is linear in the number of documents in the collection ($O(n)$, n =number of documents). As a result, the unencrypted search time is very small (0,13 ms in the Localhost setting, 2,32 ms LAN, 20,37 ms WAN) when compared to encrypted operation and mainly determined by the network latency. In contrast, encrypted searches took around

half a second (≈ 530 ms), with little variation depending on network performance, as only little data had to be transmitted.

5.3 Implications for Practical Use

Our measurements have shown that the performance penalty for using the Z-IDX searchable encryption scheme in a database is very unevenly distributed: While the performance penalty for insert queries is almost negligible in under realistic conditions (WAN profile), the penalty for search queries is tremendous by comparison. At the same time, the query performance varies greatly depending on collection size (linear effort) and filter parameters: A search query on a 10000-documents-collection in our experiments took between 219 ms ($f_p = 0.01, n = 1000$) and 4612 ms ($f_p = 0.0001, n = 100000$).

6 SECURITY

The motivation for using searchable encryption schemes such as Z-IDX is to protect the confidentiality of information that is stored on untrusted infrastructures (e.g. cloud providers). In this section, we give a qualitative evaluation of the security implications when searchable encryption schemes are used to search over encrypted data stored on a remote server. This security evaluation is generally applicable to searchable encryption schemes that correspond to the abstract model given in section 6 and therefore not specific to Goh's Z-IDX scheme (Goh et al., 2003), unless explicitly noted otherwise.

The *security* of computer systems constituted by the attributes of confidentiality, integrity and availability (as defined in the ITSEC criteria (ITSEC, 1991), see also (Avizienis et al., 2004)). As the purpose of searchable encryption is to protect the searched keywords from being disclosed to unauthorized parties, we focus our evaluation on the property of data confidentiality of search keywords.

Abstract System Model. For the security evaluation, we assume a setup as shown in figure 7 (see also (Islam et al., 2012)). A server holds a set of n documents Doc_1, \dots, Doc_n . It also holds an encrypted data structure which contains a mapping for every keyword $w \in W$ to all documents containing w . To query the encrypted index, the client generates a trapdoor T_w and sends it to the server over the network. Using this trapdoor, the server can determine all documents that contain keyword w and sends them back to the client over the network. The mapping between keywords

and trapdoors $w \mapsto T_w$ is deterministic, i.e. under the same encryption key there exists exactly one trapdoor T_w for every keyword w . These properties apply to most symmetric searchable encryption schemes.

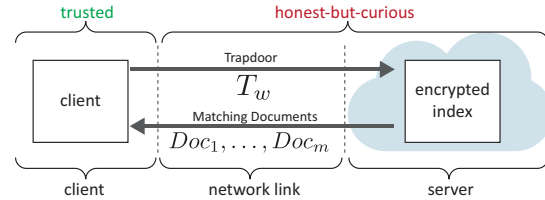


Figure 7: Encrypted Search on a remote system: Abstract Model.

Attacker Model. Attacks to learn the plaintext of keywords and their association with encrypted documents can generally be undertaken in any part of the architecture. Attacks on the client are the most dangerous, as clients hold the cryptographic key and handle unencrypted information. We assume authorized users on these clients to be trustworthy. For the operator of the network link and the server we assume a honest-but-curious attacker model (see e.g. (Lindell et al., 2008)): These operators will generally execute programs and transmit information correctly and faithfully, but can record arbitrary information and perform additional calculations on it. Under this adversarial model, data confidentiality is challenged while integrity and availability are not affected.

6.1 Threats to Keyword Confidentiality

To illustrate the threats to the confidentiality of keywords in the system we use the ADTree model (Attack-Defense-Trees, see (Kordy et al., 2012; Bagnato et al., 2012)), which build upon the concept of attack trees (Schneier, 1999). Attack trees are used to model the threats to a specific security property of a system and their logical interdependencies. Individual threats are represented as leaves of the tree and are connected by AND and OR operators to the root of the tree, which represents a specific security property. The attack of the system that corresponds to a specific threat is indicated in the model by assigning a boolean TRUE value of the node in the tree. If a combination of attacks results in a propagation of a TRUE value to the root node the security property is considered to be breached. By evaluating the attack tree, sets of possible attacks can be derived. The ADTree model extends attack trees by introducing and explicitly modeling countermeasures, which can be employed to mitigate or prevent attacks. In figure 8, an ADTree shows threats for keywords confidentiality in searchable encryption schemes and ac-

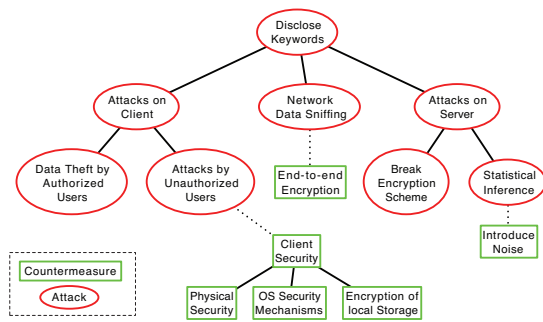


Figure 8: Attack-Defense-Tree: Threats for Confidentiality of Keywords.

cording countermeasures. Attacks to learn keywords can be undertaken on the client, on the network and the server which holds the encrypted index. In the following sections, we discuss the relevance and implications of the shown threats and their countermeasures.

6.2 Attacks on the Client

Attacks on the client are potentially severe as the client handles plaintext data and holds the cryptographic key for the searchable encryption scheme. By obtaining the key, an attacker can uncover document-keyword associations by generating valid queries and launching a dictionary attack against either the server or against intercepted trapdoors. Theft of data or keys cannot by authorized users cannot be prevented. However, in our attacker model, we assume the authorized users to be trustworthy. To protect the assets of the client systems against unauthorized users, different methods can be employed: Physical security measures can prevent unauthorized users from getting physical access to client machines. The security mechanisms of the clients operating system can ensure that only authorized users can log onto the machines directly or via network. Finally, data on the clients mass storage can be protected by hard disk encryption.

6.3 Network Data Sniffing

Interception of data exchanged by searchable encryption protocols could threaten the confidentiality of keywords as statistical properties of the trapdoor-keywords-associations can be exploited (for more detail, see section 6.4). If general security flaws of the underlying scheme become known, these could also be exploited. Data sniffing on the network can however easily be prevented by encryption of network traffic between client and server (e.g. by using *Transport Layer Security*).

6.4 Attacks on the Server

In general, threats that originate from network data sniffing also exist on the server, as the entire communication of the scheme is observable. However, as the searchable encryption scheme has to be processed on the server (i.e. matching of trapdoors to documents), an additional layer of encryption is not an option. In addition, the server also has direct access to the encrypted index, which could make attacks targeting this data structure very efficient. As the server can also monitor the program execution, side-channel attacks are theoretically possible (e.g. timing attacks). In the following, we discuss the implications of these threats.

Attacks to the Encryption Scheme. The confidentiality of the keywords depends on the trust in the chosen underlying searchable encryption scheme. In the first place, it is desirable to use algorithms that are openly published and examined by cryptographic experts. In general, searchable encryption schemes are an active field of research, with many constructions from the recent past (see section 2) that need more evaluation before they can be considered mature.

The Z-IDX scheme by Goh is among the oldest searchable schemes with no general attacks to the scheme published. The construction of the scheme is based on keyed hash functions, which are well examined and proved cryptographic tools (HMAC SHA-1 (Krawczyk et al., 1997)). The scheme fulfills three security properties suggested by (Song et al., 2000): It supports *hidden queries* as the generated trapdoors do not reveal the keyword. Valid trapdoors cannot be generated without possession of the secret key (*controlled searching*). Both properties are ensured by using a keyed hash function. Finally, the scheme fulfills the property of *query isolation* which means that the server learns nothing more than the set of matching documents about a query. This security property is formalized as the *IND-CKA* (Semantic Security Against Adaptive Chosen Keyword Attack) property: An adversary is given two documents D_0 and D_1 and an index which encodes the keywords of one of these documents. If the adversary cannot determine which documents keywords are encoded in the index with a probability significantly better than $\frac{1}{2}$ the index is considered *IND-CKA-secure*. To the best of our knowledge, no attacks that break IND-CKA-security of the Z-IDX scheme have been published to date.

Statistical Inference. Attacks using statistical inference are a possible against all searchable encryption schemes that follow the basic model outlined in

section 6. The threat of these attack is not based on weaknesses in the cryptographic constructions of searchable encryption schemes but is a direct consequence of the basic characteristics of such schemes. Under the same secret key K_{priv} , a keyword w is always mapped to the same trapdoor T_w . This allows the server to observe tuples $(w, \{D_1^w, \dots, D_m^w\})$, i.e. combinations of encrypted queries and the set of matching documents, which leak statistical information: The server can learn the frequency of certain queries as they occur over time and learn about the occurrence and frequency of distinct keywords in the document collection. While statistical information does not directly reveal keywords, it can be exploited to infer the semantics or plaintext of keyword using background knowledge about the data exchanged in the system. When handling medical data for example, very accurate assumptions about the prevalence of a specific medical condition among a population can be made using public sources of information. If this prevalence is expressed using a keyword and no other keyword in the document set possesses the same frequency, it is easy to infer the meaning of this keyword. While the given example might be trivial, statistical attack can pose a serious threat to the confidentiality of keywords. We review two practical attacks that have been published:

Search Pattern Leakage in Searchable Encryption: Attacks and New Constructions. (Liu et al., 2013) propose an attack based on the frequency of search patterns. The salient feature of the approach is that the frequency f_q at which a keyword q occurs is sampled over time, resulting in a frequency vector $V_q = \{V_q^1, \dots, V_q^p\}$ for a specific keyword. Background knowledge for a dictionary of keywords $D = \{w_1, \dots, w_m\}$ is drawn from external sources (the authors propose *Google Trends*) and represented as frequency vectors $V = \{V_{w_1}, \dots, V_{w_m}\}$. To infer the plaintext of a keyword, a distance measuring function $Dist(V, V_{w_i})$ is used to determine the vector $\in V$ with the smallest distance to V_q – the corresponding keyword is then assumed to be q . The attack is amended by an active approach, where the background knowledge is adapted to a specific scenario (e.g. healthcare) to improve accuracy. To test the accuracy of their attack, they use frequency vectors obtained from Google Trends for the 52 weeks of the year 2011 and add varying levels of gaussian noise to simulate user queries. They show that under certain circumstances (e.g. keyword dictionary size of 1000, limited level of noise) it is easy to guess the keyword with a very high accuracy. They also present mitigation strategies, which are based on inserting random keywords

along with every query, but do not consider the actual document matching on the server.

Access Pattern Disclosure on Searchable Encryption: Ramification, Attack and Mitigation. (Islam et al., 2012) propose a statistical attack which is based on the frequency at which keywords appear in the document set. As background knowledge, information about the probability of two keywords occurring in the same document is assumed. This information can be obtained by scanning public document sources for a dictionary keywords k_1, \dots, k_m . It is represented by a $m \times m$ matrix M , where $M_{i,j}$ contains the probability of keywords k_i and k_j occurring in the same document. The attacker then tries to find an order of encrypted queries q_1, \dots, q_m whose results set produce another matrix which is similar to M . This sequence that produces the matrix most similar to M is considered the result of the attack and reveals keywords by aligning the vectors of queries and keywords so that q_x corresponds to m_x . The problem can be formalized by expressing the closeness between matrices as an arithmetic distance. The authors use simulated annealing to determine a keyword sequence that minimizes this distance. The quality of the attack is the percentage of keywords that are guessed correctly. This percentage is improved if the background knowledge also includes a set of known query-trapdoor associations – this is however not required. With 15% known queries of 150 observed queries, their attack was able to infer close to 100% of a set of 500 keywords correctly. To counteract the presented attack, they also suggest the insertion of noise to hide statistical properties of the query-document associations. Encrypted index structures are considered $(\alpha, 0)$ -secure if for every keyword there are $\alpha - 1$ keywords that appear in the same set of documents – limiting an attackers probability of correctly inferring a keyword to $\frac{1}{\alpha}$ at best.

6.5 Implications for Practical Use

The threat model in section 6.1 shows that attacks on the confidentiality are possible in every part of the system. However, as shown in the previous sections, attacks by unauthorized users on the client and the network can effectively be mitigated by access control and encryption. The most relevant threat is the possibility of inferring keywords by exploiting statistical properties that can be observed by monitoring queries. The threat posed by statistical inference attacks depends strongly on the set of keywords and their distribution in the document set. Statistical inference attacks are only a minor concern if the individual key-

words exhibit very similar statistical properties, e.g. serial numbers that are evenly distributed across documents. However, attributes with statistical properties that could be available as background knowledge (e.g. medical diagnoses) to an attacker need to be treated with great caution and might require noise insertion.

7 CONCLUSION

In this paper, we evaluated the practical usability of searchable encryption for data archives in the cloud, illustrated by embedding an implementation of Goh's searchable encryption scheme into MongoDB. We found that the use of compression on the additional data structures keeps the data size at tolerable levels and relative to the number of embedded search keywords. Performance benchmarks revealed that for insert operations under typical network parameters, the additional overhead for insert operations is negligible compared to unencrypted operation. Search queries however exhibit a considerable impact for encrypted operation, as search operations are linear to the number of documents in Goh's scheme. However, the measured durations of encrypted queries could be acceptable for interactive use where the added security is required. To evaluate the security properties of searchable encryption, we presented threats to keyword confidentiality as an attack-defense-tree model, which applies to most searchable encryption schemes. The most relevant threat comes from inference attacks, which are possible if the keywords exhibit strong statistical properties which can be extracted using background knowledge. In such cases, noise insertion techniques can be used to mitigate such attacks.

Further research could investigate the performance more recent constructions of searchable encryption schemes with constant search complexity (e.g. (Kamara et al., 2012)) and schemes that provide extended search capabilities, such as range queries (see e.g. (Boneh and Waters, 2007; Wang et al., 2011)).

ACKNOWLEDGEMENTS

The authors would like to thank Martin Kreichgauer for providing the prototypical implementation of the Z-IDX scheme and the MongoDB integration.

REFERENCES

- Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., and Shi, H. (2005). Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Advances in Cryptology-CRYPTO 2005*, pages 205–222. Springer.
- Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. (2004). Order preserving encryption for numeric data. In *Proceedings of SIGMOD '04 International Conference on Management of Data*, pages 563–574. ACM.
- Arasu, A., Blanas, S., Eguro, K., Joglekar, M., Kaushik, R., Kossmann, D., Ramamurthy, R., Upadhyaya, P., and Venkatesan, R. (2013). Secure database-as-a-service with cipherbase. In *Proceedings of SIGMOD '13 International Conference on Management of Data*, pages 1033–1036. ACM.
- Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33.
- Bagnato, A., Kordy, B., Meland, P. H., and Schweitzer, P. (2012). Attribute decoration of attack–defense trees. *International Journal of Secure Software Engineering (IJSSE)*, 3(2):1–35.
- Bajaj, S. and Sion, R. (2011). Trusteddb: A trusted hardware based database with privacy and data confidentiality. In *Proceedings of SIGMOD '11 International Conference on Management of Data*, pages 205–216. ACM.
- Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.
- Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer.
- Boneh, D. and Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *Theory of cryptography*, pages 535–554. Springer.
- Byun, J. W., Rhee, H. S., Park, H.-A., and Lee, D. H. (2006). Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Secure Data Management*, pages 75–83. Springer.
- Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2006). Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88. ACM.
- Floratou, A., Teletia, N., DeWitt, D. J., Patel, J. M., and Zhang, D. (2012). Can the elephants handle the nosql onslaught? *Proc. VLDB Endow.*, pages 1712–1723.
- Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of Computing*, pages 169–178. ACM.
- Goh, E.-J. et al. (2003). Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216.

- Hore, B., Mehrotra, S., and Tsudik, G. (2004). A privacy-preserving index for range queries. In *Proceedings of the 13th International Conference on Very Large Data Bases*, VLDB '04, pages 720–731.
- Islam, M., Kuzu, M., and Kantarcioglu, M. (2012). Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Network and Distributed System Security Symposium (NDSS'12)*.
- ITSEC (1991). Information technology security evaluation criteria (itsec): Preliminary harmonised criteria. Technical report, Commission of the European Communities.
- Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage. *Financial Cryptography and Data Security*, pages 136–149.
- Kamara, S., Papamanthou, C., and Roeder, T. (2012). Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976. ACM.
- Kordy, B., Mauw, S., Radomirović, S., and Schweitzer, P. (2012). Attack-defense trees. *Journal of Logic and Computation*.
- Krawczyk, H., Bellare, M., and Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational). Updated by RFC 6151.
- Lindell, Y., Pinkas, B., and Smart, N. P. (2008). Implementing two-party computation efficiently with security against malicious adversaries. In *Security and Cryptography for Networks*, pages 2–20. Springer.
- Liu, C., Zhu, L., Wang, M., and an Tan, Y. (2013). Search pattern leakage in searchable encryption: Attacks and new constructions. Cryptology ePrint Archive, Report 2013/163.
- Popa, R. A., Redfield, C. M. S., Zeldovich, N., and Balakrishnan, H. (2011). Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100. ACM.
- Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, 32(4):169–178.
- Schneier, B. (1999). Attack trees. *Dr. Dobbs's journal*, 24(12):21–29.
- Shmueli, E., Waisenberg, R., Elovici, Y., and Gudes, E. (2005). Designing secure indexes for encrypted databases. In *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, DBSec'05, pages 54–68.
- Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE.
- Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. *Advances in Cryptology—EUROCRYPT 2010*, pages 24–43.
- Wang, S., Agrawal, D., and El Abbadi, A. (2011). A comprehensive framework for secure query processing on relational data in the cloud. In *Proceedings of the 8th VLDB Workshop on Secure Data Management*, SDM'11, pages 52–69, Berlin, Heidelberg. Springer-Verlag.
- Yang, Z., Zhong, S., and Wright, R. N. (2006). Privacy-preserving queries on encrypted data. In *Proceedings of the 11th European Conference on Research in Computer Security*, ESORICS'06, pages 479–495.

A Many-objective Optimization Framework for Virtualized Datacenters

Fabio López Pires^{1,2} and Benjamín Barán^{2,3}

¹*Itaipu Technological Park (PTI), Hernandarias, Paraguay*

²*National University of Asunción (UNA), San Lorenzo, Paraguay*

³*National University of the East (UNE), Ciudad del Este, Paraguay*
fabio.lopez@pti.org.py, bbaran@pol.una.py

Keywords: Virtual Machine Placement, Many-objective Optimization, Datacenter, Virtualization, Cloud Computing.

Abstract: The process of selecting which virtual machines should be located (i.e. executed) at each physical machine of a datacenter is commonly known as Virtual Machine Placement (VMP). This work presents a general many-objective optimization framework that is able to consider as many objective functions as needed when solving the VMP problem in a pure multi-objective context. As an example of utilization of the proposed framework, for the first time a formulation of the many-objective VMP problem (MaVMP) is proposed, considering the simultaneous optimization of the following five objective functions: (1) power consumption, (2) network traffic, (3) economical revenue, (4) quality of service and (5) network load balancing. To solve the formulated many-objective VMP problem, an interactive memetic algorithm is proposed. Simulations prove the correctness of the proposed algorithm and its effectiveness converging to a treatable number of solutions in different experimental scenarios.

1 INTRODUCTION

One of the key challenges in modern datacenters is to efficiently manage power consumption, considering electricity costs and the carbon dioxide footprints (Beloglazov et al., 2011). Most of the time, servers operate in a very low energy-efficiency region (i.e. between 10 and 50% of resource utilization), even considering that workload peaks rarely occur in practice (Barroso and Hölzle, 2007). Consequently, applying techniques for higher resource utilization can result in more energy-efficient server operation. Virtualization of computational resources is a technology that dynamically improves the utilization of available resources in a datacenter according to the existing demand, improving efficiency. Correctly locating virtual machines (VMs) into physical machines (PMs) reduces the amount of hardware in use, letting unused PMs to be in standby mode or even to shut down. This way, average resource utilization as well as energy efficiency may be improved, resulting in better economical revenue and greener datacenters.

Virtualization in modern datacenters introduces management decisions related to the placement of VMs. In this context, Virtual Machine Placement (VMP) is the process of selecting which VMs should be executed in a given set of PMs of a datacenter.

1.1 Background and Motivation

In datacenters with a considerable amount of PMs and VMs, there is a large number of possible criteria that can be considered when selecting a placement, depending on the priorities and optimization objectives. These criteria can even change from one period of time to another, which implies a variety of possible formulations of the VMP problem and objective functions to be optimized for virtualized datacenters.

According to (López Pires and Barán, 2015), the optimization of the energy consumption is the most studied objective function in VMP literature (Sun et al., 2013; Beloglazov et al., 2012). On the other hand, network traffic (Anand et al., 2013), economical revenue (Shi et al., 2013; Sato et al., 2013), performance (Bin et al., 2011) and resource utilization (Mishra and Sahoo, 2011) optimization are also very studied. For each objective function, several possible formulations can be proposed.

In the VMP context, objective functions can be studied according to the following identified optimization approaches: (1) mono-objective (MOP), (2) multi-objective solved as mono-objective (MAM) and (3) multi-objective (PMO) (López Pires and Barán, 2015). The mono-objective approach considers the optimization of only one objective or the individual

optimization of more than one objective, one at a time. On the other hand, multi-objective solved as mono-objective approach considers the optimization of multiple objectives combined into one objective (usually as a weighted sum of different normalized objectives), while a pure multi-objective approach considers the simultaneous optimization of different (possible contradictory) objectives. To the best of the authors' knowledge, there is no many-objective optimization formulation proposed for the VMP problem in the specialized literature (López Pires and Barán, 2015), i.e a multi-objective optimization problem with at least four conflicting objective functions (von Lücken et al., 2014).

Considering the large number of existing objective functions for the VMP problem, this work presents a many-objective optimization framework to be able to consider as many objective functions as needed when solving the VMP problem. As an example of utilization of the proposed framework, for the first time a formulation of the many-objective VMP problem (MaVMP) is proposed, considering the following five objective functions: (1) power consumption minimization, (2) network traffic minimization, (3) economical revenue maximization, (4) QoS maximization and (5) network load balancing optimization. In the presented formulation, a multi-level priority is associated to each VM, representing a Service Level Agreement (SLA) considered in the placement process. To solve the formulated MaVMP problem, an interactive memetic algorithm is proposed considering the issues that give place the formulated problem of many-objective optimization for Pareto-based algorithms.

This paper is structured in the following way: Section 2 presents a multi-objective optimization problem formulation, considering the issues that give place the problem of many-objective optimization. Section 3 details the proposed general many-objective optimization framework, while Section 4 summarizes a many-objective formulation of the VMP problem considering the simultaneous optimization of five objective functions and a multi-level priority of SLA. Section 5 presents a novel interactive memetic algorithm proposed for solving the formulated many-objective problem, while Section 6 presents first experimental results. Finally, conclusions and future work are left to Section 7.

2 MULTI-OBJECTIVE OPTIMIZATION

A general pure multi-objective optimization problem (PMO) includes a set of p decision variables, q objec-

tive functions, and r constraints. Objective functions and constraints are functions of decision variables. In a PMO formulation, x represents the decision vector, while y represents the objective vector. The decision space is denoted by X and the objective space as Y . These can be expressed as (Coello et al., 2007):

Optimize:

$$y = f(x) = [f_1(x), f_2(x), \dots, f_q(x)] \quad (1)$$

subject to:

$$e(x) = [e_1(x), e_2(x), \dots, e_r(x)] \geq 0 \quad (2)$$

where:

$$x = [x_1, x_2, \dots, x_p] \in X \quad (3)$$

$$y = [y_1, y_2, \dots, y_q] \in Y \quad (4)$$

It is important to remark that optimizing, in a particular problem context, can mean maximizing or minimizing. The set of constraints $e(x) \geq 0$ defines the set of feasible solutions $X_f \subset X$ and its corresponding set of feasible objective vectors $Y_f \subset Y$. The feasible decision space X_f is the set of all decision vectors x in the decision space X that satisfies the constraints $e(x)$, and it is defined as:

$$X_f = \{x \mid x \in X \wedge e(x) \geq 0\} \quad (5)$$

The feasible objective space Y_f is the set of the objective vectors y that represents the image of X_f onto Y and it is denoted by:

$$Y_f = \{y \mid y = f(x) \quad \forall x \in X_f\} \quad (6)$$

To compare two solutions in a multi-objective context, the concept of Pareto dominance is used. Given two feasible solutions $u, v \in X$, u dominates v , denoted as $u \succ v$, if $f(u)$ is better or equal to $f(v)$ in every objective function and strictly better in at least one objective function. If neither u dominates v , nor v dominates u , u and v are said to be non-comparable (denoted as $u \sim v$).

A decision vector x is non-dominated with respect to a set U , if there is no member of U that dominates x . The set of non-dominated solutions of the whole set of feasible solutions X_f , is known as optimal Pareto set P^* . The corresponding set of objective vectors constitutes the optimal Pareto front PF^* .

PMOs with more than three objective functions are known as Many-Objective Optimization Problems (MaOPs), as defined in (Cheng et al., 2014). MaOPs differ significantly from PMOs because several issues should be considered when solving problems with more than three objective functions (Farina and Amato, 2002). In case of Pareto-based algorithms, these

issues are intrinsically related to the fact that as the number of objective increases, the proportion of non-dominated elements in the population grows, being increasingly difficult to discriminate among solutions using only the Pareto dominance relation (Deb et al., 2006). Additionally, determining which solution to keep and which to discard in order to converge toward the Pareto set is still a relevant issue to be addressed (Farina and Amato, 2002). Pareto-based algorithms are still not able to provide the required selection pressure towards better solutions in order to conduct an efficient evolutionary search and, even elitism may be difficult. As the number of objectives grows, the proportion of non-comparable solutions to the total number of solutions tends to one (von Lücken et al., 2014), making more difficult to solve a MaOP. Clearly, difficulties in solving MaOPs explain why it has not yet been studied in the VMP literature.

3 MANY-OBJECTIVE OPTIMIZATION FRAMEWORK

The general many-objective optimization framework for the VMP problem proposed in this work considers that as the number of conflicting objectives of a MaVMP problem formulation increases, the total number of non-dominated solutions increases (even exponentially in some cases), being increasingly difficult to discriminate among solutions using only the dominance relation (Farina and Amato, 2002). For this reason, this work proposes the utilization of lower and upper bounds associated to each objective function $z \in \{1, \dots, q\}$ ($L_z \leq f_z(x) \leq U_z$) to be able to reduce iteratively the number of possible compromise solutions of the Pareto set approximation, when needed by the decision maker.

A VMP formulation, based on many objective functions and constraints to be detailed in Section 4, may be written as:

Optimize:

$$y = f(x) = [f_1(x), f_2(x), \dots, f_q(x)] \quad (7)$$

where:

$$\begin{aligned} f_1(x) &= \text{power consumption minimization} \\ f_2(x) &= \text{network traffic minimization} \\ f_3(x) &= \text{economical revenue maximization} \\ f_4(x) &= \text{QoS maximization} \\ f_5(x) &= \text{network load balancing optimization} \\ &\vdots \\ f_q(x) &= \text{any other considered function} \end{aligned} \quad (8)$$

subject to:

$$\begin{aligned} e_1(x) &: \text{unique placement of VMs;} \\ e_2(x) &: \text{assure provisioning of highest SLA;} \\ e_3(x) &: \text{processing resource capacity of PMs;} \\ e_4(x) &: \text{memory resource capacity of PMs;} \\ e_5(x) &: \text{storage resource capacity of PMs;} \\ e_6(x) &: f_1(x) \in [L_1, U_1]; \\ e_7(x) &: f_2(x) \in [L_2, U_2]; \\ e_8(x) &: f_3(x) \in [L_3, U_3]; \\ e_9(x) &: f_4(x) \in [L_4, U_4]; \\ e_{10}(x) &: f_5(x) \in [L_5, U_5]; \\ &\vdots \\ e_r(x) &: \text{any other considered constraint.} \end{aligned} \quad (9)$$

4 MANY-OBJECTIVE VIRTUAL MACHINE PLACEMENT

A few articles proposed formulations of a pure multi-objective VMP problem (MVMP) (Gao et al., 2013; López Pires and Barán, 2013), considering the simultaneous optimization of at most three objective functions. To the best of the authors' knowledge, this work proposes for the first time the formulation of a MaVMP problem considering the following five objective functions to be simultaneously optimized: (1) power consumption, (2) network traffic, (3) economical revenue, (4) quality of service and (5) network load balancing. In this many-objective formulation, a multi-level priority is associated to each VM, representing a SLA. Formally, the proposed offline many-objective optimization VMP problem can be enunciated as:

Given a set of PMs, $H = \{H_1, H_2, \dots, H_n\}$, a network topology G (as illustrated in Figure 1) and a set of VMs, $V = \{V_1, V_2, \dots, V_m\}$, it is sought a correct placement of the set of VMs V in the set of PMs H satisfying the r constraints of the problem and simultaneously optimizing all q objective functions defined in this formulation (as energy consumption, network traffic, economical revenue, QoS and load balancing in the network), in a pure many-objective context.

4.1 Input Data

The proposed formulation of the VMP problem models a virtualized datacenter infrastructure, composed by PMs and a network topology. The set of PMs

is represented as a matrix H of dimension $(n \times 4)$. Each H_i is represented by processing resources CPU (as ECU)¹, RAM memory [GB], storage [GB] and a maximum power consumption [W] as:

$$H_i = [Hcpu_i, Hram_i, Hhdd_i, pmax_i] \quad (10)$$

$$\forall i \in \{1, \dots, n\}$$

where:

- $Hcpu_i$: Processing resources of H_i ;
- $Hram_i$: Memory resources of H_i ;
- $Hhdd_i$: Storage resources of H_i ;
- $pmax_i$: Maximum power consumption of H_i ;
- n : Number of PMs.

As shown in Figure 1, the network topology of the virtualized datacenter is represented as:

- G : Network topology;
- L : Set of links l_a in G . For simplicity, we assume that all links are semi-duplex;
- M : Set of paths for all-to-all PM interconnections;
- K : Capacity set of the communication channel, typically in [Mbps].

The set of VMs requested by customers is represented as a matrix V of dimension $(m \times 5)$. Each V_j requires processing resources CPU (as ECU)¹, RAM memory [GB] and storage [GB], providing for them an economical revenue R_j [\$] for the provider. A SLA is also assigned to each VM to indicate its level of priority. Consequently, a V_j is represented as:

$$V_j = [Vcpu_j, Vram_j, Vhdd_j, R_j, SLA_j] \quad (11)$$

$$\forall j \in \{1, \dots, m\}$$

where:

- $Vcpu_j$: Processing requirements of V_j ;
- $Vram_j$: Memory requirements of V_j ;
- $Vhdd_j$: Storage requirements of V_j ;
- R_j : Economical revenue for locating V_j ;
- SLA_j : Service Level Agreement SLA_j of a V_j . If the highest priority level is s , then $SLA_j \in \{1, \dots, s\}$;
- m : Number of VMs.

The traffic between VMs is represented as a matrix T of dimension $(m \times m)$. Each V_j requires network communication resources [Mbps] to communicate with other VMs. These communication resources

are represented as:

$$T_j = [T_{j1}, T_{j2}, \dots, T_{jm}] \quad (12)$$

$$\forall j \in \{1, \dots, m\}$$

where:

- T_{jk} : Average network traffic between V_j and V_k [Mbps]. Note that we can consider $T_{jj} = 0$.

Figure 1 presents a basic example of a virtualized datacenter infrastructure, composed by 4 PMs $H = \{H_1, H_2, H_3, H_4\}$ and a network topology considering 6 physical network links $L = \{l_1, l_2, l_3, l_4, l_5, l_6\}$. In this example, the set of capacity for each communication channel is $K = \{100, 100, 100, 100, 1000, 1000\}$ [Mbps] respectively. Using shortest path, a path m_{12} between H_1 and H_2 uses links $\{l_1, l_2\}$, i.e. $m_{12} = \{l_1, l_2\}$. Analogously, $m_{13} = \{l_1, l_5, l_6, l_3\}$ and $m_{14} = \{l_1, l_5, l_6, l_4\}$, as shown in Figure 1.

4.2 Output Data

A calculated solution should indicate the exact placement of each VM V_j on the necessary PMs H_i , considering the many-objective optimization criteria applied. A placement (or possible solution to the formulated problem) is represented in what follows as a matrix $P = \{P_{ji}\}$ of dimension $(m \times n)$, where $P_{ji} \in \{0, 1\}$ indicates if V_j is located ($P_{ji} = 1$) or not ($P_{ji} = 0$) for execution on a PM H_i (i.e., $P_{ji} : V_j \rightarrow H_i$).

4.3 Constraints

4.3.1 Constraint 1: Unique Placement of VMs

A VM V_j should be located to run on a single PM H_i or alternatively, it could be not located in any PM if the associated SLA_j is not the highest level of priority s . Consequently, this constraint is expressed as:

$$\sum_{i=1}^n P_{ji} \leq 1 \quad \forall j \in \{1, \dots, m\} \quad (13)$$

where:

- P_{ji} : Binary variable equals 1 if V_j is located to run on H_i ; otherwise, it is 0.

4.3.2 Constraint 2: Assure SLA Provisioning

A VM V_j with the highest level of SLA (i.e. $SLA_j = s$) must necessarily be located to run on a PM H_i . Consequently, this constraint is expressed as:

$$\sum_{i=1}^n P_{ji} = 1 \quad \forall j \text{ such that } SLA_j = s \quad (14)$$

¹<http://aws.amazon.com/ec2/faqs>

4.3.3 Constraints 3-5: Physical Resource Capacity of PMs

A PM H_i must have sufficient available resources to meet the requirements of all VMs V_j that are located to run on H_i . In this work, it is not considered the overbooking of resources (Tomás and Tordsson, 2013). Consequently, this set of constraints can be mathematically formulated as:

$$\sum_{j=1}^m Vcpu_j \times P_{ji} \leq Hcpu_i \quad (15)$$

$$\sum_{j=1}^m Vram_j \times P_{ji} \leq Hram_i \quad (16)$$

$$\sum_{j=1}^m Vhdd_j \times P_{ji} \leq Hhdd_i \quad (17)$$

$\forall i \in \{1, \dots, n\}$, i.e. for all physical machine H_i .

4.3.4 Adjustable Constraints

This work proposes the utilization of lower and upper bounds associated to each objective function to reduce the number of possible solutions of the Pareto set approximation P_{known} , when needed by the decision maker. Consequently, this set of adjustable bounds can be mathematically formulated as the following constraints:

$$f_z(x) \in [L_z, U_z] \quad \forall z \in \{1, \dots, q\} \quad (18)$$

4.4 Objective Functions

A VMP problem can be defined as a many-objective optimization problem, considering the simultaneous optimization of more than three objective functions.

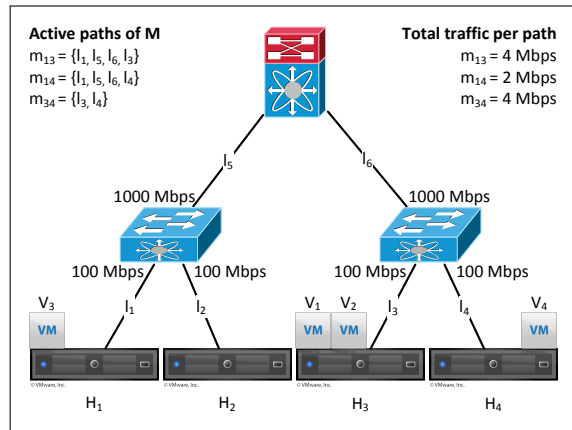


Figure 1: Example of placement in a virtualized datacenter infrastructure, composed by PMs and a network topology.

As a concrete example, this work proposes for the first time the optimization of the following five objective functions:

4.4.1 Power Consumption Minimization

Based on (Beloglazov et al., 2012) formulation, this work also proposes the minimization of power consumption, represented by the sum of the power consumption of each PM H_i :

$$f_1(x) = \sum_{i=1}^n ((pmax_i - pmin_i) \times Ucpu_i + pmin_i) \times Y_i \quad (19)$$

where:

- $f_1(x)$: Total power consumption of the PMs;
- $pmin_i$: Minimum power consumption of H_i . In what follows, $pmin_i = pmax_i * 0.6$ (Beloglazov et al., 2012);
- $Ucpu_i$: Utilization ratio of processing resources used by H_i ;
- Y_i : Binary variable equals 1 if H_i is turned on; otherwise, it is 0.

4.4.2 Network Traffic Minimization

(Shrivastava et al., 2011) proposed the minimization of network traffic among VMs by maximizing locality. Based on this approach, this work proposes equation (20) to estimate network traffic represented by the sum of average network traffic generated by each VM V_j , that is located to run on any PM, with other VMs V_k that are located to run on different PMs.

$$f_2(x) = \sum_{j=1}^m \sum_{k=1}^m (T_{jk} \times D_{jk}) \quad (20)$$

where:

- $f_2(x)$: Total network traffic among VMs;
- D_{jk} : Binary variable that equals 1 if V_j and V_k are located in different PMs; otherwise, it is 0.

The traffic between two VMs V_j and V_k which are located on the same PM H_i do not contribute to increase the total network traffic given by equation (20); therefore, $D_{jk} = 0$ if $P_{ji} = P_{ki} = 1$.

4.4.3 Economical Revenue Minimization

Based on (López Pires and Barán, 2013), this work presents equation (21) to estimate the total economical revenue that a datacenter receives for meeting the requirements of its customers, represented by the sum

of the economical revenue obtainable by each VM V_j placement that is effectively located for execution on any PM.

$$f_3(x) = \sum_{j=1}^m (R_j \times X_j) \quad (21)$$

where:

- $f_3(x)$: Total economical revenue for placing VMs;
 X_j : Binary variable that equals 1 if V_j is located for execution on any PM; otherwise, it is 0.

4.4.4 QoS Maximization

In this work, the QoS maximization proposes to locate the maximum number of VMs with the highest level of priority associated to the SLA. This objective function is proposed in equation (22).

$$f_4(x) = \sum_{j=1}^m (\hat{C}^{SLA_j} \times SLA_j \times X_j) \quad (22)$$

where:

- $f_4(x)$: Total QoS figure for a given placement;
 \hat{C} : Constant, large enough to prioritize services with larger SLA over the ones with lower SLA (see example in Section 4.4.6).

4.4.5 Network Load Balancing Optimization

This work calculates the total amount of traffic going through a semi-duplex link l_a as:

$$T_{l_a} = \sum_{i=1}^n \sum_{i'=1}^n F_{aii'} \times \left(\sum_{j=1}^m \sum_{j'=1}^m P_{ji} \times P_{j'i'} \times D_{jj'} \times T_{jj'} \right) \quad (23)$$

where:

- T_{l_a} : Total amount of traffic going through link l_a [Mbps];
 $F_{aii'}$: Binary variable that equals 1 if $l_a \in m_{ii'}$; otherwise, it is 0.

Inspired in (Donoso et al., 2005) formulation, this work calculates the Maximum Link Utilization (MLU) as:

$$MLU = \max_{\forall l_a \in L} \left(\frac{T_{l_a}}{Cl_a} \right) \quad (24)$$

where:

- MLU : Maximum Link Utilization;
 Cl_a : Channel capacity of link l_a [Mbps].

In this paper, the load balancing optimization of the network is formulated as the minimization of the MLU, i.e.,

$$f_5(x) = MLU \quad (25)$$

4.4.6 Example

The following example details how the values of $f_4(x)$ and $f_5(x)$ are calculated, considering the datacenter infrastructure presented in Figure 1. For the other three objective functions, see (López Pires and Barán, 2013).

Consider the following placement matrix P where, VMs V_1 and V_2 are executed in H_3 , while V_3 is executed in H_1 and V_4 in H_4 :

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (26)$$

Additionally, consider $\hat{C} = 100$ and the values of SLA_j and X_j presented in Table 1, $f_4(x)$ is calculated as:

$$\begin{aligned} f_4(x) &= \hat{C}^{SLA_1} \times SLA_1 \times X_1 + \dots + \hat{C}^{SLA_4} \times SLA_4 \times X_4 \\ &= 100^2 \times 2 \times 1 + \dots + 100^3 \times 3 \times 1 \\ &= 3.06 \times 10^6 \end{aligned} \quad (27)$$

It is important to remark that if \hat{C} is not sufficiently large, $f_4(x)$ could prefer a large number of VMs with lower priority. As an example, if $\hat{C} = 1$, 2 VMs with $SLA_j = 2$ result in a better figure of $f_4(x)$ than 1 VM with $SLA_j = 3$, what is not correct according to what was presented in Section 4.4.4.

On the other hand, considering that each VM V_j communicates at 1 [Mbps] with any other $V_k (k \neq j)$, the placement P results in the values of T_{l_a} and Cl_a presented in Table 2. So $f_5(x)$ is calculated as:

$$\begin{aligned} f_5(x) &= \max \left(\frac{T_{l_1}}{Cl_1}, \frac{T_{l_2}}{Cl_2}, \frac{T_{l_3}}{Cl_3}, \frac{T_{l_4}}{Cl_4}, \frac{T_{l_5}}{Cl_5}, \frac{T_{l_6}}{Cl_6} \right) \\ &= \max \left(\frac{6}{100}, \frac{0}{100}, \frac{8}{100}, \frac{6}{100}, \frac{6}{1000}, \frac{6}{1000} \right) \\ &= \frac{8}{100} \end{aligned} \quad (28)$$

5 INTERACTIVE MEMETIC ALGORITHM

A Memetic Algorithm (MA) could be understood as an Evolutionary Algorithm (EA) that in addition to

Table 1: Data for requested VMs used in basic example (see Section 4.4.6).

V_j	SLA_j	X_j
V_1	2	1
V_2	2	1
V_3	2	1
V_4	3	1

Table 2: Data of network links used in basic example (see Section 4.4.6).

l_a	Tl_a [Mbps]	Cl_a [Mbps]	Tl_a/Cl_a
l_1	6	100	6/100
l_2	0	100	0/100
l_3	8	100	8/100
l_4	6	100	6/100
l_5	6	1000	6/1000
l_6	6	1000	6/1000
MLU			8/100

the standard selection, crossover and mutation operators of most Genetic Algorithms (GA) includes a local optimization operator to obtain good solutions even at early generations of an EA (Báez et al., 2007).

This work proposes an interactive memetic algorithm for solving the VMP problem in a many-objective context, considering the proposed formulation presented in Section 4 to simultaneously optimize the five objective functions presented in the previous section. The proposed algorithm is extensible to consider as many objective functions as needed while only minor modifications may be needed if the number of objective functions changes.

It was shown in (von Lücken et al., 2014) that many-objective optimization using Multi-Objective Evolutionary Algorithms (MOEAs) is an active research area, having multiple challenges that need to be addressed regarding scalability analysis, visualization of the results, algorithm design and experimental algorithm evaluation. The interactive memetic algorithm presented in this section as a viable way to solve a many-objective VMP problem, proposes the inclusion of desirable ranges of values for the objective functions costs in order to interactively control the possible huge number of feasible non-dominated solution, as described in Section 2. The proposed interactive memetic algorithm is based on the one proposed in (López Pires and Barán, 2013) and works as follows:

In step 1, it is verified if the problem has at least one solution (considering only VMs with $SLA_j = s$) to continue with next steps. If there is no possible solution to the problem, the algorithm returns an appropriate error message. If the problem has at least one solution, the algorithm proceeds to step 2, which ge-

nerates a set of random candidates P_0 , whose solutions are repaired at step 3 to ensure that P_0 contains only feasible solutions. Then, the algorithm tries to improve candidates at step 4 using local search. With the obtained non-dominated solutions, the first set P_{known} (Pareto set approximation) is generated at step 5. After initialization in step 6, evolution begins (iterations between steps 7 and 18).

The evolutionary process basically follows the same behavior: solutions are selected from the union of P_{known} with the evolutionary set of solutions (or population) also known as P_t (step 8), crossover and mutation operators are applied as usual (step 9), and eventually solutions are repaired, as there may be infeasible solutions (step 10). Improvements of solutions of the evolutionary population P_t may be generated at step 11 using local search (local optimization operators). At step 12, the Pareto set approximation P_{known} is updated (if applicable); while at step 13 the generation (or iteration) counter is updated. At step 15 the decision maker adjust the lower and upper bounds if it is necessary, while at step 17 a new evolutionary population P_t is selected. The evolutionary process is repeated until the algorithm meets a stopping criterion (such as a maximum number of generations), finally returning the set of non-dominated solutions P_{known} in step 19.

Algorithm 1: Interactive Memetic Algorithm.

Data: datacenter infrastructure (see Section 4.1)

Result: Pareto set approximation P_{known}

```

1 check if the problem has a solution
2 initialize set of solutions  $P_0$ 
3  $P'_0$  = repair infeasible solutions of  $P_0$ 
4  $P''_0$  = apply local search to solutions of  $P'_0$ 
5 update set of solutions  $P_{known}$  from  $P''_0$ 
6  $t = 0; P_t = P''_0$ 
7 while is not stopping criterion do
8    $Q_t$  = selection of solutions from  $P_t \cup P_{known}$ 
9    $Q'_t$  = crossover and mutation of solutions of  $Q_t$ 
10   $Q''_t$  = repair infeasible solutions of  $Q'_t$ 
11   $Q'''_t$  = apply local search to solutions of  $Q''_t$ 
12  update set of solutions  $P_{known}$  from  $Q'''_t$ 
13  increment t
14  if interaction is needed then
15    | ask for decision maker action ( $L_z$  and  $U_z$ )
16  end
17   $P_t$  = non-dominated sorting from  $P_t \cup Q'''_t$ 
18 end
19 return Pareto set approximation  $P_{known}$ 

```

5.1 Population Initialization

Initially, a set of solutions (or population P_0) is randomly generated. Each solution (or individual) is represented as $C = [C_1, C_2, \dots, C_m]$. The possible val-

ues that can take each C_k for VMs with the highest value of SLA_j ($SLA_j = s$) are in the range $[1, n]$. For VMs V_j with $SLA_j < s$, the possible values are in the range $[0, n]$. Within these ranges defined by the SLA_j of each V_j , the algorithm ensures at the initialization stage that all VMs V_j with the highest level of priority will be located for execution on a PM H_i , while for VMs V_j with lower levels of priority SLA_j , there is always a probability larger than 0 that they may not be located for execution in any PM.

5.2 Infeasible Solution Reparation

With a random generation at the initialization phase (step 2 of Algorithm 1) and/or solutions generated by standard genetic operators (step 9 of Algorithm 1), infeasible solutions may appear, i.e. the resources required by the VMs located for execution on certain PMs could exceed the available resources (see Section 4.3.3), or one or more objectives functions may not meet the adjustable constraints (see Section 4.3.4).

Repairing these infeasible solutions (steps 3 and 10 of Algorithm 1) may be done in two stages: first, in the feasibility verification process, the population is classified in two classes: feasible and infeasible (Algorithm 2). Later, in the process of repairing infeasible solutions (Algorithm 3), the solutions which do not meet the feasibility criteria are repaired in three ways: (1) migrating some VMs to an available hardware, (2) turning on some PMs and then migrating VMs to them, or (3) turning off some VMs. Note that at step 3 of Algorithm 3, V_j migration to H_i' can be done to other PMs, even if they are shut down.

Algorithm 2: Feasibility Verification

Data: set of solutions P_t
Result: set of feasible solutions P_t'

```

1 while there are solutions not verified do
2   feasible = true ; i = 1
3   while i ≤ n and feasible = true do
4     if solution does not satisfy constraints (3-5)
5       then
6         feasible = false ; break
7       else
8         increment i
9     end
10  end
11  if feasible = false then
12    call Algorithm 3 (repair solution)
13  end
14 return set of feasible solutions  $P_t'$ 

```

Algorithm 3: Infeasible Solutions Reparation.

Data: infeasible solution
Result: feasible solution

```

1 feasible = false ; j = 1
2 while j ≤ m and feasible = false do
3   if it is possible then
4     migrate  $V_j$  to  $H_i'$  ( $i' \neq i$ )
5   else
6     if  $SLA_j \neq s$  then
7       turn off  $V_j$  on  $H_i$ 
8     else
9       replace solution with another solution
        from  $P_{known}$ 
10    end
11  end
12 end
13 return feasible solution

```

5.3 Local Search

Once a population contains only feasible solutions, a local search is performed (steps 4 and 11 of Algorithm 1) for improving the solutions found so far in the evolutionary population P_t . The local search pseudo-code is presented in Algorithm 4.

Algorithm 4: Local Search.

Data: set of feasible solutions P_t'
Result: set of feasible optimized solutions P_t''

```

1 probability = random number between 0 and 1
2 while there are solutions not verified do
3   if probability < 0.5 then
4     Try to turn off all the possible  $H_i$  by
        migrating all the  $V_j$  assigned to  $H_i'$  with
        available resources ( $i' \neq i$ ) and then try to
        turn on all the possible  $V_j$  (using  $SLA_j$ 
        priority order) assigning them to a  $H_i$  with
        available resources
5   else
6     Try to turn on all the possible  $V_j$  (using
         $SLA_j$  priority order) assigning them to a  $H_i$ 
        with available resources and then try to turn
        off all the possible  $H_i$  by migrating all the  $V_j$ 
        assigned to  $H_i'$  with available resources
        ( $i' \neq i$ )
7   end
8 end
9 return set of feasible optimized solutions  $P_t''$ 

```

For each individual in the population P_t , the proposed algorithm attempts to optimize a solution with a local search (step 2 of Algorithm 4). For this purpose, with probability $\frac{1}{2}$, the algorithm tries to maximize the number of located VMs with higher level of priority, locating all possible VMs that were not located so far, directly increasing $f_4(x)$ (total QoS) and $f_3(x)$ (total economical revenue) (steps 3 to 5 of Algorithm

4). On the other hand, also with probability $\frac{1}{2}$, the algorithm tries to minimize the number of PMs turned on, directly reducing $f_1(x)$ (total power consumption) (steps 6 to 8 of Algorithm 4). With the proposed probabilistic local search method, a balanced exploitation of objective functions (QoS, economical revenue and power consumption) is achieved, as experimentally verified with results presented in next section.

5.4 Fitness Function

The fitness function used in the proposed algorithm is the one presented in (Deb et al., 2002). This fitness value defines a non-domination rank in which a value equal to its Pareto dominance level (1 is the highest level of dominance, 2 is the next, and so on) is assigned to each individual of the population. Between two individuals with different non-domination rank, the individual with lower value (higher level of dominance) is considered better. To compare individuals with the same non-domination rank, a crowding distance is used. The basic idea is to find the Euclidean distance (properly normalized when the objectives have different measure units) between each pair of individuals, based on the q objectives, in a hyper-dimensional space (Deb et al., 2002). The individual with larger crowding distance is considered better.

5.5 Variation Operators

The proposed interactive memetic algorithm uses a Binary Tournament approach for selecting individuals for crossover and mutation (Coello et al., 2007). The crossover operator used in the presented work is the single point cross-cut (Coello et al., 2007). The selected individuals in the ascending population are replaced by descendants individuals.

This work uses a mutation method in which each gene is mutated with a probability $\frac{1}{m}$, where m represents the number of VMs. This method offers the possibility of full uniform gene mutation, with a very low probability (but larger than zero), which is beneficial to the exploration of the search space, reducing the probability of stagnation in a local optimum.

The population evolution in the proposed interactive memetic algorithm is based on the population evolution proposed in (Deb et al., 2002). A population P_{t+1} is formed from the union of the best known population P_t and offspring population Q_t , applying non-domination rank and crowding distance operators.

5.6 Many-objective Considerations

Given that the number of non-dominated solutions may rapidly increase, an interactive approach is recommended. That way, a decision maker can introduce new constraints or adjust existing ones, while the execution continues learning about the shape of the Pareto front in the process. For simplicity, the present work considers lower and upper bounds associated to each objective function in order to help the decision maker to reduce interactively the huge number of potential solutions in the Pareto set approximation P_{known} , while observing the evolution of its corresponding Pareto front PF_{known} .

6 EXPERIMENTAL RESULTS

To validate the proper operation of the proposed interactive memetic algorithm for solving the MaVMP on a purely many-objective context, different experiments were proposed for problem instances considering both homogeneous as well as heterogeneous hardware configurations of PMs, considering VMs instance types offered by Amazon Elastic Compute Cloud (EC2)². Experiment 1 considered 2 problem instances with homogeneous hardware configurations of PMs ($Hcpu = 4$ [ECU], $Hram = 16$ [GB], $Hhdd = 150$ [GB], $pmax = 1740$ [W]), while Experiment 2 considered heterogeneous hardware configurations of PMs according to Table 6.

A detailed description of the hardware of the VMs instance types considered for the experiments is presented in Table 3. A general description of the considered problem instances is presented in Table 4, while the complete set of datacenter infrastructure input files used for the experiments with the corresponding experimental results are available online³.

6.1 Experiment 1: Quality of Solutions

To compare the results obtained by the proposed interactive memetic algorithm and to validate its proper operation, an exhaustive search algorithm was also implemented for finding all $(n+1)^m$ possible solutions of a given instance of the VMP problem, when this alternative is computationally possible for the authors. These results were compared to the results obtained by the proposed interactive memetic algorithm after evolving populations with 100 individuals for 100 generations. Both algorithms were implemented

²<http://aws.amazon.com/ec2/instance-types>

³<https://sites.google.com/site/flopezpipes/>

Table 3: Instance types of VMs considered in experiments. For notation see equation (11).

Instance Type	Vcpu	Vram	Vhdd	R
t2.micro	1	1	0	9
t2.small	1	2	0	18
t2.medium	2	4	0	37
m3.medium	1	4	4	50
m3.large	2	8	32	100
m3.xlarge	4	15	80	201
m3.2xlarge	8	30	160	403
c3.large	2	4	32	75
c3.xlarge	4	8	80	151
c3.2xlarge	8	15	160	302
c3.4xlarge	16	30	320	604
c3.8xlarge	32	60	640	1209
r3.large	2	15	32	126
r3.xlarge	4	30	80	252
r3.2xlarge	8	61	160	504
r3.4xlarge	16	122	0	320
r3.8xlarge	32	244	0	320

Table 4: Problem instances considered in experiments, all with 50% of VMs with the highest SLAs = 2.

Input File	# PMs	# VMs	$(n+1)^m$
3x5.vmp	3	5	1024
4x8.vmp	4	8	390625
12x50.vmp	12	50	$\sim 5 \times 10^{55}$

using ANSI C programming language (gcc)⁴ and the source code is also available online³.

Considering that this particular experiment aims to validate the good level of exploration in the set of feasible solutions X_f , the local search of the algorithm was disabled, strengthening its capability of exploration rather than the rapid convergence to good solutions even in early generations of the population.

For each problem instance considered in this experiment (3x5.vmp and 4x12.vmp), one run of the exhaustive search algorithm was completed, obtaining the optimal Pareto front PF^* and its corresponding Pareto set P^* . Furthermore, ten runs of the proposed algorithm were completed, after evolving populations of 100 individuals for 100 generations at each run. The results obtained by the proposed algorithm for each run were combined to obtain the Pareto front PF_{known} and its corresponding Pareto set P_{known} . For the 3x5.vmp and 4x8.vmp problem instances, the proposed algorithm obtained every solution of the optimal Pareto set and its corresponding Pareto front. A summary of the number of elements in the corresponding Pareto sets obtained are presented in Table 5.

⁴<http://gcc.gnu.org>

Table 5: Summary of results obtained in Experiment 1 using the proposed memetic algorithm (see Section 5).

Input File Name	# P^*	# P_{known}	% Found
3x5.vmp	51	51	100%
4x8.vmp	30	30	100%

Table 6: Types of PMs considered in Experiment 2. For notation see equation (10).

PM Type	Hcpu	Hram	Hhdd	pmax
h1.medium	180	512	10000	1000
h1.large	350	1024	10000	1300

6.2 Experiment 2: Interactive Bounds

As mentioned in Section 3, this work proposes the utilization of lower and upper bounds for each objective function $f(z)$ ($L_z \leq f_z(x) \leq U_z$) to be able to reduce iteratively the number of possible solutions of the Pareto set approximation P_{known} , when needed.

For the problem instance considered in this experiment (12x50.vmp), one run of the proposed algorithm was completed, after evolving populations of 100 individuals for 300 generations. The number of generations was incremented for this experiment from 100 to 300 considering the large number of possible solutions for the particular problem considered (see Table 4). An interactive adjustment of the lower or upper bounds associated to each objective function was performed after every 100 generations in order to converge to a treatable number of solutions.

It is important to remark that the interactive adjustment used in this experiment is only one of several possible ones. As an example, we may consider: (1) automatically adjusting 10% of the lower bounds associated to maximization objective functions when the Pareto front has more than 200 elements or (2) manually adjusting upper bounds associated to minimization objective functions until the Pareto front does not have more than 20 elements, just to cite a pair of alternatives.

The Pareto front approximation PF_{known} represents the complete set of Pareto solutions considering unrestricted bounds ($L_z = -\infty$ and $U_z = \infty$). On the other hand, Pareto front approximation $PF_{reduced}$ represents the reduced set of Pareto solutions obtained by interactively adjusting bounds L_z and U_z .

In the first 100 generations, the proposed algorithm obtained 251 solutions with unrestricted bounds. A decision maker evaluated the bounds associated to power consumption and adjusted the upper bound U_1 to $U'_1 = 9000$ [W], selecting only 35 of the 251 solutions (not considering 216 otherwise feasible solutions) for the $PF_{reduced}$ as shown in Figure 2.

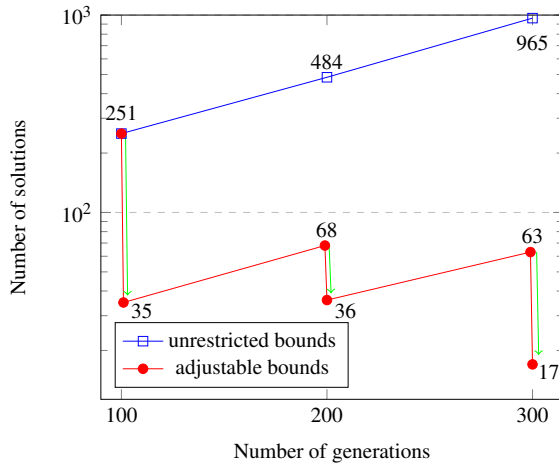


Figure 2: Summary of results obtained in Experiment 2.

After 200 generations, the algorithm obtained a total of 484 solutions with unrestricted bounds. Considering instead $U'_1 = 9000$ [W], the algorithm only found 68 solutions. The decision maker evaluated the bounds associated to network traffic and adjusted the upper bound U_2 to $U'_2 = 115$ [Mbps], selecting only 36 of the 68 solutions (not considering 32 otherwise feasible solutions) for the $PF_{reduced}$.

Finally, after 300 generations, the algorithm obtained a total of 965 solutions with unrestricted bounds. Considering $U'_1 = 9000$ [W] and $U'_2 = 115$ [Mbps], the algorithm found 63 solutions. The decision maker evaluated the bounds associated to economical revenue and adjusted the lower bound L_3 to $L'_3 = 13500$ [\$], selecting only 17 of the 63 solutions (not considering 46 feasible solutions) for the final $PF_{reduced}$ as shown in Figure 2. Clearly, at the end of the iterative process, the decision maker found 17 solutions according to his preferences instead of the untreatable number of 965 candidate solutions.

7 CONCLUSIONS AND FUTURE WORK

This work proposed a general many-objective optimization framework that is able to consider as many objective functions as needed when solving the VMP problem. As an example of utilization of the proposed framework, for the first time a formulation of the many-objective VMP problem (MaVMP) is proposed, considering the simultaneous optimization of the following five objective functions: (1) power consumption, (2) network traffic, (3) economical revenue, (4) quality of service and (5) network load balancing. In addition, a generalized multi-level of priori-

ties for the SLA associated to each VM is presented. At the same time, constraints on upper and lower limits of each objective function are also recommended as a way to interactively control a potential explosion in the number of non-dominated solutions, a well known problem of many-objective optimization problems (von Lücken et al., 2014).

A short review of the most studied objective functions was also presented to demonstrate that the number of objective functions may rapidly increase once a complete understanding of the VMP problem is accomplished for practical problems where many different parameters should be ideally considered. Based on the number of possible objective functions for the VMP, this problem can clearly be addressed as a many-objective optimization problem (MaOP).

Multiple challenges need to be considered for solving a many-objective optimization problem; therefore, an interactive memetic algorithm was proposed to solve the proposed many-objective formulation, validating the presented formulation and proving that it is solvable. By no means, the authors claim that the proposed interactive memetic algorithm is the best way to solve this problem. This proposal only illustrates that it is possible to solve the problem although the VMP problem becomes harder when increasing the number of conflicting objective functions.

To validate the proposed algorithm, it was run with different problem instances and experimental results were compared to the exact solution obtained using an exhaustive search algorithm when possible. In fact, the proposed algorithm found the complete Pareto front and Pareto set (100%) for the 2 instances of Experiment 1. To validate the effectiveness of the proposed algorithm reducing the increasing number of non-dominated solutions obtained by the Pareto dominance comparison, several problem instances were executed. In fact, considering the proposed technique of interactively adjusting lower and upper bounds associated to the values of each objective functions, the Pareto front could be reduced from 965 original candidates to a treatable 17 non-dominated solutions that satisfy the decision maker preferences, as illustrated in Experiment 2.

At the time of this writing, the authors are working on VMP formulations considering more complex network topologies with full-duplex links. Considering that this is the first formulation of the VMP problem in a many-objective context, several future works can be proposed considering all the already presented objective functions, studied so far in mono-objective as well as multi-objective contexts.

Considering that this work proposed an offline (static) formulation of the VMP problem in a many-

objective optimization context, several challenges need to be addressed for online (dynamic) formulations of the problem, considering multi-objective and many-objective approaches. At the same time, different meta-heuristics, methods and algorithms should be still tested before a real good tool is ready for massive use in commercial cloud computing datacenters.

REFERENCES

- Anand, A., Lakshmi, J., and Nandy, S. (2013). Virtual machine placement optimization supporting performance slas. In *Cloud Computing Technology and Science (CloudCom)*, 2013 IEEE 5th International Conference on, volume 1, pages 298–305. IEEE.
- Báez, M., Zárate, D., and Barán, B. (2007). Algoritmos meméticos adaptativos para optimización multi-objetivo. In *XXXIII Conferencia Latinoamericana de Informática–CLEI*, volume 2007.
- Barroso, L. A. and Hölzle, U. (2007). The case for energy-proportional computing. *IEEE computer*, 40(12):33–37.
- Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768.
- Beloglazov, A., Buyya, R., Lee, Y. C., Zomaya, A., et al. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82(2):47–111.
- Bin, E., Biran, O., Boni, O., Hadad, E., Kolodner, E. K., Moatti, Y., and Lorenz, D. H. (2011). Guaranteeing high availability goals for virtual machine placement. In *Distributed Computing Systems (ICDCS)*, 2011 31st International Conference on, pages 700–709. IEEE.
- Cheng, J., Yen, G. G., and Zhang, G. (2014). A many-objective evolutionary algorithm based on directional diversity and favorable convergence. In *Systems, Man and Cybernetics (SMC)*, 2014 IEEE International Conference on, pages 2415–2420.
- Coello, C. C., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.
- Deb, K., Sinha, A., and Kukkonen, S. (2006). Multi-objective test problems, linkages, and evolutionary methodologies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1141–1148. ACM.
- Donoso, Y., Fabregat, R., Solano, F., Marzo, J.-L., and Barán, B. (2005). Generalized multiobjective multi-tree model for dynamic multicast groups. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 1, pages 148–152. IEEE.
- Farina, M. and Amato, P. (2002). On the optimal solution definition for many-criteria optimization problems. In *Proceedings of the NAFIPS-FLINT international conference*, pages 233–238.
- Gao, Y., Guan, H., Qi, Z., Hou, Y., and Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230–1242.
- López Pires, F. and Barán, B. (2013). Multi-objective virtual machine placement with service level agreement. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 203–210. IEEE Computer Society.
- López Pires, F. and Barán, B. (2015). A virtual machine placement taxonomy. In *Proceedings of the 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society.
- Mishra, M. and Sahoo, A. (2011). On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *Cloud Computing (CLOUD)*, 2011 IEEE International Conference on, pages 275–282. IEEE.
- Sato, K., Samejima, M., and Komoda, N. (2013). Dynamic optimization of virtual machine placement by resource usage prediction. In *Industrial Informatics (INDIN)*, 2013 11th IEEE International Conference on, pages 86–91. IEEE.
- Shi, L., Butler, B., Botvich, D., and Jennings, B. (2013). Provisioning of requests for virtual machine sets with placement constraints in iaas clouds. In *Integrated Network Management (IM 2013)*, 2013 IFIP/IEEE International Symposium on, pages 499–505. IEEE.
- Shrivastava, V., Zerfos, P., Lee, K.-W., Jamjoom, H., Liu, Y.-H., and Banerjee, S. (2011). Application-aware virtual machine migration in data centers. In *INFOCOM, 2011 Proceedings IEEE*, pages 66–70. IEEE.
- Sun, M., Gu, W., Zhang, X., Shi, H., and Zhang, W. (2013). A matrix transformation algorithm for virtual machine placement in cloud. In *Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2013 12th IEEE International Conference on, pages 1778–1783. IEEE.
- Tomás, L. and Tordsson, J. (2013). Improving cloud infrastructure utilization through overbooking. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, CAC '13*, pages 5:1–5:10, New York, NY, USA. ACM.
- von Lücken, C., Barán, B., and Brizuela, C. (2014). A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, pages 1–50.

CloudMPL: A Domain Specific Language for Describing Management Policies for an Autonomic Cloud Infrastructure

Marwah M. Alansari¹, Andre Almeida², Nelly Bencomo³ and Behzad Bordbar¹

¹*School of Computer Science, University of Birmingham, Birmingham, U.K.*

²*Federal Institute of Science of Education, Science and Technology, Parnamirim, Brazil*

³*School of Computer Science, Aston University, Birmingham, U.K.*

mma809@cs.bham.ac.uk, andre.almeida@ifrn.edu.br, nelly@acm.org, b.bordbar@cs.bham.ac.uk

Keywords: Management Policies, Rule Language, Domain Specific Language, Autonomic Architecture, Cloud Infrastructure.

Abstract: To benefit from the advantages that Cloud Computing brings to the IT industry, management policies must be implemented as a part of the operation of the Cloud. Among others, for example, the specification of policies can be used for the management of energy to reduce the cost of running the IT system or also for security policies while handling privacy issues of users. As cloud platforms are large, manual enforcement of policies is not scalable. Hence, autonomic approaches for management policies have recently received a considerable attention. These approaches allow specification of *rules* that are executed via rule-engines. The process of rules creation starts by the interpretation of the policies drafted by high-rank managers. Then, technical IT staff translate such policies to operational activities to implement them. Such process can start from a textual declarative description and after numerous steps terminates in a set of rules to be executed on a rule engine. To simplify the steps and to bridge the considerable gap between the declarative policies and executable rules, we propose a domain-specific language called CloudMPL. We also design a method of automated transformation of the rules captured in CloudMPL to the popular rule-engine Drools. As the policies are changed over time, code generation will reduce the time required for the implementation of the policies. In addition, using a declarative language for writing the specifications is expected to make the authoring of rules easier. We demonstrate the use of the CloudMPL language into a running example extracted from a management energy consumption case study.

1 INTRODUCTION

Management of Cloud infrastructure supported by autonomic techniques has recently received a considerable attention (Beloglazov and Buyya, 2010; Mi et al., 2010; Maurer et al., 2013; Borgetto et al., 2012). Several existing autonomic techniques make a use of rule-based systems (Mi et al., 2010; Maurer et al., 2013; Borgetto et al., 2012; Alansari and Bordbar, 2013). The rule-based systems operate by using a set of statements in a form of "if *< condition >* then *< action >*" which are known as rules executed by a rule-engine. Rule-based frameworks have been used to automatically trigger the live-migration of virtual machine by using different types of constraints rules. The constraints rules are formulated as management policies (Borgetto et al., 2012)(Alansari and Bordbar, 2013).

In (Alansari and Bordbar, 2013), the authors propose an architectural framework for automatically ex-

ecuting management policies on Cloud infrastructure. The rule engine is integrated to work with Cloud management system to control the migration of running virtual machines among hosting nodes. Policy Rule Engine and Cloud Manager are communicated through sensors and actuators. The sensor is directly interlinked with Cloud APIs for management of virtual machines that are responsible for requesting monitoring parameters such as Estimated Energy Consumption and Current Resource Usage. Whilst the actuator uses action APIs that directly launch the management actions, such as virtual machine migration action (Alansari and Bordbar, 2013).

There are three steps defined for designing suitable policies which can be executed into the autonomic framework proposed in (Alansari and Bordbar, 2013) or similar frameworks as in (Maurer et al., 2013). These steps are Policy Authoring, Policy Implementation and Policy Deployment and Execution

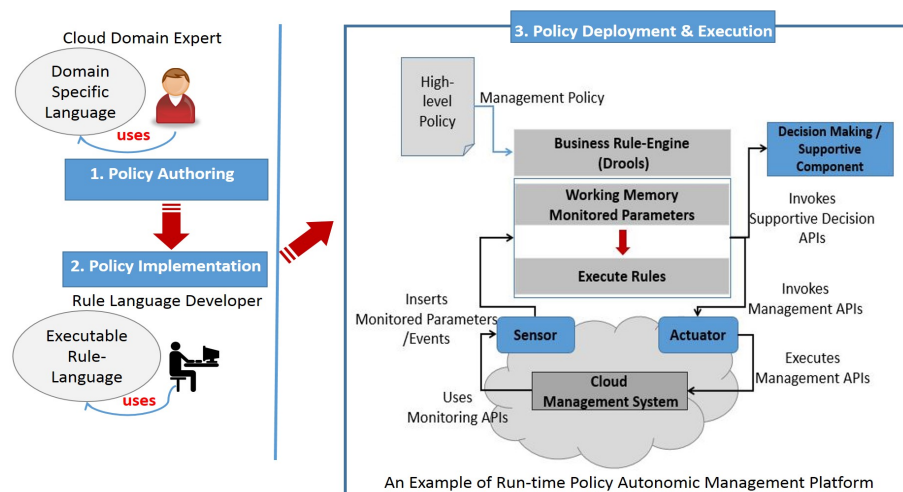


Figure 1: The steps for designing and publishing management policies.

(see Figure 1). Cloud-domain experts and rule developers are the roles which are involved during the design process of the management policies. Cloud-domain experts write the policies in simple plain English sentences “if/then”. Rule developers encode the described policies using a rule language and generating executable management policies as well as Cloud-infrastructure domain model (Alansari and Bordbar, 2014).

Management policies are updated regularly because of alteration in the technical environment, changes in the regulations, modifications of SLAs and changes in business requirements. While going thorough this process, there is a semantic gap between Policy Authoring and Implementation levels. This gap can be bridged by using a combination of *Domain Specific Language (DSL)* for authoring of the rules and *automated code generation* for producing rules which are executed in the rule-engine. The *Domain Specific Language (DSL)* supplies expressive representations, textual notions and high-level specification to describe policies written using plain English. Such a language helps the domain experts, who may have little knowledge about formulating rules, to avoid the complexity of writing management policies. On the other hand, *Code generation* is used to convert the policies captured in the DSL to be an input for executable Rule Languages such as Drools.

This paper makes two contributions. Firstly, we propose Cloud Management Policy Language (CloudMPL), a domain specific language to support Cloud-domain experts to specify policies. CloudMPL is a textual language with a specification partially inspired by the RELAX Language (Whittle et al., 2010). Secondly, we design a set of mapping rules used for automatic transformation from CloudMPL to rule

languages based on Object Patterns such as Drools (JBossCommunity, 2011) and JRules (IBM-ILOG, 2007). The objective is to build a foundation for the automatic generation of a management policy from Policy Authoring to Implementation levels and hence bridge the gap described above.

The paper is organised as follows. Section 2 has an overview about rule-based systems and domain specific languages used in Cloud environment. CloudMPL language and its specification are discussed in Section 3. The specification of management policies in an executable rule language is explained in Section 4. Section 5 includes the designed mapping rules from CloudMPL to Drools. Section 6 contains a demonstration of CloudMPL in XText for authoring migration policies in energy management case study. Finally, the conclusion is presented in Section 7.

2 BACKGROUND AND RELATED WORK

This section introduces concepts about rule engines and domain-specific languages.

2.1 Rule Engine and Rules

Policies can be executed via a rule-based engine such as Drools (JBossCommunity, 2011) and ILOG JRules (IBM-ILOG, 2007). Both Drools and JRules engines are based on the enhanced version of the RETE-Algorithm for supporting Object Oriented Pattern. RETE-Algorithm was introduced by Charles Forgy; it is one of the efficient implementation rules inference engines. The algorithm represents rules as an acyclic

graph to form a RETE-network and provides a pattern matching process (Forgy, 1982).

The architecture of the rule engine is shown in Figure 2. Both Drools and JRules rule engines include Rule-Base and Working Memory. Rule Base is a long-term memory in which rules are stored. Working Memory is a type of short-term memory, which contains Facts that need to be evaluated by the inference engine. Facts are object models that contain attributes that illustrate domain data for an application. Agenda is the place where a rule that has become active is stored for later in order to fire satisfied rules. The agenda uses resolving conflicting methodology for ordering the execution of active rules (Forgy, 1982). In order to execute policies in a rule-based engine, policies have to be written in a form of rule-sets. A rule-set consists of a number of conditions followed with a set of sequential actions in which the rule-set is expressed in the following format:

when (condition statements) **then** (action statements)

Figure 3 illustrates a sample of a constraint-based policy encoded in Drools Language. The function of this rule-set is to allow migration from Host1 to either Host2 or Host3. The action will be invoked which requests to migrate a virtual machine from Host1 when the conditions are satisfied.

2.2 Domain Specific Languages DSLs

Domain-specific languages (DSLs) are languages which are designed to be used in a specific application domain. DSLs languages provide a special feature in terms of the expressiveness and simplicity compared with general-purpose programming languages (Mernik et al., 2005). Using DSLs have several advantages. They can speed up the development time since the language is designed to be used for specific environment. In addition, the language can assist to reduce the number of domain and programming expertise which are required (Mernik et al., 2005). Furthermore, the domain-language is an extendible and

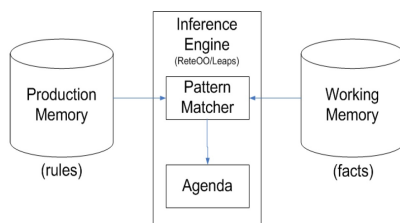


Figure 2: The architecture of OO-RETE engine.

```

rule "POLICY1"
dialect "mvel"
salience 106
no-loop
when
    Host(name=="host1" && Violation_Percentage<20 && Energy_Consumption>2000)
then
    CloudManager.MigrateHost3OrHost2()
  
```

Figure 3: A sample of a constraint management policy expressed in Drools Language.

a machine readable which allows to build auto-code generation tools in order to reduce the development time (Mernik et al., 2005). To accomplish these features provided by DSLs, designing such languages requires the experience in both domain-knowledge and language development. In our research, we focus on domain-specific languages designed for Cloud-platform.

2.2.1 DSLs used in Cloud Computing

There are extensive research for proposing many DSLs for automating the deployment of applications into Cloud-environment. One of these languages is *Crawl* which is a part of *Cloud Crawler* environment proposed for automating the execution of applications performance test in IaaS used by Cloud application developers (Cunha et al., 2013). *Crawl* is a declarative and an extensible domain-specific language (DSL) to provide a high-level specification that captures all the technical important information for executing application performance tests (Cunha et al., 2013). Instances of these information are the configuration parameters and the quantity of the resources allocated to application components (Cunha et al., 2013). The language's textual notion is described via YAML. Furthermore, the language allows using XML and JSON to define new specification of test scenarios (Cunha et al., 2013).

Neptune is also another domain specific language (DSL) designed to automate the configuration and deployment HPC applications executed in Cloud (Bunch et al., 2011). The objective of *Neptune* is to provide portability and flexibility to the developers of HPC (Bunch et al., 2011). *Neptune* is a meta-programming extension of the Ruby programming language with a flexibility to run large number of ruby's libraries which are designed to communicate with Cloud- infrastructure (Bunch et al., 2011). *Neptune* programmes allow users to write Ruby scripting code and also *Neptune* programs can also be used in Ruby programs using *neptune* keyword. The programmes of *Neptune* is composed of one or more invocations for jobs to be processed in Cloud ser-

vices(Bunch et al., 2011). The language is integrated to run into AppScale which is an open-source Cloud environment that uses Google App Engine APIs(Bunch et al., 2011).

Pim4Cloud DSL is a Platform Independent Model for Cloud-based application which is designed using a component-based approach(Brandtzæg et al., 2012). A Cloud application- designer models the application by using *Pim4Cloud DSL*. Meanwhile, at the other side the available resources for the modelled application are specified by Cloud provider (Brandtzæg et al., 2012). The *Pim4Cloud* has an interpreter which is used to match the assigned resources to application requirement. *Pim4Cloud DSL* is implemented in to Scala which includes different set of codes for modelling different topology for Cloud applications (Brandtzæg et al., 2012). The syntax of the *Pim4Cloud DSL* starts by defining the application as an abstract class which can factorise the shared entities. Each application topology can extend the abstract class. *Pim4Cloud DSL* platform supports a static analysis for the modelled application and also allow the deployment of Cloud-component to be reused (Brandtzæg et al., 2012).

3 CloudMPL LANGUAGE

CloudMPL can be applied instead of using Plain English sentences for writing the management policy at early stage. CloudMPL is a textual language that specifically is designed to be used by Cloud domain-experts to describe management policies before interpreting them to an executable rule language. CloudMPL is enriched with domain vocabularies and expressive operators for expressing conditions and actions parts which are partially inspired by the RELAX Language(Whittle et al., 2010). CloudMPL is targeting to author management policies which will be executed in Infrastructure-as-a-Service(IaaS) Cloud model.

When Cloud-domain experts specify policies, basically they are concerned about if some specific conditions are met, based on resources that compose a Cloud infrastructure, and actions that might be taken upon these conditions. Therefore, CloudMPL is designed to focus on the specification of Cloud-related resources and policies. The specification of CloudMPL includes a domain-vocabulary, a meta-model, a set of operators, which are extracted from RELAX language to describe various types of conditions might be found in a management policy, and a grammar for using the language.

3.1 CloudMPL Meta-model

Figure 4 presents the meta-model for CloudMPL language. In the meta-model described in Figure 4, a *Resource* is an element that can be managed and/or monitored which could be a host, a virtual machine, a cluster or any kind of resource that can described in term of *Properties*. Those *Properties* should represent any kind of information that can be monitored or used to describe a *Resource*. Every property has a *Type* that can be either a built-in type, such as *Number* or *String*, or could be a *Resource* created by the user. Each *Policy* is composed of several *conditions*, detailed in terms of constraints. *Constraints* are three different types which are:

1. *Time*, when an expert is interested in time related events.
2. *Location*, which is an optional element, is to check if a given resource is in a specific location.
3. *Ordinal* where raw values or a status of a given property related to a resource are checked.

In this level of abstraction, an *Action* invocation is an equivalent to a method call of any high-level programming language. The actions definition is similar to a method signature description, embedded into a *ActionManager*, which will be equivalent to an interface of a high-level programming language. To fully implement the important actions, it would require for the experts a deep knowledge on programming skills as well as specific-language information.

Table 1 gives the set of CloudMPL elements, organised into *Statement*, *Time*, *Location* and *Constraint*. The statements define the blocks of the language in terms of the main elements for expressing the policies. The *Time* and *Location* operators define conditions for time and location. The *Constraint* conditions can be applied to ordinals or status information for basic comparison.

3.2 CloudMPL Syntax

The syntax of CloudMPL expressions are defined by the grammar in Figure 5. Manageable resources are described by the *Create Resource* statement, defined by an ID and their respective properties. The *ActionManager* requires an ID and at least one action(method) to be defined. Policies are identified by an ID and composed of CloudMPL operators. As showed in Table 1 CloudMPL has *Location*, *Time* and *Constraint* operators, that can be combined using logical operator such as *AND*, *OR*, *NOT* and parenthesis. If the specified conditions are satisfied an action that belong to an *ActionManager* should be invoked.

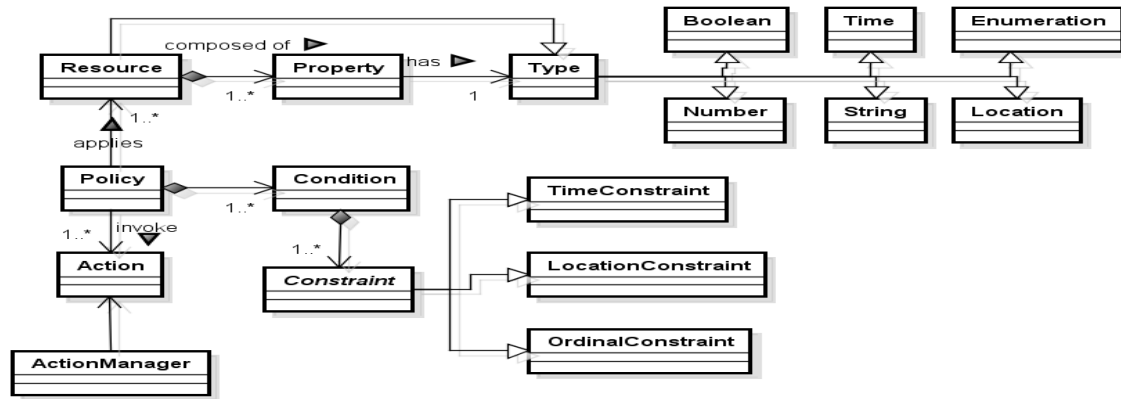


Figure 4: CloudMPL meta-model.

<CLOUDMPL>	::=	<RESOURCE> {<RESOURCE>} <ACTIONMANAGER> {<ACTIONMANAGER>} <CREATE_RESOURCE> {<CREATE_RESOURCE>} <POLICY> {<POLICY>}
<RESOURCE>	::=	DEFINE RESOURCE <ID> AS <PROPERTY> {<PROPERTY>}
<PROPERTY>	::=	<ID> AS <TYPE>
<ACTIONMANAGER>	::=	DEFINE ACTIONMANAGER <ID> AS <ACTION> {<ACTION>}
<ACTION>	::=	ACTION <ID><PARAMETERLIST>
<POLICY>	::=	POLICY <ID> IF <CONDITIONS> THEN <ACTION.INVOCATION> {<ACTION.INVOCATION>}

Figure 5: A simplified EBNF grammar for CloudMPL.

The following steps are required for applying CloudMPL:

- **Step 01:** Modelling the resources that are going to be managed and used to describe the policies.
- **Step 02:** Definition of the possible actions used in case a policy condition is true. The Actions are created by defining an *ActionManager* element.
- **Step 03:** Writing the policies using the operators and the syntax for the CloudMPL language. Each policy should be identified by a valid ID.

In order to illustrate the usage of CloudMPL, consider the following policy:

- **Policy1:** if the violation percentage rate is less than 20% in Host1 and the energy consumption is above than 2000 kwph in Host1 then it is necessary to migrate the contents of Host1 to Host3 or Host2

```

1  DEFINE RESOURCE Host AS
2  BEGIN
3  Violation_Percentage AS Number
4  Energy_Consumption AS Number
5  END
6  DEFINE ACTIONMANAGER Manager AS
7  BEGIN
8  ACTION Migrate_Alternative_Host Host origin,
9  Host alternativeA , Host alternativeB
10 END
11 CREATE host1,host2,host3 AS Host
12 POLICY policy1
13 IF ((host1.Violation_Percentage FEW_AS 20)
14     AND (host1.Energy_Consumption MANY_AS
15         2000)) THEN
16   Manager.Migrate_Alternative_Host host1 ,
17   host2 , host3

```

Figure 6: CloudMPL representation for the given policy.

Using CloudMPL grammar defined in Figure 5, Figure 6 presents *Policy* written in CloudMPL.

First it is necessary to create a resource called *Host* (lines 1 to 5) which describes a computer host in a private Cloud infrastructure. Considering the given policy the manager is interested in monitoring the energy consumption and the rate of violation(error rate). As stated in the policy definition the action that needs to be done if the conditions are met is to migrate contents of Host1 to Host2 or Host3. This action in CloudMPL is embedded in an *ActionManager*, called *Manager*, which has a single method *Migrate_Alternative_Host* (lines 6 to 10). Considering that the policy requires the usage of three host, it is necessary to create them (line 11). Finally the policy itself (lines 12 to 14) is defined in terms of the resources and properties, using *Constraint* operators. The CloudMPL language was implemented using XText(Xtext, 2014) and the current implementation is available for download at <http://www.dimap.ufrn.br/splmonitoring/adaptmcloud/index.php>

Table 1: CloudMPL Operators and Elements for Describing Management Policies.

CloudMPL Operators & Statements	Description
Statements: DEFINE RESOURCE <i>id</i> AS BEGIN <i>property</i> AS <i>type</i> END DEFINE ACTIONMANAGER <i>id</i> AS BEGIN ACTION <i>id.parameters</i> END CREATE <i>id</i> as <i>resource</i> POLICY <i>id</i> IF <i>conditions</i> THEN <i>actions</i>	Declares a Resource, by specifying its ID and properties Declares an <i>ActionManager</i> if a set of actions. Each action is declared with a ID and a set of parameters Creates a <i>Resource</i> with the given ID Declares a policy with the given ID and starts the policy conditions definition, followed by a action(s) call(s).
Time Operators: <i>resource.time</i> AFTER <i>threshold</i> <i>resource.time</i> BEFORE <i>threshold</i> <i>resource.time</i> BETWEEN <i>threshold_a</i> TO <i>threshold_b</i>	Checks if the property of <i>Time</i> type is after the given threshold Checks if the property of <i>Time</i> type is before the given threshold Checks if the property of <i>Time</i> type is in the specified interval.
Location Operator: <i>resource.location</i> IN <i>location</i>	Checks if a given <i>Resource</i> is in the specified <i>location</i> .
Constraint Operators: <i>resource.property</i> FEW_AS <i>value</i> <i>resource.property</i> MANY_AS <i>value</i> <i>resource.property</i> IS <i>status</i>	Checks if the ordinal representation for a property is less than a given value Checks if the ordinal representation for a property is greater than a given value Checks if the a given property is within one of the following status: HIGH, NORMAL or LOW

4 THE SPECIFICATION OF MANAGEMENT POLICIES IN RULE LANGUAGE

Management policies expressed in CloudMPL can be directly mapped to executable management policies by designing mapping rules between two languages. To produce such policies, any CloudMPL policy is encoded as a special form of production rules (more information about the productions rules can be found in (REWERSE, 2012)). These production rules fol-

Table 2: The syntax for conditional expressions for a management policy.

Expression	<property:>	<operator:>	<value.expr:>
Constraints-1:	monitorable.prams	Comparison	DataTerm
Constraints-2:	monitorable.pram_Level	Specification	DataTerm
SelectTargetHost:	id or name	TargetHost Selection	DataTerm
TargetHost Location:	location	Location	DataTerm
TargetHostTime:	current_time_status	Time	ObjectTerm

low a defined specification for formulating both condition and action parts in Rule Language which can be applied to either Drools(JBossCommunity, 2011) or JRules (IBM-ILOG, 2007). In this work, we briefly discuss the specification for formulating conditions and action parts for a policy in an executable rule language. The specification includes the meta-model for conditions, the classification for operators used in conditions and the types of actions used by a management policy. Both CloudMPL and rule language specification will be used for designing a translator from CloudMPL to Drools or JRules in future.

4.1 Conditions Meta-model for a Policy

Conditions for a management policy can be expressed through the meta-model presented in Figure 7 which is inspired from URML meta-model (REWERSE, 2012). This meta-model is resulted from our classification for general rules used for management purposes in Cloud.

In Figure 7, a single condition in a management policy is a boolean expression which can be composed with other conditions by using *composition operators*. From URML rule meta-model (REWERSE, 2012), we extracted some elements for modelling various conditions. These elements are *Term*, *DataTerm*, *ObjectTerm*, *uml_property*, and *Object Variable* (REWERSE, 2012).

In Figure 7, the conditional expression are classified into five types. Each expression in the condition meta-model uses a property. The property are extracted from Target Host that is running in a Cloud-platform. In addition, each expression also has a *value.expr* which might be of the following types: Data Term, Object Term, or uml property. Furthermore, suitable operators are grouped to match each conditional expression type. The operators are presented in Figure 8. Thus, by using both Figure 7 and Figure 8, each conditional expression and its syntax are explained as follows:

1. **ConstraintsExpr:** a comparative condition used to compare monitorable parameters against a

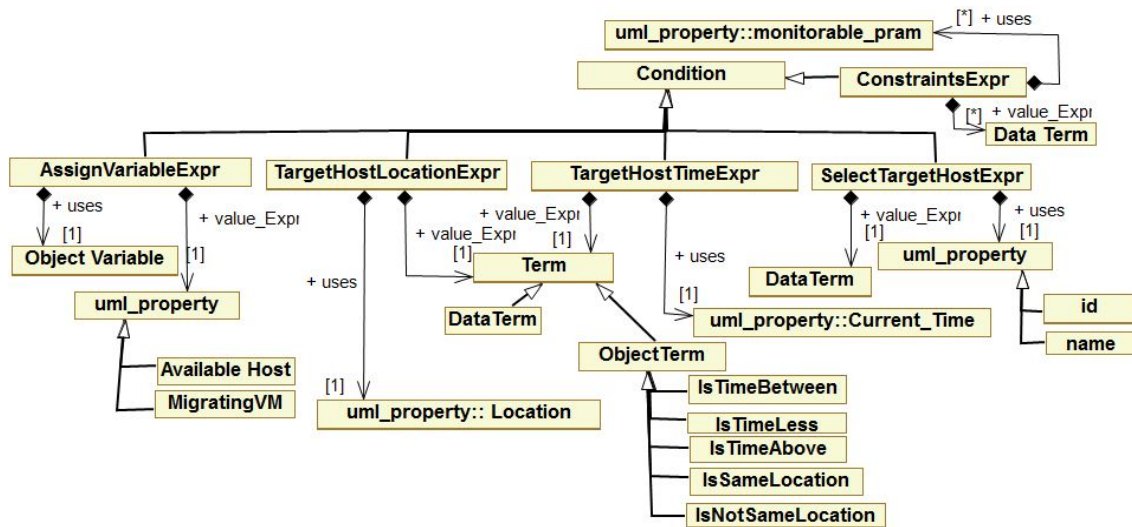


Figure 7: The meta-model for conditions of a management policy in Rule Language (Drools and JRules).

specified threshold value or to specify status of monitorable parameters. Examples for monitorable parameters are `CPU_Usage`, `Violation_Percentage` and `Energy_Consumption`. `ConstraintsExpr` has two different syntaxes, which are presented in Table 2.

2. **SelectTargetHostExpr:** an identification expression, which is used to select a targeted host. This expression is necessary to be included in a management policy. The expression syntax is shown in Table 2.
3. **AssignVariableExpr:** a selection expression, which is to get values from some properties and to assign them to an object variable. This expression can be used in a management policy for extracting variables which are required by management APIs. The expression is optional to be included in the policy. The syntax for the expression is different from the syntax for other expressions which is as follows:
$$< operator : Assignment\$ > < property : ObjectVariable > < operator : Assignment : > < property : ObjectTerm >$$
4. **TargetHostLocationExpr:** a location based expression, which is used to specify a geographical location of a target host. Since a physical Cloud host can be located at any location around the world, the policy meta-model should allow an option for such a restriction. This expression is an optional in the policy based on the requirement. The syntax for the expression is shown in Table 2 where its Data Term can be either of String type or GeoLocation which is Enumeration type. An example for TargetHostLocation is

```
location == GeoLocation.Asia.
```

5. **TargetHostTimeExpr:** a time-based expression that specifies the time status at a target host. Any target host in Cloud-platform has some operations to deal with time expression. These operations are *IsTimeBetween(< Time_Begin >, < Time_End >)*, *IsTimeLess(< Time_Literal >)*, and *IsTimeAbove(< Time_Literal >)*. The syntax for the time_based expression is presented in Table 2. The following expression is a simple expression for checking time status:
- ```
current_time_status== IsTimeBetween(16.00,23.00)
```

## 4.2 Policy Action Description

The action of a management policy in a rule language is expressed as action expressions. These action expressions can be either expressions for assigning values or expression for invocation actions. To simplify the transformation process for future, we only use invocation actions expression from the rule language which is denoted as *InvocationActionExprin* (REWERSE, 2012). In any management

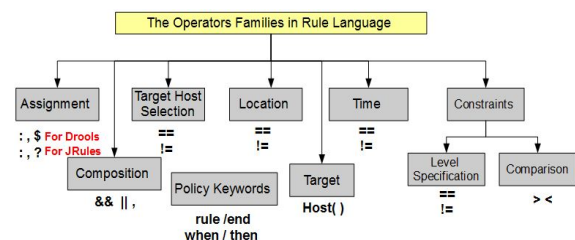


Figure 8: The possible common rule language operators families for expressing a management policy.



Table 3: Operations used by Cloud Manager instance in a management policy described in Drools.

| < OperationName >       | < Parameters : Type >                                                     |
|-------------------------|---------------------------------------------------------------------------|
| Migrate                 | Original: TargetHost                                                      |
| MigrateAlternativeHosts | Original: TargetHost , Destination1: TargetHost, Destination2: TargetHost |
| MigrateToLocation       | Original: TargetHost , LocationName: String                               |
| ReportingNoMigration    | Original: TargetHost                                                      |
| Calculate               | Original: TargetHost                                                      |

policy, the invocation action expressions include calls for management APIs/Operations specified in Cloud manager. The syntax used for expressing *InvocationActionExpr* is:

CloudManager.Operation\_Name(parameters);

In this syntax, *CloudManager* represents the instance of a Cloud manager which has a number of management operations. Each defined operations for *CloudManager* instance has a number of parameters which are necessary for migration of virtual machines, reporting information and calculating service at the target host side in Cloud-platform. Table3 shows each defined operation and its related parameters.

## 5 THE TRANSFORMATION FROM CloudMPL TO DROOLS

After presenting CloudMPL language for expressing management policies at the description level and the specification for formulating such policies at the implementation level, the transformation between CloudMPL and Drools is proposed. The objective is to build the foundation for automatically generating an executable management policy from the description level.

The transformation process requires to have a conceptual mapping from CloudMPL to Drools Language. Therefore, designing a set of mapping rules from CloudMPL to Drools for both conditions and actions parts is necessary. To design these mapping rules, we used CloudMPL meta-model as well as its syntax and Drools specification mentioned in Section 4. Firstly, the mapping step starts by presenting the

Table 4: Mapping generic syntax and keywords for a policy in CloudMPL to Drools.

| CloudMPL Generic Syntax |   | Drools Generic Syntax   |
|-------------------------|---|-------------------------|
| <b>POLICY</b> < ID >    | → | <b>rule</b> < ID >      |
| <b>IF</b>               | → | <b>when</b>             |
| < CONDITIONS >          |   | < Host(Conditions) >    |
| <b>THEN</b>             | → | <b>then</b> < Actions > |
| < ACTION INVOCATION >   |   |                         |
| {< ACTION INVOCATION >} |   | end                     |

Table 5: Mapping CloudMPL conditions to Drools conditions using Table 1 in Section 3 and Table 2 in Section 4.

| CloudMPL Expression                                                          | Drools Expression      |
|------------------------------------------------------------------------------|------------------------|
| <attribute:TIME>                                                             | TargetHostTimeExpr     |
| AFTER <value:threshold>                                                      |                        |
| <attribute:TIME>                                                             | TargetHostTimeExpr     |
| BEFORE <value:threshold>                                                     |                        |
| <attribute:TIME> BETWEEN<br>< value: threshold.a > TO<br><value:threshold.b> | TargetHostTimeExpr     |
| IN <value:ID>                                                                | SelectTargetHostExpr + |
| <value:location>                                                             | TargetHostLocationExpr |
| <attribute:monitorable> FEW_AS<br>  MANY_AS <value:threshold>                | ConstraintsExpr-1      |
| <attribute:monitorable> IS <value:status>                                    | ConstraintsExpr-2      |

mapping for the general syntax for a management policy and keywords in both CloudMPL and Drools.

Table4 shows the mapping of generic syntax and special keywords from CloudMPL to Drools. It is noticeable from the generic syntax in Table4 that any statement between *IF* and *THEN* is mapped as Conditions in Drools which it should be enclosed with Target operator mentioned in Figure 8. Furthermore, any statement after *THEN* is mapped as Actions in Drools. The mapping of both conditions and actions requires more explanation which will be appeared in the following subsections.

### 5.1 Mapping Conditions

In CloudMPL, a condition block consists of one or more condition. Therefore, any condition in CloudMPL can be structured as an attribute, an operator and a value. The attribute in CloudMPL is usually written before CloudMPL operator. Usually, the value is after CloudMPL operator. Thus, the mapping for conditions is shown in Table 5 which applies the following mapping rules:

1. The dot operator < . > in CloudMPL is mapped as '==' operator and <value:ID or Name> is mapped as <value.expr: DataTerm>.
2. *After* operator is mapped as '==' combined with <ObjectTerm:IsTimeAbove> in Drools.
3. *Before* is mapped as '==' combined with <ObjectTerm:IsTimeLess> in Drools.
4. *BETWEEN, TO* operator is mapped as '==' combined with <ObjectTerm: IsTimeBetween>.
5. < value: threshold > is mapped as <Time\_Literal> parameter for both IsTimeAbove and IsTimeLess in Drools.
6. < value: threshold.a > and <value:threshold.b> are mapped as <Time\_ Literal\_ Begin> and

<Time.Literal.End> parameters for IsTimeBetween in Drools.

7. <attribute:Time> is mapped as <property: current\_time\_status>.
8. *IN* is mapped as '==' operator and <value:location> is mapped as <value\_expr: DataTerm> which can be either String or Enumeration.
9. *FEW\_AS* or *MANY\_AS* are mapped as Comparison operators.
10. <attribute:monitorable> is mapped as <property: monitorable\_parms> and <value: threshold> is mapped as DataTerm.
11. *IS* operator is mapped as '==' operator and <value:status> is mapped as <DataTerm: Status>.
12. <attribute:monitorable> in *IS* expression is mapped as <property: monitorable\_Parms\_status>.
13. *AND* and *OR* is mapped as && and || operators in Drools receptively.

To elaborate the mapping of policy condition, we provide a sample of conditions which are written in both CloudMPL and Drools in Figure 9. These conditions are for expressing a management policy extracted from Energy Management Running Example presented in Section 6. In Figure 9, the first statement is CloudMPL expression for three conditions, which are Violation\_Percentage *FEW\_AS* 20, Energy\_Consumption *MORE\_AS* 2000, *host1*. These conditions are composed in CloudMPL by *AND* operator. The same figure also includes conditions expressed in Drools which map CloudMPL conditions. In Figure 9, the arrows represent the types of the mapping rules that can be applied.

## 5.2 Mapping Actions

In CloudMPL, any action statement is mapped as a call method for management operations in a policy expressed in Drools. The mapping rules for actions are:

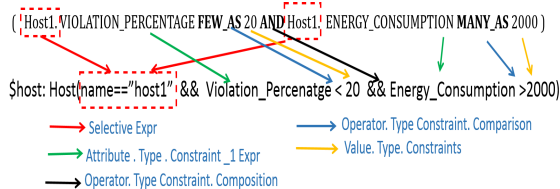


Figure 9: Using mapping rules for mapping CloudMPL condition block to Drools condition part for a policy.

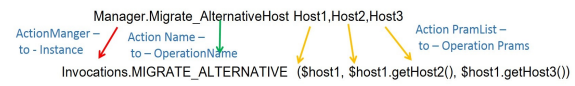


Figure 10: Using mapping rules for mapping CloudMPL action block to Drools action part for a policy.

1. Action <ID> in CloudMPL is mapped as the Name of the operation in CloudManager (See Figure 10).
2. The parameters List, which includes a set of Parameter Expression, is mapped as the operation parameters in CloudManager.
3. In CloudMPL, Parameter Expression consists of <TypeID>. Type is mapped as either <ObjectTerm> or <getObjectRef> in Operation. Whilst ID is mapped as the name of the parameter.
4. In CloudMPL, if Type is Host and it is the first parameter in the statement, then it is mapped as the Original and its type is TargetHost in Drools.
5. In CloudMPL, if Type is Host and is not the first parameter, then it is mapped as either to Destination1 or Destination2 based on ordering parameters in Drools.

Figure 10 illustrates applying the mapping rules for action from CloudMPL to Drools. The figure includes an operation defined in Table 3. The operation has three parameters which are Host1, Host2, Host3. In Drools mapping, Action ID is mapped as Migrate\_Alternative\_Host. Whilst the first parameter is mapped as \$host1. Both Host2 and Host3 are mapped as \$host1.getHost2() and \$host1.getHost3(), respectively.

The previously mentioned mapping rules for both conditions and actions parts will be used to design an interpreter to automatically or semi-automatically generate Drools codes for policies that would be executed into the architecture shown in Figure 1. The Drools code generation will be a future step.

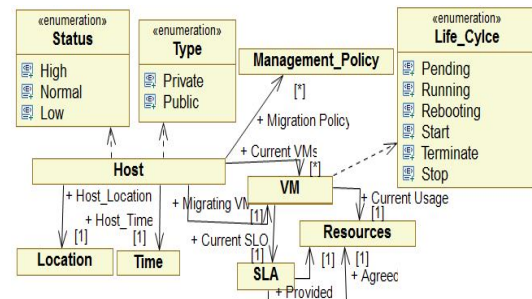


Figure 11: An UML class diagram for Cloud infrastructure used in the case study.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> 1 Define Resource Host as 2 begin 3 Violation_Percentage as Number 4 Violation_Status as Status 5 CPU_Usage as Status 6 Energy_Consumption as Number 7 Current_Time as Time 8 End 9 DEFINE ACTIONMANAGER Manager AS 10 BEGIN 11 ACTION Migrate_Alternative_Host Host origin, 12 Host alternativeA , Host alternativeB 13 ACTION NoMigrate 14 ACTION MigrateLocation Host origin, Location location 15 END 16 CREATE host1,host2,host3 AS Host </pre>                               | <pre> 18 POLICY policy1 19 IF ((host1.Violation_Percentage FEW_AS 20)AND (host1.Energy_Consumption MANY_AS 2000)) 20 THEN 21 Manager.Migrate_Alternative_Host host1 , host2, host3 22 23 POLICY policy2 24 IF ((host1.Current_Time AFTER 23:00)AND ((host1.Violation_Status IS HIGH) OR 25 (host1.CPU_Usage is HIGH))) THEN Manager.NoMigrate 26 27 POLICY policy3 28 IF ((host1.Current_Time BETWEEN 01:00 TO 07:00)AND ((host1.Violation_Status 20 IS HIGH) OR 29 (host1.CPU_Usage is HIGH))) 30 THEN Manager.Migrate_Alternative_Host host1 , host2, host3 </pre> |
| (a) CloudMPL Declarations                                                                                                                                                                                                                                                                                                                                                                                                                                                                | (b) CloudMPL Expressions_1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <pre> 32 POLICY policy4 33 IF ((host1.Current_Time AFTER 23:00)AND ((host1.Violation_Status IS HIGH) 34 OR (host1.CPU_Usage is HIGH))) THEN 35 Manager.NoMigrate 36 37 POLICY policy5 38 IF ((host1.Current_Time BEFORE 09:00)) THEN 39 Manager.NoMigrate 40 41 POLICY policy6 42 IF ((host1.Location in 'ASIA') and (host1.Current_Time Between 16:00 to 43 23:00) and ((host1.Violation_Status IS HIGH) OR (host1.CPU_Usage is HIGH))) 44 Service.MigrateLocation host1, 'ASIA' </pre> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| (c) CloudMPL Expressions_2                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

Figure 12: A sample of CloudMPL (XText) for management policies used in the case study.

## 6 CloudMPL IN PRACTICE

CloudMPL is used as a language for expressing a number of management policies extracted from Management Energy Consumption Case Study presented in (Alansari and Bordbar, 2013). The Management Energy Consumption Case Study implemented in OpenNebula(Toraldo, 2012) which is a Cloud-platform management system. In the case study, there are a set of management policies which are enforced in Drools rule-engine which periodically responses to changes in three monitored parameters. The parameters are average CPU\_usage for running VMs, Violation Rate and Average Energy Consumption for private hosts per hour. The engine automatically controls the migration action for running virtual machines deployed on three private physical hosts. The hosts runs Ubuntu OS and KVM as virtualization environment. The policies should trigger a migration action for running virtual machines and may switch off idle hosts. The policies which are provided in the case study are

written in Drools Rule Language (Alansari and Bordbar, 2013)). Figure 11 presents the UML model for the Cloud-infrastructure for the used example.

In Figure 11, the Cloud infrastructure consists of three main components, which are Host, VM images and SLA classes. The Host class runs a number of virtual machines and has a number of monitorable attributes such as CPU\_Usage, Energy\_Consumption,etc.). Furthermore, Host class has a time and also a location attribute, which represents the geographical location for the host. Each running virtual machine in the model belongs to only one Cloud consumer. Thus, each running VM instance is associated with only one SLA and also has a life cycle ( For more details about Figure 11 see (Alansari and Bordbar, 2014)).

We took some management policies encoded into Drools and we used CloudMPL to write them. Both CloudMPL Declarations and CloudMPL Policies for the case study, which are implemented using XText(Xtext, 2014), are shown in Figure 12.

Looking at Figure 12, there are six policies expressed in CloudMPL which demonstrate the usage of all operators suggested by the language. Policy 1 is constraints and it requires to use monitorable parameters and uses the CloudMPL operators *FEW\_AS* and *MANY\_AS*. Whilst policy 2, policy 3 and policy 4 are composed of both time and constraints expressions. Both policy 2 and policy 4 use the operator *After* whereas policy 3 includes the CloudMPL time operator *Between / To*. The constraints operator used in these policies is *IS*. Policy 5 has only a single time expression which uses the CloudMPL time operator *Before*. The final policy, which is policy 6, contains a location expression besides the time and constraints conditions. The location condition uses the CloudMPL operator *IN*.

### 6.1 The Interpretation of CloudMPL Policies to Drools

We applied the mapping process introduced in Section 5 to the policies of the case study. Using the designed mapping rules explained in Section 5 and the meta-model presented in Figure 11, Drools codes are generated manually. The purpose is to test the mapping rules. A sample of Drools code for policy 1, policy 3, policy 5 and policy 6 are shown in Figure 13 which are depicted into two groups. The important Drools operators used in the conditions are highlighted in Blue.

Taking policy 3 as an example, this policy is mapped as a combination of Time and Constraints.2 Expressions which are shown in management policies conditions meta-model mentioned in Section 4. The mapping for the condition part applies the rules number 1, 4, 6, 7, 11, 12 and 13 explained in Mapping Conditions at Section 5. On the other hand, all the rules proposed for mapping the action part expressed in Section 5 are applied. As a result, policy 3 will have a rule code similar to what is illustrated in Figure 13. This method is applied to all remaining CloudMPL policies captured in Figure 12.

## 7 CONCLUSION

This paper aims at reducing the existing gap between the specification of management policies for Cloud and the implementation of policies via rule-engines. We presented CloudMPL that is a domain-specific language for specifying management policies for Cloud-environments. Furthermore, we described a method of automating the creation of various CloudMPL expressions to an executable Rule

```
rule "POLICY1"
when
 $host: Host(name=="host1" && Violation_Percentage <20 && Energy_Consumption >2000)
then
 Invocations.MIGRATE_ALTERNATIVE ($host1, $host1.getHost2(), $host1.getHost3())
rule "POLICY2"
when
 $host1: Host (name=="host1" && current_time_status==IsTimeBetween(01.00, 07.00) &&
 Violation_Status ==Status. High || CPU_Usage.Status==Status. High)
then
 Invocations.MIGRATE_ALTERNATIVE ($host1, $host1.getHost2(), $host1.getHost3())
```

(a) Drools Rules "Group\_1"

```
rule "POLICY1"
when
 $host: Host(name=="host1" && Violation_Percentage <20 && Energy_Consumption >2000)
then
 Invocations.MIGRATE_ALTERNATIVE ($host1, $host1.getHost2(), $host1.getHost3())
rule "POLICY2"
when
 $host1: Host (name=="host1" && current_time_status==IsTimeBetween(01.00, 07.00) &&
 Violation_Status ==Status. High || CPU_Usage.Status==Status. High)
then
 Invocations.MIGRATE_ALTERNATIVE ($host1, $host1.getHost2(), $host1.getHost3())
```

(b) Drools Rules "Group\_2"

Figure 13: The generated Drools rules from CloudMPL policies.

Language such as Drools. CloudMPL establishes a set of operators that deal with several kinds of constraints, from ordinal, ranging through time and locations constraints, that can be applied to a specific or a set of Cloud resources. In addition, CloudMPL supports user-defined actions to deal with the consequences of conditions specified by the managers of Cloud computing infrastructure. The usage of both CloudMPL and the automated approach, which is based on designing mapping rules between CloudMPL and Drools, is illustrated with the help of an example related to management energy consumption by migrating virtual machines.

It is also important to highlight that, even without the transformation process fully implemented, the suggesting method for automating policy creation assists to cope with frequent changes in policies specified at the description level. Using both CloudMPL and the automated process for creation management policies can decrease the amount of time that requires to spend in implementing such policies. Due to the closeness of CloudMPL to natural languages and its declarative nature, it is easier for non-technical people to make a use of the language for specifying policies. As a future step, it is imperative to fully implement the transformation process and also to build an integrated environment that supporting the specification, translation and deployment of the policies.

## REFERENCES

- Alansari, M. and Bordbar, B. (2014). Modelling and analysis of migration policies for autonomic management of energy consumption in cloud via petri-nets. In *Proceedings of the The International Conference on Cloud and Autonomic Computing*. IEEE.
- Alansari, M. M. and Bordbar, B. (2013). An architectural framework for enforcing energy management policies in cloud. *2013 IEEE Sixth International Conference on Cloud Computing*, 0:717–724.
- Beloglazov, A. and Buyya, R. (2010). Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*. ACM.
- Borgetto, D., Maurer, M., Da-Costa, G., Pierson, J.-M., and Brandic, I. (2012). Energy-efficient and sla-aware management of iaas clouds. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet, e-Energy '12*, pages 25:1–25:10, New York, NY, USA. ACM.
- Brandtzæg, E., Mohagheghi, P., and Mosser, S. (2012). Towards a domain-specific language to deploy applications in the clouds. In *Cloud Computing 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 213–218.
- Bunch, C., Chohan, N., Krintz, C., and Shams, K. (2011). Neptune: A domain specific language for deploying hpc software on cloud platforms. In *Proceedings of the 2Nd International Workshop on Scientific Cloud Computing*, ScienceCloud '11, pages 59–68, New York, NY, USA. ACM.
- Cunha, M., Mendonca, N., and Sampaio, A. (2013). A declarative environment for automatic performance evaluation in iaas clouds. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 285–292.
- Forge, C. L. (1982). Rete : A fast algorithm for the many patternmany object pattern match problem. *Artificial Intelligence*, 19:17–37.
- IBM-ILOG (2007). Ilog jrules techincal. <http://logic.stanford.edu/poem/externalpapers/iRules/WP-JRules50Strengths.pdf>.
- JBossCommunity (2011). Drools tools reference guide.
- Maurer, M., Brandic, I., and Sakellariou, R. (2013). Adaptive resource configuration for cloud infrastructure management. *Future Generation Computer Systems*, 29(2):472 – 487.
- Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344.
- Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., and Yuan, L. (2010). Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *IEEE International Conference on Services Computing*, pages 514–521. Ieee.
- REVERSE (2012). Uml-based rule modeling language. <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=URML>.
- Toraldo, G. (2012). *OpenNebula 3 Cloud Computing*. PACKT Publishing, Birmingham B3.
- Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H., and Bruel, J.-M. (2010). Relax: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering*, 15(2):177–196.
- Xtext (2014). Xtext textual domain-specific language (dsl). <http://www.eclipse.org/Xtext/>.

# Dynamic Testing and Deployment of a Contract Monitoring Service

Ellis Solaiman<sup>1</sup>, Ioannis Sfyarakis<sup>1</sup> and Carlos Molina-Jimenez<sup>2</sup>

<sup>1</sup>*School of Computing Science, Newcastle University, Newcastle, U.K.*

<sup>2</sup>*Computer Laboratory, University of Cambridge, Cambridge, U.K.*

{ellis.solaiman, i.sfyarakis}@ncl.ac.uk, carlos.molina@cl.cam.ac.uk

**Keywords:** Service Agreement, Electronic Contract, Service Monitoring, Model Checking, Automated Testing, Service Oriented Computing, Cloud Computing.

**Abstract:** Internet and cloud based services involve electronic interactions that are normally regulated using service agreements (SA). Once an agreement between business partners is in place, a service can be monitored and/or enforced using an SA equivalent electronic contract. Because of the dynamic nature of such Internet and cloud based relationships, the rapidity at which electronic contracts are constructed, verified for correctness, tested, and deployed is an extremely important factor. This paper describes a model checker based framework for supporting the automated testing and deployment of electronic contracts. The central components of the framework are a contract monitoring service called the *Contract Compliance Checker* (CCC), the SPIN model checker, and EPROMELA, a language developed specifically for modeling electronic contracts. We describe how SPIN can be used to automatically generate execution sequences from an EPROMELA model of a contract, and how such sequences can then be used to test the correctness of the model equivalent electronic contract deployed to the CCC.

## 1 INTRODUCTION

Internet and cloud computing advances have made it possible for businesses to provide infrastructure and software services to their business partners and to their customers at affordable costs. Before such business relationships can commence, legal service agreements (SA) need to be negotiated and agreed. Legal agreements, explicitly define the permissible actions of the interacting parties, thus providing a legal basis for the resolution of any disputes. A Legal agreement can also be used as a guide for developing an electronic contract (Molina-Jimenez et al., 2003).

The main purpose of an electronic contract is to regulate (monitor and/or enforce) electronic service exchanges between the contracted parties, making sure that business participants adhere to the SA in place, and that performed actions comply with various message timing and sequencing constraints. Electronic contracts are not confined to the business domain, and can also be used for example to monitor/enforce SAs between the components of distributed systems in the cloud and/or the "Internet of Things".

Constructing an electronic contract that is correct (free from conflicts, and which correctly represents the requirements of the original legal document), is a

challenging and time consuming task. Cloud based business relationships can be both complex and of a highly dynamic nature (Molina-Jimenez et al., 2011) therefore it is important that the process of converting a legal document into an electronic contract that is correct, is automated as much as possible. Previous work towards this goal has been extensive, and has covered problems such as electronic contract representation and modeling (Strano et al., 2008), and contract model verification (Solaiman et al., 2003) (Abdelsadiq et al., 2011). Naturally, ensuring that a model of an electronic contract is correct, does not guarantee that the electronic contract itself is also correct. In this paper, we focus on the challenge of ensuring that an electronic contract acts correctly at run time, and that modifications and/or corrections that need to be made to the rule base of the electronic contract can be applied quickly. To this end, we develop a model checker based framework to support automatic electronic contract deployment and testing.

The central component of our framework is the *contract compliance checker* (CCC) (Fig. 1) (Strano et al., 2009) (Molina-Jimenez et al., 2012), which essentially is the System Under Test (SUT). The CCC is an independent contract monitoring service that when provided with an executable specification of a con-



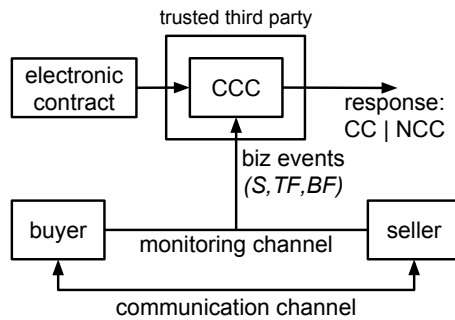


Figure 1: The CCC deployed as a contract monitor.

tract, can be deployed by the contracted parties or by a third party. The CCC is able to observe and log relevant interaction events, which it processes to determine whether the actions of the business partners are consistent with respect to the rights, obligations, and prohibitions declared in the original legal contract. Namely, the CCC declares interaction events as either contract compliant (*CC*) or non contract compliant (*NCC*).

As can be seen in Fig 1, business partners use a communication channel for exchanging their business messages. In addition they use a monitoring channel for notifying events of interest to the CCC. Notably, the figure shows that the CCC can cope with exceptions and failures, observing events that have been declared by the interacting parties as either *S* (*successful*), *TF* (*technical failure*), or *BF* (*business failure*).

The ability of the CCC to correctly declare interaction events as (*CC*) or (*NCC*) relies on an executable contract that has been specified correctly. Our goal is to provide a framework that enables; rapid testing of a deployed executable contract, and rapid update of the contract rules when testing detects errors. To do so, one must be able to exhaustively supply the CCC with execution sequences that it would be expected to observe during runtime. Our approach is to resort to model checker based testing.

Previous research into the area of model checker based testing of electronic contracts, (Abdelsadiq et al., 2010), describes the basic idea: construct a behavioural model of the SUT and validate the behaviour using a model checker. Such a validated model can then be used for generating executable test cases for the SUT.

The model checking tool we use is SPIN (Holzmann, 2003), a tool originally designed for the verification of communication protocols. SPIN's input language, Promela, provides constructs for modeling communication concepts such as messages, channels, and basic data types that include bit, byte, arrays. etc. Using these basic constructs alone for modeling elec-

tronic contracts, at a sufficiently high level of abstraction, is extremely challenging. This in turn makes the process of generating accurate execution sequences required for testing the CCC difficult.

To address these challenges, a fundamental component of our testing framework is EPROMELA, a high level language developed specifically for modeling electronic contracts (Abdelsadiq et al., 2011). EPROMELA extends Promela with constructs for expressing core electronic contract concepts contained in the CCC, thus enabling the construction of a contract model at a level of abstraction that is equivalent to the actual electronic contract.

This paper makes two contributions: 1) we describe the architecture of the CCC service, highlighting architectural improvements we have made that allow for dynamic update of rules coded in an electronic contract specification. 2) we describe how SPIN, and EPROMELA, can be instrumented with the aid of appropriate automation and message parsing tools, to produce business events that can accurately test the executable contract deployed within the CCC service.

The remainder of the paper is structured as follows: In Section 2 we describe key electronic contracting concepts with the aid of a simple example. In Section 3 we present the enhanced architecture of the CCC. Section 4 is dedicated to presenting our model checker based testing framework. In Section 5 we discuss related work. Conclusions and future work suggestions are presented in Section 6.

## 2 BACKGROUND

In order to elaborate key electronic contracting concepts, we present a simple scenario. Let us assume that Fig. 1 describes a relationship where two organisations, a Buyer and a Seller (a store), agree to a business contract. Below are some of its clauses:

1. The buyer can place a **buy request** with the store to buy an item.
2. The store is obliged to respond with either **buy confirmation** or **buy rejection** within 3 days of receiving the buy request.
  - (a) No response from the store within 3 days will be treated as a buy rejection.
3. The buyer can either **pay** or **cancel** the buy request within 7 days of receiving a confirmation.
  - (a) No response from the buyer within 7 days will be treated as a cancellation.

The clauses of such a legal agreement should take into consideration all relevant business operations (shown

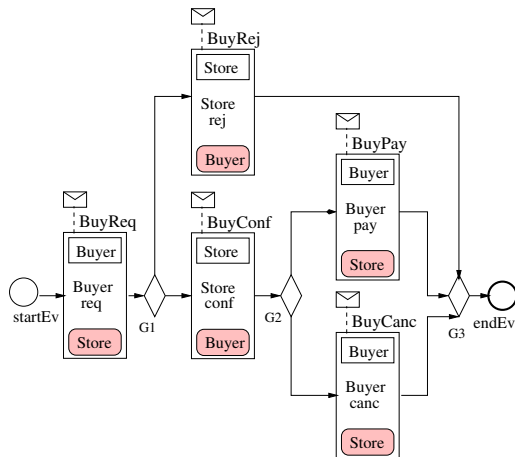


Figure 2: Correct choreography of contract example.

in bold in the contract text). A contract distinguishes operations as *Rights*, *Obligations*, *Prohibitions* (the *ROP* set). A *Right* is an operation that a party is allowed to perform under certain conditions, an *Obligation* is an operation that a party is expected to do under certain conditions, and a *Prohibition* is an operation that a party is not allowed to do under certain conditions.

To support our discussion, we will use a graphical representation of the contract written in BPMN (Business Process Management Notation) choreography language (OMG, 2011) (see Fig. 2).

The figure involves five activities, each resulting in a message (*BuyReq*, *BuyRej*, *BuyConf*, *BuyPay*, *BuyCanc*) being sent from a sender (shown as a white label in each activity), to a receiver (shown as a shaded label). These messages correspond to the five business operations (buy request, buy reject, buy confirmation, buy payment, buy cancellation) shown in bold in the English text of the contract. The diamonds in the figure are gateways. The figure includes two exclusive fork gateways (G1 and G2) and a single exclusive merge gateway (G3).

The choreography specification describes, from a global perspective, all permissible message sequences that can be exchanged between the partners, and is used by the interacting parties for two purposes: i) designing and implementing their individual parts of the business process; and ii) it is also very useful as a guide for developing the electronic contract.

The electronic contract designer is able to use the legal contract and choreography, to accurately identify and extract the ROPs attributed to the business partners, and to specify the rules which operate on the ROP set (Molina-Jimenez and Shrivastava, 2013). Rule implementation requires an appropriate specification language; contract rules for the CCC

monitoring service are currently realised using the Drools Rule Language (DRL) (RedHat, 2013).

An example of a rule that deals with receipt of a *buy request* event by the CCC, written using Drools can be seen below. Line 5 checks that the *buyRequest* operation is a right that the buyer is currently allowed to perform. If so then *buyRequest* is declared by the CCC as contract compliant (line 13). This operation is also removed from the buyer's ROP set (line 8), meaning that the buyer no longer has a right to perform this operation. At lines 10 and 11, the seller is given an obligation to perform one of 2 operations: *buyConfirm*, or *buyReject*.

```
1 rule "Buy Request Received"
2 //Verify type of event, originator, and
 responder
3 when
4 $e: Event (type=="BUYREQ",
5 originator=="buyer", responder=="store",
6 status=="success")
7 eval (ropBuyer.matchesRights (buyRequest))
8 then
9 //Remove buyer's right to place other Buy
10 Requests
11 ropBuyer.removeRight (buyRequest, seller);
12 //Add seller's obligation to either accept
13 or reject order
14 BusinessOperation[] bos = {buyConfirm,
15 buyReject};
16 ropSeller.addObligation ("React To Buy
17 Request", bos, buyer, 60,2);
18 System.out.println ("* Buy Request Received
19 rule triggered");
20 responder.setContractCompliant (true)
21 end
```

Each of the activities declared in the choreography of Fig. 2 has a rule such as the one shown above. Typically, for each activity in a choreography, each business partner can have several rights, obligations, and prohibitions in force.

Once an electronic contract specification has been completed, it can be loaded into the CCC for deployment and testing. As operations are executed, and events are received by the CCC, rights, obligations and prohibitions are granted to and revoked.

The CCC processes each event to determine if it is contract compliant (*CC*) or none contract compliant (*NCC*). The execution of a business operation (observed from the outcome event) is said to be *CC* if it satisfies the following three conditions and is said to be *NCC* if it does not: 1) it matches an operation within the set of business operations expected by the CCC, 2) it matches the ROP set of its role player (meaning, the role player that performed the operation has a right/obligation/prohibition to perform that particular operation), and 3) it satisfies the constraints stipulated in the contractual clauses. An example of a



constraint is the seven day deadline in clause 3 of the contract discussed earlier.

We also consider that the execution of a given sequence of operations is *NCC* if it includes one or more operations that are flagged by the CCC as *NCC*. A sequence of operations is also known as an *execution sequence* or *execution trace*, and drives the choreography from its initial state to a final state.

To ease the introduction of basic concepts, our legal contract example and corresponding choreography of Fig. 2, deal with successful outcome events only. However, a contract monitoring service such as the CCC should also be able to observe outcome events that include exceptional circumstances (Molina-Jimenez et al., 2009). Therefore, following the ebXML standard (OASIS, 2006), we assume that at the end of a business conversation, each party independently declares an execution outcome event from the set  $\{Success(S), BizFail(BF), TecFail(TF)\}$  as shown in Fig. 1. *Success* events model successful execution outcomes. *TecFail* models protocol related failures detected at the middleware level, such as a late, or a syntactically incorrect message. *BizFail* models semantic errors in a message detected at the business level, e.g., the credit card details extracted from the received payment document are incorrect.

Adding exceptional outcome events to the CCC's set of observable events, naturally means that the CCC has to monitor a much larger number of execution sequences. The task of generating these in order to test the CCC effectively is extremely challenging, and strengthens the case for needing to automate the testing process. Before moving on to our testing framework, let us first take a look at the new architecture of the CCC.

### 3 ARCHITECTURE OF THE CONTRACT COMPLIANCE CHECKER (CCC)

The overall architecture of the CCC is shown in Fig. 3. It consists of two layers: The *CCC Engine* (The *Logical Layer*), and the new addition is the *CCC Service* (The *Presentation Layer*). The *CCC Engine* is responsible for processing business events and for determining whether they are contract compliant or not. The *CCC Service* is an interface to the *CCC Engine*, it is used for delivering business events to the CCC, and for collecting the corresponding responses. In addition, the *CCC Service* can be used by the rule administrator for loading and editing the rules that represent the contract. The functionality of the ar-

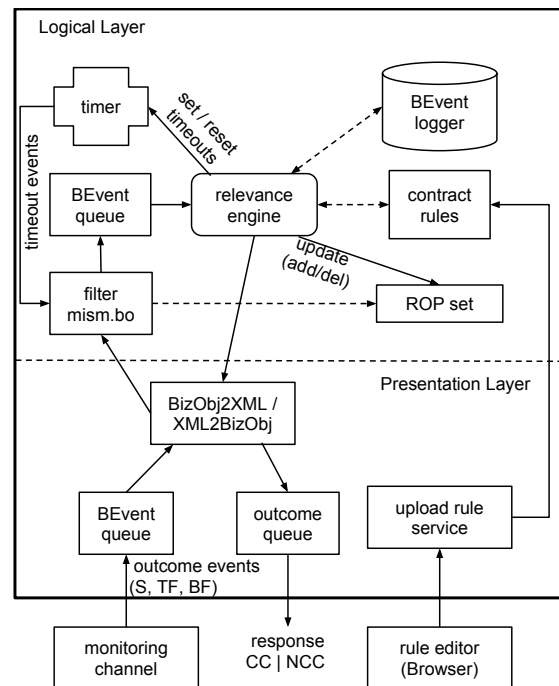


Figure 3: Architecture of the contract compliance checker.

chitecture is as follows: A business event is received through the *monitoring channel* as an XML document that includes the names of the participants, the business operation, and its outcome from the set: (*Success*, *BizFail*, *TecFail*):

```
<event>
 <originator>buyer</originator>
 <responder>seller</responder>
 <type>BuyReq</type>
 <status>success</status>
</event>
```

The event shown here is produced as a result of the implementation of a conversation synchronization protocol between the interacting parties. The protocol guarantees mutually agreed conversation outcomes. It is the responsibility of the interacting partners to apply the protocol. A detailed discussion can be found in (Molina-Jimenez et al., 2007). The XML document representing the business event is passed to the *BEvent queue*. Business events are retrieved, and converted using the *xml2BizObj/BizObj2xml Converter* from their XML format into business event objects. Events are then passed to the *CCC Engine*. The *filter mism.bo*, discards mismatched business events that are not among the permitted events defined within the ROP set. Business events that pass this filter are inserted into the *BEvent queue*. All deadlines are set and reset by the *relevance engine*, and enforced by the *timer*. Timeout events are added to the *filter mism.bo* as required by the contract, and are examined by the

filter to decide if *bevents* are mismatched. For example, receiving a *buy confirmation* event from the store after the 3 day deadline has elapsed, will be treated as mismatched. The *relevance engine* removes a business event from the head of the *bevent queue* and compares it to the rules stored in the *contract rules*. Rules that match the *bevent* under examination are triggered to determine if their conditions are satisfied. The actions of the rules whose conditions are satisfied are executed, and this may alter (*add/del*) the current state of the *ROP sets*. For our example in Fig. 2, a rule triggered by a *BuyConf* business event and finds its conditions satisfied will delete (disable) the store's right to execute another *BuyConf* business operation, and delete the store's right to execute a *BuyRej* operation. The rule also will add (impose) an obligation on the buyer to either initiate the execution of a *BuyPay*, or initiate the execution of *BuyCanc*. The *bevent* is then stored in the *BEvent logger* as a record for any future dispute resolution. The *relevance engine* eventually declares the business event either *CC* or *NCC* and produces a response as a business object, which is sent out to the *Presentation Layer*. The business object passes through the *xml2BizObj/BizObj2xml Converter*, where it is serialized into an XML message of the following format:

```
<result>
 <contractcompliant> true|false
</contractcompliant>
</result>
```

The *xml2BizObj/BizObj2xml Converter* inserts the response into the *outcome queue*, which can be accessed by the contracted parties. The *Presentation Layer* allows a "rule manager" to update the *contract rules* at run time. For this purpose, rules can be edited using the *rule editor (in a browser)* and sent to the *rule upload service* as a conventional RESTful POST operation. The *rule upload service* is responsible for producing a *drl* (Drools) file (for example *new-rules.drl*) from the payload of the POST operation, and for uploading it to the *CCC Logical Layer* to replace the *Contract Rules*.

The *CCC Logical Layer* is implemented using JBoss's Drools rules Engine (version 6.1 as of the year 2014) (RedHat, 2013). The Drools rules engine powers the decision making capabilities of the *relevance engine*. The *relevance engine*, acts as a wrapper for the Drools rule engine and its responsibilities include the initialisation of the contract, as well as the addition and processing of events received from the *Presentation Layer*.

The *Presentation layer*, a new addition to the CCC, exposes the CCC as a RESTful web service.

Its aim is to enable the exchange of XML event messages between the CCC and the contracted clients, and to ease the editing and update of the contract rules (these were previously hard coded). The *Presentation Layer* is implemented using the JBoss Enterprise Application Platform (EAP), (RedHat, 2014). The *BEvent queue* and the *outcome queue*, are implemented using JBoss's HornetQ (a message oriented middleware layer), and using the Java Message Service (JMS) API. A Message Driven Bean (MDB) receives business events from HornetQ and passes them to the *XML2BizObj/BizObj2XML converter*, which is implemented using Java. The *upload rule service* is part of the Drools Workbench— a web authoring and rules management application.

## 4 MODEL CHECKER BASED TESTING

To claim categorically that the CCC functions correctly, we need to test that it can correctly identify contract compliant and non-contract compliant executions of sequences and their constituent business operations. To this end, one needs to be able produce sequences of operations that are known to be contract compliant, and also produce sequences that include both contract compliant and non contract compliant operations. The challenge here is the production of such sequences.

### 4.1 Testing Framework

Fig. 4 shows the main elements of our testing framework. Squares with smooth corners represent humans involved in the design process. Tools are represented by solid squares with sharp corners, and dashed squares represent data.

As stated earlier, a central component of our testing framework is the SPIN model checker. SPIN models are constructed using Promela, and specifically using EPROMELA, a modeling language developed precisely for modelling electronic contracts (Abdelsadiq et al., 2011). EPROMELA is essentially a high level tool that extends Promela with constructs for expressing core electronic contract concepts contained in the CCC. Correctness properties that an EPROMELA model is expected to satisfy, can be expressed by the model designer using Linear Temporal Logic (LTL). When provided with a model of the contract and appropriate LTL properties, SPIN is able to verify the correctness of the model with respect to those properties. With the aid of tools for message parsing and automation, SPIN also can

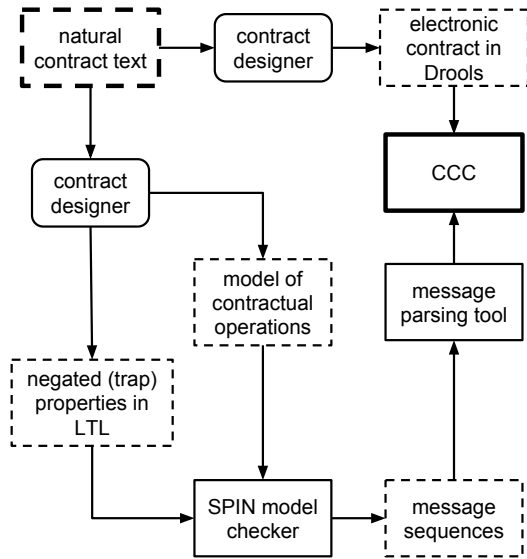


Figure 4: Model Checker based testing framework.

be instrumented to generate message sequences that can be used to test the ability of the CCC to detect contract compliant and non contract compliant message sequences, a process that we will describe next. Model checker based sequence generation follows these steps:

1. The designer constructs an abstract model of the System Under Test (SUT), and verifies that the model is correct in that it satisfies the correctness properties of interest.
2. The verified abstract model is used for generating execution sequences. This is done by presenting the verification tool with the verified abstract model, together with a negated correctness requirement in LTL (a trap property), and then challenging the verification tool to find and produce counter examples that violate the LTL.
3. Each counter example contains an execution sequence that can be extracted with the aid of a message parsing tool.

## 4.2 Example

We begin by building an EPROMELA model of our example contract presented in the Background section. To ease the task of parsing the counter examples of interest, we include within the EPROMELA model print statements that produce the required XML events. The end of each execution sequence is marked using a *reset message*).

### 4.2.1 Model Construction and Verification

Below is a section of our EPROMELA contract model, which includes the rule that deals with the *BUYREQ* operation of Fig. 2. Each of the operations for the choreography in Fig. 2 has a rule which updates the status of the ROP set belonging the participants as they transition from state to state.

```

1 RULE (BUYREQ)
2 {
3 WHEN : :EVENT (BUYREQ,
4 IS_R (BUYREQ, BUYER), SC (BUYREQ)) -> {
5 SET_X (BUYREQ, BUYER);
6 atomic{
7 printf("<originator>buyer</originator>");
8 printf("<responder>store</responder>");
9 printf("<type>BUYREQ</type>");
10 printf("<status>success</status>");
11 }
12 SET_R (BUYREQ, 0);
13 SET_O (BUYREJ, 1);
14 SET_O (BUYCONF, 1);
15 RD (BUYREQ, BUYER, CCR, CO);
16 }
17 END (BUYREQ);

```

Line 3 of the model deals with receiving a successful buy request event *SC (BUYREQ)*. *IS\_R (BUYREQ, BUYER)* is a guard that checks if the *BUYER* has a right to perform the *BUYREQ* operation. If so, then *SET\_X (BUYREQ, BUYER)* declares that this operation has been executed, and the buyer's right to execute *BUYREQ* is removed at line 11. The rule then sets an obligation to the Store to execute either *BUYREJ* or *BUYCONF* (lines 12 - 13). At line 6 we introduce the print statements required for parsing the generated execution sequences. The print statements produce XML events in the format expected by the CCC.

Each of the operations *BUYREQ*, *BUYREJ*, *BUYCONF*, *BUYPAR*, *BUYCANC*, has a rule such as the one above. Events are generated using an EPROMELA *Event Generator* module, which models the interaction between the business partners. Events exercise rules such as the one above through another EPROMELA module known simply as the *Rule Manager*. For a full description, see (Abdelsadiq et al., 2011). When the entire EPROMELA model has been constructed, SPIN can be used to verify that the model is free from any inconsistencies. Common correctness properties such as absence of deadlocks and reachability of states, can easily be checked using SPIN's configuration options. Checking for contract specific correctness properties however, requires the application of Linear Temporal Logic (LTL) formulae. Typical correctness properties of the electronic contracting domain are those that express mutual ex-

clusion of rights, obligations, and prohibitions; for example the requirement that the execution of a given operation (such as making a purchase order) is never simultaneously obliged and prohibited. Thanks to the contract constructs offered by EPROMELA, this correctness requirement can be elegantly and intuitively expressed in LTL as follows:

```
[!](IS_O(BUYREQ, BUYER) && IS_P(BUYREQ, BUYER))
```

Where `[]` is the LTL always operator. `!` is the universal not, `IS_O(BUYREQ, BUYER)` returns true if the `BUYREQ` operation is currently obliged and `IS_P(BUYREQ, BUYER)` returns true if the `BUYREQ` operation is currently prohibited. Instructing SPIN to run through the EPROMELA model using this LTL, will drive SPIN to find any examples that violate this property. If such an example is found then it is presented as a counter example to the designer, who must then correct the model.

#### 4.2.2 Generating the Test Sequences

Once the contract model has been verified for required correctness properties, it can be used as an oracle for producing sequences that can test the electronic contract. Test sequence generation is very similar to verification in that we make use of LTL properties. We can (as described in the previous section) instruct SPIN to find undesirable examples of sequences that violate a desirable property. But we also need to be able to instruct SPIN to find desirable sequences that violate a non-desirable property. The latter is done by negating a desirable LTL property converting it into a *trap* property.

As a very simple example, let us instruct SPIN to generate all sequences of messages that end with a *BUYREJECT* operation. The LTL formulae required for this task is as follows:

```
!<>IS_X(BUYREJ, STORE)
```

Where `< >` is the LTL eventually operator. The formulae states that the model will not eventually reach a state where `BUYREJ` is executed. SPIN can now be instrumented to show all sequences that do end with `BUYREJ`. From the command line we apply the following steps (*CorrectChore* is the name of the file that contains the EPROMELA model):

1. `% spin -a CorrectChore` is used for generating the verifier source code in C.
2. `% cc -o pan pan.c` is used for compiling the verifier.
3. `% ./pan -a -e -c100` instructs SPIN to produce all the counter examples (trail files) that it can find, which violate the trap property. By default SPIN produces the first one it finds and stops.

The `-c100` parameter instructs SPIN to generate the first 100 counter examples it finds. The number of counter examples requested needs to be above the actual number of counter examples that SPIN could possibly find. This number can be determined by the designer using trial and error.

4. `spin -tN -s -r -B CorrectChore` converts the  $N^{th}$  trail file into a text file that includes the XML messages involved in the execution sequence.

Given the potentially large number of trail files that can be produced by SPIN, it is advisable to mechanise the process. We use a simple shell script for this purpose. The following text represents the contents of one of the trail files produced by the Linux shell script. To ease readability, we have removed some irrelevant lines.

```
2: proc 0 (Buyer) line 35 "CorrectChore" Sent
 BuyReq,1
3: proc 1 (Store) line 71 "CorrectChore" Recv
 BuyReq,1

<originator>buyer</originator>
<responder>store</responder>
<type>BUYREQ</type>
<status>success</status>

5: proc 1 (Store) line 114 "CorrectChore"
 Sent BuyRej,1
6: proc 0 (Buyer) line 049 "CorrectChore"
 Recv BuyRej,1

<originator>store</originator>
<responder>buyer</responder>
<type>BUYREJ</type>
<status>success</status>

<originator>reset</originator>
<responder>reset</responder>
<type>reset</type>
<status>reset</status>
```

The execution sequence shown above includes a *BUYREQ* message sent from the buyer to the store, followed by *BUYREJ* sent by the store to the buyer. The status element indicates the outcome of the execution of the operation. The status in this example accounts only for successful execution outcomes (No exceptional circumstances such as technical failures are assumed), consequently, the content of this element is always *success*. The last message is the *reset* message, which we artificially include to mark the end of the sequence.

As can be appreciated from this example, the files produced by SPIN and the shell script need parsing in order to extract the XML tagged messages.

### 4.2.3 Sequence Parsing

Our parser is built using Python. It extracts all the XML tagged messages from a given sequence and stores each message as an individual XML file. The parser achieves this by creating a recursive grammar that describes the precise structure of the business events inside a sequence. As seen in the code segment below in lines 2 - 5, we first define the XML tags we want to find.

```
1 #define grammar for sequence file
2 tagOriginator = pyp.Literal("<originator>") +
 pyp.Word(pyp.alphas) +
 pyp.Literal("</originator>")
3 tagResponder = pyp.Literal("<responder>") +
 pyp.Word(pyp.alphas) +
 pyp.Literal("</responder>")
4 tagType = pyp.Literal("<type>") +
 pyp.Word(pyp.alphas) +
 pyp.Literal("</type>")
5 tagStatus = pyp.Literal("<status>") +
 pyp.Word(pyp.alphas) +
 pyp.Literal("</status>")
6 lineString = tagOriginator | tagResponder |
 tagType | tagStatus
```

The parser reads a file containing a message sequence, and searches for matches against each line according to the following rule in line 6: *If there is a line that includes a tag definition of either the originator, responder, type, or status, then the match is successful.* If the parser finds a match, then it performs the following actions: i) the parser creates a new folder with the name of the sequence, ii) it extracts the XML part that is matched according to the above rule, iii) a new XML file is created that includes the extracted business event. Thus, the folder *ExeSeq1-xml* for the sequence shown above will contain three XML files because the sequences contains three messages, namely *BUYREQ* → *BUYREJ* → *reset*.

### 4.2.4 Testing the Electronic Contract

After loading and initialising the CCC with the rules that encode the electronic contract, we can proceed with sending each of the execution sequences to the *BEvent queue*. Responses are collected from the *outcome queue* (see Fig. 3). The following lines show the results of testing the execution sequence *BUYREQ* → *BUYREJ* → *reset*:

```
1 filename: event1.xml
2 -Begin Request to CCC service-
3 BusinessEvent [originator=buyer,
 responder=store, type=BUYREQ,
 status=success]
4 -End Request to CCC service-
5
6 -Begin Response from CCC service-
```

```
7 <result>
8 <contractCompliant>true</contractCompliant>
9 </result>
10-End Response from CCC service-
11
12 filename: event2.xml
13 -Begin Request to CCC service-
14 BusinessEvent [originator=store,
 responder=buyer, type=BUYREJ,
 status=success]
15 -End Request to CCC service-
16
17 -Begin Response from CCC service-
18 <result>
19 <contractCompliant>true
 </contractCompliant>
20 </result>
21 -End Response from CCC service-
22
23 filename: event3.xml
24 -Begin Request to CCC service-
25 BusinessEvent [originator=reset,
 responder=reset, type=reset,
 status=reset]
26 -End Request to CCC service-
27 -Begin Response from CCC service-
28 <result>
29 <contractCompliant>true
 </contractCompliant>
30 </result>
31 -End Response from CCC service-
```

The operations (*BUYREQ* and *BUYREJ*) included in the sequence, are declared contract compliant by the CCC indicating that the contract rules have been coded correctly with respect to the LTL property in Section 4.2.2. The first operation is sent to the CCC in line 3, and its response *<contractCompliant>true* is shown at line 8. Similarly, *BUYREJ* operation is sent to the CCC at line 14, and its response *<contractCompliant>true* can be seen at line 19.

### 4.2.5 Testing None Compliant Events

A model that has been verified, will by default generate test sequences with events corresponding to the execution of contract compliant operations only. An EPROMELA model can be tuned to generate sequences which include unknown and noncompliant business events using the EPROMELA *Event Generator* module mentioned under Section 4.2.1. Thus we can alter the EPROMELA model to follow any variation of the choreography shown in Fig. 2. For example the modified choreography of figure Fig. 5 does not correctly reflect the original text contract.

The particularity of this diagram is that it produces contract compliant sequences such as *BuyReq* → *BuyRej*. In addition, it produces non-contract compliant sequences, for instance it allows for cancellation after payment which is not stipulated in the orig-

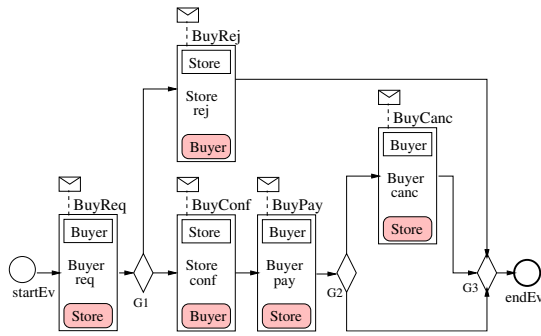


Figure 5: Incorrect choreography of contract example.

inal contract. Consequently, the execution of *BuyCanc* within the sequence *BuyReq* → *BuyConf* → *BuyPay* → *BuyCanc* should be declared non contract compliant by the CCC. The following text shows the results of the execution of the non-contract compliant sequence discussed above. The first 2 events *BUYREQ*, *BUYCONF*, were declared contract compliant by the CCC as expected. To save space we only show the outcome of the 2 events of relevance in this example (*BUYPAY* followed by *BUYCANC*):

```

1 filename: event3.xml
2 -Begin Request to CCC service-
3 BusinessEvent [originator=buyer,
4 responder=store, type=BUYPAY,
5 status=success]
6 -End Request to CCC service-
7
8 -Begin Response from CCC service-
9 <result>
10 <contractCompliant> true
11 </contractCompliant>
12 </result>
13 -End Response from CCC service-
14
15 filename: event4.xml
16 -Begin Request to CCC service-
17 BusinessEvent [originator=buyer,
18 responder=store, type=BUYCANC,
19 status=success]
20 -End Request to CCC service-
21
22 -Begin Response from CCC service-
23 <result>
24 <contractCompliant> false
25 </contractCompliant>
26 </result>
27 -End Response from CCC service-

```

The process *BUYPAY* is contract compliant (lines 3 and 8). The execution of *BUYCANC* at line 14 and the corresponding response received at line 19 indicates that the CCC has declared *BUYCANC* non-contract compliant. This is the desired behaviour from the CCC, as it has detected that this sequence of events is not consistent with the contract.

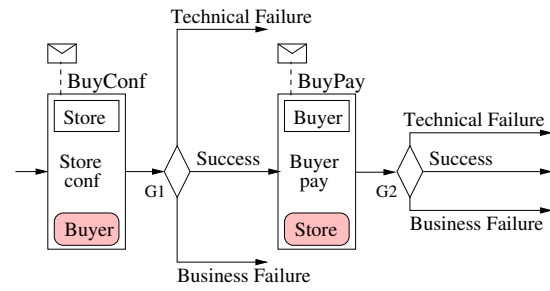


Figure 6: Execution model with success and failures.

### 4.3 Accounting for Exceptional Outcome Events

The contract example we have used so far assumes that the execution of operations always succeeds; it does not account for potential failures. More realistic examples would include the execution of activities as shown in Fig. 6, which account for successful and failed outcomes.

As discussed in Section 2, and following the ebXML standard (OASIS, 2006), we would like to be able to detect two types of failures; business failures, and technical failures. To this end, the EPROMELA modeling language has been designed with the ability to deal with these 2 types of failures. As an example of an electronic contract that can handle exceptional outcomes, we add the following clause to our original contract to account for potential semantic errors (*business failures*) in the execution of any operation:

4. *Failure handling: if after 2 attempts, an operation is not performed correctly, then the contractual interaction shall be declared terminated.*

Fig. 7 shows a partial choreography representation of the contract for three out of its five tasks only (for readability). The contract allows for a finite number of retries if business failures are encountered. The actual number of retries will normally be a configuration parameter. In the figure, *S* and *BF* stands for Success and Business Failure, respectively. Similarly, *rqBF*, *rjBF*, and *coBF*, represent counters that keep track of the number of failed executions of the operations; *BUYREQ*, *BUYREJ*, and *BUYCONF*, respectively. *N* represents an arbitrary integer that in our example allows for two failure execution ( $N = 2$ ). The execution of each activity leads to a gateway with three outgoing arrows. As an example, at *BUYREQ*, a successful (*S*) execution leads to the normal execution of the contract, namely to *G2*. Alternatively, if the execution completes in *BF* and the number of failed executions *rqBF* of the *BUYREQ* operation is less than *N*, the execution is tried again. However, if the outcome is *BF* and it has already failed *N* times, the contractual



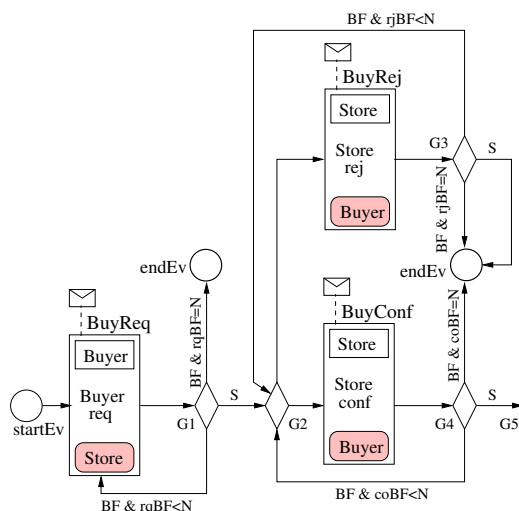


Figure 7: Contract example that accounts for failures.

interaction is terminated. Failure handling with the remaining activities is similar, except that gateways *G2* and *G5* introduce additional alternative execution paths. For instance, after failing to complete successfully *BUYREQ* at the first attempt, the initiator is allowed to choose *BUYREQ* again or alternatively can execute *BUYCONF*. Below we show how an exception such as the business failure of the *BUYREQ* operation described above can be intuitively and naturally modeled using EPROMELA. The rule for *BUYREQ* described in Section 4.2.1 can be easily enhanced as follows:

```

1 /*handle failure outcome event*/
2 :: EVENT (BUYREQ, IS_R (BUYREQ, BUYER) ,
 BF (BUYREQ)) ->{
3 atomic{
4 printf("<originator>buyer</originator>");
5 printf("<responder>store</responder>");
6 printf("<type>BUYREQ</type>");
7 printf("<status>bizfail</status>");
8 }
9 if /*1st notification of BF*/
10 :: (ReqFailBefore==NO) ->
 ReqFailBefore=YES;
11 printf("First BUYREQ-BF");
12 RD (BUYREQ, BUYER, CCR, CO);
13 /*2nd notification of BF*/
14 :: (ReqFailBefore==YES) ->
 abncoend=TRUE;
15 printf("Last BUYREQ-BF");
16 SET_R (BUYREQ, 0);
17 atomic{
18 printf("<originator>reset</originator>");
19 printf("<responder>reset</responder>");
20 printf("<type>reset</type>");
21 printf("<status>reset</status>");
22 RD (BUYREQ, BUYER, NCCR, CND); /*abnormal
 contract end*/

```

The model can now also handle *BUYREQ* events that

result in *BF* outcomes (line 2). If a failed event is received, then the rule checks if a failure of this kind has happened before. If not (line 10), then this first failure is registered, and contract execution is allowed to continue (line 12). On the other hand, if this is the second time *BUYREQ* has been received with a *BF* outcome then the rule terminates contract interaction at line 23.

The EPROMELA model includes rules like the one described above for dealing with each of the 5 business events shown in bold in our contract example. After the model has been verified using SPIN, the electronic contract deployed to the CCC can be tested, in combination with the testing framework described previously, using much more realistic execution sequences that include exceptions. A detailed description of how exceptions are handled in the CCC can be found in (Molina-Jimenez et al., 2009).

## 5 RELATED WORK

Research work on the monitoring of cross-organizational interactions between parties was pioneered by Minsky (Ungureanu and Minsky, 2000) with work on Law Governed Interaction (LGI). The notion of rights, obligations and prohibitions was introduced in (Ludwig and Stolze, 2003). A useful summary about various issues involved in contract management is provided in (Hvitved, 2010).

Linear Temporal Logic (LTL) is a powerful tool for specifying correctness properties in a model whether it is for verifying the correctness of the model, or for the generation of test sequences. However not all correctness properties can be expressed using LTL; for example it is not possible to specify that a particular property will hold for every 3rd or 4th state of the system. Such limitations are discussed in (Galton, 1987), where extensions to LTL are suggested. In addition, despite the advantages of model checker based testing, it does have its disadvantages; building a model of the SUT and describing the required LTL properties relies heavily on the skills of the technical person who must also be intimately familiar with the SUT. The advantages and disadvantages of model checker based testing are discussed in (El-Far, 2001) where the author provides a practical guide. Naturally it is difficult to ensure complete coverage of all possible system behaviors during testing with manually specified LTL properties. Therefore, it is extremely desirable to be able to systematically create complete test suites according to some test objective (Fraser et al., 2009). (Van der Aalst and Pesic, 2006) propose to automate the task of specifying LTL

properties by means of a graphical language (DecSerFlow) that is then mapped into LTL formulas. Using this language, the designer can specify a set of common or frequent correctness requirements.

It is worth noting at this point that we are in the process of developing an *LTL Manager* component for our testing framework. The *LTL Manager* includes a repository that can be populated with templates of LTL formulae for common contract related properties that must be satisfied by all contracts. For example that a business operation is not simultaneously prohibited and obliged at the same time.

Although model checker based testing techniques have been studied widely in the software engineering community (Utting and Legeard, 2006) (Pezze and Young, 2008) (Torsel, 2013), their use in the testing of a contract monitoring service has received little attention. The principles of model checker based testing of electronic contracts are investigated previously by us in (Abdelsadiq et al., 2010), however contract models in this work are built using Promela, the basic input language of SPIN. Attempting to predict how a designer would use basic Promela to model a contract is an impossible task, which makes developing tools for automating the testing process extremely difficult. An important contribution of this paper is that we highlight the benefits of developing a tool based framework that has been tailored specifically to leverage the capabilities of a domain specific modelling tool such as EPROMELA (Abdelsadiq et al., 2011), which was developed specifically for modeling electronic contracts.

## 6 CONCLUSION AND FUTURE WORK

Ensuring the correct functionality of an electronic monitoring service such as the *contract compliance checker (CCC)*, becomes more difficult as the number of execution sequences that the CCC is expected to observe increase. We have seen that cloud and Internet based interactions between business partners can indeed be extremely complex, and this is especially true when exceptional outcome events from these interactions are taken into consideration. Reproducing such complex exchanges in order to test the correct functionality of the CCC is difficult and cannot be achieved manually. In addition, it is extremely important to be able to correct and update the rule base of the monitoring service rapidly so that interruptions to the deployed service are reduced as much as possible.

In order to address these issues, we have presented a model checker based framework that includes tools

to automate the testing process. By using the SPIN model checker in combination with EPROMELA, a high level modeling language designed specifically for modeling electronic contracts, we can build verified models that accurately resemble the System Under Test (SUT) with relative ease. By using appropriate LTL formulae within an EPROMELA model, we can instrument SPIN to automatically produce contract compliant, and none contract compliant execution sequences that are capable of exhaustively testing the correct operation of the CCC. In addition, we have presented a new and enhanced architecture and implementation of the CCC, with an additional *Presentation Layer* that exposes the CCC as a web service. An important feature is that the *Presentation Layer* includes a Drools editing and upload service that enables dynamic update of the electronic contract rules at runtime.

There are a number of future research directions which we are currently exploring. Drools, the language we use for specifying electronic contracts is verbose, and not as declarative and readable as would be ideal. We have developed a contract specification language called EROP (for Events, Rights, Obligations, and Prohibitions) (Strano et al., 2009), and are in the process of completing a tool for translating EROP to Drools. Also we would like to enhance the CCC, which currently acts as a passive monitor, with the capability to act as a contract enforcer. The aim of a contract enforcement service would be to ensure that an operation is executed only if it is contract compliant.

An important item for future work is to conduct experiments in order to determine how the presented testing framework performs as the number of possible events increases. The verification and testing of the CCC can be further automated in several ways; in addition to the development of the *LTL Manager* component discussed in Section 5, we would like to create a translation tool that can produce an EPROMELA model from an electronic contract specification written in EROP automatically. This would reduce the risk of introducing unwanted errors into the contract model during construction. We believe that this goal is achievable because of the semantic similarities between EPROMELA and the electronic contracting concepts within the CCC.

## REFERENCES

- Abdelsadiq, A., Molina-Jimenez, C., and Shrivastava, S. (2010). On model checker based testing of electronic contracting systems. In *IEEE International Confer-*



- ence on Commerce and Enterprise Computing (CEC 2010). IEEE.
- Abdelsadiq, A., Molina-Jimenez, C., and Shrivastava, S. (2011). A high level model checking tool for verifying service agreements. In *The 6th IEEE International Symposium on Service-Oriented System Engineering (SOSE 2011)*. IEEE.
- El-Far, I. K. (2001). Enjoying the perks of model-based testing. In *Proc. of the Software Testing, Analysis, and Review Conference (STARWEST 2001)*.
- Fraser, G., Wotawa, F., and Ammann, P. (2009). Testing with model checkers: A survey. *Software Testing, Verification and Reliability*, pages 215–261.
- Galton, A. (1987). Temporal logics and computer science: An overview. *Academic Press*, pages ch. 1, pp. 2748.
- Holzmann, G. J. (2003). *The Spin model checker: primer and reference manual*. AddisonWesley Professional.
- Hvitved, T. (2010). A survey of formal languages for contracts. In *n Fourth Workshop on Formal Languages and Analysis of ContractOriented Software (FLACOS10)*.
- Ludwig, H. and Stolze, M. (2003). Simple obligation and right model (sorm)-for the runtime management of electronic service contracts. In *2nd Intl Workshop on Web Services, eBusiness, and the Semantic Web (WES03) LNCS*, volume 3095, pages 62–76.
- Molina-Jimenez, C. and Shrivastava, S. (2013). Establishing conformance between contracts and choreographies. In *15th IEEE Conference on Business Informatics (CBI). 2013, Vienna, Austria: IEEE Computer Society*. IEEE.
- Molina-Jimenez, C., Shrivastava, S., and Cook, N. (2007). Implementing business conversations with consistency guarantees using message-oriented middleware. In *IEEE 11th Intl Enterprise Computing Conf. (EDOC 07)*, pages 51–62.
- Molina-Jimenez, C., Shrivastava, S., Solaiman, E., and Warne, J. (2003). Contract representation for runtime monitoring and enforcement. In *2003 IEEE International Conference on E-Commerce (CEC 2003)*. IEEE.
- Molina-Jimenez, C., Shrivastava, S., and Strano, M. (2009). Exception handling in electronic contracting. In *IEEE Conference on Commerce and Enterprise Computing (CEC). 2009, Vienna, Austria*. IEEE.
- Molina-Jimenez, C., Shrivastava, S., and Strano, M. (2012). A model for checking contractual compliance of business interactions. *IEEE TRANSACTIONS ON SERVICES COMPUTING*, 5(2):276–289.
- Molina-Jimenez, C., Shrivastava, S., and Wheeler, S. (2011). An architecture for negotiation and enforcement of resource usage policies. In *IEEE International Conference on Service Oriented Computing & Applications (SOCA)*. IEEE.
- OASIS (2006). *ebXML Business Process Specification Schema Technical Specification v2.0.4*. Available: <http://docs.oasis-open.org/ebxmlbp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en.pdf>.
- OMG (2011). *Documents associated with business process model and notation (bpmn) version 2.0*. <http://www.omg.org/spec/BPMN/2.0/>.
- Pezze, M. and Young, M. (2008). *Software Testing and Analysis: Process, Principles and Techniques*. Wiley.
- RedHat (2013). "Drools". <http://www.drools.org/>.
- RedHat (2014). *JBoss Enterprise Application Platform v 6.3*. <http://www.redhat.com/en/technologies/jboss-middleware/application-platform>.
- Solaiman, E., Molina-Jimenez, C., and Shrivastava, S. (2003). Model checking correctness properties of electronic contracts. In *International Conference on Service Oriented Computing (ICSOC03)*. Springer.
- Strano, M., Molina-Jimenez, C., and Shrivastava, S. (2008). A rule-based notation to specify executable electronic contracts. In *Rule Representation, Interchange and Reasoning on the Web: International Symposium (RuleML)*. Springer-Verlag.
- Strano, M., Molina-Jimenez, C., and Shrivastava, S. (2009). Implementing a rule-based contract compliance checker. In *Software Services for e-Business and e-Society: 9th IFIP WG 6.1 Conference on e-Business, e-Services and e-Society (I3E)*. Springer.
- Torsel, A.-M. (2013). A testing tool for web applications using a domain-specific modelling language and the nusmv model checker. In *IEEE Sixth International Conference on Software Testing, Verification and Validation*.
- Ungureanu, V. and Minsky, N. H. (2000). Establishing business rules for interenterprise electronic commerce. In *14th International Symposium on Distributed Computing (DISC00)*, pages 179–193.
- Utting, M. and Legeard, B. (2006). *Practical Model-Based Testing: A Tools Approach*. MorganKaufmann.
- Van der Aalst, W. and Pesic, M. (2006). Decserflow: Towards a truly declarative service flow language. In *Bravetti M, Nunez M, Zavattaro G (eds) International Conference on Web Services and Formal Methods (WS-FM 2006)*, volume 4184, pages 1–23. Lecture Notes in Computer Science Springer-Verlag.

# ANY2API – Automated APIfication

## *Generating APIs for Executables to Ease their Integration and Orchestration for Cloud Application Deployment Automation*

Johannes Wettinger, Uwe Breitenbücher and Frank Leymann

*Institute of Architecture of Application Systems, University of Stuttgart, Universitätsstraße 38, Stuttgart, Germany*  
{wettinger, breitenbuecher, leymann}@iaas.uni-stuttgart.de

**Keywords:** Cloud Computing, DevOps, API, APIfication, Service, Web, REST.

**Abstract:** APIs are a popular means to expose functionality provided by Cloud-based systems, which are utilized to integrate and orchestrate application as well as management functionality in a programmatic manner. In the domain of application management, they are used to fully automate management processes, for example, to deploy Cloud-based Web applications or back-ends for mobile apps. However, as not all required functionality is exposed through an API natively, such processes additionally involve a multitude of other heterogeneous technologies such as scripting languages and deployment automation tooling. Consequently, combining different technologies in an efficient manner is a complex integration challenge. In this paper, we present a generic approach for automatically generating API implementations for arbitrary executables such as scripts and compiled programs, which are not natively exposed as APIs. This *APIfication* tackles the aforementioned integration challenges by unifying the invocation of heterogeneous technologies while avoiding the costly and manual wrapping of existing executables because it does not scale. We further present the modular and extensible open-source framework ANY2API that implements our APIfication approach. Furthermore, we evaluate the approach and the framework by measuring the overhead of generating and using API implementations. In addition, we conduct a detailed case study to confirm the technical feasibility of the approach.

## 1 INTRODUCTION

A remarkable amount of today's applications, especially Web applications as well as back-end systems and platforms for mobile apps, provide *application programming interfaces* (APIs) (Richardson et al., 2013). The main purpose of an API is to provide a well-defined and documented interface, which is exposed to access and utilize application functionality in a programmatic manner. APIs hide and abstract from implementation-specific details such as invocation mechanisms and data models inherited from the technology stack on which a particular application is built upon. This is the foundation for integrating and orchestrating different applications and application components, enabling systematic development and reliable operations of distributed applications, mash-up applications, and mobile apps. Furthermore, APIs are used to integrate applications with business partners, suppliers, and customers (Rudrakshi et al., 2014). Even devices can be connected and interconnected to enable the *Web of things* (Guinard et al., 2010). Technically, APIs can be exposed and utilized in different

forms. Both (i) libraries that are bound to a particular programming language and (ii) language-agnostic Web services, e.g., Web-based RESTful APIs (Richardson et al., 2013; Masse, 2011) or WSDL/SOAP-based services (W3C, 2007) are widespread forms of providing and using APIs. Popular providers such as Twitter, GitHub, Facebook, and Google offer such libraries<sup>1</sup> and Web services<sup>2</sup>. However, libraries and Web services are not mutually exclusive, meaning libraries often use Web services in the background, but adding an additional layer of abstraction to seamlessly integrate with the programming model of the corresponding language. Consequently, *Web APIs are a platform-independent and language-agnostic means for integration and orchestration purposes*, optionally enhanced by additional language-specific libraries. Regarding the terminology used in this paper, we consider a *Web API* as one particular kind of *API*. The use cases, examples, and implementations discussed in this paper mostly focus on Web APIs. However, the concepts

<sup>1</sup>Google APIs client libraries: <http://goo.gl/uVvFf>

<sup>2</sup>Google Compute Engine API: <http://goo.gl/cj0BG1>

and methods are suitably generic to be applied to other kinds of APIs, too.

The number of publicly available Web APIs is constantly growing<sup>3</sup>. As of today, the API directory *ProgrammableWeb*<sup>4</sup> lists more than 12000 APIs. Popular providers such as Google, Facebook, and Twitter are serving billions of API calls per day<sup>5</sup>. These statistics underpin the importance and relevance of APIs. Existing literature (Masse, 2011; Richardson et al., 2013) and frameworks such as Hapi<sup>6</sup> (Node.js) and Jersey<sup>7</sup> (Java) provide holistic support, best practices, and templates for building Web APIs. While this is state of the art for creating Web applications and back-ends for mobile apps, Web APIs as a platform-independent and language-agnostic means for integration and orchestration purposes are heavily utilized for automating the deployment and management of Cloud applications (Mell and Grance, 2011; Wettinger et al., 2014a), which leads to significant cost reductions and enables applications to scale: Cloud providers offer management APIs that can be programmatically used in a self-service manner, e.g., to provision virtual servers, deploy applications using platform services, or to configure scaling and network properties.

However, because such management APIs typically provide basic functionality only, they have to be combined with further configuration management systems to realize non-trivial deployment scenarios: a huge number of reusable artifacts such as scripts (e.g., Chef cookbooks (Nelson-Smith, 2013), Juju charms<sup>8</sup>, Unix shell scripts) and templates like Docker container images (Turnbull, 2014) are shared by open-source communities to be reused in conjunction with provider-supplied services. While APIs can be orchestrated easily due to well-known and common protocols (e.g., HTTP), the technical integration with these different artifacts and heterogeneous management systems is a very error-prone, time-consuming, and complex challenge (Wettinger et al., 2014a). Thus, to build, deploy, and manage non-trivial Web applications, it is of vital importance to handle the invocation of different artifacts, technologies, and service providers in a technically uniform manner to focus on the orchestration level, neglecting lower-level technical differences.

Unfortunately, many of these individual artifacts are *executables* that cannot be utilized through an API without a central middleware component (Wettinger et al., 2014a) such as a service bus that (a) maps

generic API calls into executable-specific invocations, (b) translates inputs and results of the invocation, and (c) makes them available through an API endpoint. However, this central middleware approach comes with three major drawbacks: (i) the individual artifacts are not packaged with their API to be utilized at run-time and, thus, they are not self-contained; (ii) in order to utilize the executables through an API, a central middleware component is inevitably required in addition to the individual artifacts to be invoked which results in additional costs and maintenance effort; (iii) in case a new kind of executable comes in, the central middleware has to be adapted, extended, and redeployed accordingly with potential risks such as downtime, functional failures, and unintended side effects. Today, this is nothing exceptional because open-source communities constantly share new kinds of artifacts such as Chef cookbooks, Juju charms, and Docker container images to name a few examples from the domain of application deployment automation.

The main goal of our work is overcome these drawbacks by introducing an automated approach to *generate API implementations (APIfication)* that are packaged including the corresponding artifacts such as the executable and all its dependencies in a portable manner. This makes them truly self-contained without depending on a central middleware. The generated API implementations simplify the orchestration of different kinds of artifacts and their integration with existing provider-hosted APIs. Therefore, the major contributions of this paper are as follows: (i) We present an automated *APIfication method*, respecting the requirements we derived from a use case and motivating scenario in the field of Cloud computing and deployment automation. (ii) We introduce an *APIfication framework* to implement the method we presented before and provide a prototype implementation to demonstrate the feasibility. (iii) We *validate* the proposed APIfication approach using a prototype implementation and perform an *evaluation* to analyze the efficiency of our approach. (iv) We conduct a *case study* in the field of deployment automation and discuss further use cases of the APIfication approach in other fields such as e-science.

The remainder of this paper is structured as follows: Section 2 describes the problem statement, including a use case and motivating scenario in the field of deployment automation. Based on the generic APIfication method presented in Section 3, we propose and discuss an APIfication framework in Section 4. Our prototype implementation ANY2API as well as its validation and evaluation are discussed in Section 5. Moreover, we present a case study in that section. Section 6 outlines further use cases to apply our APIfication approach.

<sup>3</sup>ProgrammableWeb statistics: <http://goo.gl/2eQ01o>

<sup>4</sup>ProgrammableWeb: <http://www.programmableweb.com>

<sup>5</sup>ProgrammableWeb calls per day: <http://goo.gl/yhggyW>

<sup>6</sup>Hapi: <http://hapijs.com>

<sup>7</sup>Jersey: <http://jersey.java.net>

<sup>8</sup>Juju charms: <https://manage.jujucharms.com/charms>

Finally, Section 7 and Section 8 discuss related work, future work, and conclude the paper.

## 2 PROBLEM STATEMENT & USE CASE

As discussed in Section 1, APIs serve as a platform-independent and language-agnostic means for integration and orchestration purposes. There are several frameworks based on different programming languages and technology stacks established to develop APIs, especially Web APIs. However, an individual API still needs to be implemented manually using these development frameworks. While this is state of the art for creating new applications such as Web applications or back-ends for mobile apps, for some use cases the individual development of an API is not feasible or even impossible. This is due to scaling issues (e.g., creating APIs for a huge amount of individual executables) or missing expertise, meaning the person, who needs to utilize certain functionality is not able to develop a corresponding API. In the following we discuss an important use case that requires API implementations to be generated in an automated manner.

### 2.1 Use Case: Deployment Automation

A major use case originates in the DevOps community (Hüttermann, 2012), proposing the implementation of fully automated deployment processes to enable continuous delivery of software (Humble and Farley, 2010; Wettinger et al., 2014b). This is the foundation for rapidly putting changes, new features, and bug fixes into production. Especially users and customers of Cloud-based Web applications and mobile apps expect fast responses to their changing and growing requirements. Thus, it is a competitive advantage to implement automated processes to enable fast and frequent releases (Hüttermann, 2012). As an example, Flickr performs more than 10 deployments per day<sup>9</sup>; HubSpot with 200-300 deployments per day goes even further<sup>10</sup>. This is impossible to achieve without highly automated deployment processes. The constantly growing DevOps community supports the implementation of automated processes by providing a huge variety of individual approaches such as tools and artifacts to implement holistic deployment automation. Reusable executables such as scripts, configuration definitions, and templates are publicly available to

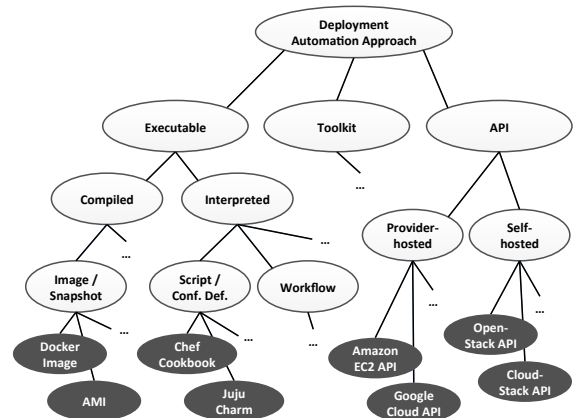


Figure 1: Deployment automation classification.

be used for deployment automation. Juju charms and Chef cookbooks are examples for these (Nelson-Smith, 2013; Sabharwal and Wadhwa, 2014). Such executables usually depend on certain tools. For instance, Chef cookbooks require a Chef runtime, whereas Juju charms need a Juju environment. This makes it challenging to reuse different kinds of heterogeneous artifacts in combination with others. Especially when systems have to be deployed that consist of various types of components, typically multiple tools have to be combined because they focus on different kinds of middleware and application components. Thus, there is a variety of solutions and orchestrating the best of them requires to integrate the corresponding tools, e.g., by writing scripts that handle the underlying lower-level invocations, parameter passing, etc. However, this is a difficult, costly, and error-prone task as many of the executables cannot be utilized through an API without relying on a central middleware component. Consequently, all artifact- and tooling-specific details (invocation mechanism, rendering input and output, etc.) have to be known and considered when integrating and orchestrating different kinds of executables. We tackle these issues with our work presented in this paper by generating APIs for individual executables. The generated APIs hide and abstract from artifact- and tooling-specific details, thereby significantly simplifying the integration and orchestration of very different kinds of artifacts.

Figure 1 shows an initial classification of deployment automation approaches. Executables are categorized in *compiled* and *interpreted* artifacts. Examples for compiled executables are pre-built virtual machine snapshots and container images such as Amazon machine images (AMI)<sup>11</sup> or Docker container images<sup>12</sup>.

<sup>9</sup>Flickr deployments per day: <http://goo.gl/VEmVqE>

<sup>10</sup>HubSpot deployments per day: <http://goo.gl/4AQy1h>

<sup>11</sup>AMIs: <http://goo.gl/S1Zx8Q>

<sup>12</sup>Docker Hub Registry: <https://registry.hub.docker.com>

In contrast to those, scripts and configuration definitions such as Chef cookbooks and Juju charms are interpreted at runtime. Beside executables, existing *APIs* can be utilized in two flavors: (i) provider-hosted APIs are offered by Cloud providers to provision virtual servers, storage, and other resources; (ii) self-hosted APIs are offered, e.g., by open-source Cloud management platforms such as OpenStack (Peppel, 2011). Our work focuses on transforming existing individual executables into self-hosted APIs by generating corresponding API implementations. As a result, full deployment automation can be achieved by integrating and orchestrating provider-hosted and self-hosted APIs without considering the tooling- and artifact-specific details of different kinds of executables. Moreover, this approach broadens the potential variety of tools and artifacts because their implementation-specific differences are completely hidden by using the generated API implementations.

Technically, the integration and orchestration of generated and existing APIs can be implemented using arbitrary scripting languages such as JavaScript, Ruby, or Python; alternatively, service composition languages such as BPMN (OMG, 2011) or BPEL (OASIS, 2007) may be used. For scripting languages, provider-independent and provider-specific *toolkits* are available to implement deployment plans that orchestrate and integrate different APIs. Examples are fog<sup>13</sup> and Google's API libraries<sup>14</sup>. Furthermore, general-purpose libraries to interact with different kinds of Web APIs are available for all major scripting languages: restler (JavaScript)<sup>15</sup>, node-soap (JavaScript)<sup>16</sup>, rest-client (Ruby)<sup>17</sup>, Savon (Ruby)<sup>18</sup>, etc.

## 2.2 Motivating Scenario: Facebook App

Considering the deployment automation use case discussed before, this section presents a comprehensive example as motivating scenario: the automated deployment of a Cloud-based Facebook application. The structure and parts of the application are shown in Figure 2. A *canvas frame*<sup>19</sup> is used to create and embed a corresponding application on the Facebook platform. The *canvas URL* points to an externally hosted Web application that is run based on a PHP runtime environment. It provides both the user interface and the underlying application logic. The PHP runtime

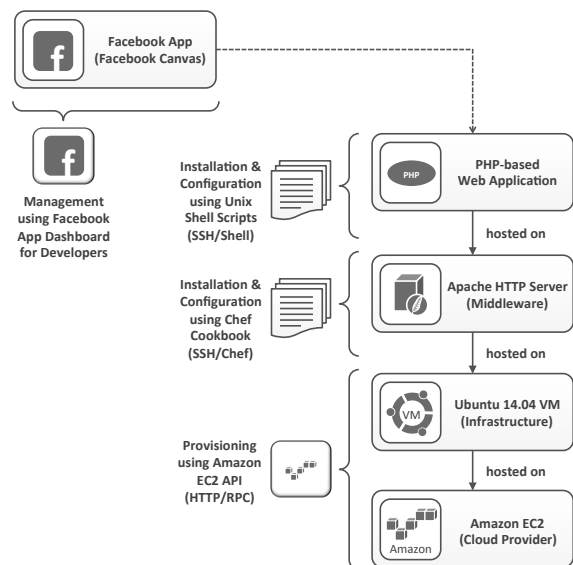


Figure 2: Facebook application stack.

itself is provided by an Apache HTTP server in conjunction with a PHP module. Both are deployed on a virtual machine, running Ubuntu 14.04 as operating system, which itself runs in the Cloud, hosted on Amazon's public infrastructure (EC2<sup>20</sup>). The scenario covers a typical setting used to deploy and run Web-based social applications as it employs and combines modern social media platforms such as Facebook as well as Cloud infrastructures such as Amazon EC2. It could be further refined, e.g., by connecting the Web application to a database that is provided by a database-as-a-service offering hosted on a different Cloud infrastructure.

To provision the complete application stack in an automated manner, different types of interfaces and invocation mechanisms have to be integrated. The virtual machine with its operating system is acquired by using the HTTP/RPC API provided by Amazon EC2. A Chef cookbook is executed on the virtual machine through an SSH connection to install the middleware of the application stack (Apache HTTP server). Furthermore, SSH is used to run custom Unix shell scripts to install and configure the actual Web application. However, remotely running executables such as Chef cookbooks and Unix shell scripts is not as straightforward as calling a well-defined API endpoint: (i) an executable needs to be placed on the virtual machine, e.g., using file transport protocols such as FTP and SCP. Moreover, (ii) the executable may require a particular runtime environment to be installed on the virtual machine such as a Chef runtime for Chef cookbooks. An SSH connection can be used to drive

<sup>13</sup>fog: <http://fog.io>

<sup>14</sup>Google APIs Client Libraries: <http://goo.gl/uVvFf>

<sup>15</sup>restler: <https://github.com/danwrong/restler>

<sup>16</sup>node-soap: <https://github.com/vpulim/node-soap>

<sup>17</sup>rest-client: <https://github.com/rest-client/rest-client>

<sup>18</sup>Savon: <http://savonrb.com>

<sup>19</sup>Facebook canvas frame: <http://goo.gl/5guKas>

<sup>20</sup>Amazon EC2: <http://aws.amazon.com/ec2>

the installation. Afterward, (iii) the execution of the scripts needs to be parameterized, which may be done by setting environment variables or storing configuration files. The final challenge is (iv) retrieving the results of the invocation, e.g., by reading, parsing, and potentially transforming the console output or files that were written to disk. In comparison to a simple API call, these steps are more complex and error-prone because lower-level implementation details such as different transport protocols and invocation mechanisms have to be considered and combined with each other. The overarching provisioning logic orchestrating all API calls as well as the preparation and invocation of the executables could be implemented by a script using a general-purpose scripting language such as Ruby or Python. However, such a script would be polluted with lower-level implementation details such as establishing SSH connections and placing files on the virtual machine. Furthermore, service composition languages such as BPEL or BPMN cannot be used without manually creating wrapping logic for the different executables involved. This is due to their focus on Web service orchestration. Consequently, the implementation details of the underlying APIs and executables directly influence which orchestration approaches can be used. This clearly contradicts with the idea of loose coupling, i.e., selecting an orchestration approach and implementing the orchestration logic without considering the implementation details of the underlying, lower-level technologies.

To tackle these challenges we propose an automated approach to generate APIs for arbitrary executables. The approach is based on the *APIfication method* we present in Section 3. In the context of our motivating scenario discussed in this section, the approach can be used to completely wrap the script invocation by generating an API that hides the (i) placement, (ii) installation of required runtime environments, (iii) parameterization and execution of the executable, as well as (iv) transforming and returning the results. Consequently, the orchestration logic deals with API calls only, without getting polluted, error-prone, or unnecessarily complex because of implementation details of the underlying executables.

### 3 APIfication METHOD

The APIfication approach presented in this section is based on the assumption that each executable has some metadata associated with it. These metadata are either natively attached and/or they are explicitly specified and additionally attached to the executable. Metadata indicate which input parameters are expected, where

results are put, which dependencies have to be resolved before the invocation, etc. The main purpose of a generated API implementation is to enable the invocation of the corresponding executable through a well-defined interface, independent from the underlying technology stack. Furthermore, a generated API implementation enables the invocation of the corresponding executable not only locally in the same environment (e.g., same server), but enables the execution using remote access mechanisms such as SSH and PowerShell in remote environments. This is to decouple the environment of an API implementation instance from the environment of the actual executable that is exposed by the API. Distributed environments as they are, for instance, used in the field of Cloud computing are thereby supported. An API call could be made from a workstation (running a script that orchestrates multiple APIs) to an API implementation instance that is hosted on premises (e.g., a local server); the actual executable (e.g., a Chef cookbook to install a middleware component) runs on a Cloud infrastructure. However, one could also run all parts on a single machine, e.g., a developer's laptop.

Figure 3 shows an overview of the *APIfication method*, outlining the individual steps and their ordering to generate API implementations in an automated manner. In the **first step**, the executable targeted for the APIfication is selected. Then, the interface type (e.g., RESTful API) and the API implementation type (e.g., Node.js or Java) is selected (**step 2 & 3**). The type of interface including the communication protocol (HTTP, WebSocket, etc.) and the communication paradigm (RPC, REST, etc.) can be chosen when generating an API implementation. This choice may be driven by existing expertise, alignment with existing APIs, or personal preferences. Similarly motivated, the type of the underlying implementation (Java, Node.js, etc.) for the generated API can be chosen when generating an API implementation. A generated API should be language-agnostic to allow the usage of arbitrary languages (scripting languages, programming languages, service composition languages, etc.) to orchestrate and integrate different APIs. Thus, Web APIs are the preferred and universal type of APIs because they can be utilized in nearly any kind of language.

After the selection part, the executable including its metadata is scanned to discover input and output parameters (**step 4**). If the scan did not discover all parameters, the following (optional) step can be used to refine the input and output parameters for the generated API (**step 5**). However, this is not required if the metadata associated with the executable are sufficient as this is, e.g., the case for many open-source deployment automation artifacts such as Chef cookbooks and Juju charms. Consequently, the method can be applied



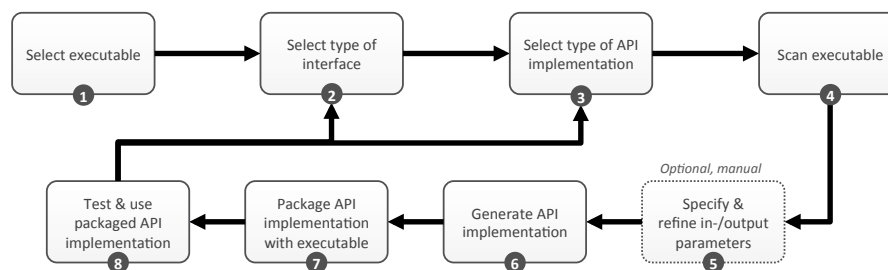


Figure 3: APIfication method.

to a huge amount and variety of such artifacts in an automated manner. Then, the API implementation is generated (**step 6**). To enable an API implementation to be hosted in different environments, it must be packaged in a portable manner (**step 7**). Thus, the implementation must be self-contained without depending on central middleware components, which dynamically provide data format transformations, parameter mappings, etc. at runtime. All these and related functionality are incorporated in the API implementation when it is generated at build time. The portability aspect is key for automated deployment processes because they need to run in very different environments (development, test, production, etc.). These environments may be hosted on different infrastructures (developer laptop, test cloud, etc.), so portability of the generated API implementations is key in this context. Technically, containerization technology (Scheepers, 2014; Turnbull, 2014) may be utilized for this purpose: each API implementation gets packaged as a portable container image that can be instantiated in different environments.

Later, the generated implementation may be refined or updated by going back to the selection steps for the interface type and the API implementation type. The APIfication method presented in this section addresses the challenges we identified in Section 2, including the deployment automation use case and the motivating scenario. However, the method itself is still abstract and can be implemented in various ways. The following section presents a modular and extensible framework to implement the APIfication method.

## 4 APIfication FRAMEWORK

In order to implement the APIfication method introduced in Section 3, we present a modular, plugin-based, and extensible framework in this chapter to support the individual steps of the method. Figure 4 shows several artifacts organized in multiple registries that are linked to the steps of the method, associated with certain actions (check, use, create). When se-

lecting an executable for its APIfication, the available *invokers* are checked (**action A**) if there is at least one invoker available that is capable of running the given type of executable (e.g., a Chef cookbook). Figure 5 outlines the registry, in which the invokers are stored: each invoker supports at least one *executable type*. For instance, the *Cookbook Invoker* can be used to run Chef cookbooks. The generator registry (Figure 6) is checked (**action B**) when selecting the interface type and the API implementation type. As an example, a Chef cookbook may be selected in conjunction with HTTP+REST as interface type and Node.js as implementation type. In this case all checks would succeed because the *Cookbook Invoker* is available and the *REST API Generator* can be used to generate an HTTP+REST interface; this is possible because the chosen generator can deal with Node.js as implementation type. Consequently, the generator uses the invoker to provide an API implementation that can run the given Chef cookbook.

Next, the given executable with its metadata is analyzed by a corresponding *scanner* (**action C**) from the scanner registry (structured similarly as invoker registry) to create an *API I/O specification* (**action D**). A scanner is a specialized module in the framework that is able to scan executables of a certain type such as a *Chef cookbook scanner* to scan cookbooks. Figure 7 shows an example for a specification (produced by a scanner) for a MySQL cookbook: it contains the input and output *parameter names*, their *data types*, and the *mapping information* to properly map between API parameters to the executable parameters at runtime. The *mode* of a parameter indicates whether this parameter is used as input or it is used to return some output of an invocation. Optionally, a *default value* can be associated with a parameter, which is used in case no value is defined at runtime for the corresponding parameter. In case the data type is *object*, a schema definition, e.g., XML schema (World Wide Web Consortium (W3C), 2012) or JSON schema (Internet Engineering Task Force, 2013) can be attached to the parameter. This is to specify the expected data structure for values (objects) of a particular parameter in more detail. The mapping of parameters spec-

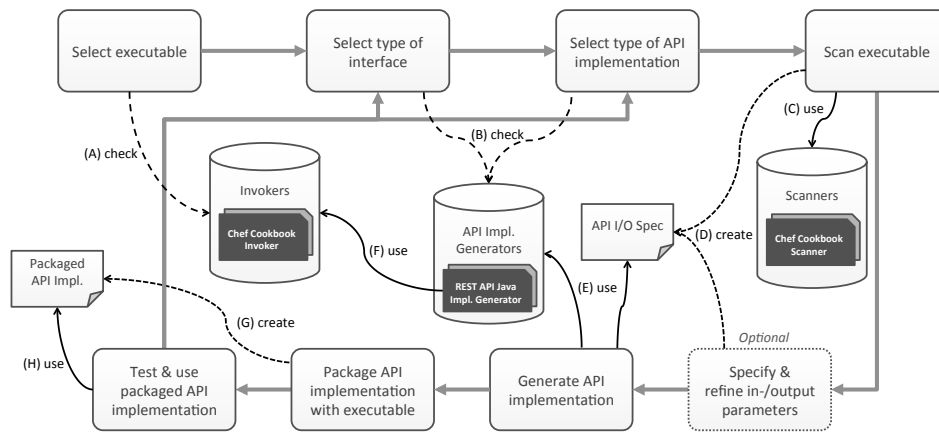


Figure 4: APIfication framework with technical examples.

Invoker	Executable Type
Cookbook Invoker	Chef Cookbook
Charm Invoker	Juju Charm
Docker Invoker	Docker Image
	Dockerfile
...	

Figure 5: Invoker registry.

ifies the target for input parameters and the source for output parameters at runtime. To refer to Figure 7: the API parameter `version` is mapped to the Chef attribute `mysql/version`, whereas the console output of the executable (STDOUT) is mapped to the API parameter `logs`. Optionally, the specification can be refined manually in the following step, which is not required if the executable's metadata is sufficient. The `invoker_config` parameter (mapped to the environment variable `INVOKER_CONFIG`) is a special one, provided by the framework; it cannot be modified or deleted during the (optional) manual refinement step. The parameter is used to configure the underlying invoker itself when using the generated API to run the executable. This is, for instance, needed to support remote access mechanisms, enabling the execution in remote environments. As an example the `invoker_config` parameter can hold the following JSON object to use SSH to run the executable remotely:

```
{
 "remote_access": "ssh",
 "remote_host": "173.194.44.88",
 "ssh_user": "ubuntu",
 "ssh_key": "-----BEGIN RSA PRIVATE KEY ..."
}
```

This sample configuration (given at runtime and transparently forwarded to the invoker) triggers the invocation of the underlying executable on the machine

Generator	Interface Type	Implementation Type
REST API Generator	HTTP+REST	Java
	HTTP+REST	Node.js
	HTTP+REST	Ruby
SOAP/WSDL API Gen.	HTTP+WSDL+SOAP	Java
	HTTP+WSDL+SOAP	Node.js
JSON-RPC API Gen.	HTTP+JSONRPC	Java
Node.js JSON-RPC API Gen.	HTTP+JSONRPC	Node.js
...		

Figure 6: Generator registry.

Param. Name	Mode	Data Type	Default	Param. Mapping
version	in	string	"5.1"	CHEF_ATTR:mysql/version
port	in	number	3306	CHEF_ATTR:mysql/port
logs	out	string	-	STDOUT
...				
invoker_config	in	object	-	ENV:INVOKER_CONFIG

Figure 7: API I/O spec for MySQL cookbook.

associated with the given IP address (`remote_host`) through SSH. Beside the special `invoker_config` parameter, the API I/O specification tells the corresponding *generator* how to create a proper API implementation (**action E**). A generator is a specialized module that performs the actual work to generate an API implementation. One part of the generation process is to put the corresponding invoker into the generated API implementation. The invoker is provided by the invoker registry to run the given executable (**action F**). Finally, the API implementation is packaged with the executable in a self-contained manner (**action G**). With this, the APIfication procedure for the given executable is finished, so the generated and packaged API implementation can be tested and used (**action H**). Figure 8 outlines the structure of a generated and packaged API implementation: the invoker (e.g., the cookbook invoker) is retrieved from the in-



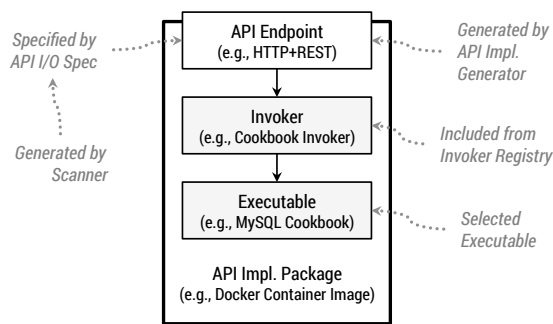


Figure 8: Generated API implementation package.

Invoker registry to invoke the selected executable such as the MySQL cookbook at runtime. The API endpoint is specified by the API I/O specification, which itself is generated by a scanner module provided by the framework. A generator module uses the specification to generate the implementation of the API endpoint. Finally, all parts are packaged in a self-contained manner, e.g., in a Docker container image. The following Section 5 presents the validation and evaluation of the APIfication method and framework we discussed in Section 3 and this section, based on a prototype implementation we provide.

## 5 VALIDATION & EVALUATION

In order to evaluate our APIfication method and framework, we developed ANY2API<sup>21</sup> as a prototype implementation. The following Section 5.1 presents and discusses the implementation. We performed experiments to measure the overhead both at build time and runtime (Section 5.2). Finally, Section 5.3 presents a comprehensive case study in the field of deployment automation.

### 5.1 ANY2API Implementation

ANY2API is a modular and extensible implementation of the APIfication framework presented in Section 4. Technically, it is based on Node.js, so most parts of it are implemented in JavaScript. Therefore, we use the Node Package Manager (NPM)<sup>22</sup> and the associated NPM registry to manage and publish Node.js modules. However, this does not imply that all parts of the framework have to be implemented in JavaScript. As an example, *invoker modules* expose several scripts that can (but do not have to) be implemented in JavaScript. Technically, these are specified as *NPM scripts*<sup>23</sup> in

<sup>21</sup>ANY2API: <http://any2api.org>

<sup>22</sup>NPM: <https://www.npmjs.org>

<sup>23</sup>NPM scripts: <http://goo.gl/0ss4NL>

the `package.json` file of a module:

```

"scripts": {
 "prepare-executable": "node ./prep-exec.js",
 "prepare-runtime": "sh ./prep-runtime.sh",
 "start": "java -jar ./invoke.jar"
}

```

Such a script can then be called using the `npm run` command, e.g., to trigger an invocation of an executable that is packaged with a generated API implementation: `npm run start`. This command is executed by the generated API implementation, which itself can be of an arbitrary implementation type such as a JAR file (Java) or a Node.js module (JavaScript). Moreover, the API implementation needs to set predefined environment variables before running the script such as `PARAMETERS` to parameterize the invocation accordingly. These environment variables contain JSON objects that are parsed and processed by the invoker. As an example, the input parameters for invoking a MySQL cookbook may be rendered as follows:

```

{ "version": "5.1", "port": 3306 }

```

At build time (i.e., when generating an API implementation) the `prepare-executable` script is triggered to prepare the packaged executable. Such preparations may include resolving all dependencies of a particular executable to package the executable in a truly self-contained manner. At runtime (i.e., when an invocation of the executable is triggered) the `prepare-executable` script is executed before the `start` script to install prerequisites required for the invoker to run such as a Java runtime environment.

*Generators* and *scanners* are implemented as Node.js modules, too. Each *generator module* exposes a `generate` function to produce an API implementation based on the given API I/O specification. Each *scanner module* exposes a `scan` function, which analyzes the given executable to generate an API I/O specification. This specification (after optional, manual refinement) can then be used in conjunction with a generator to produce an API implementation. To actually use and interact with the framework, the `any2api-cli`<sup>24</sup> module provides a command-line interface (CLI) to scan executables as well as to generate packaged API implementations:

```

any2api -o ./mysql_spec scan ./mysql_cookbook
any2api -o ./api_impl gen ./mysql_spec

```

The first command scans an existing Chef cookbook, generating an API I/O specification. Based on this specification, the second command generates a corresponding API implementation. By default, a Node.js-based API implementation exposing a RESTful interface is generated. A `Dockerfile`<sup>25</sup> (build plan

<sup>24</sup>`any2api-cli`: <https://github.com/any2api/any2api-cli>

<sup>25</sup>`Dockerfile` reference: <http://goo.gl/p5Tfdz>

to create a self-contained and portable container image) is included in each generated API implementation. Consequently, Docker can be used to create API implementation packages. Moreover, public and private Docker registries<sup>26</sup> can be utilized to store, manage, and retrieve potentially different versions of pre-built API implementations. Following this approach, a huge variety of existing tools that are part of the Docker ecosystem can be used to manage instances of generated API implementations. As an example, CoreOS<sup>27</sup> may be utilized to host API implementations in a managed cluster of Docker containers.

Currently, two *scanner modules* are implemented for analyzing Chef cookbooks and Juju charms. The *Chef invoker module* enables the invocation of Chef cookbooks, both in local and remote environments using SSH transparently. Using the *REST generator module*, Node.js-based RESTful API implementations can be generated. Further modules are currently being developed such as a Juju invoker, a Docker invoker, a Docker scanner, as well as alternative generators to support different type of interfaces (SOAP/WSDL, JSON-RPC, XML-RPC, etc.) and alternative implementation types (Java, Ruby, etc.).

## 5.2 Measurements

In order to evaluate the efficiency of our approach compared to the plain usage of the corresponding executable, we measured the overhead of the APIfication. Therefore, we generated API implementations for a selection of the most downloaded Chef cookbooks<sup>28</sup>, covering the automated installation and configuration of very common and widely used middleware components, including `mysql`, `apache2`, `php`, `nginx`, `postgresql`, and others. As an example, `apache2` and `php` are required for the automated deployment of the Facebook application we outlined in the motivating scenario (Section 2.2). First, we measured the overall duration it takes to scan the executable (Chef cookbook) and to generate a corresponding API implementation (Node.js-based RESTful API). Second, we check the additional size of the generated API implementation without the corresponding executable. This is to estimate the disk space that is additionally required at runtime when using an instance of an API implementation. Third, we measured the execution duration and memory usage for running the corresponding executable both *with* and *without* using the generated API implementation. The evaluation was run on a clean virtual machine (4 virtual CPUs

clocked at 2.8 GHz, 64-bit, 4 GB of memory) on top of the VirtualBox hypervisor, running a minimalist Linux system including Docker. The processing and invocation of a particular Chef cookbook was done in a clean Docker-based Ubuntu 14.04 container, with exactly one container running on the virtual machine at a time. We did all measurements at container level to completely focus on the workload that is linked to the executable and the API implementation.

Table 1 shows the results of our evaluation. The measured average duration to scan and generate an API implementation is in the range from 7 to 90 seconds. This duration is the overhead at build time, including the retrieval of the executable and all its dependencies. The additional size of the generated API implementation leads to slightly more disk space usage at runtime. Moreover, there is a minor overhead in terms of execution duration and memory consumption at runtime. In most of today's environments this overhead should be acceptable, considering the significant simplification of using the generated APIs compared to the plain executables. In addition, when using the plain executables directly, much of the complexity hidden by the generated API implementation has to be covered at the orchestration level. So, the overall consumption of resources may be the same or even worse, depending on the selected means for orchestration. Furthermore, instances of API implementations can be reused to run an executable multiple times and potentially in different remote environments. Through this reuse, the overhead can be quickly compensated in large-scale environments.

## 5.3 Deployment Automation Case Study

We used the presented APIfication approach to ease implementing and generating workflows for the deployment of Cloud applications based on the OpenTOSCA ecosystem (Binz et al., 2013; Kopp et al., 2013). This ecosystem is based on the TOSCA standard (Binz et al., 2014), which enables describing Cloud applications and their management in a portable fashion. To define management tasks imperatively, e.g., to migrate application components, the ecosystem employs management plans based on the workflow language BPEL (OASIS, 2007). Therefore, the orchestration of management scripts, APIs, and other executables is a major challenge. The presented APIfication approach eases developing management workflows significantly as it reduces the required effort and complexity of integrating different technologies. Using our approach, modeling management workflows requires the orchestration of APIs only, which is much more straightforward compared to the former integration of various hetero-

<sup>26</sup>Docker registry: <http://goo.gl/2lgohL>

<sup>27</sup>CoreOS: <https://coreos.com>

<sup>28</sup>Most downloaded cookbooks: <http://goo.gl/8xZUCT>

Table 1: Measurements regarding generated API implementations for Chef cookbooks.

	mysql	apache2	java	nginx	zabbix	glassfish	postgresql	php	...
<i>Avg. duration to scan and generate API impl.</i>	13s	14s	7s	25s	90s	17s	16s	29s	
<i>Add. size of generated API implementation</i>	25M	25M	25M	25M	25M	25M	25M	25M	
<i>Avg. execution duration with API impl.</i>	54s	48s	84s	45s	47s	153s	60s	123s	
<i>Avg. execution duration without API impl.</i>	54s	39s	82s	39s	42s	140s	59s	110s	
<i>Max. memory usage with API impl.</i>	556M	471M	507M	461M	429M	674M	510M	614M	
<i>Max. memory usage without API impl.</i>	343M	258M	402M	270M	212M	456M	310M	426M	

geneous technologies. Combined with the generated APIs for Chef cookbooks as discussed in Section 5.2, the integration of both the ecosystem and our APIfication approach provides a powerful means to enable a fast development of management workflows for Cloud applications.

## 6 FURTHER USE CASES

Beside the deployment automation use case (Section 2) we were focusing so far, we identified further use cases to apply our APIfication approach presented in this paper. In the *cyberinfrastructure & e-science community* (Yang et al., 2011) scientific applications are utilized, orchestrated, and run in Grid and Cloud environments to perform complex and CPU-intensive calculations such as scientific simulations and other experiments. These applications are implemented in arbitrary programming or scripting languages; they are usually run as executables directly. Consequently, they cannot be directly utilized through APIs. Existing works focus on the usage of scientific applications through Web APIs (Afanasiev et al., 2013; Sukhoroslov and Afanasiev, 2014) to ease their integration and orchestration for more sophisticated experiments, where multiple scientific applications are involved. As an example, Opal (Krishnan et al., 2009) is a framework for wrapping scientific applications, so they can be used through a Web API, abstracting from the application-specific details and differences such as invocation mechanisms and parameter passing. We tackle these challenges with our work by generating API implementations and packaging them together with the actual scientific application, i.e., the executable. This eases the integration and orchestra-

tion of different scientific application through Web APIs, without having to create API wrappers manually from scratch. As a result, running complex experiments that involve several scientific applications becomes easier.

In the previously described use cases of deployment automation and e-science, we implicitly assumed an executable to be an individual file or a collection of files (scripts, compiled executables, scientific applications, etc.). However, existing API endpoints as they are, e.g., exposed by provider-hosted Cloud APIs and social media APIs (Facebook<sup>29</sup>, Twitter<sup>30</sup>, etc.) can be considered as executables, too. This is motivated by the need for wrapping existing API endpoints to make them available through different communication protocols (e.g., wrap WebSocket by HTTP) or communication paradigms (e.g., wrap RPC by REST). As an example, Twitter provides the *users/show* endpoint<sup>31</sup> to retrieve a variety of information about a particular Twitter user. If this API endpoint needs to be utilized in a deployment workflow implemented in BPEL, a wrapper has to be implemented to make the endpoint accessible through a WSDL/SOAP-based interface (Wettinger et al., 2014a). By treating API endpoints as executables, API implementations could potentially be generated for existing endpoints to make them accessible through different protocols and communication paradigms, without relying on central middleware components such as a service bus.

<sup>29</sup>Facebook Graph API: <http://goo.gl/HKGpZG>

<sup>30</sup>Twitter REST API: <https://dev.twitter.com/rest>

<sup>31</sup>Twitter *users/show* API endpoint: <http://goo.gl/dmsJ22>

## 7 RELATED WORK

As discussed in Section 1 and Section 2, using and creating APIs is of utmost importance today (Rudrakshi et al., 2014). Consequently, a huge variety of approaches is available to simplify the creation and development of APIs. Beside API development frameworks to create API implementations manually (e.g., Hapi<sup>32</sup> and LoopBack<sup>33</sup>), there are solutions to semi-automatically create Web APIs. As an example, API specifications defined using the *RESTful API Modeling Language (RAML)*<sup>34</sup> can be utilized to generate an API implementation skeleton based on Jersey<sup>35</sup>, a Java framework to develop RESTful APIs (Masse, 2011; Richardson et al., 2013). These generated skeletons have to be refined by adding application-specific logic. Consequently, such approaches can be immediately used to develop *generator modules* for our APIfication framework: the generator produces a skeleton, which is then automatically refined by adding the logic to call a corresponding invoker to run the selected executable. Moreover, solutions such as Kimono<sup>36</sup> and Import.io<sup>37</sup> can be used to generate Web APIs for existing Web sites. These approaches provide interactive ways to extract content from HTML pages (e.g., using CSS selectors) to make them available in more machine-readable formats such as JSON. Thus, such Web page-centric approaches focus on extracting and re-formatting content, whereas our approach tackles the issue of managing the invocation of arbitrary executables. In contrast to service providers such as Kimono, our approach aims to generate self-contained, portable, and packaged API implementations that can be hosted anywhere, so they do not depend on specific provider offerings.

RPC frameworks such as Apache Thrift<sup>38</sup> and Google's Protocol Buffers<sup>39</sup> aim to ease the integration of application logic and executables that are implemented based on different technology stacks. For efficiency reasons, they typically do not rely on Web APIs but use lower-level TCP connection-based protocols. Such RPC frameworks can be perfectly combined with our APIfication approach by implementing *generator modules*. In this case, a module generates an API implementation, e.g., exposing a Thrift interface instead of an HTTP-based RESTful interface. Some of these

frameworks offer support to generate code skeletons based on interface descriptions. This functionality can be reused to ease the implementation of a corresponding *generator module*. However, by sticking to such non-standard communication protocols there are limitations on the orchestration level, meaning the same framework has to be used instead of interacting with a standards-based interface such as HTTP/REST. This is a trade-off between efficiency and interoperability that needs to be made individually based on concrete use cases. Since our framework supports both approaches, different API implementations (e.g., Thrift-based and HTTP/REST-based) can be generated and exchanged for a particular executable as needed. In the field of Web APIs, approaches such as websockify<sup>40</sup> and websocketd<sup>41</sup> can be used to expose the functionality of executables through the standards-based WebSocket protocol (IETF, 2011). Corresponding *generator modules* can be implemented to reuse these approaches in the context of our APIfication framework.

## 8 CONCLUSION

In this paper we introduced an automated APIfication approach to ease the integration and orchestration of arbitrary executables. To fulfill the requirements derived from the deployment automation use case and the motivating scenario, we presented a generic APIfication method and a corresponding framework to automatically generate API implementations. In order to confirm the practical feasibility of the presented method and framework, we published ANY2API as a modular and extensible implementation. To analyze the efficiency of our approach, we conducted an evaluation with comprehensive measurements. The measured results show a small overhead when following the APIfication approach, which is acceptable for most use cases, considering the significant simplification and convenience that our approach provides. In addition, we did a case study in the field of deployment automation to confirm the actual applicability of our approach in practice. Finally, we outlined additional use cases in different fields to apply the proposed APIfication approach.

In terms of future work, we are going to extend the APIfication framework to support an additional but optional step to refine the parameter mapping (e.g., aggregating, splitting, or transforming parameter values). For this reason we intend to enable the definition of JavaScript functions that are executed in a sandboxed

<sup>32</sup>Hapi: <http://hapijs.com>

<sup>33</sup>LoopBack: <http://loopback.io>

<sup>34</sup>RAML: <http://raml.org>

<sup>35</sup>RAML to JAX-RS (Jersey): <http://goo.gl/E39jun>

<sup>36</sup>Kimono: <https://www.kimonolabs.com>

<sup>37</sup>Import.io: <https://import.io>

<sup>38</sup>Apache Thrift: <http://thrift.apache.org>

<sup>39</sup>Protocol Buffers: <http://goo.gl/uq69p>

<sup>40</sup>websockify: <https://github.com/kanaka/websockify>

<sup>41</sup>websocketd: <https://github.com/joewalnes/websocketd>

environment at runtime. Moreover, we plan to extend and refine the ANY2API implementation. Existing scanners, generators, and invokers will be refined, and additional ones will be implemented. As an example, refinement may include authentication and authorization mechanisms for generated API implementations. The currently implemented generators can be used to create API implementations that expose Web APIs such as HTTP/REST. In future, we plan to implement generators in conjunction with alternative packaging formats to generate API libraries that can be directly used in certain programming and scripting languages such as Java and Python.

## ACKNOWLEDGEMENTS

This work was partially funded by the BMWi project *CloudCycle* (01MD11023) and the DFG project *SitOPT* (610872).

## REFERENCES

- Afanasiev, A., Sukhoroslov, O., and Voloshinov, V. (2013). MathCloud: Publication and Reuse of Scientific Applications as RESTful Web Services. In *Parallel Computing Technologies*. Springer.
- Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., and Wagner, S. (2013). OpenTOSCA – A Runtime for TOSCA-based Cloud Applications. In *Proceedings of the 11th International Conference on Service-Oriented Computing*, LNCS. Springer.
- Binz, T., Breitenbücher, U., Kopp, O., and Leymann, F. (2014). TOSCA: Portable Automated Deployment and Management of Cloud Applications, pages 527–549. *Advanced Web Services*. Springer.
- Guinard, D., Trifa, V., and Wilde, E. (2010). A Resource Oriented Architecture for the Web of Things. In *Internet of Things (IOT), 2010*. IEEE.
- Humble, J. and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional.
- Hüttermann, M. (2012). *DevOps for Developers*. Apress.
- IETF (2011). The WebSocket Protocol.
- Internet Engineering Task Force (2013). JSON Schema.
- Kopp, O., Binz, T., Breitenbücher, U., and Leymann, F. (2013). Winery - A Modeling Tool for TOSCA-based Cloud Applications. In *Proceedings of the 11th International Conference on Service-Oriented Computing*, volume 8274 of LNCS. Springer Berlin Heidelberg.
- Krishnan, S., Clementi, L., Ren, J., Papadopoulos, P., and Li, W. (2009). Design and Evaluation of Opal2: A Toolkit for Scientific Software as a Service. In *World Conference on Services I*. IEEE.
- Masse, M. (2011). *REST API Design Rulebook*. O'Reilly Media, Inc.
- Mell, P. and Grance, T. (2011). The NIST Definition of Cloud Computing. *National Institute of Standards and Technology*.
- Nelson-Smith, S. (2013). *Test-Driven Infrastructure with Chef*. O'Reilly Media, Inc.
- OASIS (2007). Web Services Business Process Execution Language (BPEL) Version 2.0.
- OMG (2011). Business Process Model and Notation (BPMN) Version 2.0.
- Pepple, K. (2011). *Deploying OpenStack*. O'Reilly Media.
- Richardson, L., Amundsen, M., and Ruby, S. (2013). *RESTful Web APIs*. O'Reilly Media, Inc.
- Rudrakshi, C., Varshney, A., Yadla, B., Kanneganti, R., and Somalwar, K. (2014). API-fication - Core Building Block of the Digital Enterprise. Technical report, HCL Technologies.
- Sabharwal, N. and Wadhwa, M. (2014). *Automation through Chef Opscode: A Hands-on Approach to Chef*. Apress.
- Scheepers, M. J. (2014). Virtualization and Containerization of Application Infrastructure: A Comparison.
- Sukhoroslov, O. and Afanasiev, A. (2014). Everest: A Cloud Platform for Computational Web Services. In *Proceedings of the 4th International Conference on Cloud Computing and Services Science*. SciTePress.
- Turnbull, J. (2014). *The Docker Book*. James Turnbull.
- W3C (2007). SOAP Specification, Version 1.2.
- Wettinger, J., Binz, T., Breitenbücher, U., Kopp, O., Leymann, F., and Zimmermann, M. (2014a). Unified Invocation of Scripts and Services for Provisioning, Deployment, and Management of Cloud Applications Based on TOSCA. In *Proceedings of the 4th International Conference on Cloud Computing and Services Science*. SciTePress.
- Wettinger, J., Breitenbücher, U., and Leymann, F. (2014b). Standards-based DevOps Automation and Integration Using TOSCA. In *Proceedings of the 7th International Conference on Utility and Cloud Computing (UCC)*.
- World Wide Web Consortium (W3C) (2012). XML Schema.
- Yang, X., Wang, L., and Jie, W. (2011). *Guide to e-Science*. Springer.

# A Modelling Concept to Integrate Declarative and Imperative Cloud Application Provisioning Technologies

Uwe Breitenbücher<sup>1</sup>, Tobias Binz<sup>1</sup>, Oliver Kopp<sup>1,2</sup>, Frank Leymann<sup>1</sup> and Johannes Wettinger<sup>1</sup>

<sup>1</sup>IAAS, University of Stuttgart, Stuttgart, Germany

<sup>2</sup>IPVS, University of Stuttgart, Stuttgart, Germany  
{breitenbuecher, lastname}@iaas.uni-stuttgart.de

**Keywords:** Cloud Application Provisioning, Automation, Declarative Modelling, Imperative Modelling.

**Abstract:** Efficient application provisioning is one of the most important issues in Cloud Computing today. For that purpose, various provisioning automation technologies have been developed that can be generally categorized into two different flavors: (i) declarative approaches are based on describing the desired goals whereas (ii) imperative approaches are used to describe explicit sequences of low-level tasks. Since modern Cloud-based business applications become more and more complex, employ a plethora of heterogeneous components and services that must be wired, and require complex configurations, the two kinds of technologies have to be integrated to model the provisioning of such applications. In this paper, we present a process modelling concept that enables the seamless integration of imperative and declarative provisioning models and their technologies while preserving the strengths of both flavors. We validate the technical feasibility of the approach by applying the concept to the workflow language BPEL and evaluate its features by several criteria.

## 1 INTRODUCTION

With the growing adoption of Cloud Computing in enterprises, the rapid and reliable provisioning of Cloud applications becomes a more and more important task. Especially the increasing number of available Cloud services offered by providers, e. g., Amazon and Google, provide powerful Cloud properties such as automatic elasticity, self-service, or pay-per-use features that are provided completely by the autonomous management capabilities of Cloud environments (Leymann, 2009). Due to this trend, more and more business applications are outsourced to the Cloud (Binz et al., 2014). As a result, Cloud-based business applications become (i) increasingly complex and (ii) employ a plethora of heterogeneous software, middleware, and XaaS components offered by different providers including non-trivial dependencies among each other. Consequently, the provisioning of such applications becomes a serious management challenge: (i) different kinds of Cloud offerings (IaaS, PaaS, SaaS, etc.) must be provisioned and (ii) complex configurations are required to setup and wire involved components. This typically requires the combination of multiple management technologies, especially if the application components are distributed

across multiple Clouds (Breitenbücher et al., 2013).

However, combining (i) proprietary APIs, (ii) non-standardized configuration management tools, and (iii) different virtualization technologies in a single automated provisioning process is a complex modelling and integration challenge using traditional approaches such as workflows. The main reason for this complexity results from the nature of technologies that have to be combined: There are *declarative technologies*, such as Chef (Opscode, Inc., 2015; Nelson-Smith, 2013) or Puppet (Puppet Labs, Inc., 2015), which only describe the desired goal state of application components without specifying the actual tasks that have to be executed to reach this state. *Imperative technologies*, e. g., scripts or workflows, explicitly specify each technical step to be executed in detail. Although there are technologies for orchestrating imperative approaches with each other homogeneously (Kopp et al., 2012), combining declarative and imperative approaches results in implementing huge amounts of wrapper code as the two flavors are hardly interoperable with each other as there is no means to orchestrate them seamlessly.

In this paper, we tackle these issues. The first contribution is a detailed state of the art analysis of declarative and imperative provisioning approaches

including a critical evaluation. To tackle the analyzed issues, we present a modelling approach that enables integrating declarative and imperative provisioning models and the corresponding technologies seamlessly. We introduce the concept of *Declarative Provisioning Activities* that allows describing declarative goals directly in the control and data flow of an imperative workflow model. Based on our approach, developers are able to model provisioning workflows that specify not only imperative statements but declarative statements as well—without polluting the model with technical integration details. The approach enables to benefit from the strengths of both flavors: Declarative models can be used to specify high-level management goals, whereas imperative logic enables modelling complex cross-cutting configuration and wiring tasks on a very low level of technical abstraction. We validate the technical feasibility of the approach by presenting a prototypical implementation, which is integrated in the standards-based Cloud management ecosystem OpenTOSCA (Binz et al., 2013; Breitenbücher et al., 2014; Kopp et al., 2013) and the imperative workflow language BPEL (OASIS, 2007). We evaluate the approach by several criteria based on the conducted analysis and discuss its limitations.

The remainder of this paper is structured as follows: In Section 2, we conduct a detailed analysis regarding declarative and imperative provisioning technologies as well as combination concepts. Section 3 presents our approach of Declarative Provisioning Activities, which is validated in terms of a prototypical implementation in Section 5 and evaluated in Section 6. Section 7 concludes the paper and gives an outlook on future work.

## 2 STATE OF THE ART ANALYSIS

In this section, we conduct a detailed state of the art analysis of declarative and imperative provisioning approaches and existing technologies including a critical evaluation. Afterwards, we discuss related work that attempts to combine the two flavors.

### 2.1 The Declarative Flavor

Declarative approaches can be used to describe the provisioning of an application by modelling its desired goal state, which is enforced by a declarative provisioning system. They typically employ domain-specific languages (DSLs) (Günther et al., 2010) to describe goals in a declarative way, i.e., only the *what* is described without providing any details about the technical *how*. For example, a declarative spec-

ification may describe that a Webserver has to be installed on a virtual machine, but without specifying the technical tasks that have to be performed to reach this goal. The main strength of declarative approaches is that the technical provisioning logic, i.e., the technical tasks to be performed, is inferred automatically by the provisioning system, which eases modelling provisionings as the technical execution details are hidden (Herry et al., 2011). One of the most prominent examples of declarative provisioning description languages is Amazon CloudFormation<sup>1</sup>. This JSON-based language enables to describe the desired application deployment using Amazon's Cloud services including their configuration in a declarative model, which is consumed to fully automatically setup the application. In comparison to such provider-specific languages, which quickly lead to a vendor lock-in, provider-independent technologies were developed such as Puppet (Puppet Labs, Inc., 2015).

Due to the automatic inference of provisioning logic, declarative systems have to understand the declared statements. This restricts declarative provisioning capabilities to standard component types and predefined semantics that are known by the runtime (Breitenbücher et al., 2014). Thus, individual customizations for the provisioning of complex application structures cannot be realized arbitrarily and have to comply with the general, overall provisioning logic. As a consequence, the declarative approach is rather suited for applications that consist of common components and configurations, but is limited in terms of deploying big, complex business applications that require specific configurations with non-trivial component dependencies. Even mechanisms to integrate script executions, API calls, or service invocations at certain points in their deployment lifecycle, as supported by many declarative approaches, do often not provide the required flexibility as the overall logic cannot be changed arbitrarily. As the integration of other technologies is often not supported natively, models get polluted by glue and wrapper code, which results in complex models including low-level technical integration details (Wettinger et al., 2014). Nevertheless, the declarative flavor is very important due to (i) native support by Cloud providers and (ii) huge communities providing reusable artifacts.

### 2.2 The Imperative Flavor

In contrast to the declarative flavor, the imperative provisioning approach enables developers to specify each technical detail about the provisioning execution

<sup>1</sup><http://aws.amazon.com/cloudformation/>

by creating an explicit process model that can be executed fully automatically by a runtime. Imperative models define (i) the control flow of activities, (ii) the data flow between them, as well as (iii) all technical details required to execute these activities. Thus, compared to declarative approaches, they describe not only *what* has to be done, but also *how* the provisioning tasks have to be executed. Imperative processes are typically implemented using (i) programming languages such as Java, (ii) scripting languages, e.g., Bash or Python, and (iii) workflow languages such as BPEL (OASIS, 2007) or BPMN (OMG, 2011). However, programming and scripting languages are not suited for the provisioning of serious business applications as they are not able to provide the robust and reliable execution features that are supported by the workflow technology (Leymann and Roller, 2000; Herry et al., 2011). Since general-purpose workflow languages do not natively support modeling features for application provisioning, we developed BPMN4TOSCA (Kopp et al., 2012), which is a BPMN extension that supports API calls, script-executions, and service invocations based on the TOSCA standard (OASIS, 2013) (a standard to describe Cloud applications). This language can be used to seamlessly integrate such tasks as it provides a separate activity-type for each of them. However, BPMN4TOSCA lacks support for the direct integration of declarative provisioning technologies, which need to be wrapped for their invocation. Thus, similar to general-purpose technologies, seamlessly integrating domain-specific technologies into one process is not possible. To wrap management technologies, we presented a management bus that provides a unified API for the invocation of arbitrary technologies (Wettinger et al., 2014). However, invoking the bus obfuscates the actual technical statements, which impedes maintaining and understanding process models.

Imperative approaches are suited to model complex provisionings that employ a plethora of heterogeneous components, especially for multi-cloud applications (Petcu, 2014). As they provide full control over the tasks to be executed, imperative models are able to automate exactly the manual steps that would be executed by a human administrator who provisions the application manually. Thus, while declarative approaches are rather suited for standard provisionings, imperative approaches enable developers to define arbitrary provisioning logic. The main drawback of the imperative approaches results from the huge amount of statements that must be modelled since the runtime infers no logic by itself. Consequently, manual process authoring is a labor-intensive, time-consuming, and error-prone task that requires a lot of low-level,

technical expertise in different fields (Breitenbücher et al., 2014; Breitenbücher et al., 2013): Heterogeneous services need to be orchestrated (e.g., SOAP-based and RESTful provider APIs), low-level technologies must be integrated, and, especially, declarative technologies must be wrapped. As currently no technology supports the seamless integration of both flavors, their orchestration results in large, polluted, technically complex processes that require multiple different wrappers to support the various invocation mechanisms and protocols (Wettinger et al., 2014). These wrappers decrease the transparency as only simplified interfaces are exposed to the orchestrating process while the technical details, which are in many cases of vital importance to avoid errors when modelling multiple steps that depend on each other, are abstracted completely. In addition, wrappers significantly impede maintaining processes as not simply the orchestration process has to be adapted, but wrapper code needs to be modified and built again, too.

## 2.3 Integration Approaches

In this section, we present related work that attempts to combine both flavors. There are several general purpose concepts that attempt to bridge the gap between imperative provisioning logic and declarative models which generate provisioning workflows by analyzing the declarative specifications (Breitenbücher et al., 2014; Breitenbücher et al., 2013; Eilam et al., 2011; Keller et al., 2004; El Maghraoui et al., 2006; Herry et al., 2011; Levanti and Ranganathan, 2009; Mietzner, 2010). These approaches are able to interpret declarative specifications modelled using a domain-specific modelling language for generating provisioning plans, which can be executed fully automatically. The advantage of these approaches is the full control over the executed provisioning steps as the resulting workflows can be adapted and configured arbitrarily. However, the complexity, lack of transparency, and the polluted control and data flows of the resulting workflows are still problems that impede extending the plans if customization is required. Thus, the approach we present in this paper may be applied to these technologies for improving the quality of the generated processes. As a result of the discussion in this section, to ensure correct operation and to ease the creation of complex provisioning processes for non-trivial business applications, it is of vital importance to employ an extensible orchestration approach that supports the seamless orchestration of imperative and declarative provisioning technologies. Therefore, the main goals of this paper are (i) a seamless integration of declarative and im-



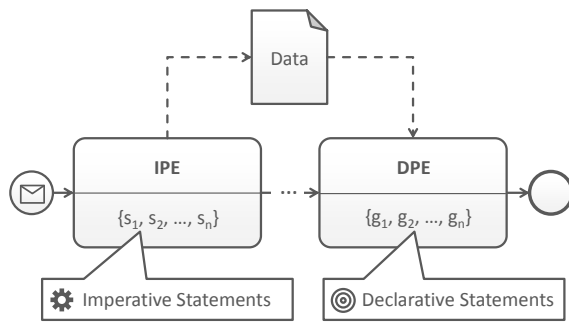


Figure 1: Concept of the direct integration approach.

perative provisioning modelling approaches as well as (ii) the orchestration of the corresponding provisioning technologies through workflow models.

### 3 INTEGRATED MODELLING

In this section, we present an approach that enables integrating declarative and imperative provisioning models seamlessly into the control and data flow of an imperative workflow. In Section 3.1, we introduce the abstract concept of the approach in a technology-independent manner and define data handling concepts in Section 3.2. In Section 4, we apply the approach to the workflow language BPEL in order to show how the concept can be realized using a concrete standardized workflow language.

#### 3.1 Declarative Provisioning Activities

The general modelling approach is shown in Figure 1 and based on extending standardized, imperative workflow languages such as BPMN or BPEL by the concept of *Declarative Provisioning Activities*. These activities enable to specify declarative provisioning goals directly in the control flow of a workflow model that describes the tasks to provision a certain application. To present the conceptual contribution independently from a concrete workflow language, we first introduce the general concept in an abstract way and show its applicability to the standardized workflow language BPEL afterwards. Therefore, in this section, we distinguish only between (i) Imperative Provisioning Activities and (ii) Declarative Provisioning Activities that abstract from concrete realizations of provisioning tasks in different workflow languages. Of course, other control and data flow constructs, such as events and gateways, are also required to model executable processes. However, these are language-specific and do not influence the presented concept.

An *Imperative Provisioning Activity (IPA)* describes a technically detailed execution of a provisioning task as a sequence of one or more imperative statements. This can be, for example, a script implemented in Python or a simple HTTP-POST request that specifies a URL and data to be sent. Thus, the term is an abstraction of several existing imperative approaches such as scripts and programs that implement a workflow activity or the invocation of an API etc. The modeling and execution of such Imperative Provisioning Activities is supported natively by many workflow languages through general-purpose concepts or by domain-specific extensions, respectively. For example, BPMN natively supports the execution of script-tasks (OMG, 2011), the BPEL extension BPEL4REST (Haupt et al., 2014) enables sending arbitrary HTTP requests, and BPMN4TOSCA natively supports orchestrating provisioning operations based on the TOSCA-standard—especially the execution of configuration scripts on a target VM (Binz et al., 2013; Wettinger et al., 2014). This enables orchestrating arbitrary provisioning tasks using workflows that describe the technical details required for the automated provisioning of complex applications.

In contrast to this, we introduce the new concept of *Declarative Provisioning Activities (DPA)* in this paper that enables specifying desired provisioning goals in a declarative manner. A DPA consists of a set of declarative statements that describe *what* has to be achieved, e.g., a desired configuration of a certain application component, but without specifying any technical details about *how* to achieve the declared goals. Similar to other activity-constructs of workflow languages, Declarative Provisioning Activities are modelled directly in the control and data flow of the process model the same way as IPAs. This enables combining Imperative and Declarative Provisioning Activities intuitively while preserving a clear understanding about the overall flow. The operational semantics of Declarative Provisioning Activities are defined as follows: If the control flow reaches the activity, the declarative statements, i.e., the modelled goals, are enforced by the runtime that executes the workflow. The activity is executed until all goals are achieved and all affected application components are in the desired state specified by the DPA. Then, the activity completes and the control flow continues following the links to the next activities. Process models that contain both provisioning activity types are called *Integrated Provisioning Models* in this paper.

Figure 2 shows an example of an Integrated Provisioning Model that contains two Imperative Provisioning Activities and one Declarative Provisioning Activity, which (i) instantiate a virtual machine,

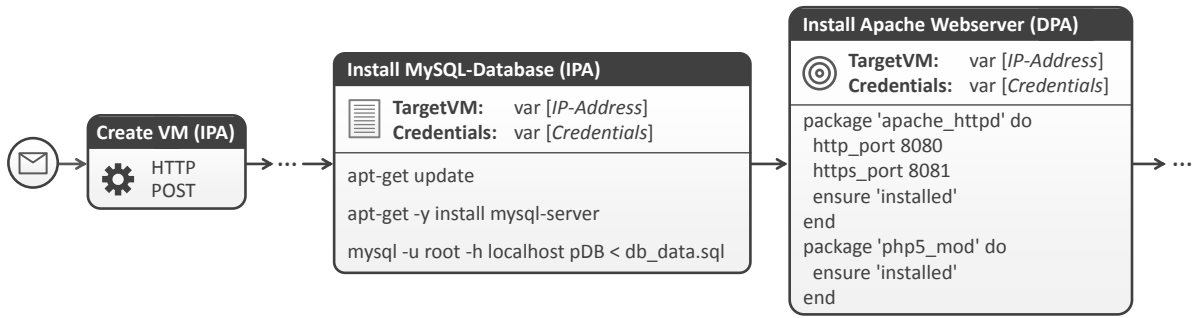


Figure 2: Simplified example of an Integrated Provisioning Model that (i) instantiates a virtual machine, (ii) installs a database, and (iii) installs a Webserver on the virtual machine (We omitted some tasks for reasons of space).

(ii) install a MySQL-database, and (iii) install an Apache Webserver on the virtual machine. The first IPA is an HTTP request to an API of a Cloud provider or an infrastructure virtualization technology that triggers the instantiation of the virtual machine. The activity specifies the request including all required configuration parameters and invokes the API correspondingly. After waiting for the successful instantiation, the IP-address and SSH credentials of the VM, which can be polled at the API, are stored in two variables of the workflow model: “IP-Address” and “Credentials”. As these are standard tasks, we omit details in the figure for reasons of space.

The second IPA installs a MySQL database on the VM: The shown activity uses a low-level Bash script that imperatively specifies statements to be executed to install the database and to import a referenced SQL-file, which is uploaded to the VM by an IPA (omitted in the figure). To copy and execute this script on the VM, the process variables that store the IP-Address and SSH credentials of the target VM are used by the IPA to access the virtual machine via SSH and to execute the imperatively specified statements.

To model the installation and configuration of the Apache Webserver on the virtual machine, the DSL of the configuration management technology Chef (Opscode, Inc., 2015) is used to declaratively define the desired installation. Consequently, a Declarative Provisioning Activity is modelled that specifies the desired goals by declaratively describing the state and configuration of the Webserver that has to be enforced when executing the activity. Similarly to the second script-based IPA, the activity employs the same process variables to access the virtual machine.

This example shows that the direct integration of declarative and imperative languages and technologies in one orchestration process provides a powerful modeling approach as the corresponding imperative programming or scripting-languages, respectively, as well as the domain-specific languages of declarative approaches can be used seamlessly in one process

model. Therefore, there is no need to write complex wrapper code or to invoke services wrapping these technologies that pollute the process model. Thus, the approach enables using the *right* technology for the *right* task while ensuring full-control over their orchestration without polluting the workflow model.

### 3.2 Data Handling

Both types of activities exchange data within the workflow. Therefore, we define three concepts including their operational semantics that enable describing the data flow between provisioning activities: (i) Input parameters, (ii) output parameters, and (iii) content injection. We continue abstracting from individual data storage concepts of workflow languages by simply referring to “process variables” and show in the next section how these concepts can be realized in the concrete workflow language BPEL.

As shown in Figure 2, the script-activity and the declarative Chef-activity reference process variables (“IP-Address” and “Credentials”) that are assigned to a “TargetVM” and a “Credentials” attribute of the activities. These attributes represent predefined activity-specific *input parameters* of the activity implementation. When the activity gets executed by the workflow, the runtime copies the content of the referenced process variables “by value” and takes them as input parameters for invoking the implementation.

To exchange produced data between DPAs and IPAs, both may specify *output parameters* that contain the results of their execution. Each output parameter is represented as a pair of (i) *activity-internal data reference* and (ii) workflow process variable. An activity-internal data reference is a reference to a data container in the language of the activity. For example, an environment variable of a script. When the execution of the statements is finished, the referenced data is copied by the activity implementation to the specified process variables “by value”.

*Content injection* enables using process variables

```

1 <extensionActivity>
2 <REST:POST ResponseVar="VMCreationResponse"
3 URL="https://ec2.amazonaws.com/?Action=RunInstances
4 &ImageId=ami-31814f58
5 &InstanceType=m1.small&..." />
6 </extensionActivity>
7 ...
8 <extensionActivity>
9 <DPA:Chef TargetVM="$bpelvar[IP-Address]" Credentials="$bpelvar[Credentials]">
10 package 'apache_httpd' do
11 http_port $bpelvar[HTTPPort]
12 https_port 8081
13 ensure 'installed'
14 end ...
15 </DPA:Chef>
16 </extensionActivity>

```

Listing 1: Snippet of a BPEL model that employs an HTTP-Request as IPA and a DPA that declares Chef statements.

directly in the declarative or imperative language of a provisioning activity. These serve as placeholders that are replaced by the current content of the referenced variable when the execution of the activity starts. For example, a script may use the variable “IP-Address” to write the IP of the VM into firewall rules to enable accessing the Webserver from the outside.

## 4 REALIZATION USING BPEL

In this section, we prove that the presented approach is practically feasible by applying the integrated modelling concept to the workflow standard BPEL. Therefore, we (i) show how Imperative Provisioning Activities can be realized using existing constructs and extensions of BPEL and how (ii) Declarative Provisioning Activities can be modelled and executed using the so called “BPEL extension activities” (OASIS, 2007). The result is a *Standards-based Integrated Provisioning Modelling Language* that supports the direct orchestration of imperative languages as well as declarative languages.

In general, we realize DPAs by applying the BPEL concept of extension activities that allow to implement custom activity types in BPEL using programming languages such as Java (Kopp et al., 2011). BPEL-workflow runtimes support registering multiple different types of extension activities including their implementations. If the control flow of a workflow reaches an extension activity-element, its implementation is executed by the workflow engine and the whole XML-content of the extension activity-element in the BPEL model is passed to the implementation of the extension activity as input. Thus, the concept en-

ables modelling arbitrary XML-definitions which are parsed and interpreted by the extension activity implementation. To select the right implementation, the element name of the extension activity-element’s first child serves as lookup key for the workflow engine. Hence, we can realize arbitrary types of DPAs by implementing small programs that are executed when the control flow reaches one of these activities.

We show how IPAs und DPAs can be realized using extension activities by conducting an example. A modeller, e.g., developers or operations personnel (Hüttermann, 2012), manually models an Integrated Provisioning Model that consists of Declarative as well as Imperative Provisioning Activities where suitable. The XML shown in Listing 1 is an excerpt of a BPEL model that instantiates a virtual machine on the Cloud-offering Amazon EC2 and installs an Apache Webserver on it. The instantiation of the VM is modelled as activity that sends an HTTP-POST request to the management API of Amazon<sup>2</sup> (lines 1-6). We employ here the BPEL4REST extension activity approach (Haupt et al., 2014), which supports defining output parameters: The Amazon API synchronously returns the instance ID of the virtual machine in the HTTP response. As the provisioning of a virtual machine takes some time, the ID can be used to poll the status of the VM instantiation. Therefore, we store the response in a process variable called “VM-CreationResponse” (line 2). The implementation of the extension activity reads this mapping and writes the content of the HTTP response as value to the VM-CreationResponse variable. This variable can be used by other activities to monitor the current VM status

<sup>2</sup>[http://docs.aws.amazon.com/AWSEC2/latest/APIReference/API\\_RunInstances.html](http://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_RunInstances.html)

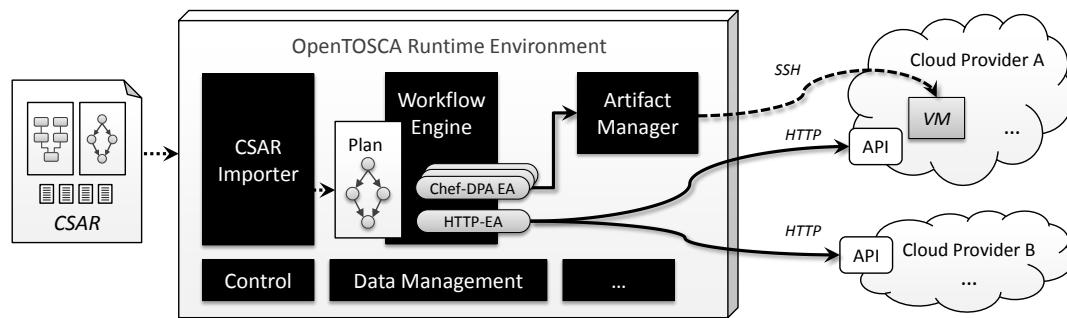


Figure 3: Prototypical implementation of the integrated concept based on the OpenTOSCA runtime environment.

and to retrieve the IP-address of the running virtual machine when the instantiation finished using similar API calls (omitted in Listing 1).

After the VM is provisioned, a Chef-DPA installs the Webserver on it (lines 8-16). This DPA defines the attributes used in our previous example with identical semantics. Similar to the HTTP extension activity, the extension activity implementation of the Chef-DPA reads its XML fragment, extracts the relevant information, and enforces the declared goals by accessing the VM using SSH, installing a Chef agent, and sending the declarative statements to this agent that enforces them. In this example, the input parameter concept is used to specify the target VM on which the Webserver has to be installed and the credentials to access the VM (line 9). The referenced BPEL variables of the workflow model are replaced by the extension activity implementation for execution. In addition, also the content injection concept is realized: In line 11, a BPEL variable is specified as configuration for the HTTP-port of the Webserver. Thus, when executing the DPA, its implementation retrieves the value of the “HTTPPort” workflow variable and replaces the placeholder before enforcing the declared configuration—similarly as for input parameters.

## 5 PROTOTYPE

To prove the technical feasibility of the presented approach, we implemented a prototype based on the OpenTOSCA ecosystem, which consists of the open-source modelling tool *Winery* (Kopp et al., 2013), the imperative runtime environment *OpenTOSCA* (Binz et al., 2013), and the self-service portal *Vinothek* (Breitenbücher et al., 2014). The system is based on the TOSCA standard (OASIS, 2013), which enables developing application packages that are portable across different platforms. TOSCA specifies a metamodel for (i) describing the application’s structure as topology model and (ii) enables using management work-

flows to provision and manage the modelled applications. In addition, TOSCA standardizes a package format called *CSAR* that contains the topology model, all management workflows, and all artifacts that are required to provision and manage the described application, e. g., application files or installation scripts. CSARs can be created using the modelling tool *Winery*. A CSAR is consumed by the OpenTOSCA runtime which deploys the workflows contained therein. Therefore, the runtime employs a workflow engine (WSO2 BPS)<sup>3</sup> to execute BPEL workflows. Using the *Vinothek*, their execution can be triggered.

Figure 3 shows a simplified architecture of the OpenTOSCA runtime environment including our prototypical realization of the integrated modelling concept. The *CSAR Importer* is responsible for consuming CSARs and processing the contained data, e. g., by storing the models in local databases. The *Control* then triggers the local deployment of all management workflows so that they can be executed by the *Vinothek* to provision a new application instance or to manage a running instance. The concept presented in this paper is realized by implementing extension activity-plugins for the workflow engine. The HTTP-extension activity, for example, can be used in BPEL workflows to invoke management APIs of providers to instantiate or manage virtual machines. As described in the previous section, DPAs can then use process variables to access these virtual machines in order to install or configure software etc. To implement these extension activities, e. g., the Chef-DPA, we delegate executing the declaratively described goals to a component called *Artifact Manager*. This plugin-based manager is able to execute various configuration management technologies such as Chef or also imperative scripts, e. g., Bash scripts (Wettinger et al., 2014). Thus, implementing IPAs and DPAs is eased by invoking this manager. Of course, arbitrary technologies can be integrated without using the manager, too. For modelling Integrated Provi-

<sup>3</sup><http://wso2.com/products/>

Table 1: Criteria Evaluation.

Feature	Declarative	Imperative	Integrated Approach
Full control		x	x
Complex deployments	(x)	(x)	x
Hybrid and multi-Cloud applications	(x)	x	x
Seamless integration			x
Component wiring	(x)	x	x
XaaS integration	(x)	x	x
Full automation	x	x	x
Straightforwardness	x		x
Extensibility	(x)	x	x
Flexibility		(x)	(x)

sioning Models, we employ the modelling tool *BPEL Designer*<sup>4</sup>. Since the prototype is based on TOSCA and BPEL, it provides an end-to-end, standards-based Cloud application management platform that enables integrating various technologies seamlessly.

## 6 EVALUATION

In this section, we evaluate the presented approach by comparing it with the plain declarative and imperative management flavors. For the comparison, we reuse the management feature criteria for comparing service-centric and script-centric management technologies (Breitenbücher et al., 2013) and additionally add criteria that are derived from the features of each flavor discussed in Section 2. As a result, the criteria represent requirements that must be fulfilled to fully automatically provision the kind of complex composite Cloud applications described in the introduction (cf. Section 1). An “x” in Table 1 denotes that the corresponding approach fully supports the criterion. An “x” in parentheses denotes partial support.

*Full control* means that provisioning may be customized arbitrarily by the process modeller in each technical detail. As declarative approaches infer the details about the execution by themselves, the general provisioning logic cannot be changed easily. In contrast to this, imperative approaches explicitly model each step to be performed and can be, therefore, customized arbitrarily. Because the integrated approach supports both, it fulfills this criterion completely.

*Complex deployments* denotes that real, non-trivial business applications that employ various heterogeneous components and services can be deployed using a technology of the flavor. Declarative approaches reach their limits at a certain point of required customizability: as the provisioning logic is inferred by a general-purpose provisioning system, only

known declarative statements can be understood and processed (cf. Section 2). Thus, if a very specific, arbitrarily customized application structure or configuration has to be deployed, declarative approaches are often not able to fulfill these rare and very special requirements completely. The integration of low-level execution code such as scripts partially solves this problem. In contrast to this, based on the full control criterion, in general arbitrary complex provisionings can be described using imperative approaches such as scripts or workflows. However, the technical complexity of the resulting processes is often hardly manageable and maintainable as the integration of technologies, as explained in Section 2, leads to a lot of glue and wrapper code, which results in many lines of process implementation code. Thus, plain imperative approaches are not ideal for handling such cases completely and are, therefore, only partially suited. The integration approach presented in this paper solves these issues as the optimal technology can be chosen without polluting the process with wrapper code.

The *hybrid and multi-Cloud applications* criteria evaluate the support for applications that are either hosted on (i) a combination of private and public Cloud services or (ii) Cloud services offered by different providers. Since many declarative approaches such as Amazon CloudFormation employ proprietary, non-standardized domain-specific languages, many of these technologies are not able to provision a distributed application as described above. General purpose technologies such as TOSCA (OASIS, 2013) allow to provision hybrid as well as multi-Cloud applications, for example, by using the TOSCA plan generator (Breitenbücher et al., 2014). However, if multiple providers are involved, typically their proprietary languages have to be used as the declarative general-purpose technologies are not able to support all individual technical features. Based on the criteria *full control* and *complex deployments*, the imperative as well as the proposed approach fulfill this criterion.

*Seamless integration* evaluates the capability to

<sup>4</sup><https://eclipse.org/bpel/>

employ arbitrary management technologies without (i) polluting the model or (ii) leading to abstracted wrapper calls (cf. Section 2). As extensively discussed in the previous sections, neither declarative nor imperative approaches natively support all required integration concepts. In contrast, the presented approach fulfills this criterion due to the introduced concept of Declarative Provisioning Activities.

The *component wiring* criterion means that multiple application components can be wired. Declarative approaches support this partially as unknown components or complex wiring tasks cannot be described in an arbitrary manner. The imperative as well as the integrated approach solve this issue as any task to wire such components can be orchestrated arbitrarily.

*XaaS integration* means the ability to orchestrate various kinds of Cloud services that represent application components. Generic declarative approaches support this only partially as complex configuration tasks are hard to model. Proprietary approaches such as Amazon CloudFormation are bound to a certain provider and, therefore, require glue code to integrate other services. The imperative and the presented approach fully support this requirement following the argumentation of *component wiring*.

The *full automation* criterion is fulfilled by all kinds of approaches, as all of them enable a fully automated provisioning of the described applications.

*Straightforwardness* evaluates, if describing the provisioning of an application can be done in an efficient manner requiring appropriate effort. The declarative approaches are typically easy to learn, as technical complexity is shifted to the provisioning systems and only the desired goals have to be specified. Imperative approaches such as scripts or workflows quickly become huge and complex due to the directly visible low-level details about the (i) control flow and the (ii) data flow. In addition, in many cases, trivial steps have to be modelled explicitly. The presented integration approach fulfills this criterion completely as the optimal technology can be selected for a certain provisioning task. Even a single DPA may be modelled that declares all provisioning goals.

The *extensibility* criterion means the ability to involve other management technologies. Declarative approaches allow this by using glue code at certain points in the inferred logic. Due to the *full control* criterion, imperative approaches are able to include arbitrary implementations at any point in the process. Thus, the integrated approach supports this feature.

The declarative approaches do not support *flexibility* due to the *full control* criterion. However, also using imperative approaches are limited in terms of flexibility: If a complex application leads to a huge

provisioning process, adapting this process is a challenging task. Therefore, imperative as well as the presented approach fulfill this criterion only partially. To tackle these issues, we conduct research on modelling situation-aware processes to increase the flexibility.

To summarize the evaluation, the presented approach profits from all benefits of the two provisioning flavors while solving drawbacks by the strengths of each other. Whereas complex application provisionings can be modelled in a flexible manner preserving the full control over the provisioning, standard tasks can be modelled easily using declarative specifications in a straightforward manner. Even distributed application structures, for example, hybrid and multi-Cloud applications can be provisioned using the integrated approach described in this paper. One of the most important criterion, the seamless integration of provisioning technologies, is solved by the concept of Declarative Provisioning Activities while imperative technologies are typically integrated already in existing languages. Thus, while the resulting process models are implemented in a standards-compliant manner, intuitive provisioning modelling helps developing and maintaining models.

## 6.1 Limitations

In this section, we discuss the limitations of the presented approach. A drawback is the tight coupling of Integrated Provisioning Models to the structure of the application to be provisioned. Imperative orchestrations to provision the components of a certain application structure are sensitive to structural changes: Different combinations of components lead to different models that must be created and maintained separately (Breitenbücher et al., 2013; Eilam et al., 2011; El Maghraoui et al., 2006). Thus, as the concept of Integrated Provisioning Models is based on imperatively orchestrating the two kinds of provisioning activities, this applies also for the approach presented in this paper. As a result, Integrated Provisioning Models for new applications often have to be created from scratch while maintaining existing processes results in complex, time-consuming adaptations (Breitenbücher et al., 2014). To tackle this tight coupling of imperative orchestrations and concrete structures, we did research on generic process fragments for application management, which can be reused for individual applications (Breitenbücher et al., 2013; Breitenbücher et al., 2013). We plan to combine this approach with the presented concept. In addition, currently only the provisioning of applications is supported by our concept. Therefore, we plan to extend the concept to support also management.

## 7 CONCLUSION

In this paper, we presented a process modelling approach that enables the seamless integration of imperative and declarative provisioning models by introducing the concepts of (i) Declarative Provisioning Activities and (ii) Integrated Provisioning Models. The approach enables intuitive provisioning modelling without handling technical integration issues of regarding different technologies and domain-specific languages that pollute the control as well as the data flow of the resulting workflow models. To prove the technical feasibility of the approach, we applied the presented concept to the workflow language BPEL and extended the standards-based application management system OpenTOSCA. In addition, we evaluated its features by several criteria. The evaluation shows that the presented approach enables to benefit from strengths of both flavors. In future work, we plan to apply the concept also for application management.

## ACKNOWLEDGEMENTS

This work was partially funded by the projects SitOPT (Research Grant 610872, DFG) and NEMAR (Research Grant 03ET40188, BMWi).

## REFERENCES

- Binz, T., Breitenbücher, U., Kopp, O., and Leymann, F. (2014). Migration of enterprise applications to the cloud. *it - Information Technology, Special Issue: Architecture of Web Application*, 56(3):106–111.
- Binz, T. et al. (2013). OpenTOSCA – A Runtime for TOSCA-based Cloud Applications. In *ICSOC 2013*, pages 692–695. Springer.
- Breitenbücher, U., Binz, T., Kopp, O., and Leymann, F. (2013). Pattern-based runtime management of composite cloud applications. In *CLOSER 2013*, pages 475–482. SciTePress.
- Breitenbücher, U., Binz, T., Kopp, O., and Leymann, F. (2014). Vinothek - A Self-Service Portal for TOSCA. In *ZEUS 2014*, volume 1140 of *CEUR Workshop Proceedings*, pages 69–72. CEUR-WS.org.
- Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., and Wettinger, J. (2013). Integrated cloud application provisioning: Interconnecting service-centric and script-centric management technologies. In *CoopIS 2013*, pages 130–148. Springer.
- Breitenbücher, U. et al. (2014). Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA. In *IC2E 2014*, pages 87–96. IEEE.
- Eilam, T., Elder, M., Konstantinou, A., and Snible, E. (2011). Pattern-based composite application deployment. In *IM 2011*, pages 217–224. IEEE.
- El Maghraoui, K., Meghranjani, A., Eilam, T., Kalantar, M., and Konstantinou, A. V. (2006). Model driven provisioning: bridging the gap between declarative object models and procedural provisioning tools. In *Middleware 2006*, pages 404–423. Springer.
- Günther, S., Haupt, M., and Splieth, M. (2010). Utilizing Internal Domain-Specific Languages for Deployment and Maintenance of IT Infrastructures. Technical report, Very Large Business Applications Lab Magdeburg, Otto von Guericke University Magdeburg.
- Haupt, F., Fischer, M., Karastoyanova, D., Leymann, F., and Vukojevic-Haupt, K. (2014). Service Composition for REST. In *EDOC 2014*. IEEE.
- Herry, H., Anderson, P., and Wickler, G. (2011). Automated planning for configuration changes. In *LISA 2011*. USENIX.
- Hüttermann, M. (2012). *DevOps for Developers*. Apress.
- Keller, A., Hellerstein, J. L., Wolf, J. L., Wu, K. L., and Krishnan, V. (2004). The champs system: change management with planning and scheduling. *Network Operations and Management Symposium, 2004*, pages 395–408.
- Kopp, O., Binz, T., Breitenbücher, U., and Leymann, F. (2012). BPMN4TOSCA: A Domain-Specific Language to Model Management Plans for Composite Applications. In *Business Process Model and Notation*, pages 38–52. Springer.
- Kopp, O., Binz, T., Breitenbücher, U., and Leymann, F. (2013). Winery – A Modeling Tool for TOSCA-based Cloud Applications. In *ICSOC 2013*, pages 700–704. Springer.
- Kopp, O. et al. (2011). A Classification of BPEL Extensions. *Journal of Systems Integration*, 2(4):2–28.
- Levanti, K. and Ranganathan, A. (2009). Planning-based configuration and management of distributed systems. In *IM 2009*, pages 65–72.
- Leymann, F. (2009). Cloud Computing: The Next Revolution in IT. In *Proc. 52th Photogrammetric Week*, pages 3–12.
- Leymann, F. and Roller, D. (2000). *Production workflow: concepts and techniques*. Prentice Hall PTR.
- Mietzner, R. (2010). *A method and implementation to define and provision variable composite applications, and its usage in cloud computing*. Dissertation, University of Stuttgart, Germany.
- Nelson-Smith, S. (2013). *Test-Driven Infrastructure with Chef*. O'Reilly Media, Inc.
- OASIS (2007). *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*. OASIS.
- OASIS (2013). *Topology and Orchestration Specification for Cloud Applications Version 1.0*.
- OMG (2011). *Business Process Model and Notation (BPMN)*, Version 2.0.
- Opscode, Inc. (2015). Chef official site: <http://www.opscode.com/chef>.
- Petcu, D. (2014). Consuming resources and services from multiple clouds. *Journal of Grid Computing*, 12(2):321–345.
- Puppet Labs, Inc. (2015). Puppet official site: <http://puppetlabs.com/puppet/what-is-puppet>.
- Wettinger, J. et al. (2014). Unified Invocation of Scripts and Services for Provisioning, Deployment, and Management of Cloud Applications Based on TOSCA. In *CLOSER 2014*, pages 559–568. SciTePress.

## **SHORT PAPERS**





# A Hedonic Price Index for Cloud Computing Services

Persefoni Mitropoulou, Evangelia Filiopoulou, Stavroula Tsaroucha,  
Christos Michalakelis and Mara Nikolaidou

*Department of Informatics and Telematics, Harokopio University of Athens, 9 Omirou str, Tavros, Athens, Greece  
{persam, evangelf, michalak, mara}@hua.gr, stsaroucha@sch.gr*

**Keywords:** Cloud Computing, Infrastructure-as-a-Service, Pricing Models, Hedonic Price Indices.

**Abstract:** Cloud computing is an innovative business model, being developed at a fast pace during the last years, offering many operational and economic benefits to both the demand and the supply side of the ICT market. Infrastructure as a Service (IaaS), which includes control of fundamental computing resources, is expected to be the fastest growing model of public cloud computing. Due to the existence of several IaaS cloud providers, there is increased competition among cloud companies, which develop different pricing models in order to meet the market demand. As a consequence, prices for cloud services are a result of a multidimensional function, shaped by the service's characteristics. The development of a suitable pricing method, based on an appropriate price index able to capture the market dynamics, is an obvious necessity. The aim of this paper is the construction of such a price index, for the IaaS model, using data from a wide range of cloud providers and a large number of price bundles. The hedonic pricing method is used to decompose cloud computing services into their constituent characteristics, obtaining estimates of the contributory value of each resource. According to the results, RAM size, CPU power and subscription turned out to be the most influential factors that shape IaaS pricing.

## 1 INTRODUCTION

During the recent years, cloud computing has gained enormous popularity across the business world as there is an increased demand for a new business model that can help companies respond faster and cheaper to their constituents' needs, not only in Europe but also worldwide. Its systems and services are being improved and developed at a fast pace, offering operational benefits to both the providers and the consumers of the technology, contributing substantially at the same time to the creation of a competitive environment in the global market (Etro 2009).

Therefore, cloud computing is considered to be a really powerful technological tool and an innovative business model, composed of three service models: Infrastructure as a Service (IaaS), which includes control of fundamental computing resources, such as memory, computing power and storage capacity; Platform as a Service (PaaS) that provides control over the deployed applications and possibly configuration settings for developer platforms and Software as a Service (SaaS), which includes the use of software services accessed through a web browser

or a program interface (Mell and Grance 2011). In addition to these, cloud computing has also four deployment models: Private cloud, provisioned for exclusive use by a single organization; Community cloud, used exclusively by a specific community of consumers from organizations that have shared concerns; Public cloud, open for use by the general public and Hybrid cloud, which is a composition of two or more distinct cloud infrastructures (Mell and Grance 2011).

Concerning the above models, public cloud computing receives more attention and the IaaS model gains increased adoption across the business world (Anderson et al. 2013). More specifically, IaaS, which is a foundational cloud delivery service and the most straightforward of the cloud models, provides flexibility and can be a very good solution for companies needing computing resources in the form of virtualized operating systems, workload management software, hardware, networking, and storage services (Hurwitz et al. 2012). Computational power and operating systems are delivered to the customers in an "on-demand" approach. An enterprise that migrates its IT system to IaaS may hire the required resources as needed, instead of buying

them (Mell and Grance 2011). According to Gartner's latest report about the public cloud (Anderson et al. 2013), it is expected that IaaS will be the fastest growing area of public cloud computing achieving a compound annual growth rate (CAGR) of 41.3% through 2016, as Figure 1 illustrates.

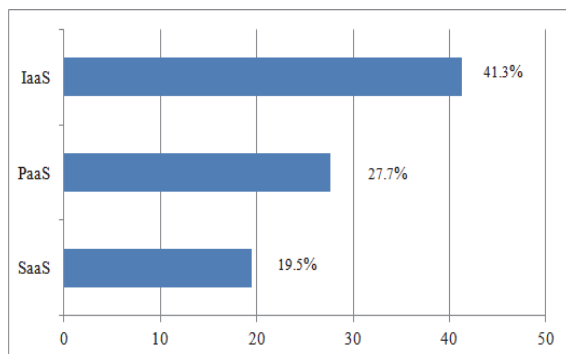


Figure 1: Public Cloud Services - five year (2011-2016) CAGRs (%), by model. Source: Gartner (February 2013).

Currently, there are several public cloud providers in the cloud computing market that provide similar services to customers. A client's choice of which cloud company would host his infrastructure-related services in the long-term depends jointly on the price it has, the Quality-of-Service (QoS) guarantees it offers to its customers and the satisfaction of the advertised guarantees (Vinu Prasad et al., Siham et al. 2012). Cloud computing works, in general, on a "pay-as-you-go" basis, giving the option to the user to pay for what they use; meaning that the customer is charged for each computing resource (e.g. RAM, CPU, storage) separately, usually per unit-hour (Martens et al. 2012). On the other hand, it is true that the battle for a dominant market share grows the competition among cloud companies and leads to the development of new pricing schemes in order to meet the market demand. As a result, several packages of different resources are offered in attractive tariffs and they are continually fitted to the changing preferences and increasing needs of customers. Furthermore, there is an option somewhere in the middle, as there are some cloud providers who offer predefined sets of some resources, usually memory and CPU are bundled, whereas users can select simultaneously on their own some other computing characteristics, such as storage size (Martens et al. 2012, Andra 2013).

Finding the right combination of the available resources is critical for a business to achieve the best value when creating its own cloud bundle of services. As any bundle consists of various characteristics, valued differently by each consumer, there are a

number of questions that arise in the cloud computing context, such as:

- How customers' choices and preferences for IaaS affect the pricing of the corresponding resources.
- Which characteristics are truly independent from one another, while at the same time being the most important for shaping the pricing of IaaS.

Towards this direction, multiple cloud providers have already developed cost estimator tools, which are used to help customers evaluate IaaS services, deciding the most suitable for their needs. In addition, another similar approach is considered to be a broker model that acts as an intermediary between consumers and providers. Both of these methods are mainly based on asking users some questions about the amount of computational power, memory, storage requirements, data transfer and they subsequently offer a monthly estimate for price, for the selected bundle (Hurwitz et al. 2012) or the most cost-efficient tariff option among many different providers in the case of a broker (Jörg et al. 2014). However, none of these tools is capable of identifying the cheapest cloud hosting provider due to the fact that this is a choice that depends exclusively on the clients' computing needs and, furthermore, price is considered to be a multidimensional function, where many factors should be taken into account (Siham et al. 2012).

Into that context, this paper tries to describe the current mechanism of pricing and reveal the dynamics which drive the pricing of cloud computing services. For this, it provides an empirical analysis of IaaS pricing, by constructing a price index based on a hedonic pricing method.

The remainder of the paper is structured as follows: Section 2 presents a brief literature review of the previous work regarding the pricing methods of cloud computing, while in Section 3 there is a theoretical approach of hedonic functions analysis and price indices. The empirical study and the evaluation of hedonic regression model together with discussion are described in Section 4, and finally, Section 5 concludes, providing directions for future research.

## 2 PRICING MODELS OF CLOUD COMPUTING

In a cloud computing environment, an Infrastructure-as-a-service demand is considered as an access to the

system resources, such as CPU, memory and disk. Resource allocation is a really challenging task because of the different pricing models the providers use. In general, cloud hosting providers must offer a good pricing model that does not bring any loss to neither the provider nor the consumer and maximizes the social surplus of the corresponding market. It is not usually easy to reach a point where both sides agree with the price set. A user is always willing to pay a lower price for the resources requested, whereas a provider should not want to go beyond the lowest price that gives him no profit (Andra 2013).

Currently, there are many cloud vendors that follow a fixed pricing strategy, meaning that customers pay for what they use but not for what the cloud services value (Vinu Prasad et al.). One of the most common pricing schemes of this category is the “pay-as-you-go” model, which charges the users for just the services they need, paying only for the required computing instances and just for the time they use them. In the case they need more resources, they simply request them from the provider (Grossman 2009). Another fixed pricing model is based on subscription, setting a standard price for the required resources according to a longer period of subscription (Al-Roomi et al. 2013). However, these static approaches of pricing have some limitations, due to the fact that they reserve computational resources in advance and it is often hard to satisfy both the cloud vendors’ and cloud users’ requirements.

Hence, dynamic pricing is usually the key solution to the above problems. It is a method in which the price for each bundle of resources is based on a number of factors, such as availability, time, the service features and according to the forces of the demand and supply of a real-time market (Andra 2013, Al-Roomi et al. 2013). Mihailescu and Teo (2010) proposed such an auction-based pricing strategy for federated clouds, in which resources are shared among many cloud service providers. Rohitratana and Altmann (2012) used an agent-based simulation of four different pricing models that indicated that the Demand-Driven (DD) pricing scheme was the best approach in ideal cases. Li et al (2011) introduced a real-time pricing algorithm for cloud computing resources. It analyzed some history utilization data and it found the final price that was mostly beneficial for the provider because it reduced its costs, allowing at the same time resources to be used more effectively. Moreover, there are some pricing methods that are mostly driven by competitors’ prices (Rohitratana and Altmann 2010) and some others based on the amount of money users

are ready to pay (Ruiz-Agundez et al. 2011). All of the above pricing methods are fair enough for the customers’ side.

Both of the above categories of pricing models, especially the dynamic one, take into consideration some of the service’s most important characteristics. The construction of price indices is generally used for this purpose, seeking to estimate the extent to which each characteristic affects the total price of a service bundle. Among the most common and widely used approaches is the hedonic pricing method (Triplett 2004). It was primarily developed seeking to capture the effect of environmental and housing attributes in the context of the housing market (Goodman 1978) and to adjust for quality change for automobiles (Griliches 1961). In (Chanel et al. 1996) a price index for paintings, based on regressions using the full set of sales, was constructed and the idea that goods are valued for their utility-bearing characteristics can be found in (Rosen 1974). As far as information and communication technologies are concerned, hedonic price indices have been widely used for personal computers (Pakes 2002, Berndt et al. 1995) taking quality changes into account and for microcomputers and printers using evidence from France (Moreau 1996). A hedonic pricing approach has been also proposed in (Jörg et al. 2014) to estimate price evolution of telecommunication services based on data across Europe and in (Siham et al. 2012) the hedonic pricing method was applied to make cloud pricing plans more transparent.

### 3 HEDONIC PRICE INDICES

Hedonic methods refer to regression models in which a product’s (or a service’s) prices are related to product characteristics and the observed price of a product (service) is considered as a function of these characteristics. The main assumption hedonic methods are based on, is that a service is a bundle of characteristics and that consumers just buy bundles of product characteristics instead of the product itself. A hedonic method decomposes the item being researched into its constituent characteristics, and obtains estimates of the contributory value of each characteristic, provided that the composite good can be reduced to its constituent parts and that the market values those constituent parts.

According to the definition of (Triplett 2004): “A *hedonic price index* is any *price index* that makes use of a *hedonic function*. A *hedonic function* is a relation between the prices of different varieties of a product,

such as the various models of personal computers, and the quantities of characteristics in them”.

These methods can be used to construct a quality-adjusted price index of a service. An informative overview of the hedonic methods and how they are constructed can be found in (Berndt 1991, Triplett 2004). Moreover, as shown in (Rosen 1974) consumer chooses from a large number of product varieties without having the ability to influence prices. As a consequence, consumers maximize utility and producers maximize profits. In hedonic studies it is possible to adjust the price of a service for its quality not quantity. All of them are based on some estimated coefficients that are inflicted on the characteristics of the products in two periods;  $m$  and  $m + 1$ . It is possible to estimate the coefficients separately, for each evaluate period of time, or consider the observations of two or all periods together and estimate a common set of coefficients, seeking to reveal the general trend.

The advantage of this method is that the necessary calculations are easy to implement. Hedonic methods are also very fast to apply but the disadvantage is that index price can change even if no new products exist, or if all prices remain the same. Among the strengths of a hedonic pricing method are that it can be used to estimate values based on actual choices and its versatility, since it can be adapted to consider several possible interactions between market goods and environmental quality.

The hedonic price indices are commonly used as approximations to find how much money a consumer would need in period  $m+1$  relatively to the amount of money required in period  $m$ , keeping the same level of utility. The solution to this problem is to determine the consumer's profile and his reaction to a varied and fast-changing supply of products. The main problem towards this direction is that each consumer has potentially different needs and requirements. No matter what profile is decided, it will be a hypothesis and an assumption that will correspond to a specific model. In addition to this, a consumer's desire is not stable, something quite reasonable since there is a great offer as technology becomes cheaper and more attractive.

A hedonic function  $f(X)$ , which relates a number of the product's characteristics with the corresponding price as:

$$P_i = f(X_i) \quad (1)$$

where  $P_i$  is the price of a variety (or a model)  $i$  of the considered product and  $X_i$  is a vector of characteristics associated with the specific variety.

The hedonic function is then used, for a number of different characteristics among the varieties of the product and the price index is calculated. As soon as the characteristics to be considered are determined then, for  $N$  varieties of the product (or service) the following equations must be evaluated:

$$P_i = b_0 + b_1 \cdot X_{1i} + b_2 \cdot X_{2i} + e_i, \quad (2)$$

$$i = 1, \dots, N$$

where  $b_i$  are the regression coefficients that have to be estimated and  $e_i$  is the regression residual of the assumed functional form. The regression coefficients value the characteristics and they are often called implicit prices, because they indicate the prices charged and paid for an increment of one unit of the corresponding characteristic. Implicit prices are much like other prices, they are influenced by demand and by supply. In some cases the natural logarithm ( $\ln$ ) of the price is considered, instead of the actual value. Furthermore, the functional form of the index can be nonlinear.

In the case that the prices span between two (or more) periods of time  $m$  and  $m + 1$ , the equations to be evaluated are

$$P_{im} = b_0 + b_1 \cdot X_{1i} + b_2 \cdot X_{2i} + e_{im},$$

$$i = 1, \dots, N$$

$$P_{im+1} = b_0 + b_1 \cdot X_{1i} + b_2 \cdot X_{2i} + e_{im+1}, \quad (3)$$

$$i = 1, \dots, N$$

In the context of this work, the vector of characteristics  $X_i$ , corresponds to the configuration of the IaaS cloud services that affects the price, including characteristics such as RAM size, number of CPUs, memory size, bandwidth etc. The description of these parameters is given in the next section.

The importance of a price index is that it can be used to determine suggested prices for combinations of the characteristics that were not included, or they were not available, when the index was constructed.

## 4 PRICE INDEX CONSTRUCTION

This section describes the empirical study design, contains the evaluation of the hedonic price index methodology for cloud computing services, the construction of a corresponding index and discussion of the results.

The price index is constructed for the IaaS cloud

computing, the most straightforward cloud service. Data collection was based on Clouddorado (<http://www.clouddorado.com>), a price comparison service of cloud computing providers. Clouddorado is also a price calculator for multiple cloud hosting providers, since the comparison is performed by calculating price for individually set server needs. It currently focuses on providing pricing bundles for IaaS providers. The number of the collected price bundles is 2354, by 25 providers, shown in Table 1.

Table 1: Cloud IaaS providers.

Amazon
atlantic.net
Bitrefinery
CloudSigma
Dimensiondata
eApps
ecloud24
Elastichosts
Exoscale
GIGENET
GOGGRID
HYVE
JoyentCloud
Lunacloud
M5
Ninefold
OPENHOSTING
Rackspace
SERVERMULE
Storm
StratoGen
Terremark
VPSNET
Windows Azure
ZettaGrid

Google is not included among the providers since it does not offer price bundles but it rather charges for each CPU and each GB of storage and memory capacity. The price bundles are specified by the resources presented in Table 2, together with the considered values. These characteristics participate as variables in the hedonic pricing model.

Data correspond to the IaaS services offered by cloud providers who use different pricing models. The study started by selecting specific computing requirements (e.g. 2xCPU, 1GB RAM, 50GB Storage, 5GB Transfer-Out, Linux) but due to the fact

Table 2: IaaS characteristics.

Characteristic	Description	Values
CPU	CPU power	2x, 4x, 6x / 3x, 5x, 7x
RAM	RAM size in Gigabytes (GB)	1, 4, 16, 32
Storage	Measured in GB	100, 1000
Transfer_Out	Number of bytes sent by server to Internet per month. (GB)	5, 10000
OS	Operating System of the server	Linux, Windows
Subscription	Indicates if there should be a subscription	No, Yes (corresponds to 1 year subscription)

that many of the providers (e.g. Amazon, Rackspace, GoGrid) use price bundling, the best package of resources, which was most close to each customer's needs, was chosen every time. Prices range between \$31 and \$3,318 per month and there are observable differentiations depending on the existence of a subscription, while the duration of the subscription does not affect the price substantially. The operating system parameter (OS) and the subscription characteristics participate as dummy variables. The values for the OS are 0 for Windows and 1 for Linux and, regarding the subscription, corresponding values are 0 for no subscription and 1 for a subscription.

Not surprisingly, the most popular geographical continent for providers is North America, with 19 out of the 24 to have datacenters located there, followed by Europe, with 13 providers. Australia and Asia follow with 8 and 6 providers, respectively, and Africa comes last with just 1 provider.

Among the limitations of the collected dataset is that there are a few more characteristics participating in the construction of the price bundling, which were not considered into this study. These characteristics are the Transfer In (the number of bytes received by server from the internet per month), the Time On (proportion of the day the server is available) and the option that the CPUs, the RAM and the storage can be distributed among more than one physical server. The value of the Transfer In characteristic does not contribute at a substantial level to the shaping of the pricing bundles, because many cloud providers such as Amazon and ecloud24 charge customers only for the outgoing traffic and the others include it as a small amount in the total price of services. Therefore, with no loss of generality, the Transfer In attribute was considered to be at 1GB per month. As far as Time On is concerned it was set at a level of 100%

availability per day. The default offered value of non-distributed resources was also considered.

Moreover, some non-functional factors, such as availability and reliability of resources or the lack of fulfilling the agreements between consumers and providers, were not considered in the price model, as variables. Inclusion of these characteristics is an interesting extension of the model, since it may reveal their potential to affect the price of a cloud offering. This needs to be empirically proven through the execution of many different scenarios.

The results of the hedonic pricing method are summarized in Table 3.

Table 3: Results of hedonic method.

Coefficients	Value
Constant	130,499*** (35,42)
CPU	14,532** (6.41)
Storage	0,249*** (0.02)
RAM	20,434*** (0.86)
OS	-16,91 (2.29)
Transfer OUT	0,076*** (0.002)
Subscription	-85,82 (20.28)

\*\*\* p<.01, \*\*p<.05, \*p<.1, n.s. not significant

The calculated  $R^2$  value is 57.91%, indicating that although a great portion of the uncertainty is described by the model, the linear form of the model may not be the most appropriate to describe the pricing index and alternative formulations could also be considered.

As observed, all parameters are significant and they contribute to the shaping of the price. Subscription is the parameter contributing more to the price index, followed by the RAM size and the CPU. The high value of the constant, which represents a fixed monthly fee, supports the finding that the subscription is a crucial parameter. Storage does not seem to affect the price very much. The choice of the operating system affects pricing at a high level, since Linux reduces the price of the bundles by a factor of 16.91.

## 5 CONCLUSIONS

The hedonic pricing method was used in this work, in order to develop a price index for the Infrastructure as

a Service cloud computing services. The evaluation of the method was based on the linear hedonic model and the data were collected for a number of 22 providers, corresponding to more than 2300 price bundles.

The results indicate that, apart from the constant parameter which indicates the importance of the subscription, a finding that is also supported by the high value of the subscription parameter, the RAM size and the CPU are also of substantial importance and significance. On the contrary, the storage and the transfer out parameters seem to affect the pricing procedure less.

As in most cases, there are some certain limitations in this work, which in turn constitute its further extension and indicate directions for future research. Among them is the use of nonlinear functional forms in the hedonic formulation, seeking to improve the accuracy of the pricing index. The value of  $R^2$  achieved indicate that it would be worth testing. Apart from the general price index considering price bundles across all providers, the construction of an index for each provider would be of particular interest, mainly for comparison reasons.

The construction of price indices for the other cloud computing models, namely the software as a service (SaaS) and the Platform as a Service (PaaS), where literature has little to present, would be another important, as well as interesting research direction.

In any case, the existence of a price index for the cloud services can provide very useful information, not only regarding the pricing schemes but also regarding the market of cloud itself and could suggest optimal pricing approaches of the cloud services.

## REFERENCES

- Al-Roomi, M., Al-Ebrahim, S., Buqrais, S. and Ahmad, I. (2013) 'Cloud Computing Pricing Models: A Survey', *International Journal of Grid & Distributed Computing*, 6(5).
- Anderson, E., Eschinger, C., Wurster, L., de Silva, F., Contu, R., Liu, V., Biscotti, F., Petri, G., Zhang, J. and Yeates, M. (2013) 'Forecast overview: Public cloud services, worldwide, 2011-2016, 4Q12 Update', *Gartner Inc., February*.
- Andra, R. S. (2013) 'Investigating Pricing and Negotiation Models for Cloud Computing'.
- Berndt, E. R. (1991) *The practice of econometrics: classic and contemporary*, Addison-Wesley Publishing.
- Berndt, E. R., Griliches, Z. and Rappaport, N. J. (1995) 'Econometric estimates of price indexes for personal computers in the 1990's', *Journal of Econometrics*, 68(1), 243-268.
- Chanel, O., Gérard-Varet, L.-A. and Ginsburgh, V. (1996)

- 'The relevance of hedonic price indices', *Journal of Cultural Economics*, 20(1), 1-24.
- Etro, F. (2009) 'The economic impact of cloud computing on business creation, employment and output in Europe', *Review of Business and Economics*, 54(2), 179-208.
- Goodman, A. C. (1978) 'Hedonic prices, price indices and housing markets', *Journal of Urban Economics*, 5(4), 471-484.
- Griliches, Z. (1961) 'Hedonic price indexes for automobiles: An econometric of quality change' in *The Price Statistics of the Federal Government*, NBER, 173-196.
- Grossman, R. L. (2009) 'The case for cloud computing', *IT professional*, 11(2), 23-27.
- Hurwitz, J., Kaufman, M. and Halper, D. F. (2012) 'Cloud Services for Dummies', *IBM Limited Edition, John Willy and Sons*.
- Jörg, G., Hiemer, J. and Hinz, O. (2014) 'A Cloud Computing Broker Model for IaaS resources', in *Proceedings of the European Conference on Information Systems (ECIS) 2014*, Tel Aviv, Israel, June 9-11,
- Li, H., Liu, J. and Tang, G. (2011) *A pricing algorithm for cloud computing resources*, translated by IEEE, 69-73.
- Martens, B., Walterbusch, M. and Teuteberg, F. (2012) *Costing of cloud computing services: A total cost of ownership approach*, translated by IEEE, 1563-1572.
- Mell, P. and Grance, T. (2011) 'The NIST definition of cloud computing'.
- Mihailescu, M. and Teo, Y. M. (2010) *Dynamic resource pricing on federated clouds*, translated by IEEE, 513-517.
- Moreau, A. (1996) 'Methodology of the price index for microcomputers and printers in France', *Industry Productivity: International Comparison and Measurement Issues*, 99-118.
- Pakes, A. (2002) *A Reconsideration of Hedonic Price Indices with an Application to PC's*, National Bureau of Economic Research.
- Rohitratana, J. and Altmann, J. (2010) 'Agent-Based Simulations of the Software Market under Different Pricing Schemes for Software-as-a-Service and Perpetual Software', *Economics of Grids, Clouds, Systems, and Services, ser. Lecture Notes in Computer Science, Altmann et al., Eds. Springer Berlin/Heidelberg*.
- Rohitratana, J. and Altmann, J. (2012) 'Impact of pricing schemes on a market for Software-as-a-Service and perpetual software', *Future Generation Computer Systems*, 28(8), 1328-1339.
- Rosen, S. (1974) 'Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition', *Journal of Political Economy*, 92, 34-55.
- Ruiz-Agundez, I., Penya, Y. K. and Bringas, P. G. (2011) *A flexible accounting model for cloud computing*, translated by IEEE, 277-284.
- Siham, E. K., Schlereth, C. and Skiera, B. (2012) 'Price comparison for infrastructure-as-a-service'.
- Triplett, J. E. (2004) *Handbook on hedonic indexes and quality adjustments in price indexes, science, technology and industry working papers*, OECD publishing.
- Vinu Prasad, G., Rao, S. and Prasad, A. S. 'A Combinatorial Auction Mechanism for Multiple Resource Procurement in Cloud Computing'.



# New Approach to Partitioning Confidential Resources in Hybrid Clouds

Kaouther Samet<sup>1</sup>, Samir Moalla<sup>1</sup> and Mahdi Khemakhem<sup>2</sup>

<sup>1</sup>*Department of Computer Science, Faculty of Sciences, University Tunis El Manar, Tunis, Tunisia*

<sup>2</sup>*Department of Telecommunications, National School of Electronics and Telecommunications,  
University of Sfax, Sfax, Tunisia*

*samet.kaouther@gmail.com, samir.moalla@fst.rnu.tn, mahdi.khemakhem@isecs.rnu.tn*

**Keywords:** Partitioning Resources, Hybrid Clouds, Confidentiality.

**Abstract:** Today, companies use more cloud environments such as hybrid clouds. Indeed, hybrid clouds give the opportunity to better manage resources mostly when companies have no space to store more resources in their private clouds. The best solution here is to allocate the required space in public cloud at a low cost. But how can resources be partitioned in hybrid clouds while assuring confidentiality of resources moved to public cloud. Many works have been done in this context. They suppose that confidentiality is assured by using encryption methods. But with this solution the cloud provider can access the resources stored on the cloud, which weakens the confidentiality of these. This work proposes an approach to the Confidential Resources Partitioning Problem in Hybrid Clouds (CRPHC) which aims at ensuring the confidentiality of resources by grouping as much as possible the most confidential resources in private cloud and resources with low degrees of confidentiality in public cloud while minimizing the size of resources to host in public cloud and consequently reducing the storage cost. This solution allows the possibility of using non-performing encryption methods which have a reduced treatment cost compared to efficient methods. Experimentally, our solution will be evaluated and compared to optimal solution given by CPLEX.

## 1 INTRODUCTION

Cloud computing has become a major concept referring to the use of memory, computing capabilities computers and servers around the world, all of them linked by a network such as the Internet. Today, companies use more cloud environments for deployment and execution of their applications. Usually, the most used type of clouds in the cloud environment is the hybrid cloud. The infrastructure of this type of cloud is composed of two or several public and private clouds. It is obvious for a company that applications must be deployed in the private cloud as resources can be provided by their cloud.

However, when the physical limit of the private cloud is reached, the company may need to use other resources (data, services or applications) from a public cloud. This occurs when applications and platforms of companies need to be enlarged and request additional resources that the private cloud is not able to provide. In this case, obtaining new resources from public cloud can solve this problem. Consequently, resources will be partitioned between private and public clouds. Among the obstacles, mentioned by authors of (Stoica and Zaharia, 2009), in cloud environ-

ment is confidentiality and the study of secure data in this environment is fairly new and has become increasingly important (Nepal and Calvo, 2014). Indeed, they consider that it is the most important obstacle in this environment. So, how can confidentiality of resources be ensured in the hybrid cloud? To ensure confidentiality in the clouds, encryption methods have been used. But to have better results, it is necessary to use performing encryption methods which are very expensive in terms of execution time and complexity (Chokhani, 2013). However these works fail to raise the problem that the public cloud providers theoretically have access to the received resources.

In this context, we propose an approach to solve the Confidential Resources Partitioning Problem in Hybrid Cloud (CRPHC) which aims at ensuring the confidentiality of resources by grouping as much as possible the most confidential resources in private cloud and resources with low degrees of confidentiality in public cloud while minimizing the size of resources to host in public cloud and consequently reducing the cost of storage. This solution allows to use a non-performing encryption methods which have a reduced treatment cost compared to efficient methods. Experimentally, our solution will be evaluated

and compared to optimal solution given by the commercial software IBM-ILOG-CPLEX 12.5 applied to an integer linear programming formulation of the CRPHC.

The rest of the paper is organized as follows: in section 2, we present an overview of works studying the partitioning problem. In section 3, we list in details the integer linear programming formulation of the CRPHC, while in section 4 we clarify our approach to partitioning confidential resources in hybrid clouds. In section 5, we evaluate and compare our solution to optimal solution given by CPLEX. Finally, we end up giving our conclusion and future works in section 6.

## 2 STATE OF ARTS

The problem of partitioning resources in cloud environments has been seen from different viewpoints, while considering different types of criteria such as confidentiality, access frequency of query execution, communication, etc.

In the following, we present some studies for confidentiality management in clouds.

### 2.1 Confidentiality Management Assured by the Public Cloud Provider

We present below approaches Schism and Birch (Zhang; and Madden, 2010) and (Ramakrishnan and Livny, 1996) treating the problem of partitioning data in databases. They try to produce the best quality clustering with the available memory and time constraints.

Authors of (Tata and Moalla, 2012) propose a new algorithm that approximates the optimal placement of services based on communication and hosting costs induced by the shifting of components towards the public cloud. This research is interested in deciding which services will be deployed to the public clouds based on communications between services within the public cloud, and communications between services of the private cloud and services of the public cloud.

In (Wang and C.Jiang, 2012) and (Wang and Guo, 2013), authors propose a model for the multi-objective data placement and use a particle swarm optimization algorithm to optimize the time and cost in cloud computing.

In works already mentioned, the authors are not interested in the confidentiality of resources moved to

the public clouds. In fact, they suppose that confidentiality is guaranteed by the public cloud provider using encryption methods applied on all the resources moved to the public cloud regardless of their degree of confidentiality (Mehrotra and Thuraisingham, 2012) and (Kantarcioglu and Thuraisingham, 2011). But, in this case, the cloud provider can consult and access to confidential resources in public cloud.

### 2.2 Confidentiality Management in Hybrid Clouds

In (Pilli and Joshi, 2013), authors present a solution approach to the data partitioning problem. They create different partitions and estimate the execution cost of the query workload for each of these partitions and check whether any monetary and confidentiality risk constraints were violated. Authors assume that all predicates have the same level of confidentiality.

Authors of (Mehrotra and Thuraisingham, 2012), (Kantarcioglu and Thuraisingham, 2011), (Marwaha and Bedi, 2013) and (Lamba and Kumar, 2014) propose approaches to ensure confidentiality in clouds based on encryption. Indeed, they suppose that resources confidentiality is assured by encryption methods. But in (Nepal and Calvo, 2014) authors consider that this solution is computationally inefficient and locates a large workload on the data owner when considering factors such as updating encryption keys. Likewise, according to (Chokhani, 2013) encryption methods have additional complexity in cloud environments which makes this operation very expensive and complex.

To solve this problem, we propose an approach to the Confidential Resources Partitioning Problem in Hybrid Clouds (CRPHC) which aims at ensuring the confidentiality of resources by keeping the most confidential resources in private cloud and moving resources with lower degrees of confidentiality into public cloud; while minimizing the size of resources to host in the public cloud.

## 3 INTEGER LINEAR PROGRAMMING FORMULATION FOR THE CRPHC

In this section we present an integer linear programming formulation for the Confidential Resources Partitioning Problem in Hybrid Cloud (CRPHC). Our aim is to:

- Ensure confidentiality by storing resources with low confidentiality in public cloud,
- Minimize resources storage cost in the public cloud while respecting a minimal size of resources to host in public cloud.

### 3.1 Problem Statement

Generally, the CRPHC can be defined on an undirected graph  $G(X, A)$  where  $X = \{1, 2, \dots, n\}$  is the set of vertices and  $A = \{[i, j], i, j \in X, i \neq j\}$  is the set of edges representing the existence of communication between two vertices  $i$  and  $j$ .

A vertex presents data, service or application. Each vertex is characterized by a confidentiality degree  $d_i$  and size  $s_i$ . Each edge is characterized by a communication frequency  $f_{ij}$  if the two vertices  $i$  and  $j$  are accessed by the same query and need some communication.

Initially, we consider that all vertices are hosted in the private cloud and the public one is empty as illustrated in figure 1. Because the incapacity of the private cloud to host all vertices, the decision maker must specify which vertices to move to the public cloud. After the partitioning vertices (resources) process, we will obtain a private cloud which contains the resources with high confidentiality and a public cloud which contains resources with low confidentiality as illustrated in figure 2.

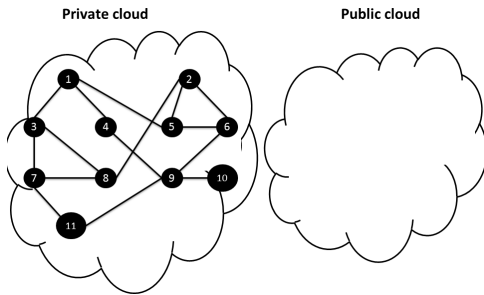


Figure 1: Hybrid cloud before partitioning process.

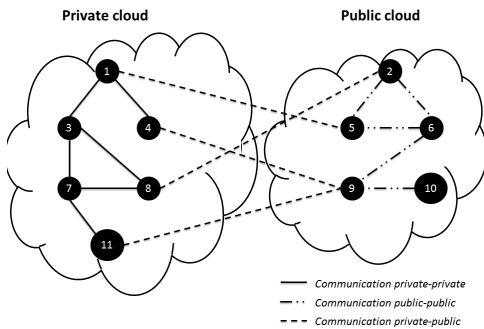


Figure 2: Hybrid cloud after partitioning process.

Usually, except confidentiality minimization, the partitioning process must also take into account the minimization of the total size of vertices affected to the public cloud and the minimization of the total communication frequency outside the private cloud (i.e. the *public-public* and *private-public* communication). In this work, we not interested to the minimization of the communication cost and thereafter we assume that  $f_{ij} = 0, \forall i, j \in X$ .

### 3.2 Mathematical Model

To formulate the mathematical model for CRPHC, we consider the following data and variables:

- $n$ : number of vertices in the graph,
- $X$ : set of vertices formed the graph,
- $d_i$ : degree of confidentiality of each vertex  $i \in X$ . The affectation of values of degree of confidentiality is performed using the opinion of an expert based on transaction historic.
- $s_i$ : size of each vertex  $i \in X$ ,
- $MS$ : the Minimal Size of ressources to host in public cloud. Indeed, the use of the public cloud is motivated by the insufficiency of the private cloud to host all vertices.
- $x_i \in \{0, 1\}$ : a binary decision variables.  $\forall i \in X$ ,  $x_i = 1$  if the vertex  $i$  is affected to the public cloud and  $x_i = 0$  if it's affected to the private one.

Initially, the CRPHC can be formulated by a 0-1 linear program:

$$\text{Min } Z = \max_{i \in X} \{d_i x_i\} + \sum_{i \in X} s_i x_i \quad (1)$$

subject to

$$\sum_{i \in X} s_i x_i \geq MS \quad (2)$$

$$x_i \in \{0, 1\}, \forall i \in X \quad (3)$$

Equation (1) represents the objective function of the CRPHC. It consists to minimize: (i) the maximum confidentiality degree between the vertices affected to the public cloud and (ii) the total size of resources to host in the public cloud. Inequality (2) represents the size constraint of the public cloud. Equation (3) represents the constraints of the decision variables.

We note that the objective function is composed by two inhomogeneous terms in term of their metrics. Indeed, the sizes and the confidentiality degrees are not belonging in the same values intervals. Henceforth, to eliminate this inconvenience, we use the normalized data  $\bar{s}_i$  and  $\bar{d}_i$  instead of  $s_i$  and  $d_i$ ,  $\forall i \in X$  where:

$$\bar{s}_i = \frac{s_i}{\max_{k \in X} s_k} : \bar{s}_i \in [0, 1] \forall i \in X$$

$$\bar{d}_i = \frac{d_i}{\max_{k \in X} d_k} : \bar{d}_i \in [0, 1] \forall i \in X$$

We also note that the proposed objective function is not linear. So, to linearize the model, in order to simplify its resolution by any mathematical models solver, we consider a new integer decision variable  $\lambda \in \mathbb{N}$  to be the maximal confidentiality degree between vertices affected to the public cloud.

$$\lambda = \max_{i \in X} \{\bar{d}_i x_i\} : \lambda \in [0, 1]$$

In the improvement model,  $\lambda$  must be minimized and each confidentiality degree of the vertices affected to the public cloud can not exceed  $\lambda$ . The modified model can be formulated as follows:

$$\text{Min } Z = \lambda + \sum_{i \in X} \bar{s}_i x_i : Z \in [0, 2] \quad (4)$$

$$\text{subject to} \quad \sum_{i \in X} \bar{s}_i x_i \geq MS \quad (5)$$

$$\bar{d}_i x_i \leq \lambda, \forall i \in X \quad (6)$$

$$x_i \in \{0, 1\}, \forall i \in X \quad (7)$$

$$\lambda \in \mathbb{N} \quad (8)$$

## 4 THEORETICAL BASIS OF CRPHC'S APPROACH

In this section, we describe our proposed approach to solve the CRPHC. To classify the resources, we are looking for grouping resources which have the closest degrees of confidentiality and to minimize size of resources which will be hosted in the public cloud. Resources classification must take into account to the already mentioned criteria such as:

- Minimizing degrees of confidentiality of resources (vertices) moved to the public cloud  $C_{pu}$ .
- The size of resources affected to  $C_{pu}$  must exceed slightly a fixed values  $MS$ . This constraint allows minimizing the storage cost of resources moved to the public cloud  $C_{pu}$ .

### 4.1 Principle

Our approach aims at partitionning  $n$  vertices to two clusters (Private Cloud  $C_{pr}$  and Public Cloud  $C_{pu}$ ) with respecting certain number of criteria (already mentioned in the previous paragraph).

**Step 1:** The initial set of vertices will be partitioned into two clusters: The first cluster  $C_{pr}$  will contain confidential resources and the second cluster  $C_{pu}$  will contain non-confidential resources.

**Step 2:** The constraint of the MS of resources to host in public cloud MS must be verified. So two cases are possibles:

*Case 1:* the Total Size of resources affected to  $C_{pu}$  ( $TS_{pu}$ ) is greater than MS,  $C_{pu}$  will be partitioned into two clusters  $CL_{pr}$  (Private CLOUD) and  $CL_{pu}$  (Public CLOUD). Then we have:  $C_{pu} = CL_{pu}$  and  $C_{pr} = C_{pr} \cup CL_{pr}$

*Case 2:* the Total Size of resources affected to  $C_{pu}$  ( $TS_{pu}$ ) is smaller than MS,  $C_{pr}$  will be partitioned into two clusters  $CL_{pr}$  and  $CL_{pu}$ . Then we have:  $C_{pr} = CL_{pr}$  and  $C_{pu} = C_{pu} \cup CL_{pu}$ .

**Step 3:** Repeat **Step 2** until MS is reached or exceeded.

Figure 3 illustrate the already described steps of the proposed approach.

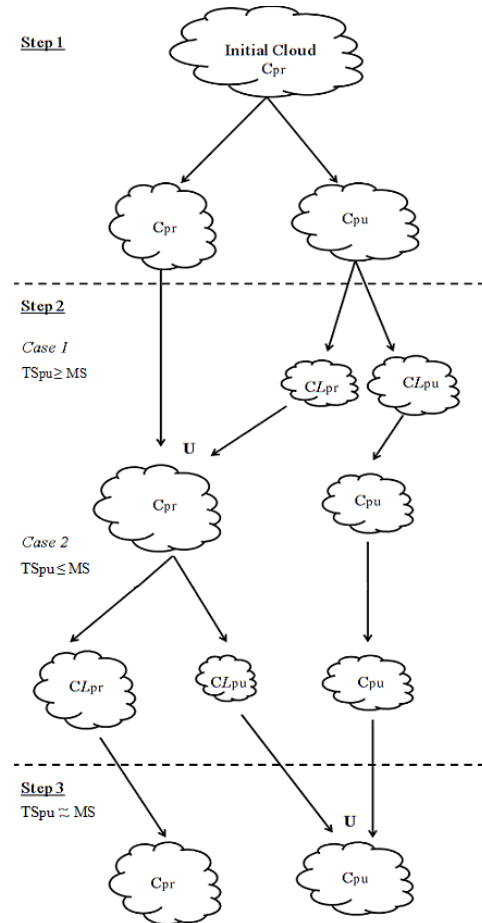


Figure 3: Illustration of approach process.

### 4.2 The CRPHC Algorithm

In this part, we present our solution to classifying resources into two clusters: private and public clusters.

So we implement **CRPHC** Algorithm to classify resources into two clusters.

First, we suppose that all vertices (resources) are hosted in the private cloud, then each vertex is affected in the right cloud taking into account the previous criteria already mentioned. However to apply **CRPHC** Algorithm, a metric must be defined to classify a set of vertices.

In our case, the distance must take into account the degree of confidentiality  $d_i$  and the size  $s_i$  of each vertex  $i \in X$ . As a result, the private cloud  $C_{pr}$  will contain vertices which have the highest degree of confidentiality while minimizing the costs of storage in the public cloud. So, the public cloud  $C_{pu}$  will contain resources having lower confidentiality degree which will minimize the costs of storage in the public cloud.

#### 4.2.1 Definition 1: (Distance)

We consider that each vertex is characterized by coordinates  $(s_i, d_i)$ . So, the distance  $\delta_{ij}$  between two vertices  $i$  and  $j$  will be defined as following:

$$\delta_{ij} = \sqrt{(\bar{s}_i - \bar{s}_j)^2 + (\bar{d}_i - \bar{d}_j)^2}$$

#### 4.2.2 Definition 2: (Centroid)

A centroid is characterized by coordinates  $(s_{\omega_k}, d_{\omega_k})$ . These coordinates are calculated as following:

$$s_{\omega_k} = \frac{\sum_{i \in C_k} \bar{s}_i}{|C_k|}, \forall k \in \{1, 2\}$$

$$d_{\omega_k} = \frac{\sum_{i \in C_k} \bar{d}_i}{|C_k|}, \forall k \in \{1, 2\}$$

#### 4.2.3 Algorithm

Our solution is based on a main algorithm (**CRPHC** Algorithm) which uses each time **Affect** Algorithm to classify different vertices of the graph into two clusters. The input of the **CRPHC** Algorithm is the graph to be partitioned. As is already mentioned, each vertex of the graph is characterized by a degree of confidentiality  $d_i$  and a size  $s_i$ . Then, the output is two clusters: private cloud  $C_{pr}$  and public cloud  $C_{pu}$ . For **Affect** Algorithm, the output is also a private cloud  $CL_{pr}$  and a public cloud  $CL_{pu}$ . Algorithm 1 describes the proposed approach to solve the **CRPHC**.

Initially, we assume that all vertices are hosted in the private cloud and the public cloud is empty. The **Affect** Algorithm will be applied to all the graph to give firstly two clusters: private cloud  $CL_{pr}$  and public cloud  $CL_{pu}$ . If the Total Size  $TS_{pu}$  of resources hosted in public cloud is greater than  $MS$  (see line 6), **Affect** Algorithm will be applied to public cloud  $CL_{pu}$ . This

---

#### Algorithm 1: CRPHC Algorithm.

---

```

input : $G(X, A) / |X| = n$.
output: C_{pr} and C_{pu} where $C_{pr} \cup C_{pu} = X$ and
 $C_{pr} \cap C_{pu} = \emptyset$.

1 $C_{pr} \leftarrow X$;
2 $C_{pu} \leftarrow \emptyset$;
3 Affect(X) /* Apply Affect (see
 Algorithm 2) */;
4 $C_{pr} \leftarrow \{CL_{pr}\}$;
5 $C_{pu} \leftarrow \{CL_{pu}\}$;
6 if $TS_{pu} \geq MS$ /* (TS: Total Size of
 resources affected to public cloud)
 */ then
7 repeat
8 $X \leftarrow CL_{pu}$;
9 Affect(X);
10 $C_{pr} \leftarrow C_{pr} \cup CL_{pr}$;
11 $C_{pu} \leftarrow CL_{pu}$;
12 until $TS_{pu} > MS$;
13 else
14 repeat
15 $X \leftarrow CL_{pr}$;
16 Affect(X);
17 $C_{pu} \leftarrow C_{pu} \cup CL_{pu}$;
18 $C_{pr} \leftarrow CL_{pr}$;
19 until $TS_{pu} < MS$;
20 end

```

---

allows to decrease the Total Size of resources hosted in  $C_{pu}$ , indeed, vertices of  $CL_{pu}$  will be moved from the public cloud  $C_{pu}$  to the private cloud  $C_{pr}$  (see lines 7-12). So the **Affect** Algorithm will be applied to public cloud  $CL_{pu}$  until the  $MS$  is reached.

Likewise, if  $TS_{pu}$  is lower than  $MS$  (see lines 13-19), the **Affect** Algorithm will be applied to  $CL_{pr}$ . Indeed, the vertices of  $CL_{pr}$  will be moved from private cloud  $C_{pr}$  to public cloud  $C_{pu}$  until the  $MS$  is reached or exceeded.

**Affect** Algorithm consists in a first step to choose two vertices  $\omega_{pr}$  and  $\omega_{pu}$  from the graph (see lines 1-2). The choice of these vertices is performed using two functions  $max()$  and  $min()$ . The function  $max()$  choose the vertex with the maximal degree of confidentiality in the graph and the function  $min()$  choose the vertex with the minimal degree of confidentiality in the graph. In this case, **Affect** Algorithm regroup vertices with higher confidentiality in one cluster (private cloud) and vertices with lower confidentiality in another cluster (public cloud).

To affect vertices to the right cluster (see lines 7-11), the idea is to compute the distance  $\delta_{(i, \omega_k)}$ , with  $k = \{pr, pu\}$ , between each vertex in the graph and each centroid. If the vertex is closest to  $\omega_{pr}$ , it will be hosted to the private cloud  $CL_{pr}$  and if the vertex

**Algorithm 2:** Affect Algorithm.

---

**input** : A set of vertices  $X$  to be partitioned.  
**output**: Tow clusters  $CL_{pr}$  and  $CL_{pu}$  for the vertices hosted (respectively) in private and public clouds.

```

1 $\omega_1 \leftarrow \max(X)$;
2 $\omega_2 \leftarrow \min(X)$;
3 repeat
4 $CL_{pr} \leftarrow X$;
5 $CL_{pu} \leftarrow \emptyset$;
6 for each $i \in X$ do
7 if $\delta_{i\omega_{pr}} \leq \delta_{i\omega_{pu}}$ then
8 $CL_{pr} \leftarrow CL_{pr} \cup \{i\}$;
9 else
10 $CL_{pu} \leftarrow CL_{pu} \cup \{i\}$;
11 end
12 end
13 $old_w_{pr} \leftarrow \omega_{pr}$;
14 $old_w_{pu} \leftarrow \omega_{pu}$;
15 $new_w_{pr} \leftarrow \text{centroid}(CL_{pr})$;
16 $new_w_{pu} \leftarrow \text{centroid}(CL_{pu})$;
17 until $old_w_{pr} == new_w_{pr}$ AND $old_w_{pu} == new_w_{pu}$;
```

---

is closest to  $\omega_{k_{pu}}$ , it will be hosted to the public cloud  $CL_{pu}$ . Then, we compute centroids of the new clusters (see lines 15-16) and we repeat this process (see lines 4-16) until we have the stability of the two clusters i.e. we reach the same centroids in two successive iterations (see line 17).

## 5 EXPERIMENTAL EVALUATION

To apply and assess our approach, we need instances for CRPHC algorithm. Then we need to vary the following parameters:

- $n$ : number of vertices of the graph,
- $d$ : degrees of confidentiality of vertices of the graph,
- $TS$ : Total Size of vertices of the graph,
- $MS$ : Minimal Size of resources to host in public cloud.

But, it is not possible to find real cases or benchmarks based on previous parameters and able to varying them. This is why we need to create several graphs according to our need. For this reason we have developed a generator of graphs. Each graph is composed by six elements: number of vertices, vector of size of each vertex, a vector of confidentiality degree

of each vertex,  $MS$ , matrix of execution frequency between two vertices and minimal degree of confidentiality in public cloud. In our work we just interest to the four first elements. The generated graphs are available on <http://goo.gl/uvO8B8>.

In this section, we will evaluate our solution with an optimal solution CPLEX. CPLEX is a computing tool of optimization (Aitha, 2014) which gives optimal solutions applied to a integer programming formulation.

So, to assess our solution we applied CPLEX to the same graphs that we used to test CRPHC Algorithm. Then we compare the results obtained with our solution and those obtained by CPLEX.

To better analyze and interpret results, we calculate the *Gap* between results given by CRPHC and those given by CPLEX. This *Gap* is given by:

$$Gap = (CRPHC\_OF - CPLEX\_OF) / cplex\_OF$$

Then, we recognize that a solution is called:

- Optimal: if the Gap associated is 0%,
- Excellent: if the Gap associated it does not exceed 15%,
- Incorrect: if the Gap associated exceeds 50%.

### 5.1 Varying Number of Vertices

These tests consist in varying the number of vertices of graphs (500, 1000, 1500, 2000, 2500 and 3000) and fixing the  $MS = 20\%$ , Total Size of the graph  $TS = 10000$  and degree of confidentiality  $d_i \in [0\% - 100\%]$ .

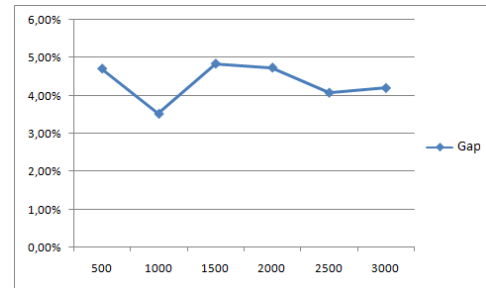


Figure 4: Gap for varying number of vertices.

Figure 4 shows that the Gap values are between 3,4% et 5% so they are so close.

We remark that the Gap between our solution and optimal solution does not exceed 5%. So, in this case, our solution can be considered excellent.

### 5.2 Varying Degree of Confidentiality

For these tests, we fixed the number of vertices  $n = 2000$ ,  $MS = 20\%$  and Total Size of the graph  $TS =$

10000. Then we varied the range of degree of confidentiality. To do this, we have chosen three ranges:

- $[0\% - 20\%]$  this range represents resources with low confidentiality,
- $[80\% - 100\%]$  this range represents resources with high confidentiality,
- $[0\% - 100\%]$  this range represents resources with low and high confidentiality.

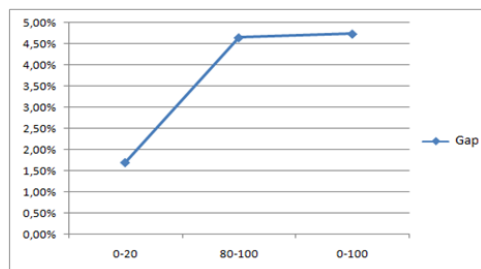


Figure 5: Gap for varying degree of confidentiality.

Figure 5 shows that the Gap for  $[0\% - 20\%]$  is 1,6%. This value is low compared to values of other range of confidentiality. So, for resources with low confidentiality, the CRPHC\_OF value is close to CPLEX\_OF. Then the graph is almost stable in the order of 4,7% between  $[0\% - 100\%]$  and  $[80\% - 100\%]$ .

The Gap between our solution and optimal solution does not exceed 4,7%. So, we can consider that our solution is excellent in this case.

### 5.3 Varying Total Size

For these tests, we fixed number of vertices  $n = 2000$ ,  $MS = 20\%$  and the degree of confidentiality  $d_i \in [0\% - 100\%]$ . Then we varied the Total Size of the graph (10000, 20000 and 30000).

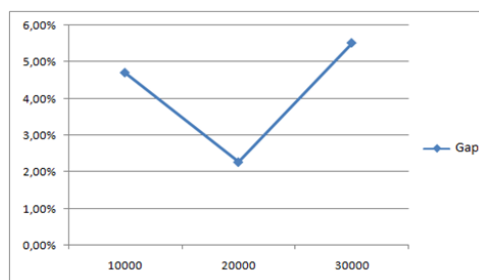


Figure 6: Gap for varying size.

Figure 6 shows that the best Gap value is given for  $TS = 20000$ .

Then we have the Gap between our solution and optimal solution does not exceed 6%. So, we can consider that our solution is excellent in this case.

### 5.4 Varying MS

For these tests, we fixed number of vertices  $n = 2000$ , degree of confidentiality  $d_i \in [0\% - 100\%]$  and Total Size of the graph  $TS = 10000$ . Then we varied MS (20%, 30% and 60%) in the public cloud.

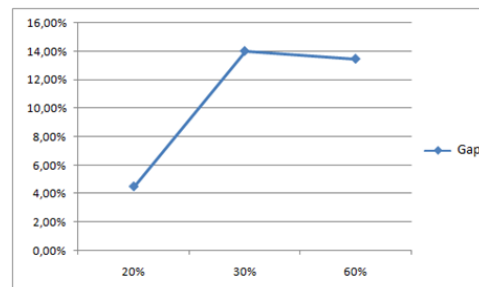


Figure 7: Gap for varying MS.

Figure 7 shows that the Gap for  $MS = 20\%$  is 4,3%. This value is the best result given by our solution compared to  $MS = 30\%$  and  $MS = 60\%$ .

Then we remark that the graph is almost stable in the order of 14% between  $MS = 30\%$  and  $MS = 60\%$ .

The Gap between our solution and optimal solution does not exceed 14%. So, in this case, our solution can be considered excellent.

## 6 CONCLUSION AND FUTURE WORK

In this paper, we tackled a new approach for partitioning confidential resources between private and public components in hybrid cloud. Our objective is to ensure confidentiality by moving confidential resources to private cloud and resources with low confidentiality to public cloud. And also, minimizing the size of resources to host in public cloud. Then we have compared the results given by our proposed solution with optimum results given by CPLEX, and we have found that our results are acceptable.

In this work, we have supposed that hybrid cloud is composed by one private cloud and one public cloud. So to enlarge our work, we hope to propose an approach to partitioning confidential resources in hybrid clouds based on multitude of criteria which managing the allocation decision of each resource to one of the classes: private clouds and public clouds. Thus we place ourselves in a melting problem of sources of information (confidentiality, capacity, degree of dependence between resource, etc.). Then we focus on the notion of dynamicity such as confidentiality and sizes of resources that will be partitioned in hybrid clouds.

## REFERENCES

- Aitha, P. (2014). Cplex tutorial handout. <http://fr.scribd.com/doc/63956075/CPLEX-Tutorial-Handout>.
- Chokhani, R. C. M. I. S. (2013). Cryptographic key management issues and challenges in cloud services. *National Institute of standards and Technology*.
- Kantarcioglu, V. K. M. and Thuraisingham, B. (2011). Secure data processing in a hybrid cloud. *Pacific Asia conference on Intelligence and Security Informatics*.
- Lamba, S. and Kumar, A. (2014). An approach for ensuring security in cloud environment. *International Journal of Computer Applications*.
- Marwaha, M. and Bedi, R. (2013). Applying encryption algorithm for data security and privacy in cloud computing. *IJCSI International Journal of Computer Science Issues*.
- Mehrotra, V. K. K. O. B. H. M. K. S. and Thuraisingham, B. (2012). Risk-aware data processing in hybrid clouds. *IEEE 5th International Conference on cloud Computing*.
- Nepal, D. T. S. C. S. and Calvo, R. (2014). Secure data sharing in the cloud. *Security; Privacy and Trust in cloud Systems*.
- Pilli, P. R. P. M. R. S. E. and Joshi, R. (2013). Improved technique for data confidentiality in cloud environment. *Networks and Communications*.
- Ramakrishnan, T. Z. R. and Livny, M. (1996). Birch : An efficient data clustering method for very large database. *SIGMOD*.
- Stoica, M. A. A. F. R. G. A. J. R. A. K. G. L. A. P. A. R. I. and Zaharia, M. (2009). Above the clouds : A berkeley view of cloud computing. In *a*.
- Tata, F. B. N. T. S. and Moalla, S. (2012). Approximate placement of service based applications in hybrid clouds. *21st International conference IEEE WET-ICE*.
- Wang, L. G. Z. H. S. Z. N. Z. J. and C.Jiang (2012). Multi-objective optimization for data placement strategy in cloud computing. *Springer*.
- Wang, X. and Guo, W. (2013). A data placement strategy based on genetic algorithm in cloud computing platform. *10th Web Information System and Application Conference (WISA)*.
- Zhang, C. C. E. J. Y. and Madden, S. (2010). Schism : a workload-driven approach to database replication and partitioning. *VLDB; 36th International Conference on Very Large Data Bases*.



# Cloud Spreadsheets Supporting Data Processing in the Encrypted Domain

D. A. Rodríguez-Silva<sup>1</sup>, L. Adkinson-Orellana<sup>1</sup>, B. Pedrero-López<sup>1</sup> and F. J. González-Castaño<sup>2</sup>

<sup>1</sup>*Gradiant, Edif. CITEXVI, Campus de Vigo, 36310, Pontevedra, Spain*

<sup>2</sup>*AtlantTIC, Escuela de Ingeniería de Telecomunicación, Universidade de Vigo, 36310, Pontevedra, Spain*  
{darguez, ladkinson}@gradiant.org, javier@det.uvigo.es

**Keywords:** Cloud Computing, Security, Privacy, Homomorphic Encryption, Spreadsheet.

**Abstract:** Security has become one of the main barriers for the adoption of cloud services. A range of legal initiatives that require support mechanisms such as access control and data encryption have been proposed to ensure privacy for data moved to the cloud. Although these mechanisms are currently feasible in situations in which the cloud acts as a mere data storage system, they are insufficient in more complex scenarios requiring processing in external cloud servers. Several new schemes have been proposed to overcome these shortcomings. Data Processing in the Encrypted Domain (DPED) permits arithmetic operations over ciphered data and the generation of encrypted results, without exposure of clear data. In such a set-up, the servers have no access to the information at any point of the process. In this paper we describe, as a case study of secure cloud data processing, a cloud spreadsheet that relies on DPED libraries to perform operations in the encrypted domain. Tests performed on local servers and in the Google cloud through the Google App Engine platform show that representative real applications can benefit from this technology. Because the proposed solution is PaaS-oriented, developers can apply the libraries to other applications.

## 1 INTRODUCTION

Security and privacy are both major concerns for Cloud Computing users. As reported in the European CIOs and Cloud Services research study (2010), around 71% of European companies are worried about security and privacy, especially when it comes to storing or processing sensitive data in the cloud. Security has thus become a significant barrier to full adoption of cloud services.

Concerns regarding security and privacy have been addressed in part by different legal initiatives within the European Union, such as Directive 95/46/EC of the European Parliament and the Council of October 24 1995 (Data Protection Directive, 1995), which proposes a set of recommendations for protecting personal data during transfer and processing. In Spain there are several specific laws to protect and regulate the management of personal and corporate data used by cloud applications, including the Data Protection Regulation, of Law 15/1999 on Personal Data Protection (LOPD, 1999) and the Royal Decree 1720/2007, which approves the development of the LOPD (RDLOPD, 2007). As an example of the

recommended proposals, the 85<sup>th</sup> article of the RDLOPD states that security measures applied to personal data in communication networks, public or not, should guarantee at least the same security level as that offered by local access systems.

Due to their very nature, data processed in the cloud will presumably be affected by international data transfers, primarily because many web applications are hosted on foreign servers. International data movement is regulated by the data protection regulation, which forbids international data transfers between countries that do not offer sufficient security guarantees according to the LOPD, although there are some exceptions explicitly indicated in the reference regulation. In addition, this regulation sets out several legal requirements, such as transfer notification to the Spanish Data Protection Agency.

Because legal procedures are slow, new technological mechanisms are required until the situation is completely regulated. Authentication on the client side and use of security mechanisms such as data encryption during data transmission are good solutions for interception attacks and servers that do not offer sufficient guarantees of reliability. To increase security, the client can cipher data using a

private key, thereby hiding the information from the server. Although this option is valid when the cloud acts as a mere storage service, there are cases in which it would be insufficient, for example when performing certain calculations on the server or when processing a query to a database with ciphered data.

To overcome the above shortcomings, new server-side schemes have been proposed, such as the use of cryptographic hardware or Data Processing in the Encrypted Domain (DPED). Cryptographic hardware can be used to perform cryptographic operations and to store keys securely, but it is expensive (specific trusted anti-tampering devices are required) and it needs to be physically integrated into the provider's infrastructure. DPED overcomes these problems, but at the expense of increased processing time. It enables operations over ciphered data that generate encrypted results, thereby allowing server-side operations without revealing the original information. This adds an additional security level to the cloud paradigm by means of complex homomorphic algorithms. The computational requirements may not be a problem thanks to the scalability and flexibility of the cloud paradigm.

In this paper we describe a cloud spreadsheet application that uses the DPED concept to perform operations in the encrypted domain. We have tested it on our local servers and in the Google cloud through the Google App Engine (GAE) platform. Section 2 discusses related work and section 3 explains the implementation details of our application. Section 4 presents the tests performed, and finally, section 5 concludes the paper.

## 2 RELATED WORK

Many office cloud applications allow users to work with spreadsheets. Some well-known examples are Microsoft Office 365, Google Drive Spreadsheets, Thinkfree Calc and Zoho Sheet. Nevertheless, none of these applications currently offers full protection mechanisms for user data, meaning that privacy, when available, is supported by external means. Indeed, most current solutions are designed for Google Drive, not for spreadsheets. Furthermore, although there are solutions that are completely integrated with the Google Drive interface that encrypt documents transparently to users (Adkinson-Orellana et al., 2010), most simply use the cloud to store the encrypted documents (CryptRoll, 2013; ZecurePC, 2011; and CloudLock, 2015).

DPED allows certain operations to be performed over ciphered data without the need to access the clear version. In particular, arithmetic operations can be performed efficiently in the encrypted domain thanks to the concept of additive and multiplicative privacy homomorphisms (Brickell and Yacobi, 1987). In 2009, Gentry presented the first fully homomorphic encryption scheme. He described public key encryption using ideal lattices (Gentry, 2009). In the same year, M. Van Dijk described a "somewhat homomorphic" encryption scheme based on elementary modular arithmetic, and used Gentry's techniques to convert it to a full homomorphic scheme (M. Van Dijk et al., 2009) that implemented addition and multiplication over integers rather than ideal lattices over a polynomial ring.

There have been other contributions in this direction. A. F. Chan formulated a privacy homomorphism for operating over ciphered data with two different encryption schemes, where data could be processed directly in an encrypted form (Chan, 2009). H. Hacigümüş, in turn, described different techniques for executing SQL queries over encrypted data (Hacigümüş et al., 2002). The strategy involves processing as much of the query as possible at the service provider site, without decrypting data. Decryption and the remainder of the query processing takes place at the client side. They also explored an algebraic framework to split the query to minimize computation at the client side.

The innovative idea in this paper is to enable DPED processing in cloud applications. We are not aware of any previous DPED-enabled complex cloud applications, although, in a previous work, we presented a toy example that demonstrated that DPED could strengthen the privacy of simple mathematical operations in the cloud (Rodriguez-Silva et al., 2011).

## 3 SECURE CLOUD SPREADSHEET

### 3.1 Arithmetic Calculations in the Encrypted Domain

The spreadsheet application is composed of two modules: a client module and a server module. The client module presents the spreadsheet interface, which is used to enter data, cipher its content, send it to the server, and decipher and present the results in the corresponding spreadsheet cell. The server

module, in turn, executes arithmetic operations on the encrypted data received from the client by means of adequate privacy homomorphisms. The supported encrypted operations are listed in Table 1.

Due to the low efficiency of complete homomorphisms in the current state-of-the-art, our implementation uses a variation of the additive homomorphic encryption described by Paillier (Paillier, 1999) as the basis of our cryptographic system. One or more additional rounds of communication between the client and the server will also be needed depending on the complexity of the operation requested.

Table 1: Encrypted operations supported by the spreadsheet.

Operation	Description	Example
AVERAGE	Average value	AVERAGE (A1:A5)
DEGREES	Degree conversion	DEGREES (A1:A5)
FFT	Fast Fourier Transform	FFT (A1:A5)
PROD	Product	PROD (A1:A5)
RADIANS	Radian conversion	RADIANS (A1:A5)
SPROD	Scalar product	SPROD(A1:A5;B1:B5)
STDEV	Standard deviation	STDEV (A1:A5)
SUM	Addition	SUM (A1:A5)
VADD	Vector addition	VADD (A1:A5;B1:B5)
VAR	Variance	VAR (A1:A5)
VPROD	Vector product	VPROD(A1:A5;B1:B5)
VSUB	Vector subtraction	VSUB (A1:A5;B1:B5)

The encryption methods used are based on asymmetric key algorithms. The libraries present different options, such as threads and JNI (Java Native Interface), thereby increasing efficiency

thanks to the use of C libraries. The encryption libraries also allow operations with scalars and vectors, unlike the version in our previous work, which only offered basic operations for unary values.

### 3.2 Ciphred Cloud Spreadsheet Implementation

The spreadsheet allows operations over a range of cells, with no limitations in terms of the number of operators involved. The implementation relies on Java technology, as this is the most common PaaS language. It uses the Java Runtime Environment (JRE) classes available to create applets and tables (JTable, TableModel, etc.), meaning that results can be easily embedded on a web page. The development environment used to create the spreadsheet and the associated technologies are the same as those used for the encrypted calculator (Rodriguez-Silva et al., 2011): Java Servlets and IDE Eclipse 3.5 (Galileo) for the server and Java applets and Oracle IDE Netbeans 6.8 for the client, with the corresponding plugin to create graphical user interfaces. Again, GAE was selected as the cloud platform to deploy the application. Due to the restrictions of this PaaS, the encryption libraries had to be adapted, since the platform has a limited support for multithreading (a characteristic that the libraries use to improve efficiency).

By default, some of the applet functionalities (e.g. reading or writing files on disk) are restricted through a security policy implemented by the security controller of the browser Java Virtual

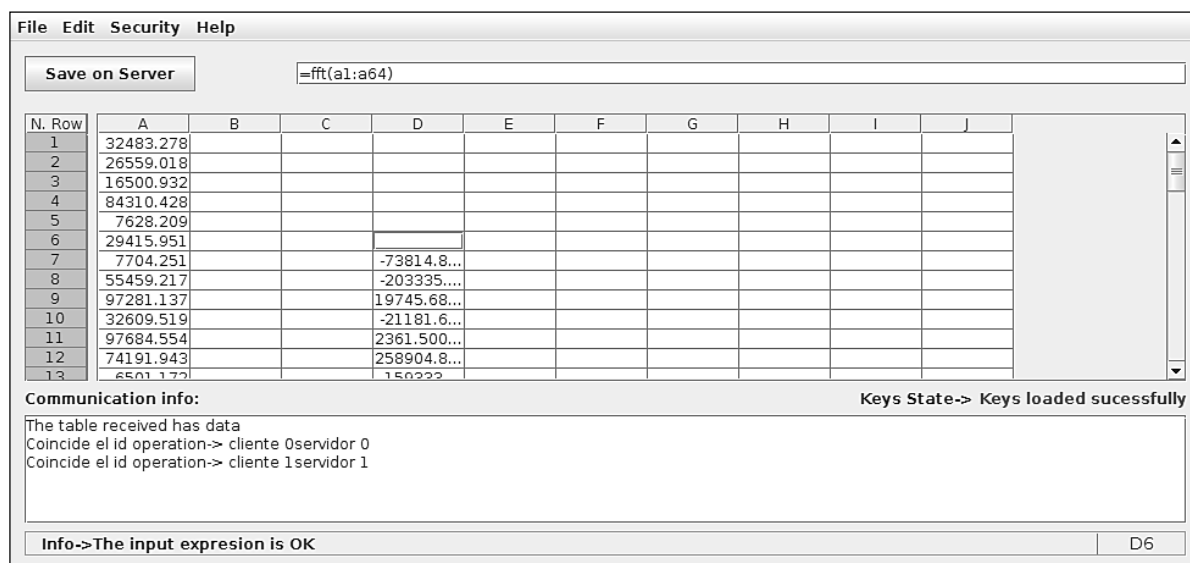


Figure 1: Ciphred cloud spreadsheet interface.

Machine (JVM) plugin. However, the spreadsheet applet needs these functionalities to store its content in a local file and to obtain and save the user keys to cipher data. For this reason, the user must accept the applet signature, granting certain restricted functionalities. All the libraries except `au.com.bytecode.opencsv` (required to save the content of the spreadsheet in .CSV format) and the DPED libraries are available on the JRE used by the browsers. Other libraries are downloaded when the applet starts.

The client is composed of two modules: one for the graphic interface and the other for parsing and communicating with the server. The graphic module is in charge of the visual interface, intercepting user events and presenting results. Its design was based on typical spreadsheet software, such as Google Spreadsheet, OpenOffice.org and Microsoft Excel (Figure 1). The client parser analyses formulae expressions to obtain the data required to perform the operations. Finally, the communication module exchanges data with the server using HTTP tunnelling.

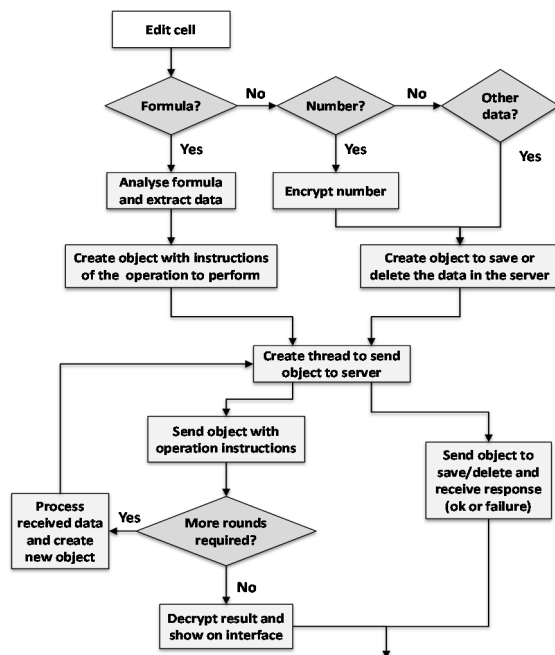


Figure 2: Client module flowchart according to user interaction.

To perform a spreadsheet operation, the first step is to start the application (applet) and load the libraries. The user then introduces his/her login and password, using his/her Google user account if the application is deployed in GAE. If access is granted, the user keys —required to cipher and decipher the data and perform encrypted operations— are loaded

from binary files. When the process is complete, the spreadsheet graphic interface is shown. At this point, the application is ready to process events generated by user interaction (see Figure 2):

- *Finish cell edition.* The application checks the type of data entered in the cell, such as formulae, a number or other data types. The formulae are analysed, the type of operation and the references to the cells are extracted, and in the case of numbers, these are ciphered. With this information a new communication object is created to be sent to the server. In the case of other data types, the cell content is directly inserted in the communication object. Once this object has been created, a new communication thread is thrown to send the object to the server. In other words, the interface thread is released to receive new user events. The communication thread remains open until confirmation is received that the number or object has been correctly received at the server side or until the operation defined by the formula has finished. This is indicated by the communication round. The result is then decrypted and displayed to the client.
- *Save on server.* The client creates an object with the order to save the spreadsheet at the server side. A new thread is created specifically to send the petition containing this object. This thread will receive a response indicating the success or failure of the request.
- *Select a menu option.* The selected option is performed at the client or the server side, depending on the actions involved, e.g., create/load user keys, add/remove rows or columns, copy/paste cells, etc.

Dependencies between the values of the cells must be taken into account. When the value of a cell changes, it can affect other related cells, resulting in the execution of multiple parallel operations. To manage this situation, a new thread is thrown for each dependent operation, creating a new client. This ensures that the different operations executed do not interfere with each other, but it requires more processing load for the application during the initialization of a new module.

We considered two possible server deployment scenarios: a private cloud (local) and a public cloud (GAE). The private cloud does not have all the resources and services offered by GAE, such as user accounts and persistent storage. In this case users are authenticated through a local mechanism and persistent storage is simulated by saving the spreadsheet in a local file, identified by the user

login. In this way, when a user enters the application, the saved spreadsheet will be retrieved and deciphered if and only if he/she has the appropriate private key.

The server is a servlet composed of two modules: the communication module and the data processing module. The communication module performs the same actions as on the client side, while the data processing module is in charge of storing or processing the data received in the encrypted domain and recovering the stored spreadsheet.

On the server side, the first step is to initialize the servlet, which will keep listening to incoming petitions from the clients. When a request is received, the thread recovers the object and retrieves the data it contains. The selected action is then executed, i.e. the data received is saved, the cell value is deleted, the module is initialized, the encrypted operation is performed, etc.

The client and server modules must store the status of each operation requested, indicating the current execution round. If several operations are requested at once, several modules for the client and the server will be instantiated, and those associated with the same operation will be identified by a unique identifier. Thus, each time an operation is requested, it will be possible to execute it in the corresponding module, avoiding result inconsistencies when several rounds are being carried out.

Communication between the client and the server takes place through a Java object, which is used to retrieve and store information. This object is sent through HTTP tunnelling, facilitating data transmission through different elements (firewalls, proxies, etc.) that typically limit connection to web resources. In addition, the object is used to send and receive different types of data, such as encrypted data to be stored by the server, information related to the operation performed and the cells involved, encrypted results received from the server, etc.

The application also supports the generation of the keys required to cipher and decipher data. These keys offer two levels of security: short-term and medium-term. While short-term security speeds up encryption and decryption, medium-term security is stronger, as it would take approximately ten times longer to break up its keys (e.g., ten years vs 1 year).

## 4 PERFORMANCE TEST

The test layout comprised a client computer (Intel

Core i3-2120 @ 3.3 GHz, 3870 MB RAM, Ubuntu 10.4) and a local server (with the same characteristics) to perform the encrypted operations under Jetty 7.5.4. The cloud application applet was executed through the Google Chrome 17 browser. We also used GAE servers equipped with Jetty to deploy the applications.

The operation selected to evaluate the performance of the spreadsheet was a Fast Fourier Transform (FFT), as this is a complex operation that permits representative performance results. The FFT was applied to vectors with lengths of 64, 128 and 512 points.

The current version of the spreadsheet encrypts and sends each data item to the server individually. When an operation (the FFT in this case) is selected at the client side, the server simply receives the operation and the cells involved, since it has already the ciphered values. Therefore, the total time needed to perform an operation comprises two times: a *data entry time*, including the management of the data in the cells in the spreadsheet and the time needed to encrypt and send each operand to the server, and a *running time*, which is the time needed to perform an encrypted operation on the server and present the result on the client side.

We used four test scenarios:

- *Local, with threads and JNI.* The encrypted FFT was executed on the local server. There were five parallel threads in the server and the client. JNI was used to improve efficiency.
- *Local, without threads or JNI.* As in the previous case, the operation was executed on the local server. Neither threads for parallel executions nor C functions were used in this scenario.
- *Local, clear FFT.* We executed the FFT using clear data on the local server. The FFT algorithm was implemented by a Java function.
- *Remote, deployed on a GAE server.* GAE does not allow the use of threads or JNI, so the server was subject to these restrictions. At the client side five threads were used, in addition to C functions through JNI.

The client was executed on the same machine and used the same browser in all four scenarios.

In each scenario, 10 FFT operations were performed for randomly generated vectors for the three lengths. We used both the short- and medium-term security levels to perform these tests. Tables 2 and 3 show the corresponding results based on the following times:

- *Entry time (ET).* Time from the moment an

operand is entered in the spreadsheet to the moment the client receives the response from the server (indicating that the encrypted data have been correctly stored).

- *Server execution time (SE)*. Time taken by the server to perform the encrypted FFT operation.
- *Running time (RT)*. The sum of communication time required to exchange data between the client and the server, the time needed by the server to perform an ciphered FFT (SE) and the time needed to decrypt the result for the client.

Each pair of values represents the average execution time ( $\bar{x}$ ) and its standard deviation ( $\sigma$ ), both in milliseconds.

The tests were unfeasible for the 512-point FFT in the remote scenario because the time needed to generate and return a response in GAE is limited to 30-60 seconds and the 512-point FFT needs longer. These results are therefore not shown in the tables.

The use of longer keys improves security considerably, but increases effective operating time. On comparing Table 2 and Table 3, we can see that the time required to perform the same operation is considerably higher for the medium-term security level. This mainly affects ciphered operation time (i.e., server execution time). The different level of security does not have an impact on data entry time, as this includes the ciphering of data but not the execution of the encrypted operations.

The use of threads and JNI considerably reduced the execution time in both cases, primarily due to the improved efficiency of the execution of the algorithm on the server and the improved efficiency of the decoding process. Although this test scenario cannot be translated to GAE, its results can give us an idea of how the performance could be improved with JNI and threads in GAE or other compatible PaaS.

The running time with GAE was much higher than in the equivalent local case without threads or JNI. Besides of the Internet delay, GAE servers took approximately 10 times longer to execute the same algorithm, probably because GAE is optimized for applications with short response times, typically of hundreds of milliseconds.

The total execution time for the best DPED scenario (local, threads and JNI) was much longer than with unencrypted data (local, clear FFT). This is obviously due to the time spent on encrypting data and decrypting the result, and the efficiency of the DPED FFT algorithm, which is over 200 times

slower than the algorithm used to calculate the FFT with unencrypted data.

Table 2: Data entry time, server execution time and running time using the short-term security level (ms).

Test scenario		64	128	512
Local, no threads or JNI	ET	$\bar{x}=524.2$ $\sigma=23.5$	$\bar{x}=897.9$ $\sigma=43.3$	$\bar{x}=6888.4$ $\sigma=114.4$
	SE	$\bar{x}=581.4$ $\sigma=152.7$	$\bar{x}=1177.9$ $\sigma=137.8$	$\bar{x}=6013.7$ $\sigma=120.1$
	RT	$\bar{x}=1206.6$ $\sigma=140.4$	$\bar{x}=2256.7$ $\sigma=135.1$	$\bar{x}=9918.4$ $\sigma=131.5$
Local, threads and JNI	ET	$\bar{x}=565.1$ $\sigma=21.9$	$\bar{x}=936.1$ $\sigma=21.3$	$\bar{x}=8642.9$ $\sigma=127.6$
	SE	$\bar{x}=76.9$ $\sigma=9.4$	$\bar{x}=140.9$ $\sigma=31.7$	$\bar{x}=545.8$ $\sigma=62.5$
	RT	$\bar{x}=239.1$ $\sigma=18.3$	$\bar{x}=356.3$ $\sigma=41.4$	$\bar{x}=1108.8$ $\sigma=103.6$
Remote (GAE)	ET	$\bar{x}=16539.3$ $\sigma=538.9$	$\bar{x}=34056.3$ $\sigma=1355.2$	---
	SE	$\bar{x}=5595.2$ $\sigma=294.3$	$\bar{x}=13793.8$ $\sigma=858.8$	---
	RT	$\bar{x}=6339.3$ $\sigma=365.7$	$\bar{x}=14688.0$ $\sigma=881.2$	---
Local, clear FFT	ET	$\bar{x}=475.6$ $\sigma=23.9$	$\bar{x}=790.3$ $\sigma=41.3$	$\bar{x}=6250.7$ $\sigma=118.3$
	SE	$\bar{x}=0.159$ $\sigma=0.086$	$\bar{x}=0.516$ $\sigma=0.307$	$\bar{x}=1.318$ $\sigma=0.422$
	RT	$\bar{x}=6.5$ $\sigma=1.5$	$\bar{x}=6.9$ $\sigma=2.5$	$\bar{x}=10.4$ $\sigma=2.0$

Table 3: Data entry time, server execution time and running time using medium-term security level (ms).

Test scenario		64	128	512
Local, no threads or JNI	ET	$\bar{x}=1132.3$ $\sigma=35.1$	$\bar{x}=1916.8$ $\sigma=36.9$	$\bar{x}=8308.8$ $\sigma=192.5$
	SE	$\bar{x}=2382.4$ $\sigma=500.8$	$\bar{x}=4538.1$ $\sigma=623.2$	$\bar{x}=20909.2$ $\sigma=742.1$
	RT	$\bar{x}=6127.7$ $\sigma=550.3$	$\bar{x}=11129.8$ $\sigma=567.9$	$\bar{x}=46873.3$ $\sigma=1314.3$
Local, threads and JNI	ET	$\bar{x}=589.9$ $\sigma=53.6$	$\bar{x}=995.1$ $\sigma=38.3$	$\bar{x}=6983.7$ $\sigma=137.4$
	SE	$\bar{x}=173.6$ $\sigma=23.2$	$\bar{x}=337.0$ $\sigma=35.7$	$\bar{x}=1558.2$ $\sigma=287.5$
	RT	$\bar{x}=597.4$ $\sigma=46.6$	$\bar{x}=1031.6$ $\sigma=35.1$	$\bar{x}=4062.7$ $\sigma=491.0$
Remote (GAE)	ET	$\bar{x}=16733.6$ $\sigma=1113.3$	$\bar{x}=34709.2$ $\sigma=2410.8$	---
	SE	$\bar{x}=22045.1$ $\sigma=915.4$	$\bar{x}=52640.6$ $\sigma=1033.4$	---
	RT	$\bar{x}=23007.0$ $\sigma=926.6$	$\bar{x}=54172.3$ $\sigma=1111.0$	---
Local, clear FFT	ET	$\bar{x}=475.6$ $\sigma=23.9$	$\bar{x}=790.3$ $\sigma=41.3$	$\bar{x}=6250.7$ $\sigma=118.3$
	SE	$\bar{x}=0.159$ $\sigma=0.086$	$\bar{x}=0.516$ $\sigma=0.307$	$\bar{x}=1.318$ $\sigma=0.422$
	RT	$\bar{x}=6.5$ $\sigma=1.5$	$\bar{x}=6.9$ $\sigma=2.5$	$\bar{x}=10.4$ $\sigma=2.0$

## 5 CONCLUSIONS

Cloud computing provides an adequate environment for deploying applications following a Software-as-a-Service (SaaS) model. However, security and privacy are key concerns when sensitive data managed by applications is moved to cloud infrastructures for processing or storage.

In this paper we have proposed, as a case study of a real-life secure cloud application, a spreadsheet capable of performing DPED operations on cloud servers. The application was tested on a private cloud and on GAE, with analysis of the time required to perform a ciphered FFT operation. Although the test results demonstrate that homomorphic encryption is a feasible solution for secure data processing on cloud infrastructures, the efficiency of current encrypted domain libraries needs to be improved to achieve commercial status. Nevertheless, although the times for encrypted operations are quite long, they are satisfactory for applications with a light processing load, such as the proposed spreadsheet. To apply this model in a PaaS, cloud providers should support DPED libraries on their servers.

This solution could be applied to other real-life applications, such as enterprise resource planning (ERP) or e-Health SaaS, where confidentiality is crucial.

## ACKNOWLEDGEMENTS

This research was supported by the SAFECLOUD grant (09TIC014CT), funded by Xunta de Galicia (Spain), and partially supported by the HIGEA grant (IPT-2012-1218-300000), funded by the Spanish Ministry of Economy and Competitiveness, the PRISMED grant (IPT-2011-1076-900000), funded by the Spanish Ministry of Science and Innovation. This research was conducted with the collaboration of GPSC research group of the University of Vigo, which provided the DPED libraries, and Fundación Barrié.

## REFERENCES

Adkinson-Orellana, L., Rodríguez-Silva, D. A., Gil-Castiñeira, F., and Burguillo-Rial, J., 2010. Privacy for Google Docs: Implementing a Transparent Encryption Layer. In *Proc. of 2nd Cloud Computing International Conference-CloudViews 2010* (pp. 20-21).

Brickell, E. F., Yacobi, Y., 1987. On Privacy Homomorphisms. In *Advances in Cryptology-EUROCRYPT 87* (pp. 117-125). Springer Berlin Heidelberg.

Chan, A. F., 2009. Symmetric-key homomorphic encryption for encrypted data processing. In *Communications, 2009. ICC'09. IEEE International Conference on* (pp. 1-5). IEEE.

CloudLock. [Online]. [Accessed 6 January 2015]. Available from: <http://www.cloudlock.com/>

CryptRoll.2013. [Online]. [Accessed 6 January 2015]. Available from: <http://cryptroll.android.informer.com/>

Data Protection Directive. [Online]. [Accessed 6 January 2015]. Available from: [http://ec.europa.eu/justice/data-protection/index\\_en.html](http://ec.europa.eu/justice/data-protection/index_en.html).

European CIOs and Cloud Services, 2010. [Online]. [Accessed 6 January 2015]. Available from: <http://www.colt.net/cio-research>.

Gentry, C., 2009. Fully Homomorphic Encryption Using Ideal Lattices. In *41st ACM Symposium on Theory of Computing-STOC* (Vol. 9, pp. 169-178).

Hacıgümüş, H., Iyer, B., Li, C., and Mehrotra, S., 2002. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (pp. 216-227). ACM.

LOPD, Ley orgánica 15/1999 de Protección de Datos de Carácter Personal, Boletín Oficial del Estado (in Spanish), 1999. [Online]. [Accessed 6 January 2015]. Available from: <https://www.boe.es/>

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology-EUROCRYPT'99* (pp. 223-238). Springer Berlin Heidelberg.

RDLOPD, Real Decreto 1720/2007, Reglamento de Desarrollo de la LOPD, Boletín Oficial del Estado (in Spanish), 2007. [Online]. [Accessed 6 January 2015]. Available from: <https://www.boe.es/>

Rodríguez-Silva, D. A., González-Castaño, F. J., Adkinson-Orellana, L., Fernández-Cordeiro, A., Troncoso-Pastoriza, J. R., and González-Martínez, D., 2011. Encrypted Domain Processing for Cloud Privacy. Concept and Practical Experience. In *Proceedings of 1st International Conference on Cloud Computing and Services Science-CLOSER 2011*.

Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V., 2010. Fully homomorphic encryption over the integers. In *Advances in Cryptology-EUROCRYPT 2010* (pp. 24-43). Springer Berlin Heidelberg.

ZecurePC. 2011. [Online]. [Accessed 6 January 2015]. Available from: <http://www.zecurex.com/>.

# SLAFM

## *A Service Level Agreement Formal Model for Cloud Computing*

Lucia De Marco<sup>1,2</sup>, Filomena Ferrucci<sup>2</sup> and M-Tahar Kechadi<sup>1</sup>

<sup>1</sup>*School of Computer Science and Informatics, University College Dublin, Belfield, Dublin 4, Ireland*

<sup>2</sup>*Department of Management and Information Technology DISTRA-MIT, University of Salerno,*

*Via Giovanni Paolo II, 132 - 84084 - Fisciano (SA), Italy*

*lucia.de-marco@ucdconnect.ie, fferrucci@unisa.it, tahar.kechadi@ucd.ie*

**Keywords:** Service Level Agreement, SLA Management, Cloud Services Logs, SLA Formal Model.

**Abstract:** Cloud Computing services are regulated by a contract called Service Level Agreement (SLA). It is co-signed between the customers and the providers after a negotiation phase, and during their validity time several constraints have to be respected by the involved parties. Due to their popularity, cloud services are enormously used and unfortunately also abused, specially by cyber-criminals. Sometimes the crimes have the consequence of violating some contractual constraints without the parties are aware of. A manner for guaranteeing more control of the SLA respect is to consider a dedicated system interacting with the cloud services and detecting the SLA violations by analysing the log files. We introduce a formal model aimed to represent the contents of such SLAs necessary in the context of an automatic mechanism for detecting SLA violations.

## 1 INTRODUCTION

Cloud services are regulated by Service Level Agreement contracts (SLA) (Mell et. al, 2011), where all the constraints among a cloud service provider and its customer(s) are detailed; the contracts are co-signed by the parties and they have legal validity in case of a court litigation (Baset, 2012; ?). Because of their importance and contents(CSA, 2013), SLAs have been being considered to be monitored to detect their violations. Many attempts to automate such monitoring there exist in literature, but to the best of our knowledge most of them do not consider cloud services logs; nevertheless the literature focuses on the monitoring of performances constraints.

In this paper a set of rules is presented in the shape of a formal model named SLAFM; we aim to formalize the necessary information needed by the SLA violation detection capability, including input, output, and intermediary computations. In addition, some diagrams will be presented in order to visualize the interactions among the components dedicated to implement our SLA violation detection capability based on our formal model.

The remainder of the paper is structured as follows: in Section 2 we synthesize the background literature about both formal modelling and automatic monitoring of SLAs; Section 3 provides an overview

about contents and structure of cloud SLAs. In Section 4 we discuss the principal objectives of our proposal, while an example of the proposal is illustrated in Section 5. The details of the formal model are explained in Section 6, and a case study based on the previous example is provided in Section 7. Some pro and con are presented in Section 8, and conclusion and future work ideas close the paper in Section 9.

## 2 BACKGROUND

Automating the management of a textual document is a big challenge. Our focus is to target such topic to the SLAs stipulated for cloud services. The final aim is to detect contractual violations. The information involved in this process are the ones concerning cloud services logs. Our approach provides a formal model that represents both SLAs and cloud logs. The analysed literature covers different arguments, such as formal representation of SLAs and automatic SLA monitoring research projects using SLA formal representations, which provide the background of our proposal. Most work provide a customized manner to structure the SLAs contents, which are then formalized via mathematical tuple, e.g. in (Czajkowski et. al, 2002; Unger et. al, 2008; Ghosh et. al, 2012; Ishakian et. al, 2011; De Marco et. al, 2014); in other



works the authors use concepts from set theory, e.g. in (Skene et. al, 2007) and (De Marco et. al, 2014); the other used formalism include derivation rules, reaction rules, integrity rules and deontic rules (Paschke et. al, 2008), or mathematical logic concepts (Unger et. al, 2008).

Other work are dedicated to manage the SLAs in terms of monitoring the respect of some constraints and to prevent their violation; for instance, a framework called DESVI (Emeakaroha et. al, 2010) is dedicated to monitor low level Cloud resources in order to detect if their measured value respects the constraints extracted from SLAs with the goal of detecting QoS violations. This work has been recovered in (Emeakaroha et. al 2012b) to demonstrate its efficiency in monitoring a single Cloud data centre, and also in (Brandic et. al, 2010), where some specific metrics are applied on the resources, whose values are required to match with a specified threshold to prevent possible contractual violations. In (Morshedlou et. al, 2014) a proactive resources allocation prototype is proposed for reducing the negative impact of SLAs' violations and for improving the level of users' satisfaction. In (Maurer et. al, 2012) a prototype for an autonomic SLA enhancement is discussed; it behaves as resource parameters reconfiguration tool at virtual machines level of cloud infrastructures, with the main advantage of reducing SLA violations and of optimizing resources utilization. Instead, in (Emeakaroha et. al, 2012a) the proposed SLA monitoring and violation detection architecture plays at the Cloud application provisioning level, where some metrics are exploited to monitor at runtime the resource consumption behaviour and performance of the application itself. In (Cedillo et. al, 2014) an approach to monitor some Cloud services non-functional SLA requirements is presented; a middle-ware interacting with services and applications at runtime is designed; it analyses the information and provides some reports as soon as an SLA violation is identified. Last but not least, SALMonADA (Muller et. al, 2014) is a platform that utilizes a structured language to represent the SLA, which is then automatically monitored to detect whether any violation occurs or not; such detection is performed by implementing a technique based on a constraint satisfaction problem.

### 3 SLA CONTENTS AND STRUCTURE

In a free trade context people have the freedom to choose for a specific need which services they prefer from a big offer pool. Once such choice is accom-

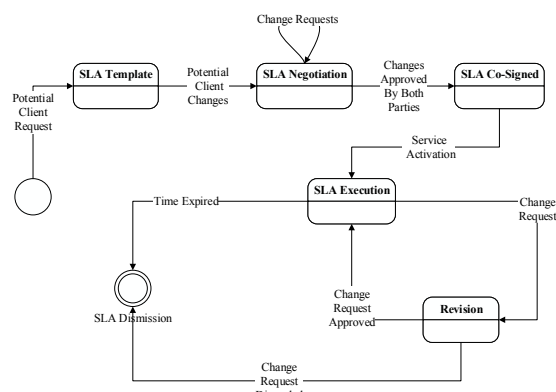


Figure 1: SLA Life Cycle.

plished, a user contacts a specific service provider. This latter is responsible of delivering the service to the user after the instantiation of a particular Service Level Agreement (SLA) contract. An SLA concerning the provisioning of IT services is defined as a *formal, negotiated, document in qualitative and quantitative terms detailing the services offered to a customer* (ITIL). To the best of our understanding among cloud services, an SLA follows the life cycle depicted in the UML (Rumbaugh et. al, 2004) state chart diagram in Figure 1.

After a potential customer request, a contractual template is customized by the cloud service provider depending on some change requests to the standard offer; subsequently a negotiation phase happens, where solutions to the change requests are included, together with information about expenses, penalties, and reports. The SLA co-signed state determines that both entities agree on the actual contractual contents, then the service provisioning can begin. The SLA has a validity time, that can be either explicitly expressed with start and end dates, or an initial date together with a time interval can be included in the document. Such validity time begins after the contract is hardly or digitally co-signed by both parties in the SLA Execution state. During its life cycle, an SLA can be subject to a revision to resolve some change requests instantiated by either party. The revision phase can provide solutions to such requests in order to continue the service provisioning and the contract validity. In case a solution is not met by the parties the service provisioning and the SLA validity are dismissed.

In an SLA regulating cloud services the duties and responsibilities of both parties are described, together with the possible presence and operations of a third party. The main contents of an SLA concern definitions and descriptions of the service, some rules and regulations about its delivering, some performance measures together with possible tolerance intervals about the levels of the services to guarantee,

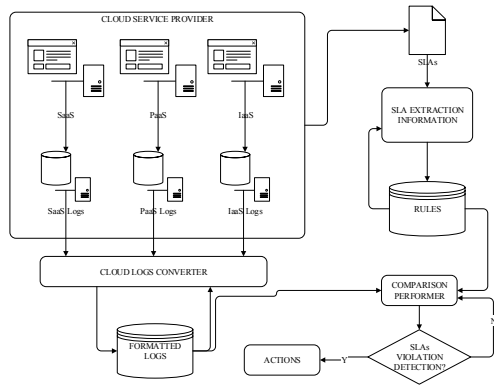


Figure 2: SLA Violation Detection Capability.

and the pricing and penalties measures in case such tolerances are not respected. For sake of monitoring and user satisfaction purposes, it is good practice to provide service levels that can be audited, managed, and measured (Larson, 1998).

## 4 SLA VIOLATION DETECTION

An SLA violation detection capability is aimed to automatically guarantee in a real time manner the respect of the contract in relation to the Cloud behaviour. This capability has two different input. They are the SLA constraints and some Cloud logs. The capability intends to perform the comparison between them. The main challenge for implementing such capability is to convert the natural language expressions of the contract in rules that a machine can manipulate. About the management of the cloud logs, instead, the effort is less, because once the structure of the logs is known they can be manipulated. In most cloud service providers such structure is documented. In Figure 2 the interactions among the components dedicated to implement our SLA violation detection capability is designed. The beginning of such capability corresponds with the SLA to guarantee starting time. The necessary contractual sections are extracted and structured in a specific format. The information gathered from the cloud services logs are translated, aggregated and indexed according to another specific format, compatible with the one the SLA sections are converted to. The capability is responsible to perform the comparison between the SLAs constraints and the cloud service behaviour structured in logs. The comparison results are stored and utilized for subsequent comparison; they also feed a decision making module, responsible to manage the consequences of the detected SLA violations.

## 5 EXAMPLE: AMAZON S3

In this section we want to provide a real-world commercial well defined SLA contract documents as example for the implementation of the SLA violation detection capability depicted in Figure 2.

In order to obtain the necessary logs, we need to access to as many information as possible, among the quantity granted by a cloud service provider. It is well documented that the amount of information accessible by cloud services customers is bigger in IaaS level, medium in PaaS, and small in SaaS (Liu et. al, 2011). Thus, we chose a IaaS type of cloud service. Moreover, we considered Amazon as cloud provider because it is considered by both business and academia as a pioneer of a complete pool of cloud services delivering. From all these considerations, our choice fell on Amazon Simple Storage Service (S3) (Amazon), which is an Infrastructure as a Service (IaaS) provided by Amazon.com.

We accessed to both the S3 public SLA and the logs, available at a customer level. Among all the legal constraints, we consider the service level expressed as Monthly Uptime Percentage. The reader has to consider this parameter as a quality attribute of the Amazon S3 cloud service. *Amazon Web Services will use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage [...] of at least 99.9% during any monthly billing cycle.*

Scanning the SLA, we can find the definition of such attribute as: *Monthly Uptime Percentage is calculated by subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.* Again, Amazon carefully defined the aforementioned Error Rates as *the total number of internal server errors returned by Amazon S3 as error status InternalError or ServiceUnavailable divided by the total number of requests during that five minute period*(Amazon).

In order to guarantee the respect of the monthly uptime percentage service level, an SLA violation detection capability has to consider the information about server responses to the HTTP requests made to S3 every five minutes. Such data is available in the Server Access Log files available in S3, which collects some information every request made to the service. An example of them is provided in Figure 3. The information in red circles are the ones necessary to calculate the Error Rates, i.e., the time and the Error Code.

In a specific five minute period time our capability has to collect the log files for all the requests made, i.e., from  $\text{Time}_0$  to  $\text{Time}_{+5min}$ . Then the Error Rate is

calculated counting the number of Error Code equals to InternalError or ServiceUnavaible, divided by the total number of requests during that five minute period.

Considering that every entry in the log has a related Error Code field, we can affirm that the total number of requests during that five minute period is obtained by the following formula:

$$5minRequests = \sum_0^{+5min} ErrorCode$$

The Error Rate of a five minute time period is obtained by the following formula:

$$5minErrorRate = \left( \sum_0^{+5min} ErrorCode = InternalError \right. \\ \left. OR ServiceUnavaible \right) / 5minRequests$$

In order to have a monthly value the capability has to compute this value along a billing month time period. Every solar day has  $60/5 * 24 = 288$  five minutes time periods; this value has to be multiplied for the number of days of a billing month. Assuming that a billing month is composed of thirty days, we will have  $288 * 30 = 8640$  five minutes time periods. Finally, the monthly uptime percentage will be obtained by the following formula:

$$AverageErrorRate = \left( \sum_0^{8640} 5minErrorRate \right) / 8640 \\ MUP = 100\% - AverageErrorRate$$

## 6 FORMAL MODEL

The SLA Formal Model (SLAFM) is aimed to provide a theoretical approach for a contractual violation detection capability for cloud computing services. A previous draft of the same formalism was discussed in (De Marco et. al, 2014), where only a specific scenario has been taken into account. We extend that formal model by adding the modelling of cloud services logs; according to that, some changes to the modelling of the SLA structure have been added. The model utilizes mathematical formalisms as tuple, set theory, functions (Ben-Ari, 1993).

### 6.1 Service Level Agreement

SLAs are contractual documents written in natural language composed of a set of information structured as service levels. An sla  $l$  is an element of the set of slas  $L$ . An sla is described by a mathematical tuple

composed of a set of service levels  $SL$ , the validity starting time  $t_{start}$  and the validity ending time  $t_{end}$ .

$$L = \{l_1, l_2, l_3, \dots, l_j\}, j \in \mathbb{N} \\ l = \langle SL, t_{start}, t_{end} \rangle; t_{end} \geq t_{start}$$

A service level  $sl$  is an element of the set of service levels  $SL$ . Each  $sl$  has a validity time interval, determined by a starting and an ending time; during this time, some indicators related to a service level attribute for a specific service resource need to be verified, hence an  $sl$  is described by the following tuple, composed of the set of indicators  $I$ , the attribute  $a$  of the resource  $r$ , and the starting and finishing times,  $t_s$  and  $t_f$  respectively.

$$SL = \{sl_1, sl_2, sl_3, \dots, sl_j\}, j \in \mathbb{N} \\ sl = \langle I, a^r, t_s, t_f \rangle; t_f \geq t_s$$

An indicator  $i$  is an element of the set of the indicators  $I$ . An indicator is described by a mathematical tuple composed of a conditioned value  $c k u$  of the attribute  $a^{sl}$  for the resource  $r^{sl}$ . A condition  $c$  belongs to the set of conditions  $C$ ; in case any condition is not expressed in the contractual text, the value of  $c$  will be set as  $=$ . A value  $k$  is related to the attribute  $a^{sl}$  of the resource  $r^{sl}$ ;  $u$  is the optional unit measure used to express the value  $k$ , belonging to the set of unit measures  $U$ . The conditioned value  $c k u$  has to be verified through the application of the metric  $m$  belonging to the set of metrics  $M$ . The metrics can be either atomic or composed, namely the value is obtained by combining more atomic metrics.

$$I = \{i_1, i_2, \dots, i_j\}, j \in \mathbb{N} \\ i = \langle cku, m \rangle$$

$$c \in C; C = \{ \leq; \geq; <; >; <>; = \}$$

$$u \in U; U = \{u_1, u_2, \dots, u_j\}, j \in \mathbb{N}$$

$$m \in M; M = \{m_1, m_2, \dots, m_j\}, j \in \mathbb{N}$$

### 6.2 Cloud Service Log

A cloud computing architecture is composed of a set of resources  $R$ , either physical or virtual.

$$P = \{R^p | R^p \subseteq R\} \\ V = \{R^v | R^v \subseteq R\} \\ R = \{r_1, r_2, r_3, \dots, r_j\}, j \in \mathbb{N}$$

```

Access to S3 file log

Bucket owner - 5927116389e7d406047097a41c8a2ef5830ad74cdaf67351d74682eaaa07ea2b
Bucket - myownlogsbucket
Time - [18/Feb/2015:10:37:23 +0000]
Remote IP - 137.43.248.70
Requester - 5927116389e7d406047097a41c8a2ef5830ad74cdaf67351d74682eaaa07ea2b
Request ID - E730ACFDEBFC0AD6
Operation - REST.GET.OBJECT
Key - myapp/AMSLogs/028578912368/elasticloadbalancing/us-east-1/2015/02/18/028578912368_elasticloadbalancing_us-east-1_MyLoadbalancer_20150218T1000Z_54.85.108.72_dzdvvvy8.log
Request URI - "GET /myownlogsbucket/myapp/AMSLogs/028578912368/elasticloadbalancing/us-east-1/2015/02/18/028578912368_elasticloadbalancing_us-east-1_MyLoadbalancer_20150218T1000Z_54.85.108.72_dzdvvvy8.log?
X-Amz-Date=20150218T103721Z&X-Amz-Expires=300&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Signature=ca71da5a8639895ffae88c18ac998b4a457559041422c5e7855d1dd460fc14d16X-Amz-Credential=ASIAJLWYBG2X77VM6P3Q/20150218/us-east-1/s3/aws4_request&X-Amz-SignedHeaders=Host&x-amz-security-token=AQoDYXdzEEQagARh3dXT/b42BuxVLeigXA/jYaewrCDf6JQFbGDUUpf88cJZylVFjh3yTwhxyAB%2BTM2hiUxQb79Y08Y30lXgUiI026%2BCW42j1623HTLFZOPzOrXMcZPfkfs2ibAcDF7TReTbr7W09rtQQO2CH7MeVdu4pJHhWFOKKQuaPqB0gfCnfOtBtZshBJt8Ki/IgHgttxbau8EWNoXhP58gusf/hD2SLndjJ/P42BQYyMqoXCVJYpiTovUu5LONJmO9ua6Vu7/thJH7JD0JBhJh%2Bxv%2BFfTa3Rz2xU2BfAaRA8ijRqfdpARRPf88%2BMy6pyG9f6yWu4QGLMU0wFuJ92g1leIILXWkacF HTTP/1.1"
HTTP Status - 200
Error code -
Bytes Sent - 331
Object Size - 331
Total time - 43 [ms. Is measured from the time your request is received to the time that the last byte of the response is sent.]
Turn-Around Time - 43 [ms. Is measured from the time the last byte of your request was received until the time the first byte of the response was sent.]
Referer - "https://s3-console-us-standard.console.aws.amazon.com/GetResource/Console.html?region=us-east-1&pageLoadStartTime=1424255801228&locale=en"
User Agent - "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111 Safari/537.36"

```

Figure 3: Amazon S3 Sever Access Log.

A resource  $r$  is described by a set of attributes  $A$ , e.g., filename, date of creation, size. Let  $a$  be an attribute, it belongs to the set of attributes  $A$ .

$$\begin{aligned}
 A &= \{a_1, a_2, a_3, \dots, a_j\}, j \in \mathbb{N} \\
 r &= \{A^r | A^r \subseteq A\} \\
 |A^r| &\neq 0 \forall r \in R
 \end{aligned}$$

**Theorem 1.** The set of attributes  $A$  cannot be empty.

*Proof.* Every resource  $r \in R$  is described by a set of attributes  $A^r$  which is subset of  $A$ .

$$\begin{aligned}
 A &= \bigcup_{r \in R} A^r \Rightarrow |A| = \bigcup |A^r| \\
 &\stackrel{by\,def.}{\Rightarrow} |A^r| \neq 0 \forall r \in R \\
 &\Rightarrow |A| \neq 0
 \end{aligned}$$

□

During the execution of a Cloud service, the value of an attribute of a resource is subject to changes via an operation  $o$ . An operation  $o$  is an element of the set of operations  $O$ . Each operation is described by a mathematical tuple composed of a sender  $s$  that is the executor of such operation, a result  $value(a^r)$  that describes the value assigned to attribute  $a$  of resource  $r$ , an operation resource  $r$ , an attribute  $a$ , and an operation time  $t_o$ . The operation value is a mathematical function that assigns a value to an attribute of a resource. The assigned value can be of numeric type; moreover it can be associated to an optional unit measure  $u$ . A sender  $s$  is an entity (process) performing operations in the Cloud, e.g., an IP address or a Dropbox process. Let  $s$  be a sender, it belongs to the set of senders  $S$ .

$$S = \{s_1, s_2, s_3, \dots, s_j\}, j \in \mathbb{N}$$

$$O = \{o_1, o_2, o_3, \dots, o_j\} j \in \mathbb{N}$$

$$o = \langle s, a^r, value(a^r), u, t_o \rangle$$

$$u \in U$$

$$value : a^r \rightarrow value(a^r) = k \in \mathbb{Z}$$

The information about the sequence of operations are stored in a cloud log  $cl$ , which is an element of the set of cloud logs  $CL$ . Every log is composed of a set of operations  $O^{cl}$  concerning an attribute  $a$  of a resource  $r$ ; the set  $O^{cl}$  is included in the set of all the operations  $O$ .

$$CL = \{cl_1, cl_2, cl_3, \dots, cl_j\}, j \in \mathbb{N}$$

$$cl = O^{cl}; O^{cl} \subseteq O$$

### 6.3 Violation Detection Capability

An SLA violation detection capability is responsible to perform a comparison between an indicator of an attribute for a resource expressed in a service level  $i^{sl} \in I$  and a set of operations of a specific cloud logs  $O^{cl}$  where the sequence of values of such attribute is contained. Such comparison is evaluated according to the metric  $m^i$  used to calculate the value. Formally, the aforementioned entities are correlated in the following tuple. A comparison  $n$  belongs to the set of comparisons  $N$ . A comparison tuple is composed of a cloud log  $cl$ , an indicator that has to be verified  $i^{sl} \in I$ , and the comparison time  $t_n$ .

$$\begin{aligned}
N &= \{n_1, n_2, n_3, \dots, n_j\}, j \in \mathbb{N} \\
n &= \langle cl, i^{sl}, t_n \rangle \\
t_e &\geq t_{ocl} \\
\forall o \in O^{cl}, \forall i \in I^{sl} (a^r)^{O^{cl}} &= (a^r)^{I^{sl}}
\end{aligned}$$

**Definition 1.** SLA Violation

Given a comparison  $n \in N$  and a service level  $sl \in SL$ , the comparison is considered an SLA violation if the values of the set of operations  $O^{cl}$  composing the cloud log  $cl$  on the attribute  $a$  of the resource  $r$  at the time  $t$  are different from the conditioned value  $cku$  of the related indicator  $i^{sl}$  about the service level  $sl$ , on the same attribute  $a$  of the same resource  $r$ . The validity has to be determined during the correct time interval, namely the times of the operations have to be included in the time interval  $t_f(i) - t_s(i)$ ; the value  $value(a^r)_i$  is obtained by the application of the metric  $m^i$ .

$$m^i(a^r) = value(a^r) \neq (cku)^i \quad (1)$$

$$t_s(i) \leq t_{value(a^r)} \leq t_f(i) \quad (2)$$

## 7 CASE STUDY: AMAZON S3

The aim of this section is to map the formal model of the previous section onto the Amazon S3 information illustrated in Section 5.

### 7.1 Service Level Agreement

The Amazon S3 SLA has a validity time that begins when a customer subscribes begins to have access to the S3 service until it decides to terminate. We consider the solar year 2015 as the S3 SLA validity time.

$$L = \{S3\}$$

$$S3 = \langle S3SL, 01/01/2015, 31/12/2015 \rangle$$

Among all the legal constraints, we consider the service level expressed as Monthly Uptime Percentage, MUP for short. *Amazon Web Services will use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage [...] of at least 99.9% during any monthly billing cycle.* We consider February 2015 as the billing month range.

$$a = \text{MonthlyUptimePercentage}$$

$$r = S3server$$

$$S3SL = \{MUP\}$$

$$MUP = \langle I, \text{MonthlyUptimePercentage}^{S3server}, 01/02/2015, 28/02/2015 \rangle$$

$$I = \{i_1\}$$

$$c = \text{atleast} = \geq$$

$$i = \langle \geq 99, MUP_m \rangle$$

The unit measure is not expressed, so we can discard it due it is defined optional.

Recalling the definition of such attribute: *Monthly Uptime Percentage is calculated by subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.* Error Rate is the total number of internal server errors returned by Amazon S3 as error status *InternalError* or *ServiceUnavailable* divided by the total number of requests during that five minute period (Amazon).

The metric for calculating MUP is a composed metric, because it needs the computation of the *AverageErrorRate* that depends on *5minErrorRate*, depending again on *5minRequests*.

$$M = \{MUP_m, \text{AverageErrorRate}_m, \text{5minErrorRate}_m, \text{5minRequests}_m\}$$

$$MUP_m = 100\% - \text{AverageErrorRate}_m$$

$$\text{AverageErrorRate}_m = \left( \sum_{0}^{8640} \text{5minErrorRate}_m \right) / 8640$$

$$\text{5minRequests}_m = \sum_{0}^{+5min} \text{ErrorCode}$$

$$\text{5minErrorRate}_m = \left( \sum_{0}^{+5min} \text{ErrorCode} = \text{InternalError} \right.$$

$$\left. \text{ORServiceUnavaible} \right) / \text{5minRequests}$$

### 7.2 S3 Server Access Log

The SLA violation detection capability collects every five minute period time the S3 Server Access Log files available in S3 where every HTTP request made to the service and its response is stored (see Figure 3). The information from those logs are mapped in our formal model in the following way.

$$R = \{S3server\}$$

$$A = \{\text{MonthlyUptimePercentage}\}$$

$$S3server = \{\{\text{MonthlyUptimePercentage}\}^{S3server}\}$$

During the execution of a Cloud service, the value of an attribute of a resource is subject to change via an operation  $o$ . An operation  $o$  is an element of the set of operations  $O$ . Each operation is described by a mathematical tuple composed of a sender  $s$  that is the executor of such operation, a result  $value(a^r)$  that describes the value assigned to attribute  $a$  of resource  $r$ ,

an operation resource  $r$ , an attribute  $a$ , and an operation time  $t_o$ . The operation value is a mathematical function that assigns a value to an attribute of a resource. The assigned value can be either a numeric or textual; moreover it can be associated to an optional unit measure. A sender  $s$  is an entity (process) performing operations in the Cloud. Let  $s$  be a sender, it belongs to the set of senders  $S$ .

$$S = \{137.43.248.70\}$$

The sender is the Remote IP address field of the log in Figure 3.

$$O = \{REST.GET.OBJECT\}$$

The operation is the Operation field of the same S3 Server Access Log file. The components of the operation are described in the following tuple.

$$REST.GET.OBJECT = \langle 137.43.248.70, \\ MonthlyUptimePercentage^{S3server},, \\ 18/Feb/2015 : 10 : 37 : 23 + 0000 \rangle$$

The value component of the tuple is empty, as well as the unit measure. More precisely, the value component is not either InternalError or ServiceUnavailable.

### 7.3 Violation Detection Capability

In order to build a log  $cl \in CL$  the SLA violation detection capability translates many S3 Server Access Log file, covering an amount of time to be decomposed in five minutes periods. From this mapping, the S3 Server Access Log file operations mapped to cloud log  $O^{cl}$  feed the metric  $MUP_m^i$  in order to determine the elements of the set of comparisons  $N$ .

## 8 DISCUSSION

SLAFM is devoted to formalize a capability to manage SLAs violation detections for cloud services. The proposed approach concerns the representation of information from both the SLAs and the cloud logs in a specific format. One of the strengths of such approach is its extensibility; because the formal model is a high level abstraction, it can be enriched with additional information.

The entity sender is included in this formal model because considered necessary to easy the eventual forensic investigation triggered once a violation happens. The senders are important to reconstruct an events time-line more efficiently and to relate the flow of the operations stored by cloud logs per authors.

An extension of the model can be the necessity of managing some legal principles, thus a specific set of formal rules can be added to the formal model. The comparison among a service level and the operations of the cloud logs have to be memorized because they can be necessary for computing comparisons of other service levels. This depends on how the SLAs relate the service levels and the indicators.

A dedicated system module reacting to the detected SLA violations is out of the SLAFM formal model duties, but it can be considered as a possible extension or integration of both the formal model and the system implementing our capability designed in Figure 2.

## 9 CONCLUSION AND FUTURE WORK

The management of Service Level Agreement contracts for cloud services provisioning is an extremely challenging and active research trend. Several proposals have been made in literature to approach the QoS levels guaranteeing issues described in the contracts with the purpose of monitoring a platform behaviour.

The main objective of such research works is to detect whether the agreed resources performances are respected, without considering the cloud services logs. In some papers, the issue is formally modelled for being subsequently implemented; in other ones, a specific framework is proposed demonstrating the manner how such monitoring is performed.

In the future we intend to prototype a system designed in Figure 2 based on the SLAFM formal model proposed in Section 6. The prototype will simulate cyber attacks to a cloud service regulated by an SLA, and it will perform comparisons among the logs and the SLA constraints. Moreover, we want to test the number of SLA formal rules violations that can be detected in a specific amount of time.

We strongly believe that such capability can enhance the security strategies of a Cloud platform, so that it can be considered as a must requirement for it, and very likely becoming a standard in the next years.

## REFERENCES

- Amazon Simple Storage Service (S3), [online] <http://aws.amazon.com/s3/sla/>, accessed on 22/02/2015.
- Baset, S.A., 2012. Cloud SLAs: present and future. *ACM*

- SIGOPS Operating Systems Review*, vol. 46, no. 2, pp. 57-66.
- Ben-Ari, M., 1993. *Mathematical Logic for Computer Science*, SPRINGER, first edition.
- Brandic, I., Emeakaroha, V.C., Maurer, M., Dustdar, S., Acs, S., Kertesz, A., Kecskemeti, G., 2010. LAYSI: A Layered Approach for SLA-Violation Propagation in Self-Manageable Cloud Infrastructures, *COMPSACW Workshop*, 2010, pp.365 - 370, IEEE.
- Cedillo, P., Gonzalez-Huerta, J., Abrahao, S., Infran, E., 2014. Towards Monitoring Cloud Services Using Models@run.time, *International Workshop on Models at run.time*, to appear.
- CSA, 2013. Mapping the Forensic Standard ISO IEC 27037 to Cloud Computing, *Cloud Security Alliance*, [online] <https://cloudsecurityalliance.org/download/mapping-the-forensic-standard-isoiec-27037-to-cloud-computing/>
- Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S., 2002. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems, *Job scheduling strategies for parallel processing*, Springer Berlin Heidelberg, pp. 153-183.
- De Marco, L., Abdalla, S., Ferrucci, F., Kechadi, M-T., 2014. Formalization of SLAs for Cloud Forensic Readiness, *Proc. ICCSM Conference*, Academic Conferences and Publishing International Limited, Reading, UK, Dr. Barbara Endicott-Popovsky University of Washington, Seattle, USA Edition, pp. 42 - 50.
- Emeakaroha, V.C., Calheiros, R.N., Netto, M.A., Brandic, I., De Rose, C.A., 2010. DeSVi: An Architecture for Detecting SLA Violations in Cloud Computing Infrastructures, *ICST Cloud Comp Conference*.
- Emeakaroha, V.C., Ferreto, T.C., Netto, M.A.S., Brandic, I., De Rose, C.A.F., 2012. CASViD: Application Level Monitoring for SLA Violation Detection in Clouds, *IEEE COMPSAC Conference*, pp. 499 - 508.
- Emeakaroha, V.C., Netto, M.A., Calheiros, R.N., Brandic, I., Buyya, R., De Rose, C.A., 2012. Towards Automatic Detection of SLA Violations in Cloud Infrastructures, *Future Generation Computer Systems*, vol. 28, issue 7, pp. 1017-1029.
- Ghosh, N., Ghosh, S.K., 2012. An Approach to Identify and Monitor SLA Parameters for Storage-as-a-Service Cloud Delivery Model, *GC Wkshps*, pp. 724-729.
- Information Technology Infrastructure Library (ITIL), <http://www.itil-officialsite.com>
- Ishakian, V., Lapets, A., Bestavros, A., Kfoury, A., 2011. Formal Verification of SLA Transformations, *IEEE World Congress on Services*, pp. 540-547.
- Larson, K. D., The role of service level agreements in IT service delivery, *Information Management & Computer Security*, vol. 6, issue 3, pp. 128-132, 1998.
- Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D., 2011. *NIST cloud computing reference architecture*, NIST special publication, 500, 292.
- Maurer, M., Brandic, I., Sakellariou, R., 2012. Self-Adaptive and Resource-Efficient SLA Enactment for Cloud Computing Infrastructures, *IEEE CLOUD Conference*, pp. 368 - 375.
- Mell, P., Grance, T., 2011. *Final Version of NIST Cloud Computing Definition*, [online], <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>
- Morshedlou, H., Meybodi, M.R., 2014. Decreasing Impact of SLA Violations: A Proactive Resource Allocation Approach for Cloud Computing Environments, *IEEE Transactions on Cloud Computing*, vol.2, no.2, pp. 156-167.
- Muller, C., Oriol, M., Franch, X., Marco, J., Resinas, M., Ruiz-Corts, A., Rodriguez, M., 2014. Comprehensive explanation of SLA violations at runtime. *IEEE Transactions on Services Computing*, vol. 7, issue 2, pp. 168-183.
- Paschke, A., Bichler, M., 2008. Knowledge Representation Concepts for Automated SLA Management, *Decision Support Systems*, vol. 46, issue 1, pp. 187-205.
- Patel, P., Ranabahu, A.H., Sheth, A.P., 2009. Service Level Agreement in Cloud Computing, [online], <http://corescholar.libraries.wright.edu/knoesis/78>
- Rumbaugh, J., Jacobson, I., Booch, G. 2004. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.
- Skene, J., Skene, A., Crampton, J., Emmerich, W., 2007. The Monitorability of Service-Level Agreements for Application-Service Provision, *Proc. International Workshop on Software and Performance*, pp. 3-14.
- Unger, T., Leymann, F., Mauchart, S., Scheibler, T., 2008. Aggregation of Service Level Agreements in the Context of Business Processes, *Proc. ICEDOC Conference*, pp. 43-52.

# Towards High Performance Big Data Processing by Making Use of Non-volatile Memory

Shuichi Oikawa

*University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, Japan  
shui@cs.tsukuba.ac.jp*

**Keywords:** Non-volatile Memory, Operating Systems, Storage, Big Data Processing.

**Abstract:** Cloud computing environments for big data processing require high performance storage. There are emerging high performance memory storage technologies, such as next generation non-volatile (NV) memory and battery backed NV-DIMM. While their performance is much higher than the current block storage devices, such as SSDs and HDDs, they provide only limited capacity. Such limited capacity makes it difficult for memory storage to be adapted as mass storage, and their uses in cloud computing environments have been severely limited. This paper proposes a method that combines memory storage with block storage. It makes use of memory storage as cache of block storage in order to remove the capacity limitation of memory storage. The proposed method inherits the high performance of memory storage and also the large capacity of block storage. Therefore, memory storage can be transparently used as a part of mass storage while its low overhead access can accelerate storage performance. The proposed method was implemented as a device driver of the Linux kernel. Its performance evaluation shows that it outperforms a bare SSD drive and achieves better performance on the Hadoop and database environments.

## 1 INTRODUCTION

The importance of big data processing increases more than ever before, and it is convincing that its importance will continue increasing in the future as well. Cloud computing environments are currently the only solution that can provide the scalability required by big data processing since they can scale out their storage capacity along with necessary computing resources. There is no doubt that cloud computing environments for big data processing require high performance storage; thus, SSDs were quickly adapted in such environments, and they are sometimes combined with HDDs to transparently enhance the performance and capacity of storage.

Now, high performance memory storage technologies, such as next generation non-volatile (NV) memory and battery backed NV-DIMM, are emerging. These new kinds of storage provide both high performance and persistency, and they are byte addressable. Since their byte addressability enables them to be accessed as memory, we call them *memory storage*. While they provide much higher performance than the current block storage devices, such as SSDs and HDDs, their capacities are limited. Such capacity limitation makes it difficult for memory storage to

be adapted as mass storage, and their uses in cloud computing environments have been severely limited.

This paper proposes a method that combines memory storage with block storage. It makes use of memory storage as cache of block storage in order to remove the capacity limitation of memory storage. Combining block storage with another faster block storage, which is typically an SSD, for higher access performance is a well known technique (Kgil and Mudge, 2006; Koller et al., 2013; Saxena et al., 2012). The technique utilizes faster block storage as cache and stores frequently accessed data in it in order to improve the average time to access data. Its open source implementation is widely available (Facebook, 2014). The existing technique employs a software layer that combines two block storage devices. Since it is possible for memory storage to emulate block storage and to use the software layer for combining block storage devices, the emulation sacrifices its performance advantage for the compatibility with the block storage interface.

The proposed method directly manages memory storage in order to make use of its high performance and byte addressability. The byte addressability of memory storage enables its direct management without a device driver; thus, the memory storage man-



agement can be integrated in a device driver that combines memory storage with block storage without an additional software layer as required by the existing method. It can effectively utilize the high performance of memory storage and also provides the large capacity of block storage. Therefore, memory storage can be transparently used as a part of mass storage while its low overhead access can accelerate storage performance.

The proposed method was implemented as a device driver of the Linux kernel, and its performance evaluation was performed by measuring the file access performance on the Hadoop distributed processing environment and also a typical benchmark performance on the MySQL database environment. Hadoop and MySQL were employed for the measurements in order to evaluate the effectiveness of the proposed method in realistic environments. The measurements were performed on a virtualized environment. The evaluation results show that the proposed method considerably outperforms a bare SSD drive and achieves better performance on the Hadoop and database environments.

The rest of this paper is organized as follows. Section 2 describes the background of the work. Section 3 describes the detailed design and implementation of the proposed method. Section 4 shows the result of the experiments. Section 5 describes the related work. Section 6 summarizes the paper.

## 2 BACKGROUND

This section describes the background of this work, which includes the overview of the block device driver layer of the operating system (OS) kernel and the existing method to combine block storage devices.

### 2.1 Block Device Driver Layer

The current storage devices, such as SSDs and HDDs, are block devices, and they are not byte addressable; thus, CPUs cannot directly access the data on these devices. A certain size of data, which is typically multiples of 512 byte, needs to be transferred between memory and a block device for CPUs to access the data on the device. Such a unit to transfer data is called a block.

The OS kernel employs a file system to store data in a block device. A file system is constructed on a block device, and files are stored in it. In order to read the data in a file, the data first needs to be read from a block device to memory. If the data on memory was modified, it is written back to a block device. A

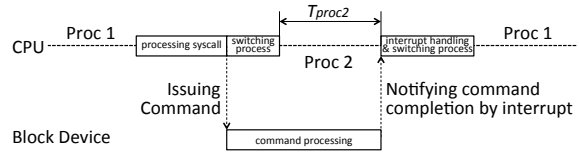


Figure 1: The asynchronous access command processing and process context switches.

memory region used to store the data of a block device is called a page cache. Therefore, CPUs access a page cache on behalf of a block device.

Since HDDs are orders of magnitude slower than memory to access data on them, various techniques were devised to amortize the slow access time. The asynchronous access command processing is one of commonly used techniques. Its basic idea is that a CPU executes another process while a device processes a command. Figure 1 depicts how it works. Process 1 issues a system call to access data on a block device. The kernel processes the system call and issues an access command to the corresponding device. The kernel then looks for the next process to execute and perform context switching to Process 2. Meanwhile, the device processes the command, and sends an interrupt to notify its completion. The kernel handles the interrupt, processes command completion, and performs context switching back to Process 1.  $T_{proc2}$  is a time left for Process 2 to run. Because HDDs are slow and thus their command processing time is long,  $T_{proc2}$  is long enough for Process 2 to proceed its execution.

The I/O request queueing mechanism that implements the asynchronous access command processing has been a right choice for the block devices. It poses high processing cost, but the cost pays off by creating additional processing times made available for other processes. Such justification for the I/O request queueing mechanism and the asynchronous access command processing is, however, no longer true when storage becomes much faster.

### 2.2 Problems to Combine Memory Storage with Block Storage

The existing method combines block storage with another faster block storage, which is typically an SSD, for higher access performance (Kgil and Mudge, 2006; Koller et al., 2013; Saxena et al., 2012). It utilizes faster block storage as cache and stores frequently accessed data in it in order to improve the average time to access data. Its open source implementation is widely available (Facebook, 2014), and the current Linux kernel includes several implementations, such as dm-cache and bcache.

The implementations of the existing method in the Linux kernel employ the device mapper mechanism to constitute a single storage device. The device mapper is implemented as a software layer in the kernel, and provides the mechanism to transfer access requests for the constituted device to appropriate underlying devices. The policy part defines how it transfers requests. There can be multiple policy implementations, and some of them combine block storage with faster storage as cache. When an SSD is used as cache storage by combining it with a HDD, it is straightforward that the combined storage provides the block storage interface and is accessed asynchronously since both of them are block storage. As its extension, it is possible for memory storage to emulate block storage and to have the device mapper to combine block storage with memory storage.

The use of the device mapper requires memory storage to emulate block storage interface since the device mapper expects it as an interface. While such emulation enables the use of the device mapper, it causes significant software overhead. The device mapper is basically a block device driver; thus, it receives access requests from the upper generic block device driver framework. It then transfers the received requests to another block storage device. The transferred requests are processed again by the generic block device driver framework, and finally the target block storage device receives them (Ueda et al., 2007). Therefore, processing in the generic block device driver framework occurs multiple times, and such processing causes a software overhead that can be hidden in the long access latency of block storage devices but becomes apparent for memory storage.

### 3 DESIGN AND IMPLEMENTATION OF THE PROPOSED METHOD

This section first describes the design of the proposed method. It next describes the implementation in the Linux kernel.

#### 3.1 Design Overview

The most considerable advantage of memory storage is its performance. In order to make use of it as much as possible and not to sacrifice it, the software overhead to access it must be minimum. The existing method to combine block storage devices is, however, inappropriate in this sense because of its inefficiency that is inherent in its use of the block storage interface

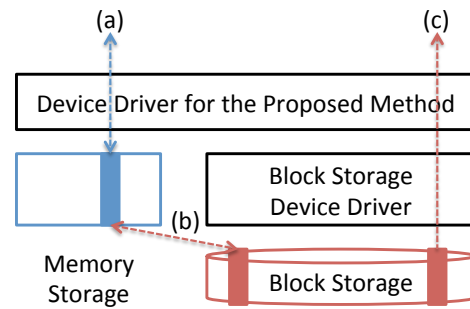


Figure 2: The design overview of the proposed architecture.

and its asynchronous access. As described in Section 2.1, the overhead of the block storage interface and its asynchronous access is significant, and it must be avoided.

The proposed method keeps its access overhead to memory storage minimum by making use of the direct and synchronous access to memory storage. Memory storage provides the memory interface, which means that there is no need to use a device driver to access it; thus, the device driver of the proposed method can directly access memory storage. Such direct access allows the least access overhead to memory storage. Moreover, because memory storage allows synchronous access, of which software overhead is much less than asynchronous access, the proposed method aggressively makes use of synchronous access.

Figure 2 depicts the design overview of the architecture of the proposed method. First, we consider reading data. There are three access paths, which are shown as (a), (b), and (c) in the figure. When data is available on memory storage, which is shown as (a), the device driver of the proposed method provides direct and synchronous access to memory storage. Such access enables the least overhead; thus it should be utilized as much as possible. In order to make it possible, data needs to be read ahead from block storage to memory storage, which is shown as (b). When reading ahead is successful, data can be continuously read from memory storage.

When data is not available on memory storage, a straightforward way is to read in the requested data from block storage to memory storage. This way, however, unnecessarily pollutes memory storage because the data that was read in to memory storage becomes useless. Therefore, the proposed method reads the data from block storage bypassing memory storage, which is shown as (c).

Second, we consider writing data. There are also three data paths, (a), (b), and (c), which are shown in the figure. Unlike reading, there is no need to read ahead into memory storage for writing since

valid data is written onto memory storage. Therefore, data can be written onto memory storage whenever free spaces are available on memory storage, which is shown as (a). The free spaces can contain valid data for reading. Unavailable spaces of memory storage are those where dirty data resides. The unavailable spaces that contain dirty data become free spaces when the dirty data is written back to block storage, which is shown as (b). When there is no free space available, data can be written to block storage, which is shown as (c). The path (c) is, however, considered to be rarely used since writing to memory storage and writing back to block storage can be processed in parallel.

The device driver of the proposed method manages memory storage and also interacts with a block storage device driver. A block storage device driver is not a part of the driver of the proposed method. By separating the management of memory storage and block storage, there is no restriction of a choice of block storage, and arbitrary block storage can be combined with memory storage.

### 3.2 Implementation in the Linux Kernel

The Linux kernel provides the device mapper mechanism, which can be used to combine multiple block storage devices. The existing method uses this mechanism as described in Section 2.2. The proposed method, however, does not use the device mapper mechanism in order to avoid the overhead of itself and also the overhead incurred by having memory storage emulate block storage.

The proposed method implements its own function that can provide the synchronous access interface depending upon the location of requested data.

```
void memory_make_request(
 struct request_queue *q,
 struct bio *bio)
```

This interface is typically used by the device driver of ramdisk, which provides synchronous access. The proposed method makes use of this interface and provides synchronous access when data is available on memory storage for reading or when a free space is available on memory storage for writing. In this case, memory storage is considered to be working just as ramdisk. When data is unavailable on memory storage for reading or when no free space is available on memory storage for writing, however, the access request is transferred to the device driver of block storage. Then, the block storage device driver asynchronously processes the request.

The device driver of the proposed method also implements the functions for reading ahead and writing

back data between memory storage and block storage. Because they need to be invoked in parallel with reading and writing data from/to memory storage, the dedicated kernel threads process them. They are invoked at appropriate timings in order to improve the efficiency of the proposed method.

## 4 EXPERIMENT RESULTS

First, file I/O throughput was measured by using the Hadoop TestDFSIO benchmark program to see performance impact on big data processing. The measured costs are compared with a sole SSD drive. Second, the performance of the database processing was measured by the TPCC-MySQL benchmark program to see performance impact on database processing.

### 4.1 Experiment Environment

Since there is no publicly available system that equips memory storage, we used DRAM to emulate it. Since MRAM, which is considered to be the best match for the proposed method, performs comparably to DRAM, the differences of results must be negligible. All measurements described below were performed on the Linux kernel 3.14.12 that includes the implementation of the proposed method. Execution times were measured using the TSC (Time Stamp Counter) register.

The system used for this experiment is a PC system equipped with the Intel Core i7-4930K 3.4GHz and 64GB of DRAM. The KVM virtualization software of Linux is employed to construct experiment environments that consist of virtual machines. Each virtual machine is configured with two CPUs, the main memory, and a dedicated block storage device. The sizes of the main memory differ to match their functionality, they are described below. The CFD S6TNHG6Q 128GB SATA SSD is used for a dedicated block storage device, and a whole SSD is assigned to a single virtual machine. When the proposed method is used for an experiment, memory storage consists of 1GB of memory.

### 4.2 Results of Hadoop TestDFSIO

This section shows the measurement results of the Hadoop TestDFSIO benchmark program. For this experiment, four virtual machines were configured to be a Hadoop cluster. One virtual machine becomes the master node, and the others are slave nodes. The main memory size of the master node is 8GB, and that of

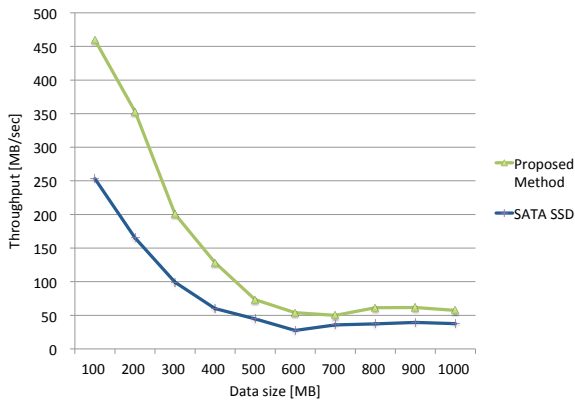


Figure 3: Comparison of read performance by Hadoop TestDFSIO.

slave nodes is 1GB. Hadoop employs Hadoop Distributed File System (HDFS) for the file service of its applications (Shvachko et al., 2010). The HDFS servers consist of the name node and data nodes, which are executed as user processes. The master node runs the name node, which locates on which data node requested data files reside upon access requests from clients. The data nodes of slave nodes manage data files.

We measured the file I/O throughput of reading files of various data sizes by using the Hadoop TestDFSIO benchmark program. Larger numbers are better as I/O throughput. The size of each file created for measurements was fixed to 100MB, and the number of files was changed from one to ten in order to change the total data sizes from 100MB to 1GB. We first executed the writing program of TestDFSIO to create files for reading. After flushing the page cache of the data nodes, we executed the reading program of TestDFSIO, and measured the costs. HDFS provides two methods for reading. One receives data from a data node through remote procedure calls (RPCs), and the other directly interacts with a local file system. The latter one is called short circuit read (SCR). Both methods were used for measurements. Figure 3 and 4 show the results without and with SCR enabled, respectively.

The measurement results show a significant performance advantage of the proposed method for the Hadoop TestDFSIO. For reading from 100MB to 1GB data sizes without SCR, it performs approximately 39.21% to 114.36% better than SSD. For reading with SCR enabled, it performs approximately 135.32% to 624.10% better than SSD. On average, the proposed method performs 78.45% better without SCR and 266.08% better with SCR than SSD.

A realistic evaluation with Hadoop shows that the proposed method provides a significant boost with the

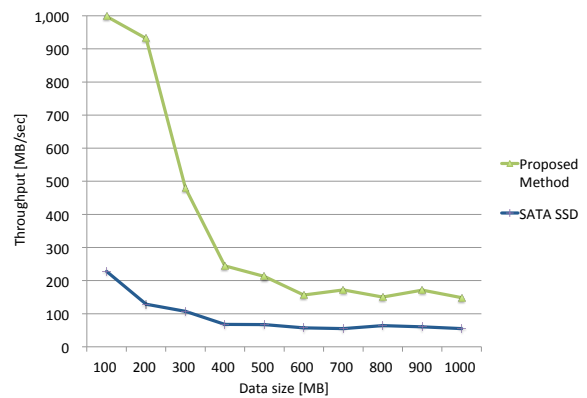


Figure 4: Comparison of read performance by Hadoop TestDFSIO with SCR enabled.

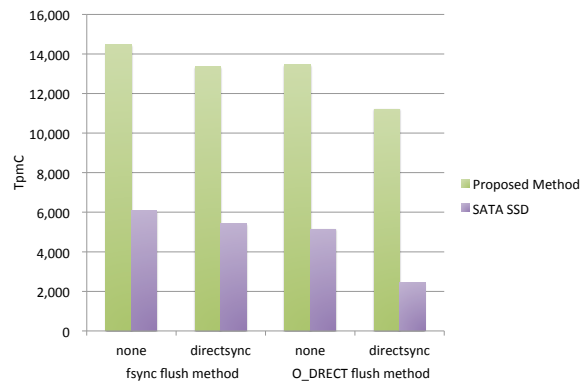


Figure 5: Comparison of TPCC-MySQL performance.

file access throughput of Hadoop. Therefore, it is certain that a wide range of applications, which involves a large amount of file access, can benefit from it.

### 4.3 Results of TPCC-MySQL

This section shows the measurement results of the TPCC-MySQL benchmark program. For this experiment, a single virtual machines with 8GB main memory was configured. The number of warehouses is 40, which constitute approximately 4GB of a database. The buffer pool size of the InnoDB storage engine is 4GB. The measurements were performed with the two flush methods of InnoDB and the two storage cache modes of KVM; thus, there are the four combinations of them. The InnoDB flush methods used for the measurement are fsync and O\_DIRECT, and the KVM storage cache modes are none and directsync. The none cache mode provides the write buffer while the directsync cache mode does not. Figure 5 shows the results.

The performance improvement enabled by the

proposed method is significant. The proposed method executes the benchmark from 2.39x to 4.60x faster than SSD. The difference between the proposed method and SSD is the largest when the combination of the O\_DIRECT Innodb flush method and the KVM directsync cache mode is used. Since this combination provides no buffering of data transfer in the OS kernel and the KVM virtualization software, the cost to write data in storage becomes the maximum among the combinations used for the experiments. The other combinations provide buffering somewhere in the OS kernel and the KVM virtualization software; thus, the differences are closer but still large, which are from 2.39x to 2.62x.

## 5 RELATED WORK

A technique to combine block storage with another block storage for higher access performance existed before SSDs become widely available and popular. DCD (Hu and Yang, 1996) first stores data in cache storage, so that it can make use of sequential access, of which performance is typically much better than random access, so that the write performance can be improved. The emergence of SSDs stimulated the research and development of various caching techniques (Kgil and Mudge, 2006; Koller et al., 2013; Saxena et al., 2012; Facebook, 2014) in order to make use of their high performance. Because SSDs are block storage, all of them combine block storage with another block storage, and provide the block storage interface. The proposed method is different from them since it combines memory storage with block storage. Because memory storage allows synchronous access, the proposed method aggressively makes use of it in order to reduce the access cost in total.

The Linux kernel provides the device mapper as the software layer to combine multiple storage devices and to constitutes a single storage device. When the device mapper is used to combine memory storage with block storage, it requires memory storage to emulate block storage since the device mapper can interact only with the block storage interface. It also causes significant software overhead since the access requests can be processed by the generic block device driver framework multiple times (Ueda et al., 2007). The proposed method does not use the device mapper mechanism in order to avoid such overheads, and implements its own function that can provide the direct and synchronous access interface to memory storage.

## 6 SUMMARY

Memory storage technologies are emerging, and they should be effectively utilized in cloud computing environments in order to accelerate storage performance for big data processing. This paper proposed a method that combines block storage with memory storage and makes use of memory storage as cache of block storage in order to remove such limitation. The proposed method effectively utilizes the high performance of memory storage and also provides the large capacity of block storage. Therefore, memory storage can be transparently used as a part of mass storage while its low overhead access can accelerate storage performance. The proposed method was implemented as a device driver of the Linux kernel. Its performance evaluation shows that it outperforms a bare SSD drive and provides better performance on the Hadoop and database environments.

## REFERENCES

- Facebook (2014). Flashcache. <https://github.com/facebook/flashcache>.
- Hu, Y. and Yang, Q. (1996). Dcd – disk caching disk: A new approach for boosting i/o performance. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 169–178.
- Kgil, T. and Mudge, T. (2006). Flashcache: A nand flash memory file cache for low power web servers. In *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '06, pages 103–112, New York, NY, USA. ACM.
- Koller, R., Marmol, L., Rangaswami, R., Sundararaman, S., Talagala, N., and Zhao, M. (2013). Write policies for host-side flash caches. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies*, FAST'13, pages 45–58, Berkeley, CA, USA. USENIX Association.
- Saxena, M., Swift, M. M., and Zhang, Y. (2012). Flashtier: A lightweight, consistent and durable storage cache. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 267–280, New York, NY, USA. ACM.
- Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA. IEEE Computer Society.
- Ueda, K., Nomura, J., and Christie, M. (2007). Request-based device-mapper multipath and dynamic load balancing. In *Proceedings of the Linux Symposium*, volume 2, pages 235–243.

# The Docker Ecosystem Needs Consolidation

René Peinl and Florian Holzschuher

*Institute of Information Systems, Hof University, Alfons-Goppel-Platz 1, Hof, Germany  
{rene.peinl, florian.holzschuher2}@iisys.de*

**Keywords:** Cloud Computing, Management Tools, Micro-services, System Integration, Docker, Container.

**Abstract:** Docker provides a good basis to run composite applications in the cloud, especially if those are not cloud-aware, or cloud-native. However, Docker concentrates on managing containers on one host, but SaaS providers need a container management solution for multiple hosts. Therefore, a number of tools emerged that claim to solve the problem. This paper classifies the solutions, maps them to requirements from a case study and identifies gaps and integration requirements. We conclude that the Docker ecosystem could help moving from IaaS and PaaS solutions towards a runtime environment for SaaS applications, but needs consolidation.

## 1 INTRODUCTION

Although lightweight operating system (OS) virtualization techniques like Solaris Zones and OpenVZ are long established, it was the release of Docker in March 2013 (Rosen, 2014) that led to mass adoption and even a hype around containerization (Kratzke, 2014). Docker aims at making container technologies easy to use and among other things encourages a service-oriented architecture and especially the micro-service architecture style (Turnbull, 2014). Containers impose less overhead than machine virtualization but still provide less isolation (Scheepers, 2014). In a Software as a Service scenario (SaaS), you therefore cannot guarantee that activities of one customer won't negatively affect other customers, if you are using containers only.

The need for a kind of "application package format" as a basis for composite SaaS offerings following the SOA principles (service-oriented architecture (Papazoglou, 2003)) was already discussed in (Mietzner et al., 2008). Originating from a Platform as a Service (PaaS) use case, Docker should be a good basis to handle the components of a composite application offered in the cloud. It provides an easy and convenient way to create, deploy and configure containers (Rosen, 2014), incl. links to dependent containers on the same host. Micro-services can be briefly summarized as a "loosely coupled service oriented architecture with bounded contexts" (Cockcroft, 2014), where loosely coupled denotes that each service should be

independently deployable and bounded contexts means that the service does not have to know anything about its surroundings, but can discover them on its own (cf. Evans, 2003). Enterprise applications are typically complex composite applications, which consist of multiple individual components (Binz et al., 2014) and therefore match micro-services. However, the Docker tools soon reach their limits when it comes to managing containers in a cluster or creating links across multiple hosts (Kratzke, 2014). To overcome those, a myriad of tools is currently in development. We found over 60 tools in "The Docker Book" (Turnbull, 2014), a special issue of the German "developer magazine" dedicated to Docker (Roßbach, 2014) and the "Docker ecosystem" mindmap ("Docker Ecosystem Mindmap", n.d.) with relevance for building an automated Docker cluster solution similar to OpenStack on the IaaS level (Peinl, 2015 for a full list). Docker Inc even counts 50,000 third-party projects on GitHub that are using Docker (Docker, Inc., 2014). Our hypothesis is, that despite the benefits of competition, the time has come to work together on a common cluster project similar to OpenStack to form a comprehensive integrated solution and stable interfaces for the required components in order to make them interchangeable, instead of building yet another tool that solves parts of the challenges, but not all of them and is not integrated with others.

Our methodology was guided by (Chauhan and Babar, 2011), so the rest of the paper is structured as follows. We briefly describe our SaaS project that

serves as a case study to derive requirements. We continue listing and explaining the requirements and then compare them to the functionality of existing tools. We categorize those tools and elaborate on consistent definitions for those categories. We conclude with remaining challenges and an outlook.

## 2 CASE DESCRIPTION

The goal of the SCHub project (Social Collaboration Hub, funded by the BMBF as part of the FHprofUnt funding, <https://www.sc-hub.de>) is to develop a distribution-like collaboration solution based on open source software (OSS) that provides end-users with a consistent experience across all systems while using a modular micro-service approach (Cockcroft, 2014). It therefore represents a composite application (Coffey et al., 2010). The solution will be available as Software as a Service (SaaS) in the cloud as well as on premise installation. In order to do that, a number of well-known OSS systems have to be migrated to the cloud and Docker is an obvious choice for supporting that. Since not all systems are capable of handling multiple tenants and customization possibilities are better that way, SCHub uses individual instances of all frontend systems (portal, groupware, ...) per tenant and only shares backend systems across tenants (database, mail server, ...). Each instance is packaged into a Docker container. To guarantee isolation between instances of different tenants, virtual machines (VM) are used additionally. The VM becomes the Docker host in this case. OpenStack serves as the basis (Sefraoui et al., 2012). Initially, there is only one VM per tenant. When resource limits of this VM are reached, additional VMs are allocated and some containers are migrated to a new host. Storage is provided by Ceph, a software defined storage solution (Koukis, 2013) as either block-level or object storage, depending on the requirements of the service.

Since off-the-shelf systems are used that are integrated with our add-ons, we wanted to change those systems as little as possible in order to stay upwards compatible and benefit from future releases. Therefore, the usage of a PaaS platform was not feasible as it would require adapting the systems to that platform. However, many components of a PaaS solution are still needed, e.g. a load balancer, a central authentication system, database as a service and so on. It turned out, that a new category of cloud offering would be ideal for this case, a kind of runtime environment for SaaS applications (RaaS). Where PaaS targets developers, RaaS targets application

administrators. The following chapter lists the requirements for such a solution.

## 3 REQUIREMENTS

From a provider's perspective, automating the management of the offered services is of vital importance, because management and operation of IT is one of the biggest cost factors today (Binz et al., 2014). Many of the required features are simply a transfer of IaaS management features to Docker. There should be a central list of containers (r1) with an overview of resource usage, IP address, open ports, dependencies and so on. You need a detail view of a container (r2) including a way to change the configuration using the Web UI concerning networking, storage and dependencies. Since the application is built from multiple services and therefore containers, it would be helpful to be able to centrally define a kind of blueprint (r3) that includes all the dependencies and to instantiate the whole solution instead of single containers (r4).

For doing so, the management solution should monitor resource usage of hosts (r5) and automatically choose one with free resources, based on an editable placement strategy (r6). Monitoring should include CPU, RAM, storage and networking, as well as application health. You should be able to configure thresholds so that high CPU utilization over a specified timeframe or low available memory trigger an alert (r7) which in turn can trigger an action like migrating a container to another host. Migration (r8) could be performed by stopping the container, unmounting the storage, starting an identical container on a new host, updating service references (r8b) and mounting the storage (r9) there. Besides storage, there should also be an easy way to pass configuration data to the application inside the container (r10). This data has to be stored in a distributed key/value store (r8a).

For communication between containers across hosts (r11), you ideally need an overlay network or a software defined network (SDN) (Jain and Paul, 2013). Its configuration should be accessible directly from the Docker management UI, e.g., for defining IP address ranges (r12). The Web UI of the SDN could be simply integrated. You need routing of external requests with URLs to tenant-specific container IPs (r13). This routing should include load balancing if multiple container instances are available (r14). There should be a way to review the list of available images (r15) including versions and ideally an association to the containers running that image. If an image is

updated, the admin should be able to trigger a mechanism that propagates the updates to the running instances (r16), e.g. analogous to the migration described above. There should be a way to access the container's console or open an SSH shell respectively (r17) and review the log files (r18), both using the Web UI.

It would also be desirable to have an integration of the underlying IaaS solution, so that you can create new hosts (VMs) from within the Docker Web UI (r19). Finally, there should be an integration with a tenant / customer management solution (r20) where both administrators and customers can review information like the list of Docker containers per tenant, the resulting resource usage, the number of total and monthly active users as well as respective billing information. It could be further argued, that an authentication solution is needed, but we are skipping this requirement, since Docker itself currently has no working mechanism for that anyway.

## 4 EXISTING SOLUTIONS

We've concentrated our analysis on open source components, although there are a few impressive commercial tools available like StackEngine. The descriptions of the tools capabilities are based on the projects' websites. We have installed and tested only the most promising systems.

### 4.1 Host Operating System

In principle, Docker can run on any modern Linux system. However, a few specialized Linux distributions have emerged that propose to bring exactly what is needed to smoothly run Docker containers and nothing more. CoreOS is the most prominent one and was launched briefly after Docker. Redhat has reacted quickly and initiated project Atomic, which is developed in close cooperation with Redhat's own PaaS solution OpenShift. Canonical has only recently announced an own solution in this field called snappy Ubuntu core. It abandons traditional package managers and uses snappy, a new tool tailored for containerized apps. Boot2Docker is based on Tiny Core Linux and seems to address developers more than cloud hosters as it provides Windows and Mac OS X integration. OpenStack ships with CirrOS as a minimal image for virtual machines. However, CirrOS brings no Docker integration by default.

### 4.2 Image Registry

Docker uses layered images as a package format. Similar to a disk image of a virtual machine, the Docker image contains all the files necessary to run the container and do something meaningful. The image registry stores them and can be used to retrieve an image, if it is not already present on the host (r15). Docker Inc. provides a public image registry called Docker Hub (<https://hub.docker.com>) and an open source implementation for running a private registry. It is not a service registry (see service discovery). Dogestry is an alternative implementation using Amazon S3 compatible storage as a backend. The OpenStack counterpart of this category is Glance.

### 4.3 Container Management

Docker itself only provides a command-line interface (CLI) and a RESTful API for managing containers. This is fine for scripting and automating things, but there is still a need for a Web UI (r1, r2), e.g. for self-service administration by a customer. As the name implies, DockerUI provides exactly that missing WebUI for Docker, while the other candidates in this category provide additional functionality like management of composite applications (Panamax, r3) or broader management of containers and VMs (mist.io and Cockpit, r19). Direct terminal access to the containers via Web UI (r17) is currently under development by mist.io and already implemented by Rancher (see section 4.10). The OpenStack counterpart is Horizon.

### 4.4 Cluster Management

While Docker itself can only list and manage containers of a single host, a cluster management solution should allow the management of a cluster of Docker hosts and all containers on them, including the resource-aware placement of new containers (r6), automatic failover and migration of containers due to resource bottlenecks (Mills et al., 2011). We found four solutions providing parts of this functionality incl. a CLI (Apache Brooklyn, Citadel, CoreOS fleet and Docker Swarm). Decking is similar, but has additional orchestration capabilities (r3). Apache Mesos was originally dedicated to hosting solutions like Hadoop and Spark. Since version 0.20 it also supports running Docker containers. Other solutions like Shipyard build on them and provide a Web UI (r1, r2). Clocker additionally provides some orchestration (r3, r4) and networking functionality (r11), so that it is getting close to the management



Table 1: Overview of Docker software tools with fulfilled requirements (parentheses means partly fulfilled).

Software	Requirements	Software	Requirements
<b>Image Registry</b>		<b>Service Discovery</b>	
Docker Hub	15	DoozerD	14
Dogestry	15	etcd	8a
<b>Container Mgmt</b>		Registrator	8b
Cockpit	1, 2, 19	SkyDNS	8a
DockerUI	1, 2	SkyDock	8b
mist.io	1, 2, 7, 19, (17, 18)	WeaveDNS	8a
Panamax	1, 2, 3	Zookeeper	14
<b>Cluster Management</b>		<b>Software Defined Network</b>	
Brooklyn	(6)	Flannel	11, 12
Citadel	(6)	Open vSwitch	11
Clocker	3, 4, 11	Pipework	11, 12
Decking	3	Socketplane	11, 12
Fleet	(6)	Weave	11, 12
Flocker	3, 4, 6, 9, (11)	<b>Load Balancer</b>	
Mesos	(6)	HAProxy	13, 14
Shipyard	1, 2	nginx	13, 14
Swarm	(6)	Vulcan	13, 14
<b>Orchestration</b>		<b>Monitoring</b>	
Compose	(3, 4)	cAdvisor	(18)
Crane		Grafana	(18)
Fig		Heapster	(18)
Helios	3	Kibana	18
Maestro		logstash	18
Maestro NG	3	<b>Management Suites</b>	
Shipper	3	CF BOSH	3, 4, 6, (9, 18)
Wire	3, 11	Flocker	3, 4, 6, 8, 9
<b>Service Discovery</b>		Kubernetes	3, 4, 6, 8
confd	10	Rancher	1, 2, (3, 4, 6, 9), 17, 18
Consul / Consul UI	8a	OpenStack Docker Driver	1, 2, 3, 4, 6, 19, (9, 11)
dnsmasq	8a		

suites (see section 4.10). Flocker doesn't provide a Web UI but also has additional functionality like basic orchestration and networking. It stands out due to its unique solution of linking storage to containers in a portable way (r9). Nova is kind of fulfilling this cluster management job in OpenStack, especially the Nova scheduler.

## 4.5 Orchestration

Service orchestration is an important feature for composite applications in an SaaS offering. When different components are deployed on different hosts to meet the scalability requirements, those separate deployments should appear as a single coherent subsystem to other components (Chauhan and Babar, 2011). BPEL and WSCI are examples of orchestration languages in SoA (Bucchiarone and Gnesi, 2006). Docker orchestration solutions mainly use YAML instead. Orchestration tools should be

able to add links between Docker containers that are distributed across multiple hosts (r3). Some tools found in literature like Crane, Fig and Maestro (formerly Dockermix) are not able to do that and concentrate on single hosts. The developers of Helios, Maestro NG and Shipper all decided not to build upon cluster management solutions and instead connect to the different hosts on their own. All three come without a Web UI. Shipper seems to be the least mature of the three. Wire is an interesting tool, as it builds on Fig as well as Open vSwitch and dnsmasq to configure interdependent containers across hosts. The OpenStack counterpart of this category is Heat.

## 4.6 Service Discovery

Service discovery has always been an issue in SOA and has never been solved satisfactory in practice (Bachlechner et al., 2006). Recently, a new proposal was made for service discovery in a cloud context

based on OpenTosca, an Enterprise Service Bus and Chef (Vukojevic-Haupt et al., 2014). Within the Docker ecosystem, the proposed tools often represent more of a service registry and leave it up to the application developer to use the provided lookup mechanism (r10).

Etd and Consul are two well-known representatives of this category. They provide a distributed key-value store in order to store ports, IP addresses or other characteristics of services running inside Docker containers. Zookeeper and DoozerD work in similar ways, but are less dedicated to Docker. Other tools like SkyDNS, dnsmasq and WeaveDNS try to solve the problem by reusing DNS for which there are discovery implementations in every OS. Tools like SkyDock or registrator automate the registration process (r8b) by monitoring Docker events and publishing information in the service registries. Confd stands out from the rest of the candidates, as it facilitates applications' usage of the configuration data from those service registries (r10). It reads data from service registries or environment variables and updates configuration files accordingly. In OpenStack, there is no dedicated service discovery tool, since it is focused on IaaS.

#### 4.7 Software Defined Network

Within Docker, every container gets a private IP only visible on the same host. Ideally, an SDN is used to connect containers between multiple hosts (Costache et al., 2014, r11). Furthermore, isolation is beneficial, so that every customer (tenant) of the SaaS solution gets an own virtual network (Drutskoy et al., 2013). In an SDN, a logically centralized controller manages the collection of switches through a standard interface, letting the software control virtual and physical switches from different vendors (ibid.). Open vSwitch is a popular SDN solution that is also used by default in OpenStack's Neutron. It is also a central part of the larger OpenDaylight initiative. Socketplane and Pipework are overlay networks that make use of Open vSwitch and are tailored for Docker. They manage IP assignment (r12) and routing of messages between networks of multiple hosts. Flannel and Weave promise to do the same, but without support for Open vSwitch and therefore with less flexibility.

#### 4.8 Load Balancer

The cloud can limit the scalability of a software load balancer due to security requirements. Amazon EC2 for example disabled many layer 2 capabilities, such

as promiscuous mode and IP spoofing so that traditional techniques to scale software load balancers will not work (Liu and Wee, 2009). HAProxy and Nginx are forwarding traffic on layer 7 which limits scalability due to SSL termination (ibid.). However they are commonly used and fulfill our requirements (r13-14).

#### 4.9 Monitoring

Monitoring is an essential part of cloud computing (Aceto et al., 2013). For monitoring Docker containers, you can use common solutions like Nagios that provide extensions for cloud scenarios, or some specialized tools like Sensu that are built for scalability from the ground up (Aceto et al., 2013). Within the Docker ecosystem, the most specialized solution is Google's cAdvisor, as it is tailored for monitoring containers. It brings its own Web UI. Logstash on the other hand is a general purpose tool for log file management (r18) and is often used in conjunction with elasticsearch as a NoSQL database and Kibana as a Web UI (Ward and Barker, 2014). Grafana is similar to Kibana and uses InfluxDB or other time series databases as data stores. It can be used in conjunction with cAdvisor since the latter can export data to InfluxDB. This seems advisable, since the Web UI of cAdvisor is limited to the latest data and does not show historical data. The combination can be further enhanced with Google Heapster which directly supports Kubernetes clusters. The container management solution mist.io does also include monitoring and seems to be the only one to support alerts based on thresholds (r7). Nagios is the default monitoring tool in OpenStack.

#### 4.10 Management Suites

Suites are the most comprehensive tools in our review and at least include cluster management and orchestration capabilities (r3, r4, r6). They either build on multiple other solutions in order to cover the required functionality (e.g. Kubernetes, which relies on the CoreOS tools etcd, fleet and flannel) or are large monolithic solutions from the micro-service perspective (e.g. BOSH). Kubernetes is popular and further supports container migration (r8). The OpenStack Docker driver allows managing Docker containers just like KVM VMs in OpenStack and therefore reusing a large part of the OpenStack modules. In principle, this is the right way to go (r19). However, it does not match our use case very well (Docker inside KVM-based VMs), since you have to decide per host which hypervisor to run (KVM, Xen

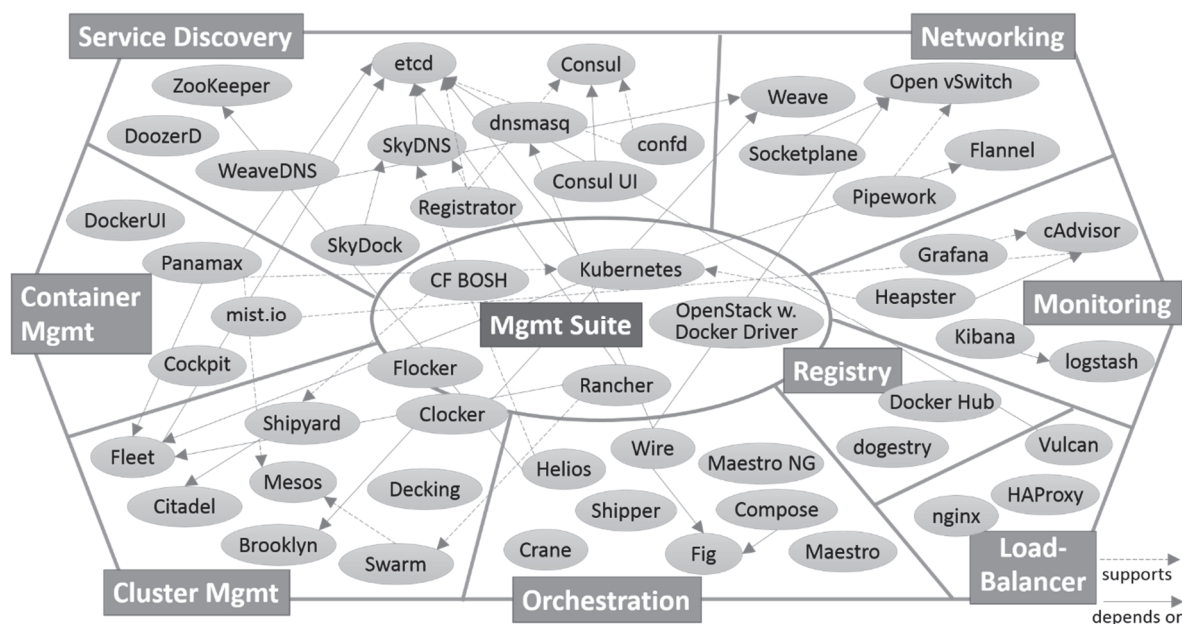


Figure 1: Docker ecosystem with dependencies (own illustration).

or Docker). Furthermore, not all modules are already updated to be used with Docker. A promising but still premature (alpha) candidate is Rancher.io. It aims at solving the multi-host problems of Docker by providing a Web-based UI, storage and networking capabilities. Version 0.3 from January 2015 allows starting and stopping containers on multiple hosts, linking containers across hosts and assigning storage (r9). They are also dedicated to support Docker Swarm and have a terminal agent that offers Web-based terminal sessions with containers (r17). BOSH is part of Pivotal's PaaS solution Cloud Foundry. It is able to start and stop containers on multiple hosts, but seems to be missing an overlay network component.

Figure 1 gives an overview of tools in the Docker ecosystem. The ellipses represent tools. The thick lines demarcate areas of functionality that are labeled in the rectangles. A position closer to the center of the figure within one category indicates that the tool has more functionality than others in the outer areas. Solid arrows represent dependencies between tools. Dashed arrows indicate that the tool is directly supported. It becomes obvious, that there already are some dependencies and interactions between tools. However, it is far from ideal and the most promising candidates of different categories are often developed side-by-side instead of hand-in-hand. Table 1 summarizes our findings more formally.

## 5 DISCUSSION

Despite the fact that the Docker ecosystem is huge, there still are requirements not fulfilled by any of the tools (r16, r20) and some are only fulfilled by a single tool (r7, r8b, r9). Many tools emerged quite recently and therefore must be considered premature.

Managing tenant data is maybe the most important missing part. (Lindner et al., 2010) argue, that there should be a complete supply chain for the cloud starting with deployment and monitoring and ending with accounting and billing. The economic part of this supply chain is currently not present in the Docker ecosystem. Updating a container can be emulated with a couple of Docker commands replacing it, since containers should be immutable. Still there should be a way to automate this. Registering a service in the registry is also a neglected requirement. Some tools do it, but our impression is that it is a better idea to use IP addresses and an SDN for routing instead of relying on one of the service discovery solutions when containers are migrated to another host. Storage is handled quite well by Flocker, but in our setup with Docker inside VMs there is still a problem. Volumes have to be mounted by the VM and mapped into the container. If the container moves, the volume has to be unmounted from the VM and mounted on the new host of the container. That means, that every container needs its own volume. If the space on the volume runs low,

growing it won't be easy. The Linux Device Mapper with its thin provisioning strategy can attenuate that problem but is not an ideal solution.

## 6 CONCLUSIONS

A Docker-based open source cloud environment to easily run composite applications as SaaS offerings would be a good basis for initiatives like the Open Cloud Alliance (Crisp Research, 2014) that aim at simplifying the process of bringing your applications to the cloud while preserving the freedom of choice and openness of the offering. In our paper, we have shown that many components are needed to fulfill the requirements for such a solution, which we dubbed runtime environment for SaaS applications (RaaS). It is similar to an IaaS environment, as we have shown with OpenStack, and includes some components from PaaS like load balancing and logging, but also has unique features like service orchestration and discovery. Not all requirements are currently fulfilled and despite first integration approaches, there is a need for closer cooperation within the Docker ecosystem. We plead for an embracing ecosystem project that serves as a coordination center for the tools that contribute to mastering the Docker management challenge. From our tests, Kubernetes with etcd, fleet and flannel seems the most usable combination right now. Mesos also seems a solid basis and integrations from other tools are currently in development (e.g. Compose/Swarm).

## REFERENCES

- Aceto, G., Botta, A., De Donato, W. and Pescapè, A. (2013), "Cloud monitoring: A survey", *Computer Networks*, Vol. 57 No. 9, pp. 2093–2115.
- Bachlechner, D., Siorpaes, K., Fensel, D. and Toma, I. (2006), "Web service discovery-a reality check", *3rd European Semantic Web Conference*, Vol. 308.
- Binz, T., Breitenbücher, U., Kopp, O. and Leymann, F. (2014), "TOSCA: Portable Automated Deployment and Management of Cloud Applications", *Advanced Web Services*, Springer, pp. 527–549.
- Bucchiarone, A. and Gnesi, S. (2006), "A survey on services composition languages and models", *International Workshop on Web Services-Modeling and Testing (WS-MaTe 2006)*, p. 51.
- Chauhan, M.A. and Babar, M.A. (2011), "Migrating service-oriented system to cloud computing: An experience report", *Cloud Computing (CLOUD) 2011, IEEE Int. Conf. on*, IEEE, pp. 404–411.
- Cockcroft, A. (2014), "State of the Art in Microservices", DockerCon Europe 14, Amsterdam, The Netherlands.
- Coffey, J., White, L., Wilde, N. and Simmons, S. (2010), "Locating software features in a SOA composite application", *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, IEEE, pp. 99–106.
- Costache, C., Machidon, O., Mladin, A., Sandu, F. and Bocu, R. (2014), "Software-defined networking of Linux containers", *13th RoEduNet Conf.*, IEEE.
- Crisp Research. (2014), *Open Cloud Alliance - Openness as an Imperative* (Strategy paper), Crisp Research, available at: <http://bit.ly/1ArYcyc>.
- "Docker Ecosystem Mindmap". (n.d.). *MindMeister*, available at: <http://bit.ly/1BjDgtW>.
- Docker, Inc. (2014), "About", *Docker Homepage*, available at: <http://bit.ly/1OjEBLL>.
- Drutskoy, D., Keller, E. and Rexford, J. (2013), "Scalable network virtualization in software-defined networks", *Internet Computing, IEEE*, Vol. 17 No. 2, pp. 20–27.
- Evans, E. (2003), *Domain driven design: Tackling Complexity in the Heart of Software*, Addison-Wesley, Boston.
- Jain, R. and Paul, S. (2013), "Network virtualization and software defined networking for cloud computing: a survey", *Communications Magazine, IEEE*, Vol. 51 No. 11, pp. 24–31.
- Koukis, V. (2013), "Flexible storage for HPC clouds with Archipelago and Ceph", *8th Workshop on Virtualization in High-Performance Cloud Computing*, ACM.
- Kratzke, N. (2014), "Lightweight Virtualization Cluster How to Overcome Cloud Vendor Lock-In", *J. of Computer and Communications*, Vol. 2 No. 12, pp. 1–7.
- Lindner, M., Galán, F., Chapman, C., Clayman, S., Henriksson, D. and Elmroth, E. (2010), "The cloud supply chain: A framework for information, monitoring, accounting and billing", *2nd Int. Conf. on Cloud Computing*.
- Liu, H. and Wee, S. (2009), "Web server farm in the cloud: Performance evaluation and dynamic architecture", *Cloud Computing*, Springer, pp. 369–380.
- Mietzner, R., Leymann, F. and Papazoglou, M.P. (2008), "Defining composite configurable SaaS application packages using SCA, variability descriptors and multi-tenancy patterns", *ICIW 2008*, IEEE.
- Mills, K., Filliben, J. and Dabrowski, C. (2011), "Comparing VM-placement algorithms for on-demand clouds", *Cloud Computing Technology and Science (CloudCom), IEEE 3rd Int. Conf. on*, IEEE, pp. 91–98.
- Papazoglou, M.P. (2003), "Service-oriented computing: Concepts, characteristics and directions", *Web Information Systems Engineering (WISE 2003). 4th Int. Conf. on*, IEEE, pp. 3–12.
- Peinl, R. (2015), "Docker ecosystem on Google Docs", available at: <http://bit.ly/1DJ0eS4>.
- Rosen, R. (2014), "Linux containers and the future cloud", *Linux Journal*, Vol. 2014 No. 240, p. 3.
- Roßbach, P. (2014), "Docker Poster", *Entwickler Magazin Docker spezial*, Vol. 2014 No. Docker spezial.
- Scheepers, M.J. (2014), "Virtualization and Containerization of Application Infrastructure: A Comparison", 21st Twente Student Conference on IT,

- University of Twente, Twente, The Netherlands.
- Sefraoui, O., Aissaoui, M. and Eleuldj, M. (2012), "OpenStack: toward an open-source solution for cloud computing", *International Journal of Computer Applications*, Vol. 55 No. 3, pp. 38–42.
- Turnbull, J. (2014), *The Docker Book: Containerization is the new virtualization*, James Turnbull.
- Vukojevic-Haupt, K., Haupt, F., Karastoyanova, D. and Leymann, F. (2014), "Service Selection for On-demand Provisioned Services", *Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International*, IEEE, pp. 120–127.
- Ward, J.S. and Barker, A. (2014), "Observing the clouds: a survey and taxonomy of cloud monitoring", *Journal of Cloud Computing: Advances, Systems and Applications*, Vol. 3 No. 1, p. 40.

# Container-based Virtualization for HPC

Holger Gantikow<sup>1</sup>, Sebastian Klingberg<sup>1</sup> and Christoph Reich<sup>2</sup>

<sup>1</sup>*Science & Computing AG, Tübingen, Germany*

<sup>2</sup>*Cloud Research Lab, Furtwangen University, Furtwangen, Germany*

*gantikow@gmail.com, {sebastian.klingberg, christoph.reich}@hs-furtwangen.de*

**Keywords:** Container Virtualization, Docker, High Performance Computing, HPC.

**Abstract:** Experts argue that the resource demands of High Performance Computing (HPC) clusters request bare-metal installations. The performance loss of container virtualization is minimal and close to bare-metal, but in comparison has many advantages, like ease of provisioning.

This paper presents the use of the newly adopted container technology and its multiple conceptional advantages for HPC, compared to traditional bare-metal installations or the use of VMs. The setup based on Docker (Docker, 2015) shows a possible use in private HPC sites or public clouds as well. The paper ends with a performance comparison of a FEA job run both bare-metal and using Docker and a detailed risk analysis of Docker installations in a multi-tenant environment, as HPC sites usually are.

## 1 INTRODUCTION

Applications in the domain of High Performance Computing (HPC) have massive requirements when it comes to resources like CPU, memory, I/O throughput and interconnects. This is the reason why they are traditionally run in a bare-metal setup, directly on physical systems, which are interconnected to so-called clusters.

Such a cluster infrastructure offers the best performance, but of disadvantage is the time for setting up: a) The operating system, usually some Linux flavor, must be installed using automatic mechanisms like PXE and Kickstart to install a basic installation ready to log in and get customized. b) All the applications required for computation and general HPC related libraries, like MPI (MPI, 2015), have to be installed and fine tuned in the customization phase. This is usually done by configuration management tools like Chef (Chef, 2015) or Puppet (Puppet, 2015). c) Before the first computational jobs can be started, the installed systems have to be finally integrated in some job scheduler like GridEngine (Oracle, 2015), LSF (IBM, 2015), or TORQUE (Adaptive Computing, 2015) which ensures proper resource management and avoids over-usage of resources.

Even though these steps can be automated to the great extent, the whole process until being finally able to start a job is quite time consuming and leaves the system in a rather static setup difficult to adapt to dif-

ferent customer needs. Often different applications, or even different versions of the same one, have conflicting environmental requirements, like a specific Linux version or specific library version (e.g. *libc*). This leads to the risk of putting the consistency of the whole cluster at stake, when adding a new application, or a newer version. Libraries might have to be updated, which might imply an upgrade of the whole Linux operating system (OS). Which in return can lead to old versions which are usually required for the ability to re-analyze previous calculations not being functional.

Now given a scenario where applications need computing environment changes frequently, the setup might take several hours. Even when using disk images, this approach does not pay off for jobs only running a rather limited time. One would like to have high configuration flexibility, with low application interference on the same cluster and optimal resource utilization. Isolation is the key solution for this. The use of different virtual machines (VMs), as they offer a feasible solution for tailoring a suitable environment for each type of workload and even providing a portable entity for moving HPC jobs to the cloud, is a trend that is gaining more and more momentum. With such a setup compute servers are no longer used for bare-metal computing, but turned into host systems for virtualization instead, reducing the installation time and making the systems much more flexible for different workloads, as they only have to offer the

minimum environment to host a VM, whereas all the application specific environment is encapsulated inside the VM.

Even though the use of *hypervisor-based virtualization* using VMs is highly common, it comes with quite a few trade-offs performance-wise, which make them less suitable for demanding HPC workloads.

Our work makes the following contributions:

- We present *container-based virtualization* with *Docker* as a superior alternative to VMs in the field of HPC.
- We provide a comparison of the concepts of VMs and containers and their use for HPC.
- We evaluate its performance using a finite element analysis (FEA) job with ABAQUS (Abaqus FEA, 2015), a widely used application in the field of computer aided engineering (CAE).
- We discuss possible risks of container-based virtualization.

The rest of the paper is organized as follows: Section 2 explores possible options for container-based virtualization. Section 3 discusses their advantages over VMs for HPC and describes the most viable Linux solution (*Docker*) for containers. Section 4 evaluates the performance overhead over native execution with a real life industrial computational job. Section 5 takes a look at possible security implications by using *Docker*. Section 6 concludes the paper.

## 2 RELATED WORK

The use of container-based virtualization for all sorts of workloads is not new, as the underlying concepts such as namespaces (Biederman, 2006) are mature. The core concept of isolating applications is seen in any Unix-like OS, with BSD Jails (Miller et al., 2010), Solaris Zones (Price and Tucker, 2004) and AIX Workload Partitions (Quintero et al., 2011) being available for years. Linux, the operating system that powers most HPC clusters and clouds, as opposed to Solaris and AIX, offers a similar solution called Linux Containers (LXC), with its initial release back in 2008. Even though LXC offers good performance it never really caught on in the HPC community. Another option for Linux based containers is *systemd-nspawn*, which hasn't seen any widespread use so far. The most interesting option we are considering as a VM alternative for HPC in this paper is *Docker*, which recently became the industry standard for Linux containers, due to its ease of use, its features, like layered file system images and the ecosystem supporting it.

There have been several studies comparing VM to bare-metal performance (Matthews et al., 2007), (Padala et al., 2007) which have lead to much improvements in hardware support for virtualization and VM technology as such (McDougall and Anderson, 2010). Especially the two open-source Type-1 hypervisor solutions *Xen* (Barham et al., 2003) and the *Kernel Virtual Machine (KVM)* (Kivity et al., 2007), that turns the whole Linux kernel into a hypervisor, have seen lots of performance improvements, for example by combining hardware acceleration with paravirtualized I/O devices using *virtio* (Russell, 2008). This papers discusses the advantages of container-based virtualization for HPC, the amount of performance-overhead added by *Docker* when running a FEA compute job inside a container instead bare-metal and takes a closer look at the security-related implications when using *Docker* in a multi-tenant environment.

## 3 CONTAINERS FOR HPC

Whereas VMs still offer the most mature and reliable technology for isolating workloads, both in terms of security and stability for encapsulating applications and their dependencies in a portable entity, their performance loss still remains, as overhead is added by a hypervisor, also known as *Virtual Machine Manager (VMM)*, running on top of the hardware to control the VMs.

While hypervisor-based virtualization can still mean up to 25% reduction in turnaround time in certain scenarios (Stanfield and Dandapanthula, 2014), large compute clusters continue to still run bare-metal setups, even though this means a huge trade-off in flexibility. The same applies for many of the commercial HPC-on-demand offerings, for example addressing the CAE sector. Because they're based on bare-metal setups they frequently can't offer the elasticity customers are accustomed to from mainstream cloud offerings like Amazon EC2, as there are certain minimum quantities that have to be booked for making the re-installation pay off.

By using a container-based virtualization approach this might change to a certain extend.

### 3.1 Container Virtualization vs. Hypervisor

The technical concept of container-based virtualization differs quite a bit from hypervisor-based virtualization. The containers run in user space on top of an operating system's kernel, while the hypervisor is

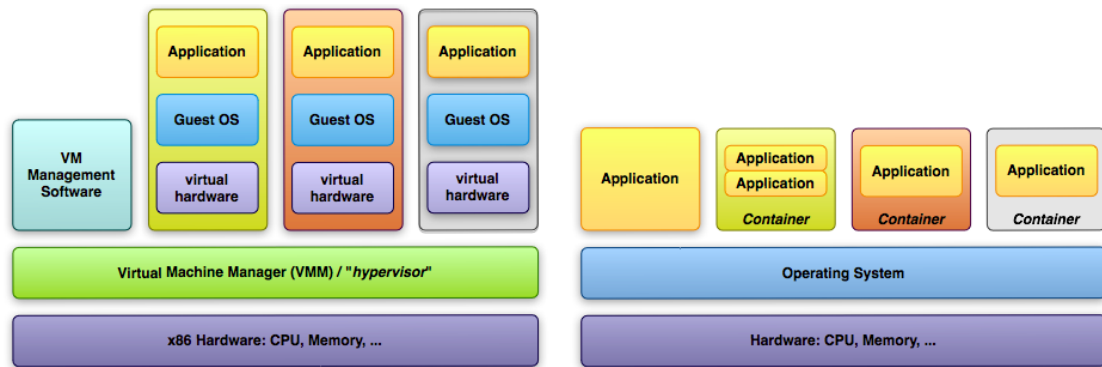


Figure 1: Hypervisor- (left) vs. Container-Based Virtualization (right).

scheduling system calls of the virtualized operating system as seen in Figure 1.

As the operating system kernel is the exact same for all *containers* running on one host and can not be changed, containers are limited to the same host operating system. VMs can run other operating systems e.g. Windows on top of Linux. This is a valid point for many scenarios, but the majority of HPC clusters and IaaS clouds tend to homogeneously run Unix (e.g. Linux). Container virtualization still offers the required flexibility to run e.g. an Ubuntu Linux container on top of a Red Hat Enterprise Linux host. Which also means a containerized process appears to be running on top of a regular Linux system.

Container-based virtualization relies on two Linux kernel features of the host system to provide the required isolation:

- *kernel namespaces* (Biederman, 2006): enabling individual views of the system for different processes, which includes namespaces for processes, file systems, user ids, etc. and enables the creation of isolated containers and limiting access of containerized processes to resources inside the container.
- *kernel control groups (cgroups)*: this subsystem puts boundaries on resource consumption of a single process or a process group and limits CPU, memory, disk and network I/O used by a container.

As mentioned before there are several options for using containers with Linux, with Docker having gained most attention recently due to its features and ease of use. Docker is released by the team at Docker, Inc. under the Apache 2.0 license.

### 3.2 Docker and HPC

Docker offers a good solution containerizing an application and its dependencies. As seen in Figure 2

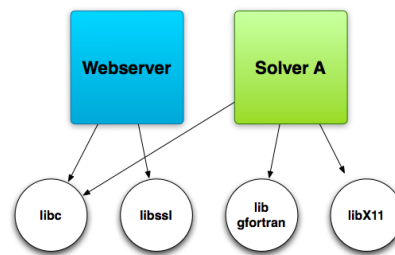


Figure 2: Applications and library dependencies.

applications usually share several libraries.

With container virtualization it is possible to isolate libraries as seen in Figure 3 to allow coexistence of special or incompatible library versions or even an *outdated* Linux distribution in a shippable entity easily. This solves the problem of running legacy code (might be needed for verifying old results of a computation) on a modern system, without the risk of breaking the system. As long as the code is not dependent on a special version of the kernel, as the kernel can not be changed inside a container.

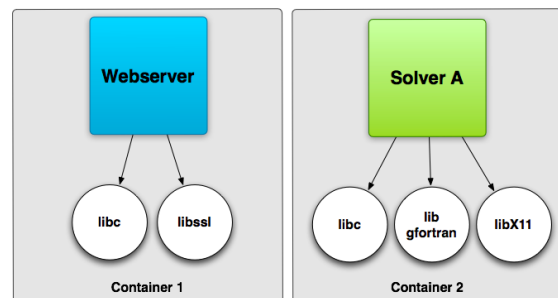


Figure 3: Applications and library dependencies encapsulated in containers.

Containers equipped with all tools and libraries for a certain tasks can be easily deployed on own clusters, systems of a related party for evaluation without



having to rebuild the whole computational work-flow or at a complete third party if additional resources are needed. All that is required is a suitable runtime for starting the container.

This also cuts down the complete re-purposing of compute resources to a simple start a new container image, as compute nodes only have to offer the Docker runtime and have access to the container files.

Compared to virtual machines this significantly reduces the amount of resources required. Both in memory footprint, as containers share a lot of resources with the host system as opposed to VMs starting a complete OS and in terms of storage required. Startup time is reduced from the time booting a full OS to the few seconds it takes till the container is ready to use.

For reducing storage requirements Docker makes use of *layered file system images*, usually *UnionFS* (Unionfs, 2015) as a space conserving mechanism, which is lacking in several other container solutions. This allows file systems stacked as layers on top of each other (see Figure 4), which enables sharing and reusing of base layers. For example the base installation of a certain distribution, with individually modified overlays stacked on top, can provide the required application and configuration for several different tasks.

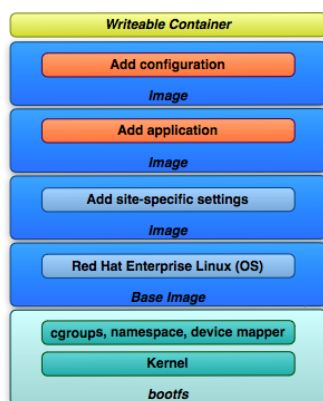


Figure 4: The layers of the Docker file system.

If a weakening of complete isolation is acceptable, it is also possible to pass directories, for example containing job input data into a container, so not all the required files for a compute job have to be included in the container image. One has to consider that strong isolation is desired, if providing a multi-tenant environment.

One popular VM feature it currently lacking. Compared to hypervisor-based setups Docker can not live-migrate running workloads to another host (Clark et al., 2005), which might be desirable for planned

system downtime. Even though this might not be required for most HPC environments, as killing and recreating a container might be faster, the *Checkpoint/Restore In Userspace* (CRIU) project is currently working on at least providing checkpoint and restore functionality (CRIU-Project, 2015). This feature would be much more required for high availability (HA) clusters than for HPC clusters.

## 4 EXPERIMENTAL EVALUATION

Performance-wise, without all the overhead added by hypervisor and VMs, containers as a light-weight virtualization mechanism can achieve almost the same performance as native execution on a bare-metal system does, as other benchmarks (Felter et al., 2014), (Xavier et al., 2013) underline.

As we were interested in the amount of overhead generated by containerizing a HPC workload, we decided to benchmark a real-world ABAQUS example in two different storage scenarios, comparing the containerized execution time to the native execution. ABAQUS (Abaqus FEA, 2015) is frequently used for finite element analysis (FEA) and computer aided engineering (CAE) for example in the aerospace and automotive industries, as it provides wide material modeling capability and multi-physics capabilities. ABAQUS jobs tend to be CPU and memory intense, requiring lots of scratch space too.

The application was installed to local disks on a CentOS 7 SunFire X2200 server with the following hardware configuration:

- CPU: 4x Dual-Core AMD Opteron (tm) 2220
- RAM: 16GB
- HDD: local conventional disks without RAID
- Infiniband: Mellanox MT25204

The job used for evaluation is the freely available *s4b* from the *samples.zip* package included in the ABAQUS 6.12-3 installation. It was installed onto local disks and the software licensing was done using a local license file.

As the server provided access to a Lustre parallel distributed file system, which is frequently used for large-scale cluster computing, we decided to execute the job one time with the folder for temporary data located on local disks and the other time on the Lustre file system. Both storage configurations were used with bare-metal execution and inside a Docker container. Docker was installed using packages shipped with CentOS 7.

There were no limits imposed on Docker using cgroups to ensure maximum performance. The resources available to the container were only limited by the actual resources on the host. When using one host for multiple containers simultaneously usage should be limited as expected. We used only one container per host for not distorting the outcome.

To rule out side effects, the job was run twenty times in each configuration and the *Total CPU Time* (in seconds), which is the sum across all involved CPUs of the time spent executing ABAQUS (user time) and the time spent by the OS doing work on behalf of ABAQUS (system time), from the job output file was taken to ascertain the total runtime of the simulation (see Figure 5).

The results, diagrammed in Figure 5, show the following:

- Docker offers near native execution speed
- there is constant but minimal overhead added
- average runtime (native vs Docker)

**local disk** 114s vs 116,5s - overhead: 2,21%

**Lustre** 117,3s vs 118,5s - overhead: 1,04%

To be clear: this means only two up to five seconds longer total execution time to complete the s4b example job, which is a fraction of the time a VM would even need to boot.

A point worth commenting on is the reason for the lesser overhead when accessing Lustre. The lower difference in overhead when using Lustre can be explained by the fact, that the container uses the RDMA stack for IBofP as directly as the host does, while accessing a local disk obviously needs to be passed through a UnionFS technology which affects the I/O flow here, at a small but mentionable minimum.

## 5 CONTAINER VIRTUALIZATION RISKS

As mentioned before the cornerstones of the performance and security isolation offered by Docker are cgroups and namespaces, both very mature technologies, which do a good job of avoiding Denial of Service (DoS) attacks against the host and limiting the view of what a container can see and has access to. Recent analysis on Docker (Bui, 2015) shows that the internal concepts of containers, even when using default configuration, are reasonable secure.

When deploying Docker in a multi-tenant environment, certain security aspects have to be considered: **Container vs VM.** When it comes to security aspects, isolation of filesystem, devices, IPCs, network

and management as described in Reshetova's paper (Reshetova et al., 2014) important. Generally it can be said, that containers have been seen as less secure than the full isolation offered by hypervisor virtualization, which is still true.

**Vulnerabilities.** Recent research by Ian Jackson and his colleague George Dunlap (Jackson, 2015) compared the number of Common Vulnerabilities and Exposures (CVE) in 2014 for paravirtualized Xen, KVM + QEMU, and Linux containers, that could lead to either privilege escalation (guest to host), denial of service attacks (by guest of host) or information leak (from host to guest) and showed that the risk for any of the three is higher when using containers.

One reason is that every additional interface available is a possibility for additional vulnerabilities. A hypervisor provides an interface similar to hardware and so-called hyper-calls (Xen offering 40). These are only very few calls a VM can interact with, compared to the much wider Linux system call interface used by containers. Making use of hardware virtualization for hypervisor-based setups may even add additional risks, as a detailed study (Pék et al., 2013) shows.

**Docker Daemon.** Since running Docker containers requires the *Docker Daemon* to run as root on the host, special attention should be given to its control, what leads to certain good practices for using Docker from a security point of view.

**Misuse.** Because it is possible to pass through file systems available on the host, only trusted users should be allowed access to start images or pass arguments to the Docker commandline-tool *docker*. Great harm can be done, if for example the hosts' complete file system is granted access to from inside the container. Membership of the *docker*-group is usually enough to invoke a command like `docker run -v /:/tmp myimage rm -rf /tmp/*` which would start a container, pass the hosts' filesystem to */tmp* inside the container and directly delete it.

**NFS.** This risk intensifies in a NFS-environment, where someone with unrestricted access to Docker can bind to an existing share, for example containing user homes, and circumvent access control based on numeric user IDs (UID) by creating a container for spoofing his UID. This can be mitigated by offering only NFSv4 shares to Docker hosts, where Kerberos as an additional authorization layer is available.

**Application Container vs System Container.** When using Docker for HPC applications the best thing to do is utilizing the container as an *application container* that directly starts the computing job after the container starts running. This eliminates the possibility for a user to pass parameter to the *docker-*

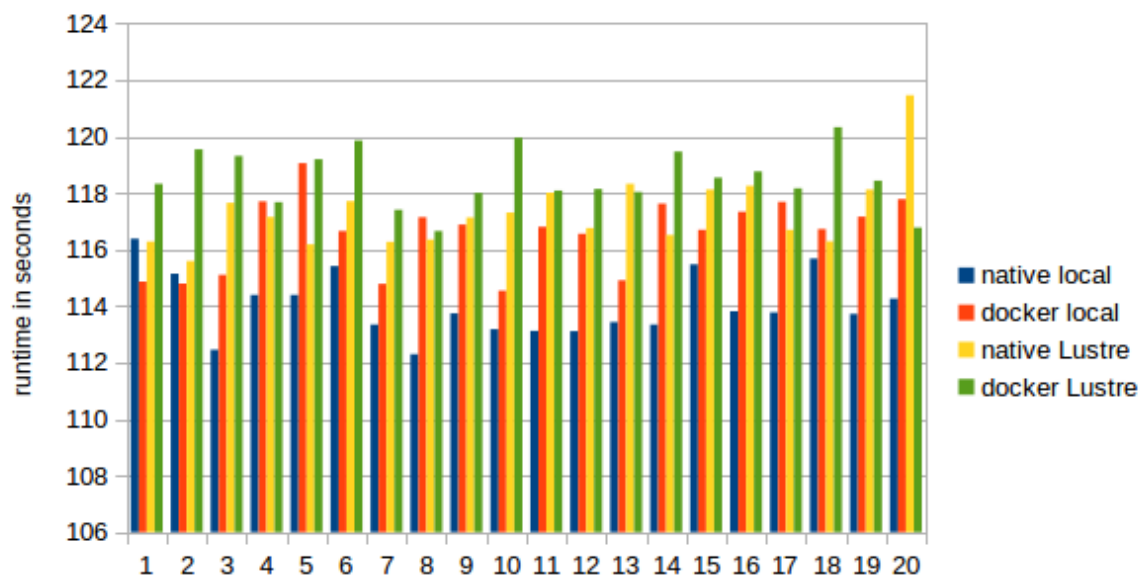


Figure 5: Total CPU Time (seconds).

command-line and to utilize the container as a more VM-style *system container*. As this disables the possibility to interactively use and explore the system, offering only pre-defined, parameter-controlled containers greatly reduces the risk of wanted or accidental misuse.

**Docker Security.** According to (Jérôme Petazzoni, 2013) development focusing on security is on the way, which will limit the Docker daemons' root privileges to certain sub-processes like virtual network setup. Further improvements aim at a possibility to map the root user of a container to a non-root user on the host system, reducing the impact of a possible privilege escalation. These new *user namespaces* will also optimize sharing file systems, as users within a container can be mapped to users outside the container. LXC already uses this feature (?), so it should be only a matter of time until Docker implements *user namespaces*.

**Image Security.** Attention should be paid to the source the container images are being pulled from. Docker Inc. offers a convenient way (called *Docker Hub*) to access thousands of preconfigured images which are ready to deploy. These images are for users who quickly want to set up an Apache web server or development environment for testing and do not offer any HPC related applications. But these images might be used as base for creating own HPC images. As security researchers state (Rudenberg, 2014) Docker includes a mechanism to *verify* images, which does imply that the checksum of the downloaded image has been validate. But this is actually not the case and offers possibilities for attacks. Docker solely checks

for a signed manifest and never actually verifies the image checksum from the manifest. Another potential attack arises from a vulnerability when dealing with malformed packages (Jay, 2014), as malformed packages can compromise a system. The advice in this case is to only download and run images from trusted sources, at best an internal image registry, which might be best-practice after all, not just for HPC clusters behind a corporate firewall.

## 6 CONCLUSION

Container-based virtualization using Docker solves many problems of bare-metal HPC, when flexibility to rapidly change the installed software, deploy new versions or use applications with conflicting dependencies on the same cluster is key.

Environmental details for a job, like a certain Linux distribution with a special compiler version could be included in a field in the job description, like required CPU and memory are nowadays. The workload manager then would pick a host fulfilling the hardware requirements, pulls the workload-specific image and starts the container that runs the job.

As our evaluation with an ABAQUS test job has shown, Docker offers near native execution speed, generating a mean loss in performance of 2,21% in our scenario with local disk I/O and 1,04% when accessing a Lustre clustered file system.

The point that Docker performs almost on the same level as the bare-metal execution shows that the Docker engine has almost trivial overhead and thus

offers performance traditional hypervisor-based VMs can not offer at the moment.

Since our research only focused on conceptual advantages and single host performance and many HPC applications rely on MPI for distributed computing future testing should be done in this area, taking a look at the performance of the Docker engine in distributed multi-host, multi-container scenarios and with other applications from the HPC field.

From a security standpoint VMs offer a more secure solution at the moment, but whether containers offer enough security depends on the overall HPC work-flow and the security requirements. A cloud provider offering a multi-tenant self-service solution with several customers on one cluster or even one host might want to implement an additional layer of security. In a regular HPC environment this might not be needed, as long as the necessary precautions are taken and users are not allowed to directly interact with Docker to provision potentially malicious containers but through a middle-ware like a job scheduler or parameter-controlled *sudo* scripts, that do careful parameter checking.

When it comes to patch management Docker could even provide an advantage over VMs, as the kernel is out of the focus of a container and shared among all hosts, meaning that if a kernel vulnerability is found only the Docker host has to be patched, which might be even done on the fly using tools like Ksplice.

Security of container-based solutions will further increase over time, with lot's of development being already underway. Linux containers have gotten a lot of attention over recent time and more people utilizing it will lead to closer examination and continuous improvements.

## REFERENCES

- Abaqus FEA, S. (2015). ABAQUS. <http://www.simulia.com>.
- Adaptive Computing (2015). TORQUE. <http://www.adaptivecomputing.com/products/open-source/torque/>.
- Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP '03*, pages 164–177, New York, NY, USA. ACM.
- Biederman, E. W. (2006). Multiple instances of the global linux namespaces. In *Proceedings of the 2006 Ottawa Linux Symposium*, Ottawa Linux Symposium, pages 101–112.
- Bui, T. (2015). Analysis of docker security. *CoRR*, abs/1501.02967.
- Chef (2015). Chef: Automation for Web-Scale IT. <https://www.chef.io/>.
- Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. (2005). Live migration of virtual machines. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, NSDI'05*, pages 273–286, Berkeley, CA, USA. USENIX Association.
- CRIU-Project (2015). Checkpoint/Restore In Userspace (CRIU). <http://www.criu.org/>.
- Docker (2015). Docker. <https://www.docker.com/>.
- Felter, W., Ferreira, A., Rajamony, R., and Rubio, J. (2014). An updated performance comparison of virtual machines and linux containers. *technology*, page 28:32.
- IBM (2015). LSF. <http://www-03.ibm.com/systems/platformcomputing/products/lsf/>.
- Jackson, I. (2015). Surviving the zombie apocalypse – security in the cloud containers, kvm and xen. <http://xenbits.xen.org/people/iwj/2015/fosdem-security/>.
- Jay, T. (2014). Before you initiate a docker pull. <https://securityblog.redhat.com/2014/12/18/before-you-initiate-a-docker-pull/>.
- Jérôme Petazzoni (2013). Containers & Docker: How Secure Are They? <https://blog.docker.com/2013/08/containers-docker-how-secure-are-they/>.
- Kivity, A., Kamay, Y., Laor, D., Lublin, U., and Liguori, A. (2007). kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, Ottawa, Ontario, Canada.
- Matthews, J. N., Hu, W., Hapuarachchi, M., Deshane, T., Dimatos, D., Hamilton, G., McCabe, M., and Owens, J. (2007). Quantifying the performance isolation properties of virtualization systems. In *Proceedings of the 2007 Workshop on Experimental Computer Science, ExpCS '07*, New York, NY, USA. ACM.
- McDougall, R. and Anderson, J. (2010). Virtualization performance: Perspectives and challenges ahead. *SIGOPS Oper. Syst. Rev.*, 44(4):40–56.
- Miller, F., Vandome, A., and John, M. (2010). *FreeBSD Jail*. VDM Publishing.
- MPI (2015). Message Passing Interface (MPI) standard. <http://www.mcs.anl.gov/research/projects/mpl/>.
- Oracle (2015). Grid Engine. <http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html>.
- Padala, P., Zhu, X., Wang, Z., Singhal, S., Shin, K. G., Padala, P., Zhu, X., Wang, Z., Singhal, S., and Shin, K. G. (2007). Performance evaluation of virtualization technologies for server consolidation. Technical report.
- Pék, G., Buttyán, L., and Bencsáth, B. (2013). A survey of security issues in hardware virtualization. *ACM Comput. Surv.*, 45(3):40:1–40:34.
- Price, D. and Tucker, A. (2004). Solaris zones: Operating system support for consolidating commercial workloads. In *Proceedings of the 18th Conference on Systems Administration (LISA 2004)*, Atlanta, USA, November 14-19, 2004, pages 241–254.

- Puppet (2015). puppet: Automate IT. <http://puppetlabs.com/>.
- Quintero, D., Brandon, S., Buehler, B., Fauck, T., Felix, G., Gibson, C., Maher, B., Mithaiwala, M., Moha, K., Mueller, M., et al. (2011). *Exploiting IBM AIX Workload Partitions*. IBM redbooks. IBM Redbooks.
- Reshetova, E., Karhunen, J., Nyman, T., and Asokan, N. (2014). Security of os-level virtualization technologies: Technical report. *CoRR*, abs/1407.4245.
- Rudenberg, J. (2014). Docker image insecurity. <https://titanous.com/posts/docker-insecurity>.
- Russell, R. (2008). Virtio: Towards a de-facto standard for virtual i/o devices. *SIGOPS Oper. Syst. Rev.*, 42(5):95–103.
- Stanfield, J. and Dandapanthula, N. (2014). HPC in an OpenStack Environment.
- Unionfs (2015). Unionfs: A Stackable Unification File System. <http://unionfs.filesystems.org>.
- Xavier, M., Neves, M., Rossi, F., Ferreto, T., Lange, T., and De Rose, C. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 233–240.

# **Towards Self-Protective Multi-Cloud Applications**

## ***MUSA – a Holistic Framework to Support the Security-Intelligent Lifecycle Management of Multi-Cloud Applications***

Erkuden Rios<sup>1</sup>, Eider Iturbe<sup>1</sup>, Leire Orue-Echevarria<sup>1</sup>, Massimiliano Rak<sup>2</sup> and Valentina Casola<sup>3</sup>  
<sup>1</sup>*TECNALIA, ICT-European Software Institute, Parque Tecnológico de Bizkaia, C/ Geldo Edificio 700, E-48160, Derio, Spain*

<sup>2</sup>*Dipartimento di Ingegneria Industriale e dell'Informazione, Seconda Università di Napoli, via Roma 29, Aversa (CE), Italy*

<sup>3</sup>*Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, da Università di Napoli, via claudio, Napoli, Italy*

*{erkuden.rios, eider.iturbe, leire.orue-echevarria }@tecnalia.com, massimiliano.rak@unina2.it, casolav@unina.it*

**Keywords:** Multi-cloud, Security-by-design, Cloud SLAs, QoSec, Distributed Deployment, DevOps.

**Abstract:** The most challenging applications in heterogeneous cloud ecosystems are those that are able to maximise the benefits of the combination of the cloud resources in use: multi-cloud applications. They have to deal with the security of the individual components as well as with the overall application security including the communications and the data flow between the components. In this paper we present a novel approach currently in progress, the MUSA framework. The MUSA framework aims to support the security-intelligent lifecycle management of distributed applications over heterogeneous cloud resources. The framework includes security-by-design mechanisms to allow application self-protection at runtime, as well as methods and tools for the integrated security assurance in both the engineering and operation of multi-cloud applications. The MUSA framework leverages security-by-design, agile and DevOps approaches to enable the security-aware development and operation of multi-cloud applications.

## **1 INTRODUCTION**

Cloud computing is an emerging promising paradigm for enabling new business models and economies of scale based on on-demand provisioning of IT resources (both hardware and software) over a network as metered services, where consumers are billed only for what they consume. A recent IDC Cloud forecast shows that the investment on public cloud services is expected to be more than €77,287 million in 2017 (IDC Cloud research, 2013).

Nevertheless, enterprises consider security as the #1 inhibitor to cloud adoptions (Waidner, 2009) (Expert Group Report. European Commission, 2010). Companies are reluctant to adopt cloud computing because of the difficulty in evaluating the trade-off between cloud benefits and the additional security risks and privacy issues it may bring. Most concerns are related to data protection, regulations compliance (Symantec, 2013) (Bitcurrent cloud computing survey, 2011) and other issues due to lack of insight (of controls and governance

processes) in the outsourcing of data and applications: data confidentiality, trust on aggregators, control over data and/or code location, and resource assignment in multi-tenancy (Expert Group Report. European Commission, 2010). Businesses that want to exploit cloud computing need to be vigilant in understanding the potential privacy and security breaches in this new environment (Hubbard & Sutton, 2010).

Secure cloud environments are even more challenging today, since they are becoming more and more complex in reference to the number of *cloud resource types* that are available “as a service”. Besides the traditional three service models defined by the NIST (Mell & Grance, 2010) (IaaS, PaaS and SaaS), new models are showing up such as Network as a Service specified by the ITU-T or Data as a Service defined in ISO/IEC 17826:2012 (ISO/IEC 17826:2012, 2012).

As the number of cloud models, cloud resources and cloud service providers grow in the market, it becomes theoretically easy (but not necessarily technically) for the cloud consumer to deploy and

use multiple cloud solutions at the same time in an integrated way (Miller, 2013). This means that despite the diverse characteristics of the cloud resources such as own management APIs and own service level offerings (both functional and security), all need to be monitored and managed as an integrated working entity.

The most challenging applications in heterogeneous cloud ecosystems are those that are able to maximise the benefits of the combination of the cloud resources in use: multi-cloud applications. For the context of this paper, a *multi-cloud application* is understood as *a distributed application over heterogeneous cloud resources whose components are deployed in different cloud service providers and still they all work in an integrated way and transparently for the end-user*.

Multi-cloud application solutions have to deal with the security of the individual components as well as with the overall application security including the communications and the data flow between the components. Even if each of the cloud service providers offered its own security controls, the multi-cloud application has to ensure an integrated security across the whole composition. Therefore, the overall security depends on the security properties of the application components, which in turn depend on the security properties offered by the cloud resources they exploit. For instance, the database component in charge of storing sensitive data cannot ensure a high confidentiality if the cloud storage resource in which it is deployed does not use strong encryption algorithms. Consequently, the whole multi-cloud application may be not sufficiently safe.

The paper is structured as follows. Section 2 describes the intended advances over the state of the art. Section 3 explains the MUSA approach and introduces the MUSA framework. Section 4 explains the future validation of the framework in industrial case studies. Finally, section 5 discusses the future work.

## 2 STATE OF THE ART

As outlined in introduction, the main purpose of the MUSA framework is to offer a solution to build security-aware multi-cloud applications. This research activity involves many different open research aspects, among them we focus on the following questions: *How do we identify the security requirements of a multi-cloud application? How can we ensure security of a multi-cloud application even*

*when control over some of its components is not granted?? How do we deploy a multi-cloud application maintaining the promised security features?*

The following three subsections try to offer a brief summary of the state of the art of the existing replies to such questions.

### 2.1 Security-by-design in Multi-Cloud Applications

Security by design (SbD) was first positioned by Gartner (Kreizman & Robertson) and pointed out the importance of incorporating security into the enterprise architecture process since the beginning, i.e. by including security requirements in the design process. In addition, Gartner recently defined the Runtime Application Self-Protection (RASP) (Gartner) security concept which is a security technology capable of controlling the application execution and detecting and preventing real-time attacks. The concept behind this idea would be that the application itself is able to control and manage security mechanisms embedded in the application or which can be invoked as a service by the application.

Security Control frameworks are widely adopted tools used to identify the security controls required to ensure the protection of an ICT system. A security control is a safeguard or a countermeasure prescribed to protect a system and meet a set of defined security requirement. Control Frameworks are a structured list of security controls that help a security expert to select the checks to perform in order to guarantee the respect of security requirements of a given system. Example of such Control Frameworks are the NIST Control Framework (NIST 800-53r4, 2013) and the ISO/IEC 27001 (ISO/IEC 27001).

In Cloud environments such frameworks are of limited use since they mostly miss specific cloud related security controls. Nevertheless, several attempts to address these issues have been made in (NIST SP500, 2010), (Cloud Data Protection Cert, 2013), (Cloud Security Alliance, 2014).

### 2.2 Security Aware SLAs in Multi-Cloud Applications

As stated above, in this paper, multi-cloud applications are distributed applications that run consuming cloud resources. Such an approach implies that the multi-cloud application developer and owner (i.e. the one that runs it and offers its

services) have no control over the real execution environment of the application. This inhibits the correct evaluation of the security controls.

The approach almost universally followed to define guarantees for users of a service is the introduction of Service Level Agreements (SLAs). An SLA is a formal agreement between a service provider and its end user that describes functional and non-functional aspects of the provided target service, together with clearly defined responsibilities of the involved parties.

The most well-known machine-readable SLA models are the Open Grid Forum's Web Services Agreement (WS-Agreement) (Hubbard & Sutton, 2010) and IBM's Web Service Level Agreement (WSLA) (Vukolić, 2010). The WS-Agreement specification proposes a domain-independent and standard way to create SLAs while its predecessor WSLA seems to be deprecated.

SLAs appear as a successful method to guarantee common Quality of Service parameters, like availability and performance indicators. As stated in many recent works, such as (Kandukuri, Paturi, & Rakshit, 2009), in order to deal with security requirements in the Cloud ecosystem, SLAs should be actually used to define target service security parameters.

Security Service Level Agreements (often named SecLA), are recognized as a promising way to model security issues between Cloud Service Providers and their users. ENISA, in (Dekker & Hogben, 2011), has also identified the importance of SecLAs in the Cloud computing field, pointing out that, in many circumstances, customers are not aware of many acquired services security aspects.

As introduced in (Almorsy, Grundy, & Ibrahim, 2011) and in (Luna et al, 2013), the current dearth of reasoning techniques on Security SLAs is preventing the diffusion of these approaches in production environments. Nevertheless, currently, many efforts are being made to fill this gap. For example, in (Luna et al, 2013), authors aim to outline techniques to quantitatively reason about Cloud Security SLAs, defining security metrics and a proof of concept semi-automated framework in order to assess cloud security of different providers.

Several European projects have worked or are working in this subject focusing mainly on SecSLA negotiation (SPECS Project, 2014), the creation of a security-aware SLA based language and related cloud security dependency model (CUMULUS project) and on the accountability for cloud-based services (A4Cloud Project, 2014).

## 2.3 Security Driven Dynamic Deployment of Multi-Cloud Applications

Multi-cloud applications have complex composition, provisioning and deployment requirements, and the application design becomes even more complex at the time an additional aspect such as security enters in the equation. Therefore, several initiatives are running in order to support this type of activities.

CloudML (CloudML project, 2013) (Ferry et al, 2013) developed a domain-specific language to support the specification of provisioning, deployment and adaptation concerns related to multi-cloud systems at design-time and their enactment at runtime. CloudML's background is PIM4Cloud language, defined in REMICS project (REMICS Consortium, 2012) (Ferry, Chauve, Rossini, Morin, & Solberg, 2013).

Based on CloudML, different approaches (ARTIST Consortium, 2013) (ModaClouds consortium, 2013) (PaaSage Consortium, 2014) and versions of CloudML have been recently released to provide means to the design of cloud based applications deployment. In this context where there are multiple CloudML versions, a joint task force has been started by MODAClouds, PaaSage and ARTIST projects which goal is to define a unique common CloudML specification (ARTIST Consortium, 2013).

Another approach that can be followed includes TOSCA (OASIS, 2013). The TOSCA specification aims to enhance the portability of cloud applications and services by using a language for defining both the service components of distributed applications and the service management interfaces (Antonescu, Robinson, & Braun, 2012). This approach is currently being followed by SeaClouds (SeacLOUDS consortium, 2013).

## 3 MUSA APPROACH: THE MUSA FRAMEWORK

Multi-cloud solutions represent a new challenging field in order to add value to overall cloud client experience (Vukolić, 2010). In order to exploit multi-clouds potentialities, different architectural approaches can be adopted (Bohli et al, 2013):

- (i) replication of applications, i.e. the same system is deployed in more than one provider and malicious attacks can be easily discovered comparing operation results;



- (ii) partition of application system into tiers, that allows to separate logic from data;
- (iii) partition of application logic into fragments, that obfuscates the overall application logic to providers;
- (iv) partition of application data into fragments, that makes impossible to a single provider to reconstruct data, safeguarding confidentiality.

MUSA aims at ensuring the security in all multi-cloud environments including those that combine multiple scenarios as described above. To this aim, MUSA approach combines i) a preventive security approach, promoting Security by Design practices in the development and embedding security mechanisms in the application, and ii) a reactive security approach, monitoring application runtime to mitigate security incidents, so multi-cloud application providers can be informed and react to them without losing end-user trust in the multi-cloud application.

In order to ensure the preventive oriented security to be embedded and aligned with reactive security measures, MUSA supports an integrated coordination of all phases in the application lifecycle management.

### 3.1 The MUSA Framework

The MUSA framework presented in this paper is intended to provide support the integration of the security within the multi-cloud application lifecycle, as illustrated in Figure 1. MUSA supports the first phase of the multi-cloud application lifecycle, the *development phase*, through the MUSA IDE, which helps in both specifying the end user security requirements and integrate such requirements in the application development.

The MUSA Decision support tool and MUSA Distributed deployment tool support the multi-cloud application *deployment phase*, helping in the choice of the cloud service provider and deployment of the multi-cloud application deployment.

The MUSA security assurance platform (SaaS) supports the last phase of the multi-cloud application lifecycle (*execution phase*), monitoring the application execution and, when needed, applying correction actions to grant the security features.

The MUSA framework aims to define a set of best practices and guidelines for the integrated management of Security by Design mechanisms in the lifecycle of multi-cloud secure applications, based on DevOps and agile (AgileManifesto, 2001) methodologies' principles. The practices are supported by the different automation tools provided

in the MUSA framework, which enable the coordination between programming and deployment infrastructure worlds, ensuring the continuous alignment of multi-cloud application security requirements specification (both at composition and SLA levels), implementation, monitoring and enforcement.

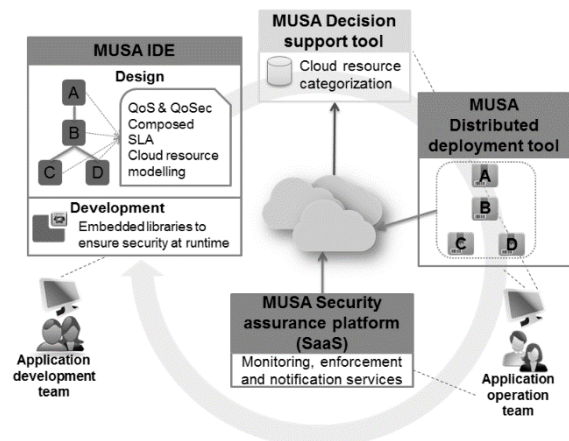


Figure 1: MUSA approach.

Following section describes the proposed tools in more detail.

#### 3.1.1 Multi-cloud Secure Applications Design

For an effective design of multi-cloud secure applications an integrated development environment (IDE) is needed. To solve this requirement, MUSA framework intends to deliver an IDE that allows the design of application components taking into account security requirements. This IDE will be based on existing open-source solutions and will include three main modules.

The first module is a security requirements specification tool for multi-cloud applications, taking into consideration multi-cloud SLA definition and composition. The tool will allow expressing in the multi-cloud application SLA the security requirements (QoSec) together with functional and business requirements (QoS). To this aim, the needed components' and cloud resources' SLA composition shall be computed.

The second module supports the design of the breakdown of multi-cloud application into components based on the combination of *functional, business and security properties* that the multi-cloud application should offer. The tool will be based on existing standards such as CloudML (CloudML project, 2013) and TOSCA (OASIS, 2013), and will help application developers to design the

architecture and the components composition taking into account the SLAs of the cloud resources in which the components will be deployed;

The last tool has the goal of design the provisioning and the deployment configuration of the needed heterogeneous cloud resources at multiple clouds layers (IaaS, PaaS).

Moreover, the MUSA IDE will include a set of security libraries that, embedded in the multi-cloud application, will enable the activation of security mechanisms and security controls without modifying the programming model. The security libraries aim at proposing a non-intrusive approach to introduce security in multi-cloud applications. These libraries will be inserted into the multi-cloud application components at design time and will be the responsible for ensuring the overall security at runtime. This software will include detection of non-compliant behaviour and enforcement mechanisms to be executed at runtime.

### **3.1.2 Multi-cloud Secure Applications Deployment**

Once the application has been successfully designed, it has to be deployed. Deploying multi-cloud applications on distributed and heterogeneous resources encompasses several challenges that are not addressed currently in the existing tools. To solve this challenge, MUSA aims to provide a secure multi-cloud deployment tool that offers a distributed deployment service based on the dynamic selection of the cloud service providers (CSPs) that match with the application risk analysis, the subsequent security requirements as well as functional and business needs. In order to achieve it, the MUSA framework bases its offering mainly over three components. The first one is the cloud resource categorization of CSPs based on the measures of the security and functional properties at real time. The second component, a decision support tool, allows the selection of the cloud resources which combination is compliant with the security and functional requirements specified in the multi-cloud application composite SLA, after a previous simplified process of risk analysis. Finally, the third component allows an automated deployment of the multi-cloud secure application, distributing each of the application components' packages towards the matched cloud resource.

### **3.1.3 Multi-cloud Secure Applications Runtime**

Monitoring multi-cloud applications at runtime

involves collecting metrics of QoS and QoSec parameters of both the components of the application and the cloud resources provisioned. MUSA aims to provide a monitoring service capable of collecting such measurements by using standard APIs (if they are used by the cloud service providers (CSP)), cloud interoperability frameworks such as jclouds (Apache, 2012), or measures provided by MUSA security embedded libraries.

Whenever an incident occurs, MUSA sends alerts to notify the application provider about detected security relevant incidents. Moreover, MUSA send alerts when an application is in risk of not fulfilling its SLA, and some preventive action needs to be taken in order to keep security parameters well counterweighted with performance or within the margins specified in the SLA, e.g. a redeployment of application components across a different combination of cloud resources.

Finally, the enforcement service offered by MUSA ensures that the multi-cloud application respects the security requirements in its SLA.

All three services (monitoring, notification and enforcement) are delivered in the form of a security assurance platform, packaged as a SaaS product. The MUSA SaaS security assurance platform collaborates closely with the embedded libraries to enforce the security protection of the multi-cloud application user's data, through mechanisms such as authentication, authorisation, data encryption, data location assurance, etc. when the cloud resources used do not offer such mechanisms.

The MUSA SaaS will store monitored security parameters over the cloud resources and components, and manage the necessary notifications and alerts, so as the multi-cloud application provider can early react to possible security breaches. Contract verification processes will require a mapping between low-level resource metrics and high-level security parameters of the cloud services. This process will be done at runtime by MUSA SaaS, which will provide real-time assessment supported by complex processing of composed measures of low-level metrics.

## **4 MUSA FRAMEWORK VALIDATION**

The economic viability, user acceptance and practical usability of the MUSA framework is expected to be validated through piloting the solution in realistic industry environments representing highly relevant services for the

European economy: airline flight scheduling systems and urban smart mobility services.

In the following, we summarize the research challenges faced in both case studies.

#### **4.1 Case Study A: Airline Flight Scheduling Multi-Cloud Application**

For our first case study we have selected NetLine/Sched product by Lufthansa Systems to demonstrate how MUSA framework benefits the integrated security management of this application that exploits a number of heterogeneous cloud resources.

The product NetLine/Sched supports all aspects of flight schedule development and management.

In this case study, we are particularly interested in researching on how to:

- (i) allow NetLine/Sched developer declare the options regarding data localisation (e.g. location country of the files), data retention and deletion, data integrity, confidentiality, access control and availability, etc. and make possible that such policies are embedded in the application specification.
- (ii) enable security properties are embedded into the deployed application artefacts (security-by-design) for their continuous control at operation.
- (iii) allow deployment into secure multi-cloud and multi-provider environments.
- (iv) provide automated security assurance, supported by continuous monitoring, enforcement and notification mechanisms.
- (v) keep the NetLine/Sched operator informed about the discrepancies and/or adapts to such requirements even in those cases that a change in the architecture or composition of the clouds underneath is needed.

#### **4.2 Case Study B: Smart Mobility Multi-Cloud Application**

Our second case study is an urban smart mobility multi-cloud application in Tampere city in Finland. Tampere Region has almost half a million inhabitants with a modal share of: 16% public transport, 27% pedestrians and cyclists, 57% private cars.

Tampere City Council has a number of services exposed to allow companies and individual

developers to develop, test and productize own traffic applications using public data. The services can be publicly accessed via Intelligent Transport Systems and Services (ITS) platform (Wikipedia ITS, 2014), which includes the public transport services APIs, other traffic related APIs, traffic data, etc.

In this case study Tampere University (TUT) will take the role of many entrepreneur citizens and companies (generally SMEs) that create innovative applications by combining freely available open services and datasets in the Web to create business. The multi-cloud application by TUT aims at supporting the energy efficient and sustainable multi-modal transit of Tampere citizens when commuting from home to work and vice versa.

The major challenges for MUSA in this case study are the following:

- (i) Enhance security capabilities of innovative services in transportation and public infrastructure in Tampere.
- (ii) Enable entrepreneurs and citizens willing to develop innovative services based on IST Factory to be able to easily integrate security-intelligence into their applications through the use of MUSA IDE.
- (iii) Empower operators of the multi-cloud applications that integrate IST Factory cloud-based services to ensure security of data storage and exchange at runtime through the use of MUSA assurance tools.
- (iv) Allow evaluating new service multi-cloud deployment implications by checking service dependencies on other network and cloud resources.

## **5 FUTURE WORK**

Application growth, rise in complexity and need for interoperability create market opportunity for cloud integrators and multi-cloud providers by offering new capabilities in the existing complex cloud landscape (North Bridge in partnership with GigaOM Research, 2013).

Taking profit of this opportunity window, MUSA aims at contributing to building up the innovation capacity and technology excellence of the European software and service industry by proposing a solution to master the security-intelligent lifecycle of multi-cloud applications based on novel DevOps and security-by-design approaches.

In this paper we have presented the MUSA framework whose main goal is to support the security-intelligent lifecycle management of multi-cloud applications. There are a number of major challenges in the path:

- (i) Enable the security aware design of distributed applications over heterogeneous cloud resources.
- (ii) Automatic discovery and decision support system of combinations of cloud services that best match the required balance between security and functional properties.
- (iii) Security assurance through continuous monitoring and integrated methods in both engineering and operation of multi-cloud applications.

The MUSA project which will lead to the development and validation of MUSA framework was launched on January 2015 and will last 36 months. Future publications on the progress of the framework are expected both online ([www.musa-project.eu](http://www.musa-project.eu)) and in future papers.

## ACKNOWLEDGEMENTS

The project leading to this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644429.

## REFERENCES

- A4Cloud Project. (2014). Accountability For Cloud and Other Future Internet Services. Retrieved from Accountability For Cloud and Other Future Internet Services.: [www.a4cloud.eu/](http://www.a4cloud.eu/)
- AgileManifesto. (2001, February 17). Manifesto for Agile Development. Retrieved December 8, 2013, from Manifesto for Agile Development: <http://agilemanifesto.org/>
- Almorsy, M., Grundy, J., & Ibrahim, A. S. (2011). Collaboration-based cloud computing security management framework. IEEE International Conference on Cloud Computing (CLOUD) (pp. 364-371). IEEE.
- Antonescu, A.-F., Robinson, P., & Braun, T. (2012). Dynamic Topology Orchestration for Distributed Cloud-Based Applications. NCCA, (pp. 116 - 223).
- Apache. (2012). Apache jclouds. Retrieved April 2014, from Apache jclouds: <http://jclouds.apache.org/>
- ARTIST Consortium. (2012). ARTIST Project. Retrieved April 15th, 2014, from ARTIST Project: <http://www.artist-project.eu/>
- ARTIST Consortium. (2013, September). Deliverable 7.2.1. Cloud services modelling and performance analysis framework. Retrieved April 2014, from Deliverable 7.2.1. Cloud services modelling and performance analysis framework: [http://www.artist-project.eu/sites/default/files/D7.2.1%20Cloud%20services%20modeling%20and%20performance%20analysis%20framework\\_M12\\_30092013.pdf](http://www.artist-project.eu/sites/default/files/D7.2.1%20Cloud%20services%20modeling%20and%20performance%20analysis%20framework_M12_30092013.pdf).
- ARTIST Consortium. (2013, September). Deliverable D4.3.1 Dissemination report. Retrieved April 2014, from Deliverable D4.3.1 Dissemination report: [http://www.artist-project.eu/sites/default/files/D4.3.1%20Dissemination%20report\\_M12\\_01102013.pdf](http://www.artist-project.eu/sites/default/files/D4.3.1%20Dissemination%20report_M12_01102013.pdf).
- Bitcurrent cloud computing survey. (2011). Bitcurrent cloud computing survey 2011. Bitcurrent cloud computing survey 2011.
- Bohli, J. et al. (2013). Security and Privacy Enhancing Multi-Cloud Architectures.
- Cloud Security Alliance. (2014). Cloud Controls Matrix. Retrieved April 2014, from Cloud Controls Matrix: <https://cloudsecurityalliance.org/research/ccm>.
- Cloud Data Protection Cert. (2013). Cloud Data Protection Cert. Retrieved April 2014, from Cloud Data Protection Cert: <http://clouddataprotection.org/cert>.
- CloudML project. (2013). Model-based provisioning and deployment of cloud based systems. CloudML project. Retrieved April 2014, from Model-based provisioning and deployment of cloud based systems. CloudML project: <http://cloudml.org>.
- CUMULUS project. (n.d.). Certification infrastructure for Multi-Layer cloud Services. Retrieved from Certification infrastructure for Multi-Layer cloud Services: <http://cumulus-project.eu/>
- Dekker, M., & Hogben, G. (2011). Survey and analysis of security parameters in cloud SLAs across the European public sector. Retrieved April 2014, from Survey and analysis of security parameters in cloud SLAs across the European public sector: <http://www.enisa.europa.eu/activities/Resilience-and-CIIP/cloud-computing/survey-and-analysis-of-security-parameters-in-cloud-slas-across-the-european-public-sector>.
- Expert Group Report. European Commission, I. S. (2010). The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010.
- Ferry, N. et al. (2013). Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. CLOUD 2013: IEEE 6th International Conference on Cloud Computing, (pp. 887-894).
- Ferry, N., Chauve, F., Rossini, A., Morin, B., & Solberg, A. (2013). Managing multi-cloud systems with the CloudML framework. NordiCloud'13: 2nd Nordic Symposium on Cloud Computing & Internet Technologies. Oslo, Norway.
- Gartner. (n.d.). Gartner IT Glossary - Runtime Application Self-Protection (RASP). Retrieved April 2014, from <http://www.gartner.com/it-glossary/runtime->

- application-self-protection-rasp (Retrieved April 2014).
- Hubbard, D., & Sutton, M. (2010). Top Threats to Cloud Computing V1.0. Cloud Security Alliance.
- IDC Cloud research. (2013, September). IDC Cloud research. Retrieved March 2014, from IDC Cloud research: <http://www.idc.com/getdoc.jsp?containerId=prUS24298013>.
- ISO/IEC 17826:2012. (2012). ISO/IEC 17826:2012 Information technology -- Cloud Data Management Interface (CDMI).
- ISO/IEC 27001. (n.d.). ISO/IEC 27001 Information Technology – Security Techniques – Information Security management Systems – requirements.
- Kandukuri, B., Paturi, V. R., & Rakshit, A. (2009). Cloud security issues. SCC'09. IEEE International Conference on Services Computing, 2009., (pp. 517-520).
- Kreizman, G., & Robertson, B. (n.d.). Incorporating Security into the Enterprise Architecture Process. Retrieved April 2014, from Incorporating Security into the Enterprise Architecture Process: [http://www.gartner.com/DisplayDocument?ref=g\\_search&id=488575](http://www.gartner.com/DisplayDocument?ref=g_search&id=488575).
- Luna, J., et al. (2013). Negotiating and Brokering Cloud Resources based on Security Level Agreements. CLOSER 2013, (pp. 533-541).
- Mell, P., & Grance, T. (2010). The NIST definition of cloud computing. In ACM (Ed.), Communications of the ACM, 53, no. 6, p. 50.
- Miller, P. (2013, September). Sector RoadMap: Multicloud management in 2013.
- ModaClouds consortium. (2013, September). Deliverable 4.2.1 MODACloudML development – Initial version. Retrieved April 2014, from Deliverable 4.2.1 MODACloudML development – Initial version: [http://www.modaclouds.eu/wp-content/uploads/2012/09/MODAClouds\\_D4.2.1\\_MODACloudMLDevelopmentInitialVersion.pdf](http://www.modaclouds.eu/wp-content/uploads/2012/09/MODAClouds_D4.2.1_MODACloudMLDevelopmentInitialVersion.pdf).
- NIST 800-53r4. (2013). 291 NIST Security and Privacy Controls for Federal Information Systems and Organizations. Retrieved April 2014, from 291 NIST Security and Privacy Controls for Federal Information Systems and Organizations: [nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf](http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf).
- NIST SP500. (2010). 291 NIST Cloud Computing Standards Roadmap. Retrieved April 2014, from 291 NIST Cloud Computing Standards Roadmap: [http://www.nist.gov/itl/cloud/upload/NIST\\_SP-500-291\\_Version-2\\_2013\\_June18\\_FINAL.pdf](http://www.nist.gov/itl/cloud/upload/NIST_SP-500-291_Version-2_2013_June18_FINAL.pdf).
- North Bridge in partnership with GigaOM Research. (2013). The future of cloud computing, 3rd annual survey 2013. Retrieved March 2014, from The future of cloud computing, 3rd annual survey 2013: <http://www.northbridge.com/2013-cloud-computing-survey>.
- OASIS. (2013). Topology and Orchestration Specification for Cloud Applications Standard. Retrieved April 2014, from TOSCA standard by OASIS: [www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=tosca](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca).
- PaaSage Consortium. (2014, April 30). Deliverable D2.1.2: CloudML Implementation Documentation (First version). Retrieved from Deliverable D2.1.2: CloudML Implementation Documentation (First version): [http://www.paasage.eu/images/documents/paasage\\_d2.1.2\\_final.pdf](http://www.paasage.eu/images/documents/paasage_d2.1.2_final.pdf).
- REMICS Consortium. (2012). Deliverable 4.1 PIM4Cloud. Retrieved March 2014, from Deliverable 4.1 PIM4Cloud: [http://www.remics.eu/system/files/REMICS\\_D4.1\\_V2.0\\_LowResolution.pdf](http://www.remics.eu/system/files/REMICS_D4.1_V2.0_LowResolution.pdf).
- SeacLOUDS consortium. (2013). SeacLOUDS project. Seamless adaptive multi-cloud management of service-based applications. Retrieved from SeacLOUDS project. Seamless adaptive multi-cloud management of service-based applications: <http://www.seacclouds-project.eu/project.html>.
- SPECS Project. (2014). Secure Provisioning of Cloud Services based on SLA management. Retrieved from Secure Provisioning of Cloud Services based on SLA management: <http://specs-project.eu/>.
- Symantec. (2013). Choosing a Cloud Hosting Provider with Confidence. Retrieved April 2014, from Choosing a Cloud Hosting Provider with Confidence: <http://www.itwhitepapers.com/content20287>.
- Vukolić, M. (2010). The Byzantine empire in the intercloud. 41(3), 105-111.
- Waidner, M. (2009, November). Cloud computing and security. Lecture Univ. Stuttgart (November 2009). Retrieved from Cloud computing and security. Lecture Univ. Stuttgart (November 2009).
- Wikipedia ITS. (2014). Intelligent Transport Systems and Services (ITS) Factory Wiki. Retrieved April 2014, from Intelligent Transport Systems and Services (ITS) Factory Wiki: [http://wiki.itsfactory.fi/index.php/ITS\\_Factory\\_Development\\_Wiki](http://wiki.itsfactory.fi/index.php/ITS_Factory_Development_Wiki).

# High Performance Virtual Machine Recovery in the Cloud

Valentina Salapura<sup>1</sup> and Richard Harper<sup>2</sup>

<sup>1</sup>IBM T. J. Watson Research Center, 1101 Kitchawan Rd, NY, Yorktown Heights, U.S.A.

<sup>2</sup>IBM T. J. Watson Research Center, Research Triangle Park, NC, U.S.A.  
{salapura, reharper}@us.ibm.com

**Keywords:** Cloud Computing, High Availability, Virtualization, Automation, Enterprise Class.

**Abstract:** In this paper, we outline and illustrate concepts that are essential to achieve fast, highly scalable virtual machine planning and failover at the Virtual Machine (VM) level in a data center containing a large number of servers, VMs, and disks. To illustrate the concepts a solution is implemented and analyzed for IBM's Cloud Managed Services enterprise cloud. The solution enables at-failover-time planning, and keeps the recovery time within tight service level agreement (SLA) allowed time budgets via parallelization of recovery activities. The initial serial failover time was reduced for an order of magnitude due to parallel VM restart, and to parallel VM restart combined with parallel storage device remapping.

## 1 INTRODUCTION

Cloud computing is being rapidly adopted across the IT industry as a platform for increasingly more demanding workloads, both traditional and a new generation of mobile, social and analytics applications. In the cloud, customers are being led to expect levels of availability that until recently were available only to the largest of enterprises.

Cloud computing is changing the way high availability (HA) of a data center can be implemented. It is widely recognized that the standardization, virtualization, modularity and cross system management capabilities of cloud computing offer a unique opportunity to provide highly resilient and highly available systems. Resilience techniques can build on a well-defined and uniform framework for providing recovery measures for replicating unresponsive services, and recovering failed services to respond to disaster scenarios. Since virtualization allows packaging of workloads — operating system, applications, and data — into a portable virtual machine image container, it facilitates transfer of workloads from one server to another. High availability features can migrate a VM image from one physical server to another within the same data center if the original server suffers any failure, performance loss, or to perform scheduled maintenance.

However, clouds and the workloads that run on them are big. Many high availability systems were

originally designed for smaller managed environments, and do not scale well as the system size and complexity increases. Detecting failures, determining appropriate failover targets, re-mapping storage to those failover targets, and restarting the virtual workload have to be carefully designed and parallelized in order to meet the service level agreement (SLA) for large systems.

This paper describes a highly scalable parallel virtual machine planning and recovery method that enables high availability at the Virtual Machine (VM) level for large data centers comprising many high-capacity servers, many VMs, and a large number of disks in a storage area network (SAN). The system enables on-the-fly failover planning and execution for a compute environment with a large number of servers and storage devices.

The functionality described in this paper has been released as part of IBM's enterprise cloud offering known as CMS (Cloud Managed Services), where it was used to provide scalable HA for the AIX Logical Partitions (LPARs) running on the CMS Power Systems (Sinharoy *et al.*, 2015) servers. To stay within this context, the paper will continue to use the Power LPAR terminology. However, the concepts described here apply equally well to any platform that is similarly structured. While in this paper we focus only on the infrastructure level resiliency, CMS cloud implements all application level high availability approaches. However, they are not in scope of this paper, and will not be discussed here.

## 2 BACKGROUND AND POSITION STATEMENTS

### 2.1 Virtual Machine-Level and Application-Level High Availability Are Complimentary

There are multiple approaches to provide a high availability solution in a virtual environment. One approach is to provide HA at the application level, using what are commonly known as HA Clustering techniques. Another approach is to provide availability at the infrastructure level, using VM-level HA.

Application-level high availability techniques are built around application clustering technology. These solutions are used to improve the availability of applications by continuously monitoring the application's resources and their physical server environment, and invoking recovery procedures when failures occur. These solutions typically use multiple virtual machines which are working together in order to ensure that an application is always available. These VMs are arranged in active-passive or active-active configuration. When one VM fails, its functionality is taken over by the backup VM in the cluster. Examples of these solutions are IBM PowerHA (IBM, 2008), Microsoft Clustering Services (Microsoft, 2003), Veritas Storage Foundation, and LinuxHA.

HA solutions at the infrastructure level are designed to ensure that the virtual resources meet their availability targets. This is accomplished by continuously monitoring the infrastructure environment, detecting a failure, and invoking recovery procedures when a failure occurs. Typically, such recovery procedures involve restarting the failed VM, either on the same or a different physical server.

Although this paper will not discuss application-level HA in detail, we have found that application-level HA and infrastructure-level HA can operate beneficially together with no mutually destructive effects. A tidy separation of concerns exists - infrastructure-level HA restarts VMs when appropriate (sometimes on alternate servers), while application-level HA sees these restarts as simple system crashes and recoveries, which it is designed to tolerate anyhow. In addition, recovery of the VMs in a cluster on another server after the originating server fails restores the redundancy that the application-level HA cluster relies upon, minimizing

the time during which that cluster is operating with degraded resiliency.

### 2.2 Dynamic Storage Mapping Is Preferable to Static Mapping

Virtualized infrastructures can be designed such that either all physical servers in a server pool are statically mapped to all the storage devices that may be used by the virtual machines, or all physical machines are dynamically mapped to only the storage devices that are needed to support the virtual workload running on the respective physical machines. The first design choice has the merits of being simpler to operate, since no remapping of storage is required as virtual machines migrate or failover within the pool. However, it is unsuitable for high-scale cloud environments where the pool may consist of hundreds or more servers, supporting thousands of virtual machines, which in turn use even more storage devices. In this environment, the architectural and design limits of the hypervisor running on each physical server cannot support the huge number of simultaneous connections required to support all possible VM-storage device mapping. Instead, it is desirable to have a physical server only possess storage mappings for those VMs that are actually running on that physical server, and this is the design point utilized in this paper. The disadvantage of this approach are that, if it is necessary to migrate or failover a VM from one server to another, it is necessary to map that VM's storage to the destination physical server, and unmap that storage from the source physical server.

### 2.3 Parallelization of Recovery Is Critical to Maintaining SLAs

Complex recovery activities consist of a number of sequential steps that must often be executed using tools, processes, and infrastructure elements that have limited recovery performance and concurrency. Given the large scale of a recovery operation (recovery of potentially thousands of virtual machines across dozens of physical servers), it is absolutely necessary to judiciously parallelize these recovery actions and eliminate bottlenecks to meet tight SLAs. The limited space herein does not permit a full exposition of these position statements, but we will partially illustrate them using an implemented case study based on the IBM Cloud Managed Services (CMS) architecture.

### 3 CMS POD ARCHITECTURE

CMS is a cloud computing offering for enterprise customers. It is designed to bring the advantages of cloud computing to the strategic outsourcing customers of IBM. It provides standardized, resilient, and secure IBM infrastructure, tools, and services with full ITIL management capabilities (Cannon, 2011). CMS offers functions such as consumption-based metrics and automated service-management integration.

The design of the CMS is based upon a unit called the point of delivery (PoD). A PoD contains many physical managed resources (server, storage, and network) that are virtualized, and provided to customers as an infrastructure offering. A CMS PoD contains Intel-based servers to support virtual and bare metal Windows and Linux workloads, and IBM Power servers to support virtual AIX workloads. The Power virtual machines are called Logical Partitions, or LPARs. This paper focuses on the recovery of the AIX workloads, contained in LPARs, in the event that a Power server fails.

A PoD is designed to be highly available, with the physical infrastructure architected to eliminate single points of failure. The customer is offered selectable availability SLAs, which are contractual obligations and may include penalties for noncompliance. These availability agreements are only for unplanned outages and refer to Virtual Machine availability. CMS supports multiple levels of availability ranging from 98.5% to 99.9%. A more detailed description of the CMS can be found in (Salapura, 2013).

PoDs also contain a number of managing servers which host management tools for storage management, backup, and performance monitoring.

#### 3.1 Fault Model: Permanent Failure of a Power Server

The remainder of this paper will describe the architecture we have created for recovering LPARs on other physical servers when one or more Power Systems physical servers hosting those LPARs has failed.

In this failure mode, a Power Server suffers a hardware failure from which it cannot recover in a short time (for example, 10 minutes) and for which maintenance/repair is required. In this case, the failover process will restart all affected LPARs on another Server. The function implementing this recovery process is called Remote Restart. The

recovered LPARs need to use the same network storage disks – referred to as LUNs (logical unit number) that the original Server was using. Restarts are prioritized by SLA. Recovery from other types of outages and transient failures are covered by means not described in this paper.

### 4 REMOTE RESTART ARCHITECTURE

The architecture of the Remote Restart solution used in CMS PoDs is illustrated in Figure 1. There are one or several managing servers, indicated in the upper part of the figure, and a number of managed servers with storage are illustrated in the lower part of the figure. The managing servers host tools for controlling, provisioning, managing and monitoring of the workload on managed servers. Relevant managing tools are Provisioning engine, which uses a DB to maintain all the PoD management information, a Storage management engine, and a Hardware Maintenance Console (HMC) for server management. The Remote Restart software and collected configuration data resides on a management server for Virtualization management.

The managed servers host LPARs running customers' AIX workload. Each managed Power server also contains dedicated LPARs called Virtual I/O Servers (VIOS) that virtualize external storage and present it to the customer's LPARs.

#### 4.1 Overview of Recovery Procedure

The tasks that the Remote Restart solution performs are as follows:

*Periodic data gathering and persistence:* configuration and status of LPARs in a PoD is collected periodically. The time interval for data gathering is configurable, and is given later in this paper. There are two sources of collecting needed information:

- information about physical servers in the PoD, all LPARs and their hosts, and their storage and network configuration; this information is collected via HMC;
- SLA availability information for all LPARs; this information is obtained by querying the Provisioning engine database.

*Server failure detection:* the health of all servers in a PoD is monitored in order to detect their failure. A failure of a server is detected via HMC when it returns an ERROR state.



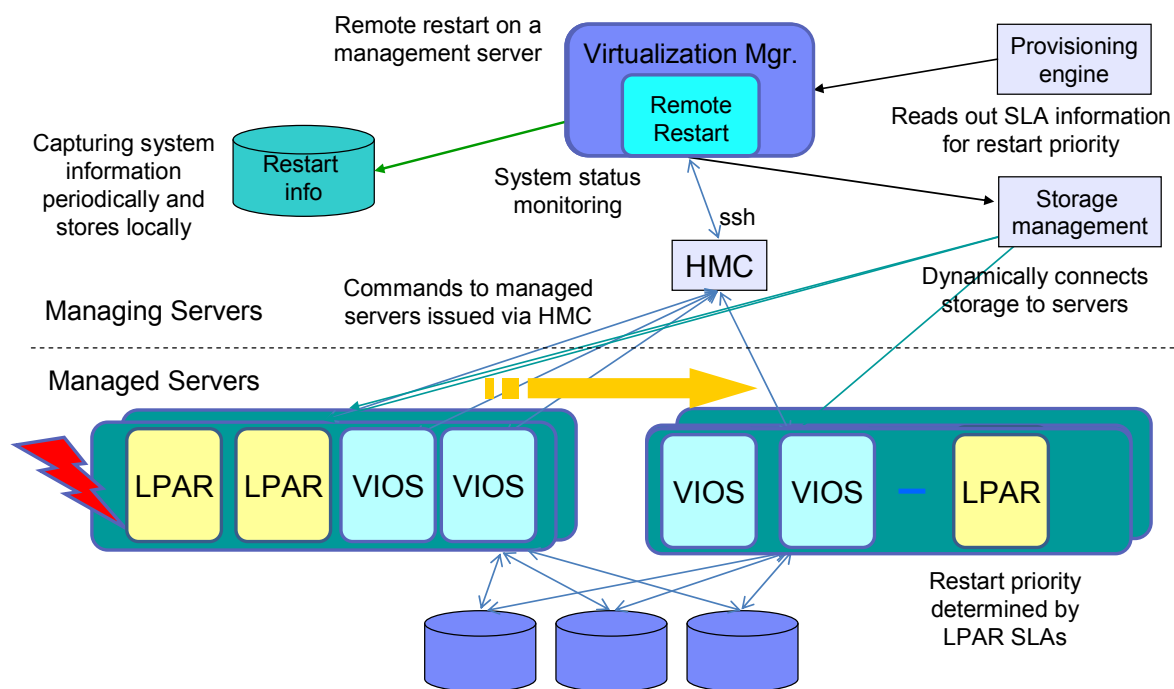


Figure 1: Remote Restart architecture.

*Server fencing:* once a server is determined faulty, it is powered off via HMC commands.

*Failover planning:* provides an evacuation plan. Our Remote Restart implementation uses a “Dynaplan” (Harper, 2011) algorithm to determine the optimal failover targets.

*VIOS configuration for failover:* in this step, virtual SCSI devices are created via HMC on the failover server for LPARs to be restarted.

*SAN configuration for failover:* LUNs are not connected to all servers in a PoD, and the connecting of LUNs to the failover servers according to the evacuation plan is performed in this step.

*LPAR restart:* once virtual SCSI devices are created and LUNs and connected to the failover server, an LPAR is restarted on the failover server via HMC commands.

The Remote Restart scripts performs these steps by issuing `ssh` commands to the HMC, via database queries to the Provisioning engine, and by issuing commands for storage configuration.

## 4.2 Failover Planner

Failover planning is based on a parallelized algorithm evolved from the prior dynamic resource planner described in (Harper, 2011). The planner formulates a schedule to restart a large collection of

interdependent VMs on a large collection of resources. There are a number of constraints the planner has to meet, for example that recovery time objective is met, that the maximum number of the most important dependency groups is started, that VMs within a dependency group are started in the proper order, and that the capabilities of the environment (e.g., restart bandwidth and capacities) are not exceeded.

Restart priority is a partial ordering of all VMs into priority classes. Within a given priority class, all VMs can be restarted in parallel, subject to restart parallelism constraints of the physical environment and application start order dependencies. A “restart rules” language allows customization of the restart priority based on restart rules. A restart rule template can be automatically populated by discovery tools and/or manually edited.

The restart priority is automatically and dynamically determined based on a number of VM properties, such as SLAs, application priority, application topology, and other rules as determined by the dynamic restart priority calculator and a given set of rules. Priority aggregation rules convert the various restart rules into the VM restart partial priority order while taking into account application dependencies.

The cost of running the planner is low, so it is run at failure-handling time. In addition, the failover

planner is run once per day for each server in a PoD to determine any resource constraint, for example to determine if there are capacity problems so that not all LPARs can be hosted on the remaining hosts. If this condition is detected, a warning notification is sent to the cloud administrators for the purposes of planning.

## 5 IMPLEMENTATIONS AND RESULTS

### 5.1 Initial Implementation: Serial Restart

The restart priority of LPARs is based on their SLA. Thus, in case of failover, the highest SLA workloads would be restarted first followed by the next highest SLA. Within the same SLA level, restart priority is random. In an early CMS release, restart capability was needed only for workloads with the two highest level SLAs. This initial Remote Restart implementation was implemented as a single process which, after the failure of a server is detected, and the need for a failover process was determined, would initiate the failover process.

For each LPAR on the affected server, the failover planner determines a destination server, and the restart process starts. The failover process is performed for the highest priority LPARs first, configuring the storage and network for these LPARs to their destination servers, and restarting them at the destination server. After all LPARs with the highest restart priority are restarted at their target servers, the next lower priority level LPARs are processed.

There are two significant time components to executing the restart. The first is the process of unmapping the LUNs from the (failed) original server and mapping them to the designated failover server. This time is proportional to the number of LUNs connected to the LPAR. The second time component is the process of restarting the LPAR on the designated failover server.

In this early CMS release, each LPAR was allowed to have up to two LUNs. For the case where only the top two SLAs were to be restarted, with up to two LUNs per LPAR, the SLA time budget was readily met.

However, in the subsequent releases of CMS, the number of disks per LPAR was continuously increased. In addition, it was necessary to extend restart capabilities to all SLA levels. With these

increases, it was clear that we needed a solution for Remote Restart which would handle restarts for a larger number of LPARs containing more LUNs, within the SLA time limits.

### 5.2 Parallel Restart

The requirement for an increased number of LUNs per LPAR, and the increased number of LPARs which need to be restarted motivated us to improve the Remote Restart solution using parallel processes. We chose to use server-level parallelism in which the level of parallelism depends on the number of operational servers in the PoD.

In our parallelization scheme, one restart process is launched for each destination server. For example, in a PoD with 6 servers, and one failed server, there would be up to 5 destination failover servers. One restart process is initiated for each destination server. LPARs assigned for restart on that particular server are restarted sequentially, starting with the highest priority LPARs in that group. For each LPAR, storage is mapped, storage and network drivers are reconfigured for the target server, and the LPAR is restarted at the destination server. Once all highest priority LPARs assigned to that destination server are restarted, the next SLA priority level LPARs are processed. A similar process is performed in parallel for all destination servers.

These parallelization steps ensured that the failover time was well within the allowed SLA for the subsequent releases of CMS.

### 5.3 Parallel Disk Mapping

However, the disk capacity in CMS continues to increase. For the current release, each LPAR can have up to 24 LUNs and up to 96 TB of storage. For a large number of LPARs on a single server, this can lead to the case where a very large number of storage LUNs has to be mapped to different servers in short time.

Analysis indicated that the procedure that was taking the most amount of time was the process of mapping disks to the destination server, so our next improvement focused on parallel disk mapping. In this implementation, in addition to the number of parallel failover processes that is started, we also initiate the mapping of multiple disks attached to a single LPAR in parallel. We limit the number of simultaneous mappings of disks for a single failover stream to four to avoid potential bottleneck at the storage management interface. By measuring the time needed for restarting individual LPARs with a

different number of LUNs and time measured for parallel failover streams, we analyzed failover time needed for parallel disk restart.

The analysis shows that adding this additional level of parallel processing brings the failover time requirements well within the available time budget for the worst-case configuration known to date.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a highly scalable parallel virtual machine planning and failover method that enables high availability at a VM level in a data center. This solution is efficient for large data centers comprising many high-capacity servers, many VMs, and a large number of disks. The solution is implemented and used in IBM's CMS enterprise private cloud.

The system enables at-failover-time failover planning and execution for a compute environment with a large number of servers and storage. The described system keeps the recovery time within limits to a service level agreement (SLA) allowed time budget. With this design, we reduce the initial failover time requirements by more than an order of magnitude by using parallel failover and parallel storage mapping implementation.

As our future work, we plan to explore the applicability of this solution for disaster recovery (DR), where a whole PoD needs to be restarted at a failover data center within the allowed recovery time objective (RTO).

## REFERENCES

- Cannon, D. 2011. ITIL Service Strategy 2011 Edition, The Stationery Office, 2nd edition, 2011.
- Harper, R., Ryu, K., Frank, D., Spainhower, L., Shankar, R., Weaver, T., 2011. DynaPlan: Resource placement for application-level clustering, 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, pp. 271 – 277, 2011.
- IBM, 2008. Implementing PowerHA for IBM i, IBM Corporation, Armonk, NY, USA, 2008. [Online]. <http://www.redbooks.ibm.com/abstracts/sg247405.html>.
- Microsoft, 2003. Introducing Microsoft Cluster Service (MSCS) in the Windows Server 2003 Family, Microsoft Developer Network. [Online]. <https://msdn.microsoft.com/en-us/library/ms952401.aspx>.
- Salapura, V., Harper, R., Viswanathan, M., 2013. Resilient cloud computing, IBM Journal of Research and Development, vol. 57, no. 5, pp. 10:1-10:12, 2013.

- Sinharoy, B. *et al.*, 2015. IBM POWER8 processor core microarchitecture. IBM Journal of Research and Development, vol. 59, no. 1, pp. 2:1-2:21, 2015.

# Adopting an Agent and Event Driven Approach for Enabling Mutual Auditability and Security Transparency in Cloud based Services

Moussa Ouedraogo<sup>1</sup>, Eric Dubois<sup>1</sup>, Djamel Khadraoui<sup>1</sup>, Sebastien Poggi<sup>2</sup> and Benoit Chenal<sup>2</sup>

<sup>1</sup>*Luxembourg Institute of Science and Technology, 5 Avenue des hauts Fourneaux, L4362 Esch/Alzette, Luxembourg, Luxembourg*

<sup>2</sup>*Victor Buck Services S.A, L-8308 Capellen, Luxembourg, Luxembourg*

*{moussa.ouedraogo, eric.dubois, djamel.khadraoui}@list.lu, sebastien.poggi@victorbuckservices.com benoit.chenal@learch.lu*

**Keywords:** Cloud, Security Transparency, Mutual Auditability, Monitoring, Event Specification and Detection.

**Abstract:** We propose an event-driven approach for the automated audit of cloud based services security. The proposed approach is a solution to two of the intrinsic security issues of cloud based services, notably the need of security transparency and mutual auditability amongst the stakeholders. We leverage a logic based event specification language to represent patterns of events which occurrence can be evidence of security anomaly or breach or simply a sign of a nefarious use of the cloud infrastructure by some of its users. The use of dedicated algorithms for the detection of composite events coalesced with the definition of primitive events structure based on XCCDF format ensures the reuse and interoperability with security audit tools based on the Security Content and Automation Protocol-SCAP. The implementation and application of the approach on a cloud service dealing with electronic archiving have demonstrated its feasibility and viability.

## 1 INTRODUCTION

For most businesses and individuals, Cloud based services are the alternative to achieving cost-efficiency in the provisioning and consumption of services. However companies dealing with security and/or privacy critical data, have often shown some reluctance to fully embrace the trend, even if there is evidence that the trend is starting to sift at least for the banking and financial sector (<http://www.businesscloudnews.com/2014/06/02/cloud-in-financial-services-what-is-it-not-good-for/>). Several factors could explain such an attitude towards the cloud: In the cloud, the data and the mechanisms necessary for its processing may reside in the provider's premises. This leads to some devolution of security matters about such data and processes to the cloud provider whose capability and/or due diligence to deal with the security issues may be mistrusted or simply feeble. The uncertainty on the actual location of the data is also exacerbated by the complexity of the chain of provider-consumer. In fact, although a CSP may be registered in a given country, the chain of provider-consumer may be such that the actual data centre used by the CSP is located elsewhere. Given the stored information may be subject to the legislation of the country where it is stored physically, this may also pose serious privacy

management challenges. In fact there may be ambiguity in understanding which regulation applies for a data about a third country citizen (which should normally be subjected to national regulation) but stored in another country, where regulation towards privacy may be well different. The multi-tenancy aspect that is most often used to characterize cloud computing also introduces a new risk unique to cloud services, the possibility of attacks from other consumers, who may be competitors or simply hackers, co-located on the same infrastructure, e.g., servers, hard disks, virtual machines. This is well exemplified by "Amazon Zeus botnet" incident involving Amazon EC2's infrastructure (McAfee and Guardian Analytics, 2012), whereby cybercriminals, by initially hacking into a service hosted by Amazon cloud infrastructure, were able to install command-and-controls infrastructure with the aim to infect client computers and steal their banking credentials. This incident is a reminder that the security of the cloud service is only as good as at its weakest link given that a vulnerability at a tenant application may result in the jeopardy of the whole service. This status quo calls for techniques that help to foster more security assurance in the cloud realm. Security assurance being the ground for confidence that security deployed and/or managed by a third party is correctly implemented and also effective against the

risks (Ouedraogo et al., 2012). In third party services such as the cloud, security assurance can be practically met by probing the security of the CSP through audits and by gaining more visibility on its security policy and operation through security transparency mechanism (Winkler, 2011). Consequently, achieving a wider adoption of cloud based services would depend on how effective issues related to mutual auditability and security transparency can be addressed (Chen, 2010; Ouedraogo et al., 2013; Nuñez et al., 2013; Sunyaev and Schneider, 2013). Techniques and approaches tailored in that vein of idea should enable the Cloud Service Consumer (CSC), provided the existence of contractual clauses with the Cloud Service Provider (CSP), to gather evidence that corroborate or challenge compliance, performance and security claim made by the CSP, while at the same time enabling the latter to monitor the activity and traffic of the users to ensure no abuse and nefarious use of the cloud is made.

This paper's contribution can be summarized as an effort to leverage Event-driven computing (Luckham, 2005; Etzion and Niblett, 2010) and Multi agent systems-MAS- (Ganzha and Paprzycki, 2014) to foster more security transparency and enable mutual-auditability in a cloud setting. To achieve this, we resort to a tree based specification of security events of interest by the CSC and CSP while within the infrastructure, software agents are generated for capturing such events in case they materialize. We amalgamate real time security related event detection and logic-based rules for empowering both the CSC and CSP, with effective means of depicting and promptly detecting anomalies and security or QoS breaches. An event is here considered as a happening of interest (related to security or quality of service procurement) to the CSP or CSC.

The paper is organized as follows: Section 2 analyses the related work. Section 3 provides a description of the adopted architecture. In Section 4 we specify monitor-able event using a logic based language. Section 5 shows how audits and transparency are enforced, while Section 6 presents an application case. Section 7 provides some concluding remarks.

## 2 RELATED WORKS

Initiatives purporting to address the issue of mutual trust and transparency in the cloud have mainly revolved around the topic of audit, Virtual machine introspection and Service level agreement. Audits standards including SSAE16 ([www.ssae16.com](http://www.ssae16.com)),

and its international version ISAE3402 (<http://isae3402.com/>) rely in a large part on the words and assessment of the CSP, information that cannot be guaranteed to be immune from bias. Recent efforts in cloud audits have leaned towards automation. For instance, the CSA cloud-audit (<http://cloudaudit.org/CloudAudit/Home.html>) pur-ports the automation of standard audit and related assurance and compliance effort by providing a controlled set of interfaces to allow CSCs or their representatives to assess their services. Dolitzscher et al. (2013) propose a cloud audit methodology based on the usage of MAS for conducting the audit of virtual machines dynamically allocated to clients to account for changes within the cloud infrastructure.

Rak et al. (2011) adopts APIs derived from the mOSAIC project (<http://www.mosaic-project.eu/>) to build up an SLA-oriented cloud application that enables the management of security features related to user authentication and authorization. An extension of the work of Rak et al. can be found through the EU FP7 project Specs aiming to deliver a platform for providing a security services based on SLA management. The SLA monitoring in SLA@SOI relies on EVEREST+ (Lorenzoli and Spanoudakis, 2010), which is a general-purpose engine for monitoring the behavioural and quality properties of distributed systems based on events captured from them during the operation of these systems at runtime. The major problem with the adoption of SLA management as a means to enhance security transparency is primarily on its practicality. Indeed the academic notion of SLA appears to be far more extensive than it is in reality. In the context of this work, the authors have approached a number of CSPs in Luxembourg with the aim to get a glimpse on the set of items that were part of their SLA. Most often, such documents were restricted to the sole aspects of allocated bandwidth, storage capacity, etc.; while the only security aspect included was related to service availability. Clearly, the items included in those specifications were those the companies were confident they could deliver on. Their argument on the most pressing and challenging issues such as security was that stringent mechanisms were in place for its guarantee as evidenced by their certifications.

Unlike the existing initiatives, our approach leverages events processing to enable mutual audit between the CSC and CSP. While existing commercial and open source solutions for event analytics such as Splunk (Carasso, 2012), Arcsight (<http://www.arcsight.net/>) and Graylog2 (<https://www.graylog.org/graylog2-v0-92/>) are based on log analysis, our initiative is based on near-real time

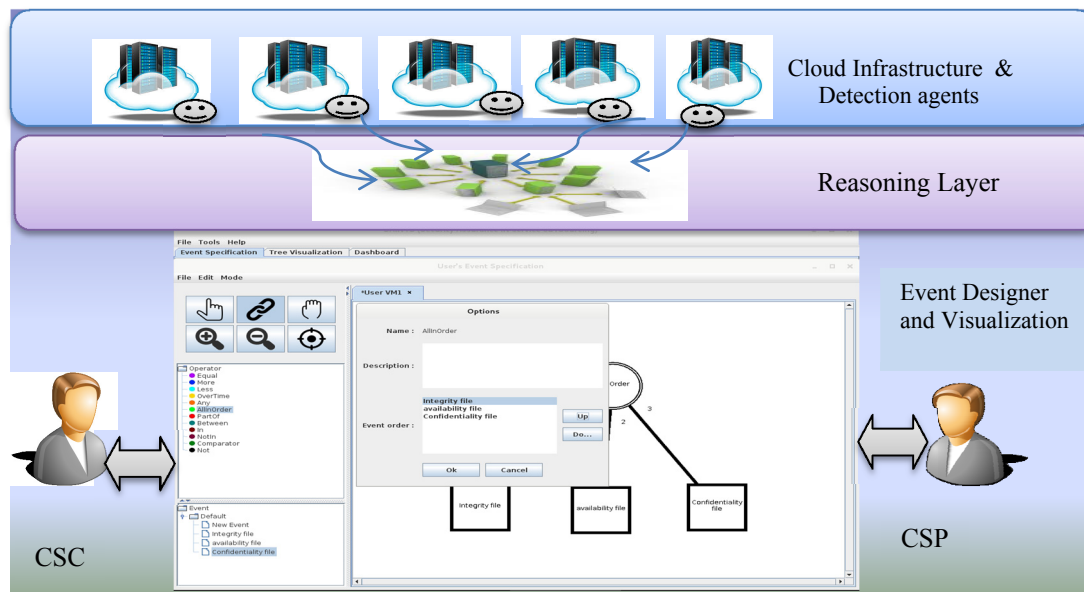


Figure 1: Architecture of the event specification monitoring for transparency and mutual auditability.

detection of primitive events followed by a reasoning on whether the specified composite event has materialized. Thus, allowing the concerned CSC/CSP to promptly take mitigating actions. Also, provided the existence of a contractual agreement, both the CSP and CSC can specify and launch a monitoring of the security of the other. Additionally our approach is that the event driven specifications provides a good expressivity for capturing events of interest emanating from an SLA, an internal policy or external regulations while allowing the reuse of existing security management tools.

### 3 HIGH LEVEL ARCHITECTURE OF THE APPROACH

Our event driven approach and tool is made of three main components as depicted in Figure 1.

The first component is a case tool or *Event designer* that offers a graphical design interface to the cloud stakeholders, for specifying patterns of events that are of interest. Upon the design of the events composites that could result from combining different patterns to monitor, the stakeholder provides technical details for each event using an event specification format of choice. In the context of our work, the Extensible Configuration Checklist Description (XCCDF) format (Waltermire et al., 2011) is adopted as further elaborated in Section 4.1. The case tool also serves as a dashboard for the visualization of the events status once the monitoring is triggered.

The second component relates to the elaboration of a multi-agent system embedded within the cloud infrastructure (CI) with the purpose of detecting each primitive events specified within the Event Designer console. The definition and management of the detection agents are performed using JADE platform (Bellifemine et al., 2008). The peculiarity of the agent structure and organization is adopted from the work of Ouedraogo et al. (2014).

The third component is the event processing layer. Individual atomic events captured from the cloud infrastructure are sent by specialized agents to the reasoning layer where dedicated algorithms detailed in Section 5 will be resorted for informing the stakeholder when a specified pattern has materialized.

In the following we further elaborate on how those three components play a role in practically delivering mutual audit and help foster better transparency.

### 4 SPECIFICATION OF MONITORABLE EVENTS

The decision to adopt an event driven approach to audit and monitor the security in the cloud is underscore by the argument that events provide a powerful construct to capture current state of a system and deviations from expectation and to predict future security or QoS related issues (Luckham, 2005; Etzion and Niblett, 2010). Additionally a well-defined architecture can support event based moni-

toring in ensuring the prompt dissemination of its occurrence to the interested parties who would make judgment on the course of action to adopt. Amongst others, it may be a way to hold cloud providers accountable for a security breach that may have stemmed from a lax in their security; a breach of SLA or other escrows between the two parties. The set of patterns and the detection algorithms associated could also constitute a powerful tool for a cloud provider concerned with activities of its clients.

A prerequisite for effective event patterns detection is the definition of a clear event structure coupled with the adoption of a pattern specification language that is expressive enough to capture the realm of events of interest and their propensity. Only after that one can begin to implement the required strategy for the ensuing detection. This section provides an insight into our event based approach.

#### 4.1 A Primitive Event Structure to Ensure Re-Use of Existing Security Tools

Commonly an event is defined as an occurrence of interest within a system or domain (Etzion and Niblett, 2010). Subsequently, dealing with events could purport the monitoring of a system or process with the intent to flag exceptional or anomalous behaviors. Alternatively, event analysis could be the baseline for (i) predicting a major event before they actually take place as in fraud detection application, financial market trends and natural disaster; (ii) diagnosing a problem based on deductive reasoning after the observation of symptomatic events. While most event structure includes header information that provides meta-information about the event (identification number, occurrence time, description, category, etc...), attributes related to the event payload is intrinsically linked to the intent sought for their processing. For instance, for the description of an event structure pertinent to a credit card fraud, the geographical locations where individual purchase takes place and the amount of money involved are very salient information to capture. Owing to the fact that this audit emphasis on anomalies related to security, the idea was then to adopt an event structure that could allow the re-use of existing security audits tools given the area of network and system audits is already beaming with a plethora of tools. The adoption of an Extensible Configuration Checklist Description Format (XCCDF) like format as a baseline for the primitive events structure was thus to ensure reuse and interoperability with Security Content and Automation Protocol (SCAP) tools. XCCDF is an

XML based format used to specify security checklists and benchmarks amongst others. The overarching purpose of the format is to provide a uniform expression of security checklists, benchmarks, and other configuration guidance, and thereby foster more widespread application of good security practices. With analogy to the XCCDF format, we specify an event with the following attributes:

```
<Name>..\<Name>
<Identification>..\<Identification>
<Description>..\<Description>
<Category>..\<Category>
<Time stamp>..\<Time stamp>
<Value>..\<Value>
<Frequency>..\<Frequency>
<Rule>..\<Rule>
<Probe>..\<Probe>
```

In the context of this work, the most relevant attributes associated to a primitive event include the rule attribute which is the underlining policy based on which the associated probe (either a SCAP tool or an in-house program) could interpret and carry out the specificities of the rule, leading to the detection of the primitive event. Furthermore, the rule attribute encompasses and provides reasoning about the expected and exceptional behaviours and states. The field associated to a rule takes as input a path leading to a file, thus allowing one to define a comprehensive set of policy that should drive the detection of primitive events of interest. A rule could be specified in a logical language such as Etalis (Anicic et al., 2012) and Drools fusion (<http://www.drools.org/>) or programming language including Python as in the case of our implementation. The category attribute is a field we added, for allowing the user to systematically classify events based on their typology and/or interest for the stakeholders. Secondary event attributes such as name, identification, description, timestamp, frequency, denote respectively, the given name unique identifier of the event, a succinct description of it, the time at which the event was created, the frequency at which the tool should be probing the event.

#### 4.2 Event Patterns Specification

We hereafter use the term *event pattern* to refer to any composite event whereby primitive events (leaves of the tree) are associated through a logical operator or connector. An event tree can thus be a simple event patterns or a combination of patterns of different semantic leading to a much complex event specification. Adopting a tree based representation also allows leveraging relevant graph theories for the efficient processing of the event nodes. Our event-based cloud audit and monitoring is based on event

patterns specification using the Yet Another Language for Event Specification or YALES logic (Zhang and Unger, 1996). The choice of YALES lies on its expressiveness but also to the fact that it allows to capture synchronous and asynchronous events and also provides event counter and calendar constructs. YALES distinguishes three types of events: temporal events denoting the explicit time instants in real life and can be deemed relative temporal event when the offset is equal to span. Calendar events express the periodical activities that occur regularly or irregularly many times in terms of real life time measurements and primitive events. A primitive event is considered as an explicit event taking place at a discrete or during a time span.

More precisely, our framework and tool allows both the cloud client and provider to declare and detect the following event patterns or event composite (note that this list of pattern is not exhaustive and new one can be added for detection purpose):

**Disjunction of Events,  $E1 \vee E2 \vee \dots \vee E_n$  or any Pattern:** This pattern of events, occurs when one or more of primitive events  $E1, \dots, E_n$  occurs. The detection algorithm for the ANY pattern is later provided.

**Conjunction of Events,  $A(m, \{E1, E2 \dots E_n\})$**   
**Part of:** This pattern of event occurs if all the constituent events occur. With a slight extension and with the notation of  $A(m, \{E1, E2 \dots E_n\})$  where  $m < n$ , we can express the idea that if at least  $m$  out of  $n$  events occur, the event pattern has happened.

**Sequences of Events,  $E1; E2; \dots; E_n$  | All in Order:** A sequence of events pattern expresses the requirement that occurrence of composite events be strictly in order in time, i.e. no adjacent events that occur at the same time are counted.

**Event Counter,  $C(E, n^+ | n | n+ | n-)$ :** **Over-time, Equals, More, Less:** A way to tell how many times an event has occurred is useful. The event counter as an event pattern is designed to provide this kind of mechanism. Event counter,  $C(E, n^+)$ , will occur upon every  $n$ th occurrence of  $E$ . In this case  $C(E, n^+)$  may be triggered more than once;  $C(E, n)$  is validated at when at the  $n$ th occurrence of  $E$ . Thus  $C(E, n)$  is only triggered once.  $C(E, n+)$  is validated when  $E$  has occurred not less than  $n$  times; while  $C(E, n-)$  will be flagged when  $E$  has occurred less than  $n$  times but at least once.

**Moving Window,  $W(n, E, span)$ :** A moving window uses a moving interval with a fixed span to provide aggregate event information. Here,  $W(n, E, span)$  is used to mean that if there are more than  $n$  of  $E$  occurrences during a time period, then the composite event should be raised.

**Put the Occurrence of an Event in Context of a Sequence of Other Events  $E: C | [E1, E2]$  and  $E \text{ IN } C | [E1, E2]$ :** Events in periods or intervening events are those that occur in a period marked by two reference events  $E1$  and  $E2$ . Two alternative cases can be considered: The first supposes a left-closed and right-open interval that we refer to as *BETWEEN event pattern*; and the second one considering a left-closed and right-closed interval referred to as *IN event pattern*. While both detection patterns *BETWEEN* and *IN* would require the event of interest to occur after the initiating event ( $E1$ ), the *IN* pattern is detected only after the right-end event has occurred as opposed to the *BETWEEN* event.

## 5 FROM EVENTS SPECIFICATION TO AUDIT

The previous section provided the foundation for specifying the patterns of events relevant to the security audit. In this section, we depict how in practice patterns are used and analyzed to support mutual auditability and increase cloud transparency. The mutual audit process and the ensuing increase in transparency between the CSC and the CSP is supported by two main steps once the composite events have been specified: the generation and triggering of software agents for conducting detecting the primitive events and the reasoning on whether an overall event pattern of interest has taken place. This will ultimately lead to the generation of reports and alerts on the dashboard of the cloud stakeholder of interest.

### 5.1 Generation and Triggering of Agents

To conduct the audits, an organization of software agents is used. The first type of agent or *probe agents*, purport to conduct the detection of the primitive events within the pattern specified by the cloud stakeholder. The second type of agent is a single agent in some cases, referred to as *Event receiver* which role is to filter and aggregate the set of inputs that arrive from the probe agents before they are passed to the reasoning engine. In order to ensure the intrinsic link between a primitive event and a probe agent, we adopt the following reference format for agent during their generation: *Agent<EventName>@<Domain>*, where domain refers to the name given to the audit platform; and eventName- a unique name given to the event in the event structure provided in Section 3.1. In practice,



the probes agents actually carrying out the instruction within the *Rule* field of the primitive event, thus triggering an existing security tool or launching a pseudo code before collecting the results of the check. The creation and management of the agents can be done through a MAS platform such as JADE.

## 5.2 Reasoning and Detecting a Pattern

Primitive events alone may not always be significant to portray the emergency of a situation. In contrast, considering a pool of events from the same or different sources within the infrastructure of the CSC/CSP may reveal a pattern that relate to an anomaly or impending risk for the service stakeholder. As mentioned in the previous section, since we have adopted a tree based representation and validation of event patterns, efficiently detecting a pattern of events depends on how the tree is processed, and individual events are handled upon their detection. With this in mind, we have tailored for each of the event pattern specified in Section 3.2, a dedicated algorithm that is resorted by the reasoning engine to determine whether the composite event has materialized. Owing to page limitation, we only provide a description of some amongst them:

**Any:** this operator has a list of events and is validated if one of those events occurs. If the operator above this operator is a count operator, then this operator will reset only the event that validated it. This ensures that any other subsequent event in the list that was partially validated, will still be accounted for.

```
Algorithm:
boolean valid = false
for each event in list {
 if event is validated {
 add event to happenedList
 valid = true
 }
}
return valid
```

**Part Of:** This operator has a list of events and a trigger number. It is validated when the number of events validated in the list is equal or above to the trigger number. If the operator above this operator is a count operator, this operator will reset only the events of the list that are validated. Therefore if another event in the list was partially validated, it will not lose its current state.

```
Algorithm :
clearhappenedList
for each event in list {
 if event is validated {
 add event to happenedList
 }
}
if happenedList size >= trigger {
```

```
 return true
}
return false
```

## 5.3 Audit Reporting

Upon the processing of the events by the reasoning engine, it sends the result to the Report Generator which subsequently displays it at the HMI or Dashboard. Given the adoption of a tree based specification of the composite events to monitor for, the Event designer could also serve the purpose of visual dashboard whereby events detected by the agents get automatically highlighted in a different colour. For further details information on the detected events, another dashboard could be added, giving details on the event's name, primary identifier, date of creation, the probe agent tasked with detecting it, and its status (for instance *Occur or not happened*). Such information could then allow the CSP and CSC to respectively, to detect any nefarious and malicious use of its service by a CSC, detect any security anomaly within their virtual machines and/or held the CSP accountable for a breach of contractual agreement related to security and quality of service.

## 6 APPLICATION TO A REAL SCENARIO

The scenario described thereafter is a real use case we have worked on though the names of the companies involved have been altered.

L-BANK which is actually affiliated to an international Bank, has decided to use the service of D.CLOUD owing to the fact that the local branch was closing down. As information related to the bank customers could not be moved beyond the boundary of the country, it was therefore imperative to found a reliable service that could carry out the archiving in accordance with the legal framework. D.CLOUD was chosen primarily for the fact that archiving was the core of its business and also because of the security reliability (supported by certifications). This means a considerable amount of archives (bank customers' information) were ready to be transferred to D.CLOUD through SFTP connection mode for fast integration. L-BANK was then given an access to D.CLOUD's standard web portal. D-CLOUD could create new archives, search, restore, and consult individual archives. This portal is accessible directly over Internet (HTTPS) or through a VPN. At the end of the retention period, the

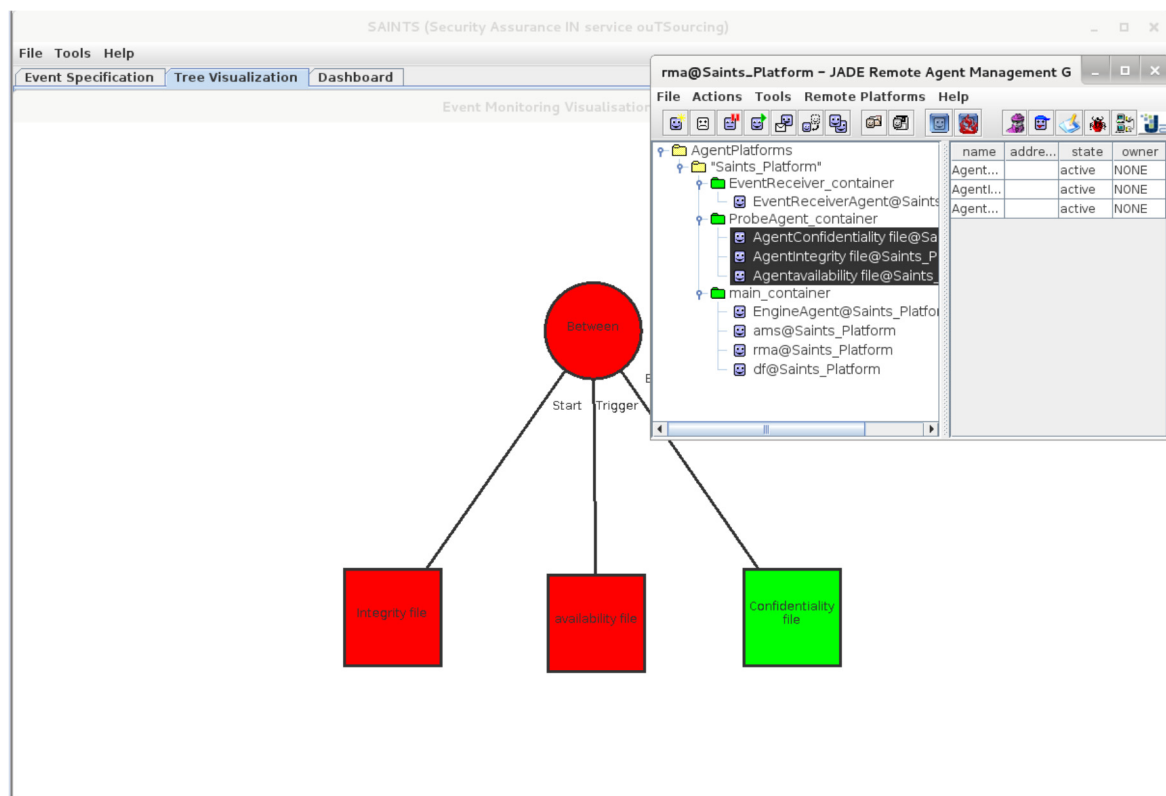


Figure 2: Launch of a JADE based agents and visualization of archive's related events.

documents are available for deletion or archiving. From L-BANK's view point the following events were identified as being relevant for monitoring:

**-Confidentiality of the ANY (C1, C2) where**

C1: An attacker or the cloud provider staff may access the archive stored on the cloud provider server without authorisation.

C2: An error in storage or a bad virtual separation may cause a breach of confidentiality.

**- Integrity of the Archives ANY (I1, I2, I3)**

I1: Unauthorised modification by hackers may occur when the archive is stored on the CSP's server.

I2: A loss of integrity of the archive due to a corruption of data may occur when the cloud client consults the archive through the archive portal.

I3: In the documents management configuration, when the client consults the archive, an error may cause a loss of integrity or a deletion of the archive.

**-Availability of the Archives ANY (A1, A2) where**

A1: The platform may be unavailable due to a traffic overloading or a denial of service attack.

A2: Transmission and communication errors may occur when the archive is sent to the cloud client.

The D-CLOUD in turn was concerned with unwarranted modification and activities from L-BANK with consequences on its services:

ANY (M1, M2) where:

M1: The cloud client may intentionally or not upload a malicious file as an archive.

M2: an error in the system or archive maintenance may result in unwanted changes.

Noteworthy, the specification of composite events combining events from the three groups is possible as it was the case during the application to the use case (Figure 1-2). Similarly most of the events above listed can be further specified as composite events of their own, but due to page restriction such an exercise will not be conducted in this paper.

## 7 CONCLUDING REMARKS

This paper has reported on an initiative for addressing security transparency and allowing mutual auditability within a cloud realm. Given that the use of raw logic based language may prove a challenge for some stakeholders wishing to engage in the systematic audit and monitoring of security and QoS related parameters, the choice of a tree based representation was made. Another key aspect of the approach is the adoption of the XCCDF format, which enables the reuse of existing security management tools.

The initial prototype based on the concepts and algorithms presented has been validated on an electronic archiving platform with the event specification and detection console allocated to a dedicated virtual machine, while the multi agent system platform JADE has been adopted for the specification and management of agent entrusted with the role of detecting primitive events as can be seen in Figure 2. NAGIOS plugins (Pervilä, 2007) along with other tailored programs were developed for the detection of primitive events within the Infrastructure.

The initial results were very encouraging as most of the security events of concerns provided by the SaaS provider and consumer and specified using the Event designer were detecting, by simulating alterations and attacks targeting the archived files. Furthermore, the capacity of the VM required for hosting the whole application (Event Designer and multi-agent detection platform) was confined to a 2 Go of RAM and in single CPU. Nonetheless, further applications are envisaged for better appraising the effect of deploying simultaneously a multitude of agents for detecting and reporting events of interest.

## ACKNOWLEDGEMENTS

This work has been conducted in the context of the SAINTS project, financed by the national fund of research of the Grand Duchy of Luxembourg (FNR) under grant number C12/IS/3988336. The authors also thank Maimouna Seck and Charles Hubert Duthilleux for their work on implementing the tool.

## REFERENCES

- Anicic D., Rudolph S., Fodor P., Stojanovic N.: Stream reasoning and complex event processing in ETALIS. *Semantic Web* 3(4): 397-407 (2012).
- Bellifemine F., Caire G., Poggi A., Rimassa G. 2008 JADE: A software framework for developing multi-agent applications. Lessons learned. *Information & Software Technology* 50(1-2): 10-21.
- Carasso D. (2012) Exploring Splunk, CITO Research, New York.
- Chen Y, Paxson V, Katz RH (2010) What's New About Cloud Computing Security? Report EECS Department, University of California, Berkeley, <http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html>.
- Dölitzscher F., Knahl M., Reich C., Clarke N.L. 2013 Anomaly Detection in IaaS Clouds. In *proceedings of CloudCom* (1) 387-394.
- Etzion O., Niblett P. 2010. Event Processing in Action. Manning Publications Company 2010, ISBN 978-1-935182-21-4, pp. I-XXIV, 1-360.
- Lorenzoli D., Spanoudakis G. 2010 EVEREST+: Runtime SLA Violations Prediction. In: *Proceedings of the 5th Middleware for Service-oriented Computing Workshop*, ACM.
- Luckham D. C. (2005) The power of events - an introduction to complex event processing in distributed enterprise systems. ACM 2005, ISBN 978-0-201-72789-0, pp. I-XIX, 1-376.
- Ganzha M, Paprzycki M. (2014): Agent-oriented computing for distributed systems and networks. *J. Network and Computer Applications* 37: 45-46 (2014). McAfee and Guardian Analytics. 2012. Dissecting. Operation High Roller. Accessed 10 December 2014. From: <http://www.mcafee.com/us/resources/reports/rp.operation-high-roller.pdf>.
- Núñez D., Fernandez – Gago C., Pearson S., Felici M. 2013 A Metamodel for Measuring Accountability Attributes in the Cloud. In: *Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013)*, IEEE.
- Ouedraogo M., Khadraoui D., Mouratidis, H. and Dubois E. (2012): Appraisal and reporting of security assurance at operational systems level. *Journal of Systems and Software* 85(1): 193-208 (2012).
- Ouedraogo M, Mouratidis M (2013) Selecting a cloud service provider in the age of cybercrime, *Computers & Security*, vol.38, pp.3-13 Special issue on Cybercrime in the Digital Economy, Elsevier.
- Ouedraogo M., Kuo C.T, Tjoa S., Preston D, Dubois E., Simões P., Cruz T.: Keeping an Eye on Your Security Through Assurance Indicators. In *proceedings of SECUREPT 2014*: 476-483.
- Pervilä, M.A., 2007. Using Nagios to monitor faults in a self-healing environment. In: *Seminar on Self-Healing Systems*. University of Helsinki.
- Rak M, Liscardo L, Aversa R 2011. A SLA-based interface for security management in cloud and GRID integrations. In: *Proceedings of the 7th International Conference on Information Assurance and Security (IAS)*, pp.378-383, IEEE.
- Robert J. Zhang, Elizabeth A. Unger (1996) Event Specification and Detection Technical report TR CS-96-8, 1996, Kansas State University.
- Sunyaev A., Schneider S. 2013. Cloud services. certification. *Communication of the ACM* 56(2): 33-36, ACM digital Library.
- Waltermire D., Schmidt, C., Scarfone K.
- Winkler V. (2011) Securing the cloud- cloud computer. security techniques and tactics. Syngress.
- Ziring N. 2012. Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2, *NIST Interagency Report 7275Revision 4*, National Institute of Standards and Technology Gaithersburg, MD 20899-89.

# **MOBILE CLOUD COMPUTING AND SERVICES**



## **FULL PAPERS**



# The Case for Visualization as a Service

## *Mobile Cloud Gaming as an Example*

Abdelmounaam Rezgui<sup>1</sup> and Zaki Malik<sup>2</sup>

<sup>1</sup>*Department of Computer Science & Engineering, New Mexico Tech, Socorro, NM, U.S.A.*

<sup>2</sup>*Department of Computer Science, Wayne State University, Detroit, MI, U.S.A.  
rezgui@cs.nmt.edu, zaki@wayne.edu*

**Keywords:** Mobile Cloud Gaming, Visualization as a Service, GPUs, Offloading.

**Abstract:** In recent years, significant progress has been made to improve the power efficiency of mobile devices. In particular, new GPU architectures have made it possible to run compute-intensive applications directly on battery-powered mobile devices. In parallel, research is also being conducted in the area of application offloading, the process of running compute-intensive tasks on cloud servers and delivering the results of these computations to mobile devices through their wireless interfaces. It is important to understand the power consumption implications of each of these two options. In this paper, we use *mobile cloud gaming* as an example to evaluate and compare these two alternatives (running games on the cloud or on mobile devices.) Based on this comparison, we introduce the concept of *Visualization as a Service* (VaaS) as a new model to design and implement graphics-intensive applications for mobile devices. In this model, advanced visualization capabilities (e. g., interactive visualization of high resolution videos/images) would be provided to mobile users as a service via the Internet. We show through actual hardware specifications that, despite the recent introduction of ultra low power GPUs for mobile devices, it remains far more power efficient to offload graphics-intensive tasks to the cloud. The associated latency can still be tolerated in most applications.

## 1 INTRODUCTION

The growth in the use of mobile devices (mobile phones, tablets, and ultra mobile PCs) is driving a phenomenal market shift. A 2013 Gartner report (Table 1) predicted that, by 2017, device shipments will reach more than 2.9 billion units, out of which 90% will be mobile devices (Milanesi et al., 2013). This growth is accompanied by an equally phenomenal boom in mobile applications. According to the research firm MarketsandMarkets, the total global mobile applications market is expected to be worth \$25 billion by 2015 (up from about \$6.8 billion in 2010) (MarketsandMarkets, 2010). A 2012 study by the Application Developers Alliance found that 62% of the U. S. online population owned app-capable devices and that 74% of those device owners use mobile applications. As the rendering capabilities of mobile devices improves, mobile applications are becoming increasingly graphics-intensive. This requires intensive computations that quickly drain the device's battery.

Several solutions are being developed to reduce power consumption in graphics-intensive mobile applications. Some solutions are to be used at develop-

ment time while others are to be used when the application is running. The former focus on tools that help developers estimate power consumption at development time. For example, in (Thompson et al., 2011), the authors present SPOT (System Power Optimization Tool), which is a model-driven tool that automates power consumption emulation code generation. In (Hao et al., 2013), the authors use program analysis during development time to estimate mobile application energy consumption. The latter type of solutions focus on reducing power consumption of hardware components such as the GPU or NIC at run-time. Examples include the *racing to sleep* technique (that sends data at the highest possible rate), wide channels, and multiple RF chains (Halperin et al., 2010).

A third alternative is application offloading, the process of running compute-intensive tasks on servers (often in the cloud) and delivering the results of these computations to mobile devices through their wireless interfaces. However, these wireless interfaces also may consume substantial amounts of power when receiving large amounts of data as is typical in many modern, interactive, graphics-intensive mobile applications. It is therefore important to understand the



Table 1: Worldwide Devices Shipments by Segment (Thousands of Units) (Milanesi et al., 2013).

Device Type	2012	2013	2014	2017
PC (Desk-Based and Notebook)	341,263	315,229	302,315	271,612
Ultramobile	9,822	23,592	38,687	96,350
Tablet	116,113	197,202	265,731	467,951
Mobile Phone	1,746,176	1,875,774	1,949,722	2,128,871
<b>Total</b>	<b>2,213,373</b>	<b>2,411,796</b>	<b>2,556,455</b>	<b>2,964,783</b>

power consumption implications of the two alternatives: running the graphics-intensive application on the cloud or on the mobile device itself. In this paper, we use *mobile cloud gaming* as an example to analyze and compare these two alternatives in terms of power consumption. We show through actual hardware specifications that, despite the recent introduction of ultra low power GPUs for mobile devices, it remains far more power efficient to offload graphics-intensive tasks to cloud servers. To make our discussion concrete, we focus on two cases of mobile devices: (i) notebooks and (ii) smartphones. In both cases, we only consider gaming using the device's WiFi interface not its cellular interface. The reason for this is that the high latency and high cost make mobile cloud gaming using cellular networks (UMTS, LTE, etc.) an impractical alternative for most consumers. We will elaborate on this in Section 3.

This paper is organized as follows. We first give an overview of mobile cloud gaming. In Section 3, we contrast cellular-based and WiFi-based mobile cloud gaming from the perspectives of power consumption, throughput, latency, and cost. In Sections 4 and 5, we present power consumption trends in modern mobile GPUs and 802.11 network cards. In Section 6, we quantitatively evaluate and compare power consumption of a gaming session in the two previously mentioned scenarios in the context of notebooks. We repeat the same analysis for smartphones in Section 7. In Section 8, we give the conclusions from our study and suggest directions for future research.

## 2 WHAT IS MOBILE CLOUD GAMING?

Mobile cloud computing (MCC) is the process of offloading compute-intensive tasks from mobile devices to cloud servers (Soliman et al., 2013; Shiraz et al., 2013). The purpose is often to save power on the mobile device and/or access servers with much higher computing power. A prime example of MCC is *mobile cloud gaming* which is the process of providing video games on-demand to consumers through the use of cloud technologies. One benefit is that

the cloud, instead of the user's device, carries out most of the computations necessary to play the game, e. g., complex graphical calculations. This is obviously a tremendous advantage in case the player uses a battery-powered, mobile device. Even when power is not a critical issue for the user's device, cloud gaming still provides other cloud services, e. g., storage. Cloud gaming enables power savings also on the cloud itself as it makes it possible that several players simultaneously share cloud GPUs. For example, Nvidia's VGX Hypervisor manages GPU resources to allow multiple users to share GPU hardware while improving user density and the utilization of GPU cycles (Nvidia, 2015a). To illustrate, a single cloud gaming-capable Nvidia VGX K2 unit requires 38 Watts per cloud user (Nvidia, 2015c), whereas a comparable single-user Nvidia GTX 690 consumer unit requires 300 Watts to operate (Nvidia, 2015b). In this case, cloud gaming can reduce the overall graphics-related power consumption by 87%.

## 3 CELLULAR-BASED VS. WiFi-BASED MOBILE CLOUD GAMING

Mobile cloud gaming may be achieved using cellular connections or WiFi connections. While both options are technically possible and relatively comparable in terms of power consumption, the WiFi option seems much more attractive when we consider throughput, latency and cost. In this section, we present results from recent studies analyzing power consumption, throughput, latency, and cost in both scenarios:

### 3.1 Power Consumption and Throughput

In (Carroll and Heiser, 2010), the authors analyze power consumption of smartphones. In particular, they studied power consumption of the two main networking components of the device: WiFi and GPRS (provided by the GSM subsystem). The test consisted of downloading a simple file via HTTP using wget. The files contained random data, and were 15

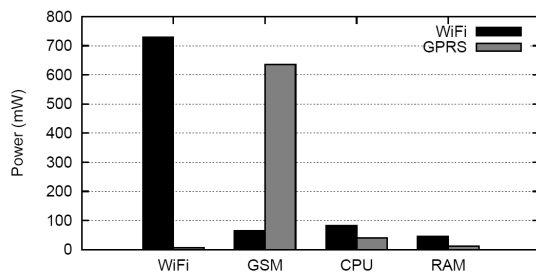


Figure 1: Power Consumption of WiFi and GSM Modems, CPU, and RAM (Carroll and Heiser, 2010).

MiB for WiFi, and 50 KiB for GPRS. While the test was not a gaming session, it still gave valuable insights. The experiments showed that WiFi achieved a throughput of  $660.1 \pm 36.8$  KiB/s, and GPRS  $3.8 \pm 1.0$  KiB/s. However, they both show *comparable* power consumption far exceeding the contribution of the RAM and CPU (Figure 1). The experiments also showed that, with the increase in throughput possible using WiFi, CPU and RAM power consumption also increases reflecting the increase in the cost of processing data with a higher throughput.

### 3.2 Latency

In the context of mobile cloud gaming, latency refers to the timespan between a user's action and the corresponding reaction (Lampe et al., 2013), e. g., time between the action of pressing a button and seeing a character in the game move as a result of that action. High latency is a real challenge in mobile cloud gaming. Wireless connections (WiFi and cellular) and even wired residential end host round trip times (RTTs) can exceed 100 ms (Lee et al., 2014). To many gamers, this is the point when a game's responsiveness becomes unacceptable. A recent effort to reduce latency in mobile cloud gaming is Outatime, a speculative execution system for mobile cloud gaming that is able to mask up to 250 ms of network latency (Lee et al., 2014). It produces speculative rendered frames of future possible outcomes, delivering them to the client one entire RTT ahead of time.

While latency is an issue in both cellular-based and WiFi-based mobile gaming, WiFi connections typically have much less latency than cellular connections (Lampe et al., 2013).

### 3.3 Cost

Cost is also a major factor in favor of WiFi-based mobile cloud gaming. For example, in (Lampe et al., 2013), the authors give an analytical assessment that shows that the cost (from cellular data transfer) of a

gaming session of one hour would be about 2.36 Euros without including the likely additional usage fee to be paid to the cloud gaming provider.

As we may conclude from the previous discussion, WiFi-based mobile cloud gaming is currently more practical than cellular-based mobile cloud gaming. We, therefore, limit our discussion to this option in the remainder of this paper.

## 4 POWER CONSUMPTION TRENDS IN MODERN MOBILE GPUS

It is currently generally true that GPUs offering a good rendering capability consume much power for operation and cooling. To illustrate the current power consumption trends of mobile GPUs, we list in Table 2 some modern notebook GPUs and their respective power consumptions. The table suggests that playing a game on a notebook equipped with one of the listed GPUs may not be a viable option. For example, the Dell Precision M6700 mobile workstation (which Dell touted as the “world’s most powerful 17.3” mobile workstation”) is equipped with the Nvidia Quadro K5000M GPU. The configuration can pull 98 Watts of power when running on battery under a heavy CPU or GPU load. This means that it would be possible to drain the system battery in about an hour (Notebook Review, 2015). Even with this limited ability to support long running, compute-intensive applications, this configuration costs more than \$2K. Better battery life may be possible but with much more expensive configurations. Efforts are underway to develop mobile devices with power efficient computing components (e. g., multicore CPUs and ultra low power GPUs) and batteries that can run compute-intensive applications (e. g., games and other graphics-intensive applications) for many hours. For example, Nvidia is introducing Tegra 4, a mobile GeForce GPU with up to 72 custom cores, a quad-core ARM Cortex-A15 processor with a fifth Companion Core that further improves performance and battery life. According to Nvidia, a battery of a capacity of 38 watt-hours would be sufficient to operate a Tegra 4 mobile device running a gaming application between 5 and 10 hours. This corresponds to a power consumption (for the entire device) of 4 to 8 Watts (Hruska, 2013). However, mobile devices with these high-end configurations will remain beyond the reach of average users for the foreseeable future.

Table 2: Energy Consumption of Some Modern Notebook GPUs.

GPU Card	Power Consumption (W)
NVIDIA GeForce GTX 680M SLI	2 x 100
AMD Radeon HD 7970M Crossfire	2 x 100
NVIDIA GeForce GTX 680MX	122
NVIDIA GeForce GTX 675M SLI	2 x 100
GeForce GTX 680M	100
Quadro K5000M	100
AMD Radeon HD 7970M	100

## 5 POWER CONSUMPTION TRENDS IN MODERN NOTEBOOK NICs

The original 1997 release of the IEEE 802.11 standard operated in the 2.4 GHz frequency band and provided a data bit rate of 1 to 2 Mb/s. The standard release approved in February 2014 (known as 802.11ad) operates in the 2.4/5/60 GHz frequency bands and provides a data bit rate of up to 6.75 Gbit/s. While higher bit rates often translate into higher power consumption, this is less true in recent ultra-low power 802.11 standards. For example, today's fastest 3 antenna 802.11n device can achieve 450 Mbps. A single antenna 802.11ac device can achieve a similar bit rate with similar power consumption. This means that a typical tablet with single antenna 802.11n 150Mbps WiFi can now support 450 Mbps with 802.11ac without any increase in power consumption or decrease in battery life (Netgear, 2012).

## 6 GRAPHICS-INTENSIVE APPLICATIONS: GPUS VS. NICs

To assess the benefits of using a mobile GPU versus offloading to the cloud, we consider gaming as it is a typical example of graphics-intensive mobile applications. Specifically, we consider four modern games that rely heavily on GPUs. We compare two scenarios in terms of power consumption. In the first scenario, the game is run entirely on the mobile device and uses only its GPU. In the second scenario, we consider an execution where the game is run on a cloud server and the mobile device only receives and renders sequences of frames produced by the server. We analytically evaluate power consumption in these two scenarios and show that, with modern wireless technology, offloading is a far better alternative to running graphics-intensive applications using the device's GPU. To make the comparison even

more in favor of the GPU-based alternative, we ignore the power consumption of the device's disk. We assume that, when a graphics-intensive application is run on a mobile device, most of the power is consumed by the device's GPU. This is becoming increasingly true with the wide availability of mobile devices with solid-state disk drives.

To compare power consumption in the two scenarios, we first present a simple model that captures the interactions between the player and the gaming application. We will assume that, during a given gaming session of duration  $t$ , the player takes an action after every  $r$  seconds in average. We call  $r$  the *reactivity* of the player. To respond to the player's action, the application generates a video stream of length  $v$  seconds.<sup>1</sup> So, during the entire session, the application generates  $t/r$  video sequences whose length is  $v$  seconds each. In total, the application generates  $tv/r$  seconds of video during the given gaming session.

### 6.1 Scenario 1: Gaming using the Mobile Device's GPU

To assess the power consumed by a notebook's GPU in a gaming session, we used the benchmark presented in (NoteBookCheck, 2014). The benchmark has a large number of notebook GPUs and a number of popular games. For each combination of game and GPU card, the benchmark gives the average number of frames per second (fps) that the GPU card achieves with four different resolution levels: Low (L), Medium (M), High (H), and Ultra (U). The benchmark considers that a frame rate of 25 fps is sufficient for fluent gaming. For the purpose of this study, we considered four GPU cards and four 2014 games, namely GRID Autosport, Watch Dogs, Titanfall, and Thief. Table 3 gives the frame rates ob-

<sup>1</sup>This is to simplify our discussion. In practice, the application likely generates two video sequences of different lengths in response to two different actions.

tained in the given combinations<sup>2</sup>. The resolutions in the table are as follows: Low (1024x768), Medium (1366x768), High (1920x1080 for the first two games and 1366x768 for the last two games), and Ultra (1920x1080). Table 3 also gives power consumption for the four GPU cards.

As an example, consider a mobile device equipped with a GPU of type Nvidia GeForce GTX 850M. As shown in Table 3, this GPU card will consume between 40 and 45 Watts in one hour. We will show that offloading to the cloud (Scenario 2) brings an order of magnitude reduction in terms of the power consumed by the mobile device.

## 6.2 Scenario 2: Mobile Cloud Gaming

We now evaluate the required data bit rate that the NIC card of a notebook would have to support to achieve the same game fluency (i. e., 25 fps) for one of the four GPU cards of Table 3. As an example, consider again the Nvidia GeForce GTX 850M (which is the best of the four GPUs in terms of power consumption.) For the game GRID Autosport and for low resolution, the Nvidia GeForce GTX 850M is able to support 166.65 fps which is:  $166.65 \times 1024 \times 768 \times 8 = 1048471142.4$  bits/second (assuming a color depth of 8 bits/pixel). Thus the NIC card would have to operate at a bit rate of about 1.05 Gb/s. A similar computation for the Ultra high resolution level gives us a bit rate of:  $34.7 \times 1920 \times 1080 \times 8 = 575631360$  bits/second. Thus, to support the same gaming fluency at the Ultra-high resolution level, the NIC would have to operate at 575 Mb/s. Note that the required bit rate at the Ultra-high resolution level is almost half of that of the required bit rate at the low resolution level because the GPU supports a lower frame rate at the Ultra-high resolution level. To support these bit rates, the mobile device's NIC would have to be 802.11ad compliant. The 802.11ad standard is able to support bit rates up to 6.77 Gbit/s.

To evaluate the power consumed by the device's wireless networking card during the considered gaming session, we will assume a model of a wireless networking card that consumes  $\rho_{tx}$  watts when in transmit mode and  $\rho_{rx}$  watts when in receive mode. With single-antenna 802.11 devices, the devices cannot send and receive simultaneously. This normally implies that one has also to take into account the cost of frequently switching the device's radio between the transmit and the receive mode. This, however, is changing as mobile devices are now increas-

ingly being equipped with MIMO (multiple-input and multiple-output) technology enabling the use of multiple antennas at both the transmitter and receiver. In fact, Mobile Experts predicts that the use of MIMO technology will reach 500 million PCs, tablets, and smartphones by 2016 (Madden, 2011). As a result, we will only take into account power consumption due to transmission, reception, and idling. We will note the power consumption of the radio during idling by  $\rho_{id}$ .

Let  $\mu_t$  and  $\mu_r$  be the transmission and reception rates respectively. Let  $l$  be the length of the packet sent to the application when the player takes an action. The time needed to transmit this packet is then:  $l/\mu_t$ . Let  $t$  be the length of the entire gaming session (in seconds). During the time  $t$ , the device transmits  $t/r$  times where  $r$  is the player's reactivity (defined earlier). The total time during which the device transmits is therefore:

$$\frac{tl}{r\mu_t} \text{secs.} \quad (1)$$

The corresponding power consumption during the period of time  $t$  is:

$$P_{tx} = \frac{\rho_{tx}tl}{r\mu_t} \quad (2)$$

To evaluate the power consumed by the device's receiver, recall that our model assumes that, to respond to each player's action, the application generates a video stream of length  $v$  seconds. The device spends  $v/\mu_r$  seconds to receive each of these video streams. Since we have  $t/r$  of these video streams during the considered time period of length  $t$ , the device's NIC receives video streams during:

$$\frac{tv}{r\mu_r} \text{secs.} \quad (3)$$

Let  $P_{rx}$  be the power that the device's NIC consumes to receive the  $t/r$  video sequences.  $P_{rx}$  can be given by:

$$P_{rx} = \frac{\rho_{rx}tv}{r\mu_r} \quad (4)$$

The device's NIC is in the idle mode when it is not transmitting and not receiving. This occurs during:

$$t - \frac{tl}{r\mu_t} - \frac{tv}{r\mu_r} \text{secs.} \quad (5)$$

The power consumed by the device's NIC while idling is therefore:

$$P_{id} = \rho_{id}t(1 - \frac{l}{r\mu_t} - \frac{v}{r\mu_r}) \quad (6)$$

<sup>2</sup>The missing value in the last row corresponds to a test that could not be run because the GPU card could not support a sufficiently acceptable frame rate.

Table 3: Average Frame Rate of Some Combinations of GPU cards, Games, and Resolutions.

GPU Card	GRID Autosport				Watch Dogs				Titanfall				Thief			
	L	M	H	U	L	M	H	U	L	M	H	U	L	M	H	U
GeForce GTX 770M (75 Watts)	199.6	130.3	92.6	46.5	80.7	66.1	27.7	19.8	60	60	59.3	48.3	57.1	51.3	46.8	26.6
GeForce GTX 860M (60 Watts)	192.15	109.65	88	47.2	71.2	60.7	27.7	18.9	60	60	59.5	42.4	60.5	52.7	44	23.95
GeForce GTX 850M (40-45 Watts)	166.65	99.33	68.3	34.7	61.8	52.3	20.75	14.7	60	59.7	53.25	34.3	46.45	39.6	36.65	18.2
GeForce GTX 765M (50-75 Watts)	191.9	130.7	74.1	34.8	81.3	56.9	21.1		60	59.7	54.3	35.6	58.2	43.1	37	19.1

Table 4: Power Consumption for the Intel Dual Band Wireless-AC 7260 802.11ac, 2x2 Wi-Fi Adapter (Hewlett Packard, 2013).

Mode	Power (mW)
Transmit	2000
Receive	1600
Idle (WLAN associated)	250
Idle (WLAN unassociated)	100
Radio Off	75

Let  $P_{NIC}(t)$  be the power consumed by the wireless NIC during the  $t$ -second gaming session.  $P_{NIC}(t)$  is then:

$$\begin{aligned}
 P_{NIC}(t) &= P_{tx} + P_{rx} + P_{id} \\
 &= \frac{\rho_{tx} t l}{r \mu_t} + \frac{\rho_{rx} t v}{r \mu_r} + \rho_{id} t \left(1 - \frac{l}{r \mu_t} - \frac{v}{r \mu_r}\right)
 \end{aligned}$$

In practice, one must consider values for  $\rho_{rx}$  that accommodate high reception rates (for high definition gaming) and values for  $\rho_{tx}$  that correspond to low transmission rates since the user's actions usually translate into short packets.

To illustrate, we consider the case of an HP Elite-Book Folio 1040 G1 Notebook PC. This notebook is equipped with the Intel Dual Band Wireless-AC 7260 802.11ac Wi-Fi Adapter whose power consumption is given in Table 4) (Hewlett Packard, 2013). Assume that the NIC card is 80% of the time in reception mode, 10% of the time in transmit mode, and is idle (but associated) 10% of the time. If we apply our power model to this WiFi adapter, power consumption in one hour would be (approximately):

$$\begin{aligned}
 P_{NIC}(t) &= P_{tx} + P_{rx} + P_{idle} \\
 &= 0.1 \times 2000 + 0.8 \times 1600 + 0.1 \times 250 \\
 &= 1505 \text{ milliwatts}
 \end{aligned}$$

assuming the highest Rx and Tx power levels.

Considering the example of a notebook equipped with a GPU of type Nvidia GeForce GTX 850M (Section 6.1), we can estimate that, in one hour, the GPU card will consume about between  $0.8 \times 40$  W and  $0.8 \times 45$  W, i.e., between 32W and 36W, assuming a GPU utilization of 80% similar to our assumption of the NIC card being in the Rx mode 80 % of the time.

From the results obtained in the two scenarios, it is clear that using the wireless networking interface in a gaming session consumes much less power than using a modern GPU card installed on the same device. Specifically, the power consumed using the wireless card would be around  $(1505 / 34000) \times 100$ , i.e., around 4.42% of the power consumed by the on-device GPU.

## 7 MOBILE CLOUD GAMING USING SMARTPHONES

We now compare power consumption between GPU-based gaming and cloud-based gaming on smartphones.

### 7.1 Power Consumption of GPU-based Gaming on Smartphones

In (Kim et al., 2015), the authors measured power consumption of a Qualcomm Adreno 320 GPU in a Google Nexus 4 smartphone. They used two games in their tests: Angry Birds (2D game) and Droid Invaders (3D game). The authors report results for a gaming session that lasted 560 seconds for Angry Birds and 505 seconds for Droid Invaders. Throughout the two gaming sessions, power consumption remained approximately at around 1750 mW for Angry Birds and at around 2000 mW for Droid Invaders. We will use the average of these two numbers (1875 mW) as an estimate of the average power consumption of both 2D and 3D games.

Table 5: Frame Rates for the Adreno 320 GPU on a Google Nexus 4 and on a Samsung Galaxy S4 using the Manhattan Benchmark (GFXBench, 2015)

Smartphone Model	GPU	Resolution	Frame Rate
Google Nexus 4 (LG E960)	Adreno 320	1196 x 768	9.2
Google Nexus 5	Adreno 330	1794 x 1080	10.1
Samsung GT-I9507 Galaxy S4	Adreno 320	1920 x 1080	5.4
Samsung GT-I9515 Galaxy S4 Value Edition	Adreno 320	1920 x 1080	5.1
Samsung Galaxy S4 Active (GT-I9295, SGH-I537)	Adreno 320	1920 x 1080	5.1
Samsung Galaxy S4 (GT-I9505, GT-I9508, SC-04E, SCH-I545, SCH-R970, SGH-I337, SGH-M919, SPH-L720)	Adreno 320	1920 x 1080	5.1

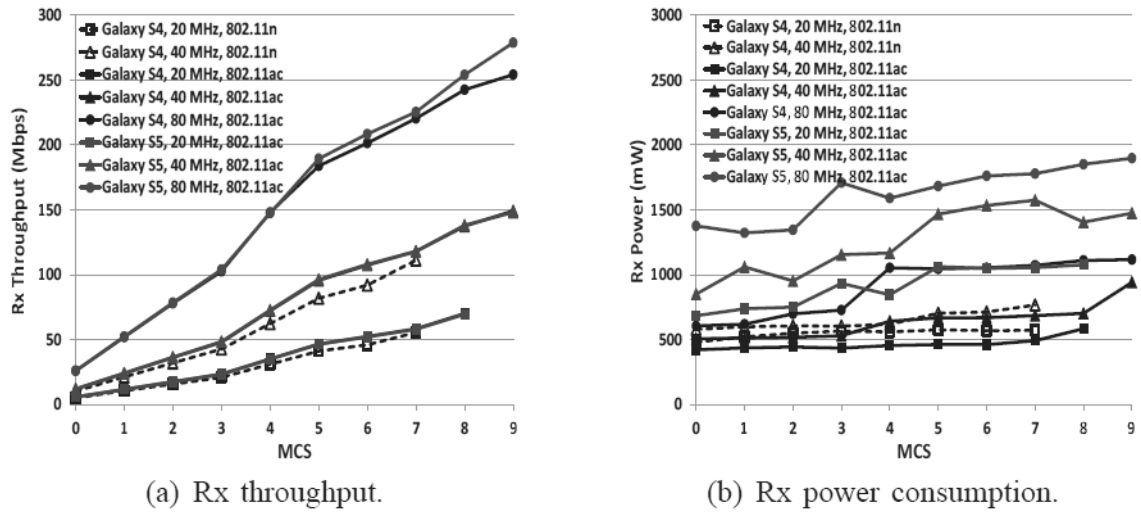


Figure 2: 802.11ac Throughput and Power Comparison for Galaxy S4 and Galaxy S5 with a Channel Width of 20/40/80 MHz and FA on. (Saha et al., 2015).

## 7.2 Power Consumption of Cloud-based Gaming on Smartphones

To compare power consumption of cloud-based gaming with GPU-based gaming, we first need to evaluate the NIC bit rate that would be necessary to provide a gaming experience comparable to the one achieved through GPU-based gaming. For this, we used results from the GFXBench 3.0 benchmark, a cross-platform OpenGL ES 3 benchmark designed for measuring graphics performance, render quality and power consumption on several types of devices including smartphones. In particular, the benchmark has battery and stability tests that measure the devices battery life and performance stability by logging frames-per-second (fps) performance and expected battery running time while running sustained game-like animations (GFXBench, 2015). We focused on results for the Adreno 320 GPU on a Google Nexus 4, which is the same configuration used in the GPU-based scenario of the previous section.

Table 5 shows the frame rate for several tests using

the Manhattan benchmark (GFXBench, 2015). Row 1 of the table shows that the Adreno 320 GPU on a Google Nexus 4 achieved a frame rate of 9.2 fps. Considering this frame rate and the given resolution (1196 x 768), the NIC bit rate that would be necessary to achieve a similar gaming experience can be derived as:  $9.2 \times 1196 \times 768 \times 24 \text{ (bits/pixel)} = 202810982.4 \text{ bps} \approx 203 \text{ Mbps}$ .

We now turn to evaluating the power needed on the NIC to sustain this bit rate. For this, we use the results from (Saha et al., 2015) where the authors experiment with a variety of smartphones supporting different subsets of 802.11n/ac features. In particular, the authors measured throughput and power consumption in a Galaxy S4 using different configurations. Based on their findings for the Galaxy S4 used in the experiment, only 802.11ac offers Rx throughput levels sufficient for the considered gaming bit rate (of 203 Mbps). Figure 2 (reproduced from (Saha et al., 2015)) shows that the best Rx throughput with 802.11ac was about 250 Mbps. Power consumption in this case was about 1100 mW.

Table 6: Power Consumption (in mW) in Non-Communicating Modes. (Saha et al., 2015).

Configuration	PSM	Idle
802.11n, 20 MHz, SS	$24 \pm 16$	$398 \pm 7$
802.11n, 40 MHz, SS	$25 \pm 5$	$413 \pm 2$
802.11ac, 20 MHz, SS	$22 \pm 9$	$374 \pm 7$
802.11ac, 40 MHz, SS	$20 \pm 9$	$425 \pm 3$
802.11ac, 80 MHz, SS	$19 \pm 10$	$529 \pm 11$

The authors did not provide measurements for the throughput and power consumption in transmit mode with 802.11ac. They, however, measured throughput and power consumption in transmit mode with 802.11n. Figure 3 shows their results. In particular, the results show that it is possible to achieve a Tx throughput of more than 40 Mbps with as little power as 800 mW. Note that, in a cloud-based gaming session, a Tx throughput of 40 Mbps is typically sufficient. The authors also measured power consumption of the Galaxy S4 when it is in non-communication modes, i.e., power saving mode (PSM) or idle. Their results (Table 6) show that the highest 802.11ac power consumption in PSM was 31 mW and that the highest 802.11ac power consumption when idle was 540 mW. The relatively high idle mode power consumption of larger channel widths (80 Mhz) has also been observed by other studies (e. g., (Zeng et al., 2014)).

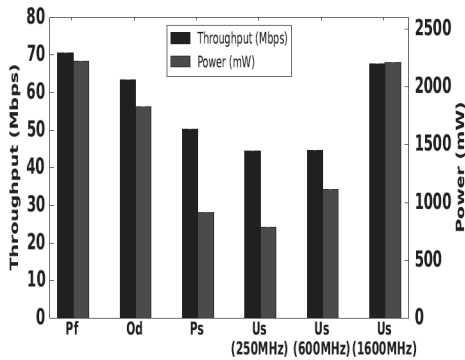


Figure 3: Comparison of Different CPU Governors/Frequencies for Galaxy S4 (802.11n) (Saha et al., 2015).

Based on all the previous results from (Saha et al., 2015) and assuming that, in a cloud-based gaming session, the device's 802.11 adapter spends 80% of the time receiving, 10% of the time transmitting, and 10% of the time idle, the total power consumed in one hour by the 802.11 adapter would be:

$$\begin{aligned}
 P_{NIC}(t) &= P_{Tx} + P_{Rx} + P_{idle} \\
 &= 0.1 \times 800 + 0.8 \times 1100 + 0.1 \times 540 \\
 &= 1014 \text{ milliwatts}
 \end{aligned}$$

Comparing power consumption in the two scenarios: using GPU-based gaming (which is 1875 mW as derived in Section 7.1 and cloud-based gaming (which is 1014 mW as derived in this section), we conclude that, in the considered smartphone configuration, cloud-based gaming can potentially result into a power saving of about 46%.

## 8 CONCLUSION

We presented a comparative analysis between two scenarios of mobile gaming, one that relies entirely on the GPU of the mobile device and one where the gaming application runs on the cloud. We analytically evaluated and compared power consumption in these two scenarios. Based on our analysis, we argue that the idea of Visualization-as-a-Service (VaaS) is a viable computing model that enables the users of mobile devices with limited power capabilities to still use long running graphics-intensive applications. In this model, advanced visualization capabilities would be provided to users as a service via the Internet.

Two research directions are worth studying: (i) the impact of protocol (TCP/UDP/IP) overhead and (ii) the impact of the CPU overhead for processing the large number of packets typical in cloud gaming. We believe that considering these two types of overhead will provide a more accurate assessment of the benefits of cloud-based gaming over GPU-based gaming.

## ACKNOWLEDGEMENTS

The first author gratefully acknowledges travel support from the Institute for Complex Additive Systems Analysis (ICASA) at New Mexico Tech for presentation of this paper at the CLOSER'2015 Conference.

## REFERENCES

- Carroll, A. and Heiser, G. (2010). An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10*, pages 21–21, Berkeley, CA, USA. USENIX Association.
- GFXBench (2015). Gfxbench 3.0 directx. <http://www.gfxbench.com>.
- Halperin, D., Greenstein, B., Sheth, A., and Wetherall, D. (2010). Demystifying 802.11n Power Consumption. In *Proceedings of the International Conference on Power-Aware Computing and Systems, HotPower*, Vancouver, BC, Canada.

- Hao, S., Li, D., Halfond, W. G. J., and Govindan, R. (2013). Estimating Mobile Application Energy Consumption using Program Analysis. In *Proceedings of the the International Conference on Software Engineering (ICSE)*, San Francisco, California.
- Hewlett Packard (2013). HP EliteBook Folio 1040 G1 Notebook PC. Technical report.
- Hruska, J. (2013). Nvidia's Tegra 4 Demystified: 28nm, 72-core GPU, Integrated LTE, and Questionable Power Consumption. <http://www.extremetech.com>.
- Kim, Y. G., Kim, M., et al. (2015). A Novel GPU Power Model for Accurate Smartphone Power Breakdown. *ETRI Journal*, 37(1).
- Lampe, U., Hans, R., and Steinmetz, R. (2013). Will mobile cloud gaming work? findings on latency, energy, and cost. In *Proceedings of the 2013 IEEE Second International Conference on Mobile Services, MS '13*, pages 103–104, Washington, DC, USA. IEEE Computer Society.
- Lee, K., Chu, D., Cuervo, E., Kopf, J., Grizan, S., Wolman, A., and Flinn, J. (2014). DeLorean: Using Speculation to Enable Low-Latency Continuous Interaction for Mobile Cloud Gaming. Technical report, Microsoft Research.
- Madden, J. (2011). MIMO Adoption in Mobile Communications Forecast: Devices by Operating System and User Type, Worldwide, 2010-2017, 1Q13 Update. Technical report, Mobile Experts.
- MarketsandMarkets (2010). World Mobile Applications Market - Advanced Technologies, Global Forecast (2010 - 2015). Technical report, MarketsandMarkets.
- Milanesi, C., Tay, L., Cozza, R., Atwal, R., Nguyen, T. H., Tsai, T., Zimmermann, A., and Lu, C. K. (2013). Forecast: Devices by Operating System and User Type, Worldwide, 2010-2017, 1Q13 Update. Technical report, Gartner.
- Netgear (2012). Next Generation Gigabit WiFi - 802.11ac. Technical report.
- Notebook Review (2015). Dell precision m6700 owner's review. <http://forum.notebookreview.com/dell-latitude-vostro-precision/679326-dell-precision-m6700-owners-review.html>.
- NoteBookCheck (2014). Computer games on laptop graphic cards. <http://www.notebookcheck.net/Computer-Games-on-Laptop-Graphic-Cards.13849.0.html>.
- Nvidia (2015a). Building Cloud Gaming Servers. <http://www.nvidia.com/object/cloud-gaming-benefits.html>.
- Nvidia (2015b). GeForce GTX 690 Specifications. <http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-690/specifications>.
- Nvidia (2015c). Grid GPUs. <http://www.nvidia.com/object/grid-boards.html>.
- Saha, S. K., Deshpande, P., Inamdar, P. P., Sheshadri, R. K., and Koutsonikolas, D. (2015). Power-Throughput Tradeoffs of 802.11n/ac in Smartphones. In *Proc. of the 34th IEEE International Conference on Computer Communications (INFOCOM)*, Hong Long, Spain.
- Shiraz, M., Gani, A., Khokhar, R., and Buyya, R. (2013). A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing. *IEEE Communications Surveys Tutorials*, 15.
- Soliman, O., Rezgui, A., Soliman, H., and Manea, N. (2013). Mobile cloud gaming: Issues and challenges. In Daniel, F., Papadopoulos, G. A., and Thiran, P., editors, *Mobile Web and Information Systems - 10th International Conference, MobiWIS 2013, Paphos, Cyprus, August 26-29, 2013. Proceedings*, volume 8093 of *Lecture Notes in Computer Science*, pages 121–128. Springer.
- Thompson, C., Schmidt, D. C., Turner, H. A., and White, J. (2011). Analyzing Mobile Application Software Power Consumption via Model-driven Engineering. In Benavente-Peces, C. and Filipe, J., editors, *PECCS*, pages 101–113. SciTePress.
- Zeng, Y., Pathak, P. H., and Mohapatra, P. (2014). A First Look at 802.11ac in Action: Energy Efficiency and Interference Characterization. In *Pros. of the 13th IFIP International Conferences on Networking*, Trondheim, Norway.



# Cloud-side Execution of Database Queries for Mobile Applications

Robert Pettersen, Steffen Viken Valvåg, Åge Kvalnes and Dag Johansen

Department of Computer Science, University of Tromsø, The Arctic University of Norway, Tromsø, Norway  
{robert, steffenv, aage, dag}@cs.uit.no

Keywords: Mobile, Cloud, Performance.

**Abstract:** We demonstrate a practical way to reduce latency for mobile .NET applications that interact with cloud database services. We provide a programming abstraction for location-independent code, which has the potential to execute either locally or at a satellite execution environment in the cloud, in close proximity to the database service. This preserves a programmatic style of database access, and maintains a simple deployment model, but allows applications to offload latency-sensitive code to the cloud. Our evaluation shows that this approach can significantly improve the response time for applications that execute dependent queries, and that the required cloud-side resources are modest.

## 1 INTRODUCTION

Use of cloud-provided services is integral to the operation of modern distributed and mobile applications. In particular, cloud databases simplify application logic by serving as highly available repositories for critical state. For improved scalability and availability these databases are commonly NoSQL, with limited support for tabular relations and transactions and with a more relaxed consistency model than a conventional relational database. Queries are issued through a programmatic interface, rather than a domain-specific, high-level query language.

This promotes a usage pattern where multiple, consecutively-issued queries implement a single logical transaction. For example, an atomic update can be implemented as a read of the old value, followed by a conditional write of the new value, with the predicate that the old value remains unchanged. Or a collection of related records can be retrieved in multiple steps, by manually following foreign key references, rather than using higher-level features like joins and subqueries.

When the database is hosted in the cloud, issuing a sequence of dependent queries entails multiple round-trips of communication, and network latency becomes an important concern. For example, we have measured a latency of 50 – 350ms for accessing the Amazon DynamoDB (DeCandia et al., 2007) cloud database from a mobile device (Pettersen et al., 2014), whereas a study covering 260 global vantage points reports an average round-trip time (RTT) of 74ms for

accessing Amazon EC2 instances (Li et al., 2010). Issuing a sequence of queries to the cloud can result in unwanted delays that are perceptible by users.

This paper demonstrates a practical way to significantly reduce completion-latency when mobile applications execute dependent queries against a cloud database. Our approach is to provide a programming abstraction—*satellite execution*—for location-independent code, which has the potential to execute either locally on a device, or be offloaded to the cloud. If an application experiences high latency, or needs to issue a long sequence of database queries, the latency-sensitive code can be offloaded to the cloud and executed in close proximity to the database service. This ensures low-latency database access on demand, while preserving the programmatic style of database access.

Our system, called Dapper, significantly extends and integrates the functionality of two previous systems: Rusta (Valvåg et al., 2013) and Jovaku (Pettersen et al., 2014). Rusta is a platform for developing cloud applications that can utilize client-side storage and processing capacity, while the Jovaku system provides a distributed infrastructure for caching of cloud database data through the ubiquitous DNS service.

A goal with Rusta was to express computations in a location-independent way, allowing for opportune execution in the cloud or at client-side devices. This was accomplished by expressing computations in the Scala programming language and using built-in closure features to create transferable execution contexts. Dapper uses .NET reflection to achieve the

same, thereby approaching the problem of transferability in a more general manner; any part of the execution context of a program written in any .NET supported language can be made transferable.

Jovaku's application-transparent interfacing with Amazon's DynamoDB through DNS was in part made possible by a cloud-side relay-node. Software on this node bridged DNS with DynamoDB by turning DNS requests into database queries. Dapper not only supports use of DNS for caching of data on behalf of a mobile application, but also transforms and extends the Jovaku cloud-side node into a platform for hosting and executing offloaded .NET code.

To illustrate the benefits of our approach, we quantify latency savings when cloud database queries are executed from the cloud rather than at the client-side device. We examine communication traces of popular phone applications to determine the practicality of our approach, and we measure the performance of the Dapper cloud-side platform to assess added costs.

The rest of the paper is structured as follows. Section 2 elaborates on the background and context of our work, motivating our general approach. Section 3 describes Dapper, and the programming abstractions that enable cloud-side execution of queries. Section 4 contains our performance evaluation, with measurements of typical reductions in latency, and the maximum query processing throughput that can be achieved in various configurations. Section 5 discusses related work, and Section 6 concludes.

## 2 BACKGROUND

The desire to reduce latency for mobile applications tends to encourage a split application architecture, where parts of the application logic executes on the device, and other parts execute in the cloud. Higher-level operations such as submitting a comment or generating a news feed are delegated as a whole to the cloud, to avoid multiple round-trips of communication.

The split between frontend and backend also has a tangential benefit: it allows a variety of frontends, often tailored for different devices, to access the same backend service. For example, an on-line chess service will typically offer both a web-based frontend, as well as clients for various mobile devices and platforms. Users should be able to switch seamlessly between client devices, e.g. moving from their laptop to their phone, so the state of on-going games must be maintained by the backend. This requires frequent communication with the cloud to retrieve and update

application state.

Many frameworks and platforms aim to ease the development of mobile applications that are factored into separate backend and frontend components. One example is Parse (Parse, 2015), which provides a backend-as-a-service solution that offers backend cloud storage, as well as the ability to deploy application modules that execute in the cloud, close to the data. One common downside of these approaches is that the device-specific and cloud-specific parts of the application are deployed independently, through different channels. This increases the risk of breakage, when old versions deployed on devices interact with the newest version deployed in the cloud.

We approach the problem differently; rather than explicitly deploying parts of applications in the cloud, we empower applications to offload latency-sensitive code on demand, in a dynamic manner. Offloaded code will execute in close proximity to the backend storage service, where latency is low. Thus, we address the main motivating concern—improving application responsiveness as experienced by users—without dictating a static deployment model for applications.

A key idea underlying this work is to move computations closer to the data that they touch, which is a well-known technique that finds diverse applications. When processing streams of data, the demand for network bandwidth can be reduced by filtering streams closer to the source, pushing computations upstream. When processing stored data, similar gains can be made by scheduling computations to execute locally on the storage nodes, using functional programming models like MapReduce (Dean and Ghemawat, 2004) for location independence.

Our experience from mobile agents (Johansen et al., 2001; Johansen et al., 1999) and MapReduce-style distributed data processing have inspired some key aspects of this work. As in Cogset (Valvåg et al., 2013), we promote a functional programming model using the visitor pattern, where latency-sensitive code has the ability to *visit* the backend storage service as desired. In this case, a visitor also resembles a mobile agent; although restricted to moves back and forth between a client device and the cloud, it retains the defining ability to carry state.

## 3 DAPPER

Instead of statically partitioning applications into client- and cloud-side components, Dapper enables individual objects to move dynamically between the client and the cloud. This is accomplished by ex-

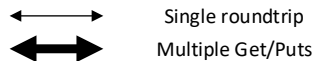
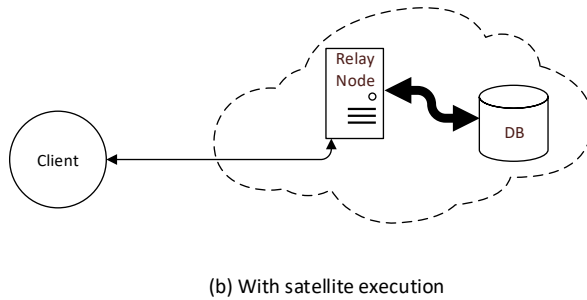
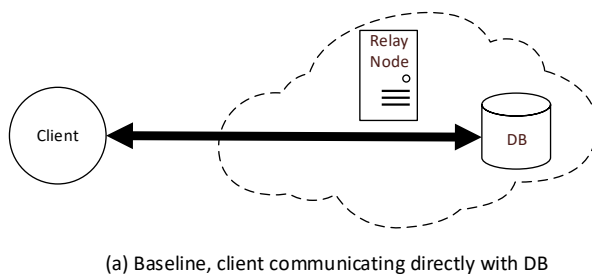


Figure 1: How *satellite execution* is applied to eliminate extraneous round-trips of communication between a client and the cloud, reducing latency.

tending the mobile platform with a *satellite execution* environment hosted on a cloud-side relay-node. Dapper implements the relay-node and provides programming abstractions for an application to temporarily execute an object at the satellite.

The decision to deploy an object for satellite execution is taken at run-time. Deployment involves moving an object's code (i.e., its class) and its current state. Incurred state changes while executing remotely are included when the object is moved back to the client. Objects can move repeatedly between the client and the cloud, for example in response to changes in application environment or state.

In this work, we demonstrate how satellite execution can reduce completion-latency for cloud database queries. Queries are expressed as objects that interact programmatically with the database. Through satellite execution the objects are deployed in close proximity to the targeted cloud database. This approach preserves the advantages of a programmatic database interface; for example, objects can perform computations, transformations, cryptographic operations, and any other manipulations of parameters and intermediate results that may be required when performing a sequence of queries. Figure 1 illustrates our approach,

```
public interface IContext
{
 Task<object> Get(string key);

 Task<List<object>> GetMany(string key);

 Task<bool> Put(string key, object value);
}
```

Figure 2: Dapper interface to key/value databases or stores.

showing how multiple round-trips between a client and the cloud can be replaced with a single round-trip to the relay-node and multiple low-latency intra-cloud interactions with the database.

Our implementation targets Amazon DynamoDB, a popular NoSQL cloud database service, but it can easily be adapted to work with any similar services. The relay-node is an Amazon EC2 instance in the same availability zone as the DynamoDB service, running an unmodified Windows Server 2012 image. The relay-node software is written in C# and uses an asynchronous programming model to efficiently handle a large number of clients and database connections.

Amazon's official C# API is used to perform DynamoDB operations. But Dapper exposes this API to the programmer indirectly, through a database context object providing general database operations, implementing the *IContext* interface shown in Figure 2. This indirection separates application logic from the particular database, promoting customization flexibility. For example, an application can be run fully client-side by providing a context object that binds to a client-side database.

To be eligible for satellite execution, a class must implement the *ISatellite* interface. Figure 3 shows an example implementing a bag-of-queries capable of satellite execution, given a database context. Queries are added to the bag by invoking *AddQuery()*; the queries are aggregated in the *\_queryList* field. *Execute()* issues the queries via the database context object and stores results in the *\_responseList* field. The client can retrieve query results by invoking *GetResponses()*.

An object is moved for execution at a relay-node when the application invokes the Dapper run-time's *ExecuteAt()* function, shown in Figure 4, specifying the object and the particular satellite execution environment. *ExecuteAt()* transfers the object, in a serialized state, to the relay-node, where the object is deserialized and its *Execute()* function is invoked. After the *Execute()* function completes, the object is moved back to the client. Dapper exposes .NET task-based asynchronous programming primitives, as shown in

```

[Serializable]
public class QueryBag : ISerializable, ISatellite
{
 private List<string> _responseList;
 private List<string> _queryList;

 public async Task Execute(IContext ctx)
 {
 foreach (var query in _queryList)
 {
 var queryResponse =
 await ctx.GetMany(query);
 if (_responseList == null)
 _responseList = new List<string>();

 _responseList.AddRange(queryResponse);
 }
 }

 public void AddQuery(string query)
 {
 if (_queryList == null)
 _queryList = new List<string>();

 _queryList.Add(query);
 }

 public List<string> GetResponses()
 {
 return _responseList;
 }
}

```

Figure 3: Implementation of a bag-of-queries that can execute remotely in the cloud via satellite execution.

```

async Task<object> ExecuteAt(object obj,
 Node location = null)

```

Figure 4: Interface for requesting remote execution.

figures 3 and 4, for the application to determine completion of remote execution and for the remote environment to efficiently handle actual execution.

Dapper employs several techniques to reduce the amount of data communicated between the client and the relay-node. One technique is to cache previously received assemblies at the relay-node. Thus, if instances of the same class are moved repeatedly, the corresponding byte codes need only be communicated once. Assembly-versioning determines the validity of a cached assembly. Another optional optimization is to communicate only changed state back from the remote environment—when remote execution completes, Dapper determines the difference in object state before and after execution and communicates that difference back to the client for object reconstruction. This is implemented using a custom serialization protocol; default serialization offers more convenience and is employed unless otherwise speci-

fied.

A user typically assigns different levels of trust to applications hosted on the same mobile device. For example, the user could entrust one application with access to data such as a contact list, but deny that access to another application. It is important for satellite execution not to weaken enforcement of this differentiated trust. For example, if the relay-node provides no isolation between execution environments, code executed on behalf of one application could potentially access code and data belonging to another application, thereby compromising trust assigned by the user at the mobile device.

Dapper relies on .NET application domains (Application Domains, 2015) to create separate and isolated execution domains for received assemblies at the relay-node. Each of these domains is configured with a whitelist of library-assemblies that are available to the hosted assembly. Also, some library-assemblies are made partially available subject to call-interception and approval.

## 4 EVALUATION

Our relay-node was hosted on two types of Amazon EC2 instances in our experiments. The first type was t1.micro, equipped with 613 MB memory and a single-core 64-bit vCPU operating at 1.85 GHz. The second type was t2.medium, equipped with 4 GB memory and a two-core vCPU operating at 2.50 GHz. Both types of instances were running Microsoft Windows Server 2012 R2. We used Amazon’s DynamoDB as the cloud-side database, instantiated in the same availability zone as our relay-node.

Dapper runs on a variety of Microsoft Windows platforms, including phone, store, and desktop. We used two different client-side platforms for the experiments: (1) a phone with 2 GB memory and a four-core Qualcomm Snapdragon 800 2.2 GHz CPU and (2) a desktop machine with 64 GB memory and a four-core Intel Xeon E5-1620 3.7 GHz CPU. The phone ran Windows Phone 8.1 and communicated over 4G, whereas the desktop machine ran Windows 10 and was connected to a LAN.

We first report on a black-box examination of the cloud communication patterns of some popular mobile device applications. Here we sought to discover patterns consistent with sequences of dependent queries, with the motivation that satellite execution could be used in place of such interactions. We configured our phone platform to communicate through an access point instrumented to capture all ingress and egress packets. We then inspected packet traces look-

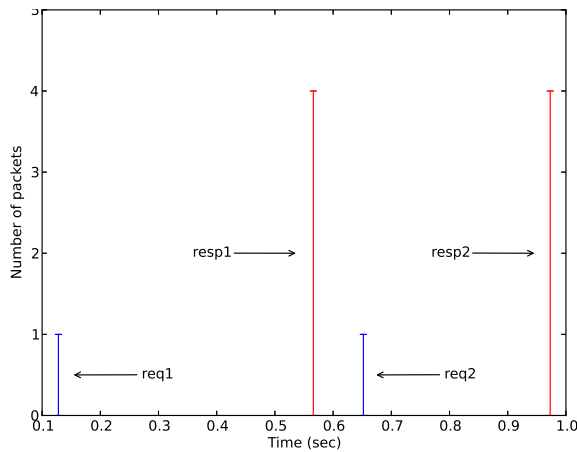


Figure 5: Example communication pattern between mobile device and cloud assumed to be of a request/reply type.

Table 1: Summary of cloud interactions during phone application startup.

Application	# request/reply	# connections
Social networking	1	1
	2	1
Instant messaging	1	4
	2	3
	7	1
Short messaging	1	7
	2	3
	4	1
	6	1
Picture exchange	1	1
	2	2

ing for what appeared as consecutive request/reply cloud interactions without intervening user actions. The particular pattern we looked for is exemplified in Figure 5, which shows two interactions assumed to be of a request/reply type.

Our findings for cloud interactions during startup of four popular applications are summarized in Table 1. We observed that the applications communicate over a number of separate network connections, ranging from 2 for the social networking application to 12 for the short messaging application. Most of these connections are to different services within the same cloud, but some are external, typically in support of content distribution such as Akamai (Nygren et al., 2010). The number of assumed request/reply interactions varied across applications and connections, with the instant- and short messaging applications respectively having as many as 7 and 6 consecutive interactions. These findings suggest satellite execution could be effective if applied in these popular applications.

We continue with an experiment that quantifies latency when a client issues cloud database queries di-

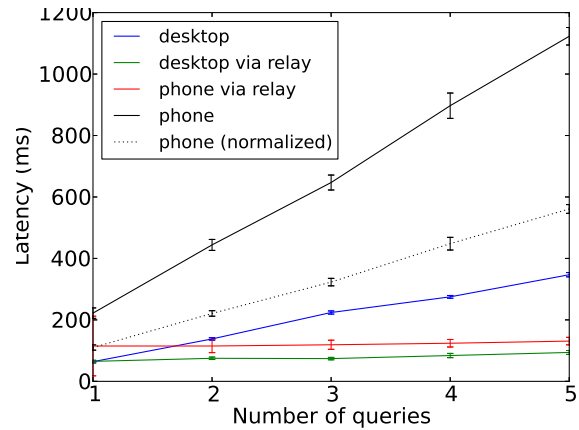


Figure 6: Increasing number of queries executed with and without satellite execution.

rectly and when utilizing the cloud-side relay-node. For this we used the bag-of-queries implementation outlined in Figure 3 to issue queries to the cloud database. Latency when the bag contained between 1 and 5 queries is shown in Figure 6. The figure presents results, averaged over 1000 runs, for both phone and desktop with the relay node hosted on a t1.micro instance. As shown, there are significant latency savings when the bag contains more than one query. This is because latency between the relay-node and the database is low, and the round-trip latency between the client and the cloud—approximately 64 ms for desktop and 105 ms for phone—overshadows the low cost of serializing and transferring the query bag.

The DynamoDB library uses the HTTP 100-continue feature when interacting with the cloud database. Use of this feature adds a communication round-trip to database interaction, needlessly inflating latency, as described in (Pettersen et al., 2014). We therefore used platform interfaces to disable this HTTP feature on desktop. Similar interfaces do not exist on Windows Phone, however. The results in Figure 6 consequently include one additional round-trip latency for phone, compared to desktop. To better convey the latency difference between phone and desktop, the figure also includes results where one round-trip latency has been subtracted from phone. Even after this normalization, phone has significantly higher latency than desktop, demonstrating the relative importance of our satellite execution technique for the mobile platform.

The data on popular applications in Table 1 only indicates that latency savings are possible; determining the degree to which the interaction could exploit satellite execution would require access to application source code. To approximate the savings that could be experienced in a deployed application we reconstruct

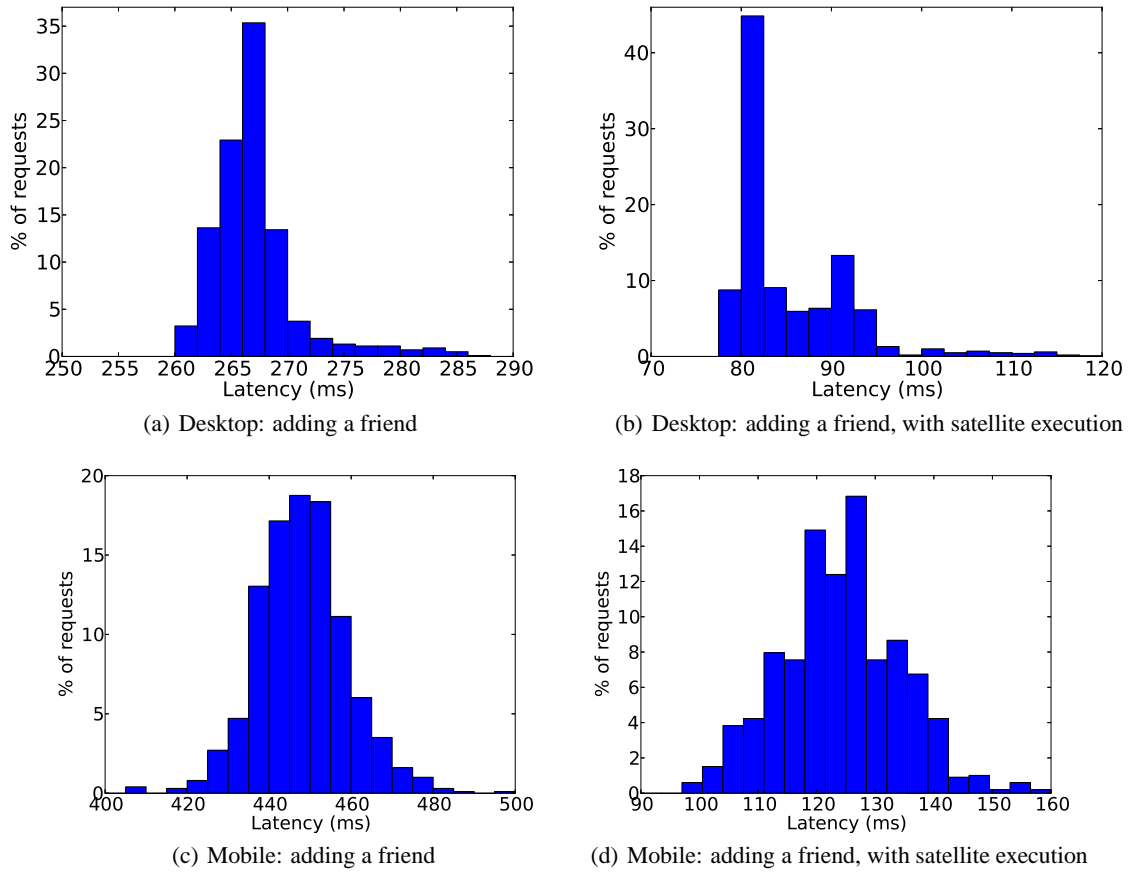


Figure 7: Latencies when adding a friend to a social network, with and without Dapper’s satellite execution.

a scenario where a friend connection is established in the MSRBook, a social networking application based on Deuteronomy (Levandoski et al., 2011). The addition of a friend in this network involves friend and news feed updates for both concerned parties, for a total of 4 queries. Equivalent queries were placed in our bag-of-queries and we ran the friend-add action 1000 times on both the desktop and the mobile platform, with and without satellite execution. Figure 7 illustrates latency savings. Savings due to satellite execution are pronounced; on desktop latency drops from around 265 ms to approximately 100 ms, while it drops on mobile from around 450 ms to approximately 125 ms.

On a mobile device such as a smartphone, a person uses around 24 different applications every month (Nielsen, 2014). Even the modest resource allocations available to the Amazon t1.micro instance used in our experiments are likely to be ample for a relay-node dedicated to a single mobile device. But if the relay-node functionality was a service offered by the cloud database provider, in a fashion similar to the Parse application module service (Parse, 2015),

the relay-node would likely be shared among many mobile devices and its capacity would be an issue. We therefore last consider an experiment where the relay-node serves an increasing number of mobile devices.

In the experiment we configured each client (i.e. mobile device) with a 4-query bag at the relay-node. Queries in these bags were repeatedly executed, ensuring high contention for relay-node resources. We then increased the number of clients, in an attempt to reveal relay-node capacity. Results for the t1.micro instance are shown in Figures 8(a)–(c). From the figures we observe that the t1 instance is capable of completing around 50 bags per second before performance tapers off. This maximum performance is likely due to CPU becoming a bottleneck, as indicated by the data in Figure 8(c). This is corroborated by t2.medium instance performance, which is shown in Figures 8(d)–(f). The t2.medium instance has approximately twice the CPU capacity of the t1.micro instance, and also completes bags at twice the rate of the t1.micro instance.

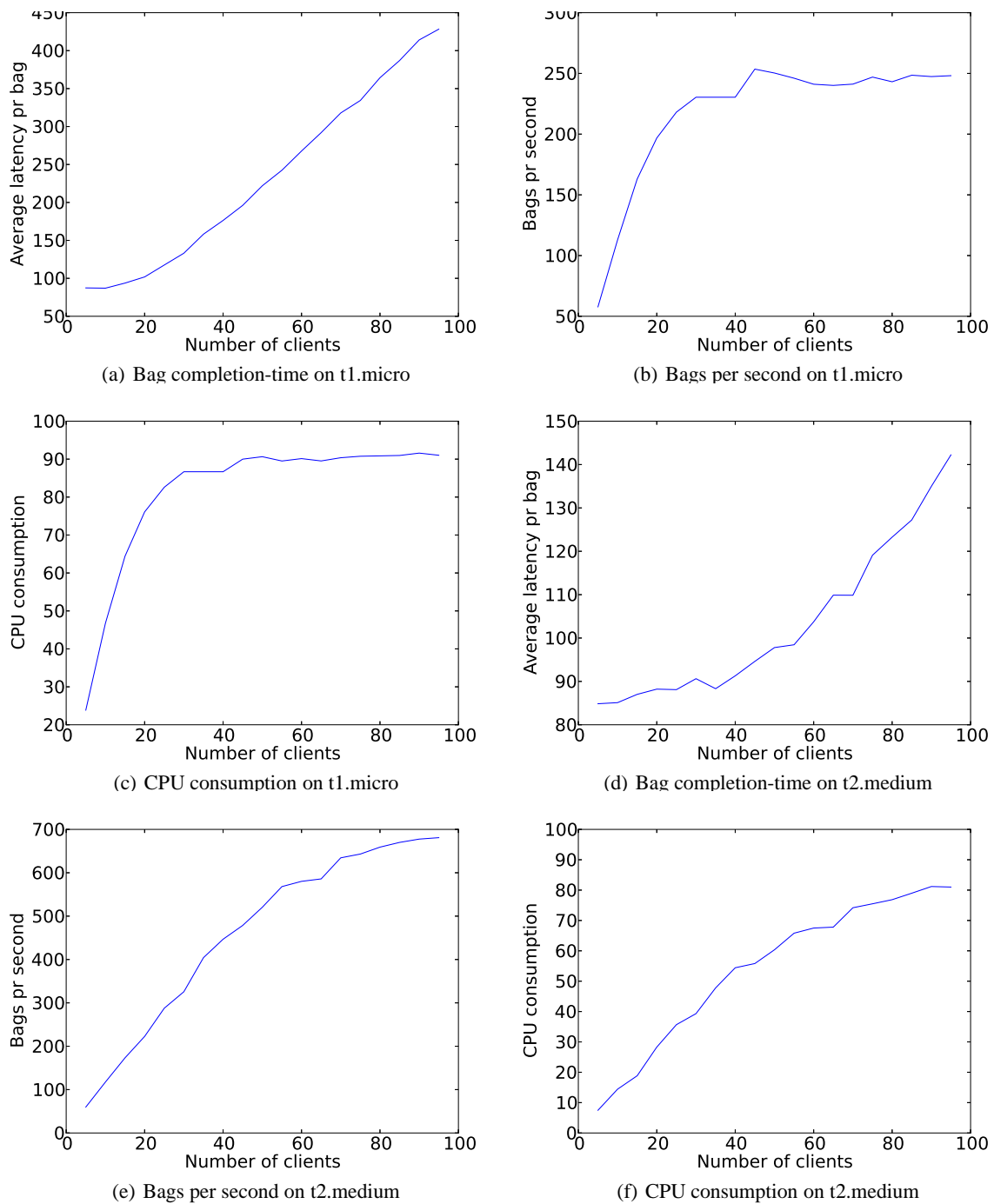


Figure 8: t1.micro and t2.medium relay-node performance

## 5 RELATED WORK

The complexity of developing and deploying applications that span a variety of mobile devices, personal computers, and cloud services, has been recognized as a new challenge. Users expect applica-

tions and their state to follow them across devices, and to realize this functionality, one or more cloud service must usually be involved in the background. Sapphire (Zhang et al., 2014) is a recent and comprehensive system that approaches this problem by making deployment more configurable and customizable,

separating the deployment logic from the application logic. The aim is to allow deployment decisions to be changed, without major associated code changes. Applications are factored into collections of location-independent objects, communicating through remote procedure calls. Like Sapphire, Dapper provides a location-independent programming abstraction, but preserves a monolithic application structure, which allows the application to be installed in its entirety on a single device through a regular distribution channel like an app store. Code is then transferred on demand from the device to the cloud, as objects move to the cloud to enjoy low-latency execution. The decision to visit the cloud or stay on the local device can be made dynamically, at run time.

With Dapper, we introduce relay-nodes in the cloud as an architectural tier between the cloud and mobile devices. Similar middle tiers have been proposed for example with Cloudlets (Satyanarayanan, 2013), and are implemented in code-offloading systems like COMET (Gordon et al., 2012), MAUI (Cuervo et al., 2010), and CloneCloud (Chun et al., 2011). However, the goal of these systems is often to augment mobile devices with additional computing power, or to conserve energy (Tilevich and Kwon, 2014), so the added tier may be located close to the devices, on local server machines, or wherever cheap computing power is available. In contrast, our motivation is not to offload work, but to reduce the latency of accessing cloud services, and thus the new tier sits as close to the cloud services as possible.

Concretely, Dapper reduces latency by eliminating extraneous round-trips of communication to the cloud. An alternative way to achieve that is by having cloud databases support more expressive query languages, so that more sophisticated transactions can be submitted as a single operation. Indeed, relational databases with full SQL support are part of the offerings of major cloud providers like Amazon. However, the ability to access the database via a general-purpose programming language remains appealing for its generality and flexibility. This is a lesson learned from programming models like MapReduce (Dean and Ghemawat, 2004), Oivos (Valvåg and Johansen, 2008), and Cogset (Valvåg et al., 2013), where data is accessed programmatically through user-defined visitor functions that can integrate easily with legacy code and libraries. The programming model in Dapper follows a similar philosophy, with the difference that user-defined functions are visiting a database in the cloud rather than a partition of data in a cluster.

## 6 CONCLUSION

This work focuses on the general issue of latency as a concern for applications that interact with the cloud, and looks specifically at scenarios where multiple consecutive queries are issued to a database in the cloud. Intuitively, latency can be reduced by shortening communication distances, so our idea is to move the location where queries are issued closer to the database. Since cloud databases commonly have programmatic interfaces, we implement a general mechanism for code-offloading to support this pattern.

Having a relay-node in the cloud, located in close proximity to the database service, has already proven to be a useful technique for caching, and beneficial for read-mostly workloads (Pettersen et al., 2014). Here, we extend the relay-node with functionality for *satellite execution*, allowing code that has moved temporarily from a mobile device to execute in an environment with low-latency database access. This gives benefits for additional workloads, which may include updates.

The key characteristic that a workload must exhibit to benefit from our approach is dependencies between queries. For example, if the results from one query are used to shape the next query, there is a dependency between the two. So long as there is no need for user interaction, a whole sequence of dependent queries can be offloaded to the cloud. By eliminating extraneous round-trips of communication, this improves response times.

To estimate the potential for improvement in real applications, our evaluation examines the communication patterns of some popular applications through a black-box technique. This has yielded some indications that dependent queries occur in practice, since sequences of up to 7 requests were observed back-to-back over the same connection on startup. Looking at a concrete implementation of a social networking application from (Levandoski et al., 2011), we found specific examples. For example, a friend request results in 4 dependent queries; when offloaded to the cloud from a phone, the completion time of a friend request dropped from 450 ms to approximately 125 ms.

Our implementation handles the practicalities of transferring assemblies of .NET code, serializing and deserializing objects, and sandboxing code that executes on the relay-node. Our evaluation gives some data points on performance: a single Amazon t1.micro instance can serve hundreds of queries per second. One such instance can thus easily handle load imposed by a large number of applications. So, we can dramatically reduce latency without disrupting



application architectures and with minimal requirements for resources in the cloud.

## REFERENCES

- Application Domains (2015). <http://msdn.microsoft.com/en-us/library/cxk374d9%28v=vs.90%29.aspx>.
- Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., and Patti, A. (2011). Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems, EuroSys '11*, pages 301–314, New York, NY, USA. ACM.
- Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. (2010). Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services, MobiSys '10*, pages 49–62, New York, NY, USA. ACM.
- Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th symposium on Operating Systems Design and Implementation, OSDI '04*, pages 137–150. USENIX Association.
- DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Voshall, P., and Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41:205–220.
- Gordon, M. S., Jamshidi, D. A., Mahlke, S., Mao, Z. M., and Chen, X. (2012). Comet: code offload by migrating execution transparently. In *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, OSDI'12*, pages 93–106, Berkeley, CA, USA. USENIX Association.
- Johansen, D., Lauvset, K. J., van Renesse, R., Schneider, F. B., Sudmann, N. P., and Jacobsen, K. (2001). A TACOMA retrospective. *Software - Practice and Experience*, 32:605–619.
- Johansen, D., Marzullo, K., and Lauvset, K. J. (1999). An approach towards an agent computing environment. In *ICDCS'99 Workshop on Middleware*.
- Levandoski, J. J., Lomet, D. B., Mokbel, M. F., and Zhao, K. (2011). Deuteronomy: Transaction support for cloud data. In *CIDR*, pages 123–133. [www.cidrdb.org](http://www.cidrdb.org).
- Li, A., Yang, X., Kandula, S., and Zhang, M. (2010). Cloud-Cmp: comparing public cloud providers. In *ACM SIGCOMM*, pages 1–14.
- Nielsen (2014). <http://www.nielsen.com/us/en/insights/news/2014/smartphones-so-many-apps-so-much-time.html>.
- Nygren, E., Sitaraman, R. K., and Sun, J. (2010). The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19.
- Parse (2015). <http://www.parse.com>.
- Pettersen, R., Valvåg, S. V., Kvalnes, A., and Johansen, D. (2014). Jovaku: Globally distributed caching for cloud database services using DNS. In *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 127–135.
- Satyanarayanan, M. (2013). Cloudlets: at the leading edge of cloud-mobile convergence. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pages 1–2. ACM.
- Tilevich, E. and Kwon, Y.-W. (2014). Cloud-based execution to improve mobile application energy efficiency. *Computer*, 47(1):75–77.
- Valvåg, S. V., Johansen, D., and Kvalnes, A. (2013). Cogset: A high performance MapReduce engine. *Concurrency and Computation: Practice and Experience*, 25(1):2–23.
- Valvåg, S. V. and Johansen, D. (2008). Oivos: Simple and efficient distributed data processing. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, HPCC '08*, pages 113–122. IEEE Computer Society.
- Valvåg, S. V., Johansen, D., and Kvalnes, A. (2013). Position paper: Elastic processing and storage at the edge of the cloud. In *Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services, HotTopiCS '13*, pages 43–50, New York, NY, USA. ACM.
- Zhang, I., Szekeres, A., Aken, D. V., Ackerman, I., Gribble, S. D., Krishnamurthy, A., and Levy, H. M. (2014). Customizable and extensible deployment for mobile/cloud applications. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 97–112, Broomfield, CO. USENIX Association.

## **SHORT PAPER**



# Telco Clouds

## Modelling and Simulation

Jakub Krzywda<sup>1</sup>, William Tärneberg<sup>2</sup>, Per-Olov Östberg<sup>1</sup>, Maria Kihl<sup>2</sup> and Erik Elmroth<sup>1</sup>

<sup>1</sup>*Dept. of Computing Science, Umeå University, SE-901 87 Umeå, Sweden*

<sup>2</sup>*Dept. of Electrical and Information Technology, Lund University, SE-223 63 Lund, Sweden*  
*{jakub, p-o, elmroth}@cs.umu.se, {william.tarneberg, maria.kihl}@eit.lth.se*

**Keywords:** Mobile Cloud Computing, Telecommunication Infrastructure, Cloud Infrastructure, Modelling, Simulation.

**Abstract:** In this paper, we propose a telco cloud meta-model that can be used to simulate different infrastructure configurations and explore their consequences for system performance and costs. To achieve this, we analyse current telecommunication and data centre infrastructure paradigms, describe the architecture of the telco cloud, and detail the benefits of merging both infrastructures in a unified system. Next, we detail the dynamics of the telco cloud and identify the components that are the most relevant from the perspective of modelling performance and cost. As a number of well established simulation technologies exist for most of the telco cloud components, we survey existing models in an attempt to construct a suitable composite meta-model. Finally, we present a showcase scenario to demonstrate the scope of our telco cloud simulator.

## 1 INTRODUCTION

Recent technological developments have enabled a union of telecommunication and cloud computing. Joint management of telecommunication infrastructure, such as Radio Base Stations (RBS), and Data Centres (DC) may help to achieve better performance of hosted applications and reduce the operation costs. In this paper we refer to this paradigm as telco cloud computing (Bosch et al., 2011).

Despite the interest in this paradigm, there are no simulation models capable of simultaneously modelling the dynamics of Mobile Devices (MD), placement and capacity of DCs, and network infrastructure. Understanding of these relations is important for telco cloud stakeholders, e.g., Infrastructure Providers (IP) that can use that knowledge to reduce infrastructure costs while still delivering competitive performance.

We propose a meta-model of the telco cloud that facilitates experimentation and evaluation of possible configurations, such as placement and capacity of DCs in a joint telecommunication-cloud infrastructure. The meta-model uses existing, well established simulation models, e.g., for Radio Access Networks (RANs) or DCs, for modelling of individual parts of the infrastructure behaviour.

The contributions of this paper are: describing the dynamics of the telco cloud, including Quality of Service (QoS) and the associated costs of this paradigm

(Section 4); surveying existing models of telco cloud building blocks (Section 5); and establishing a meta-model that captures the described dynamics using existing and composite models (Section 6).

This paper is structured as follows. Section 2 outlines the architecture of the proposed telco cloud. Next, the simulation motivations, challenges, and requirements that the meta-model has to fulfill are presented in Section 3. Section 4 describes the telco cloud dynamics, followed by a survey of existing models in Section 5. Section 6 introduces the telco cloud meta-model. Section 7 presents a showcase simulation scenario, using a prototype implementation of the introduced meta-model. In Section 8 we list a few relevant research topics that can be explored using the proposed model and conclude the paper.

## 2 THE TELCO CLOUD ARCHITECTURE

In this section we present issues with current telecommunication and cloud infrastructures, an overview of the proposed telco cloud topology, and how the telco cloud paradigm can help to remedy these issues.

## 2.1 Current Infrastructure

Currently, telecommunication and cloud computing infrastructures are separated and managed independently. The telecommunication infrastructure is placed in close proximity to end users and is built using specialised hardware. The cloud computing infrastructure consists of remote DCs that are significantly geographically separated from end users, consists of commodity hardware, and is connected with the telecommunication infrastructure via the Internet.

We identify several issues of the current infrastructure. Performance (especially latency) of cloud services is not predictable, which makes computation offloading difficult. All data processed in the cloud are sent over the Internet to DCs, which adds communication latency. For the Internet of Things, with millions of sensors generating huge amounts of data, the volume of traffic can cause network congestion. Moreover, specialised telecommunication hardware is expensive and hard to upgrade.

As a consequence of the performance bottlenecks, particularly latency sensitive applications such as industrial process control and augmented reality context recognition applications have mostly not yet been cloudified. The low latency, jitter free, and high throughput connections required by such applications cannot be provided by the existing telecommunication and cloud infrastructures (Barker and Shenoy, 2010). Moreover, compute and battery resources in MDs are limited and coupled. An approach to augmenting MD's capabilities is to offload the execution of applications to a cloud infrastructure. Such performance augmentation can only be seamless if the communication latency is low enough, and both network bandwidth and service availability are high.

Devices are at an increasing rate gaining access to the Internet (Atzori et al., 2010). Anything from small sensors to petrol pumps, flowerpots, helmets, tumblers, windows, and spark plugs is being connected to the Internet to communicate and monitor its quantified performance metrics. Most of the connected devices are generating correlated contextual information, incurring large amounts of Wide Area Network's (WAN) traffic. The traffic typically converges to a handful of DCs for analysis and processing which introduces congestion in the capacity-sparse and increasingly congested RANs, core networks, and WANs. Additionally, a portion of the transported information is only locally relevant and will add no or very little entropy to a service in a Remote DC.

Wireless access network virtualisation and cloudification of telecom equipment and services proposed by (Wang et al., 2013), requires careful placement of

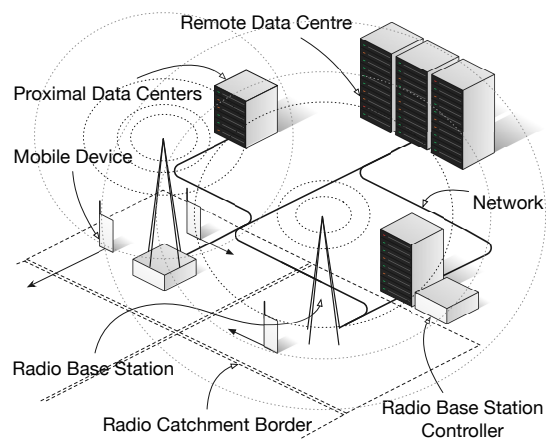


Figure 1: Overview of telco cloud.

compute capacity as not to introduce significant propagation delay. The placement of the compute nodes has to reflect the demand for telecom- and cloud-services in the geographic area which the RAN covers. Current telecom signalling standards and remote DCs do not interoperate well and are often not able to meet telecom latency requirements.

## 2.2 Introducing the Telco Cloud

A telco cloud is an infrastructure consisting of MDs (known also as user equipment), stationary devices (sensors), access networks, intermediate WANs (connecting the access networks to the backbone networks), backbone (Internet) networks, and DCs. We here include two main types of DCs: Remote Data Centres (large DCs located far from the access networks) and Proximal Data Centres (smaller DCs located close to the access networks).

The telco cloud topology paradigm proposes a closer integration between access networks and a cloud infrastructure than the current topological model where the telecom and cloud infrastructures are unaware of each other. When coexisting with cloud infrastructure, telecommunication functionality can be virtualised and augmented to the adjacent DCs. When virtualising RAN functions large portions of an RBS and the RAN control functions can be executed in a DC (Baroncelli et al., 2010).

Cloud capacity will reside in geographically distributed DCs. The telco cloud topology is composed of multiple DCs that are dispersed in a mesh structure, ranging from complete adjacency with the telecom infrastructure, Proximal DCs, to more traditional, Remote DCs, as depicted in Figure 1.

A group of telco cloud DCs can be provisioned and load balanced as one resource or act as indepen-

dent DCs. In the former case, a control plane will need to coordinate services and the shared resources by for example optimising locality and proximity to all entities by geographically placing services accordingly. Proximal DCs will thus have to be managed in a distributed and coordinated manner.

The cloud services hosted in Proximal DCs are accessed through the RAN. The MDs are assumed to possess varying degrees of mobility, with an equivalent likelihood of passing between a particular set of macro/micro-RBS<sup>1</sup> over time. In an effort to be able to enforce DC management constraints and to globally avoid unnecessary migration, an MD is not strictly associated with the closest DC or the DC that its current RBS cell is associated with.

The telco cloud infrastructure topology will need to reflect the capacity and latency objectives of the virtualised RBSs, and to give access to the telco cloud-hosted services given the networks local capability and diversity at a sufficient service level. The prevailing 4G/LTE topology is centred around hierarchical macro- and micro-cells that very much resembles that of its predecessors. The telco cloud topology will evolve with mobile access technology generations, shifting and/or dispersing compute capacity at various levels of mobile, Metropolitan Area Network (MAN), and WAN networks to best suit the prevailing services and throughput channels.

### 2.3 Benefits of the Telco Cloud

We identify four main benefits of the telco cloud. First, it provides cloud applications with better and more predictable performance. Second, it supports computation off-loading for resource-bounded MDs. Third, the telco cloud reduces network utilization by processing part of data closer to its producer or consumer. Fourth, it enhances cost-efficiency and flexibility of telecommunication infrastructure.

Thanks to a geographically distributed cloud infrastructure, application developers and telecommunication operators can take advantage of the significantly lower round-trip times. Moreover, we expect that the average network throughput will increase when communicating with Proximal DCs in comparison to Remote DCs. Additionally, users offloading MD applications will benefit from a low latency communication with a DC, where the code is executed.

The large amount of information generated by sensors can be filtered through the intermediate hierarchical cascade of DCs to prevent congestion in the intermediate WAN. Data that is only locally relevant

can be kept and consumed locally, while redundant and highly covariant information can be more easily identified in a local context and discarded.

Through the telco cloud traditional proprietary hardware-bound telecom services can be virtualised, migrated, and executed in Proximal or Remote DCs. Multiple RBSs can be consolidated to increase the aggregate utilisation of the infrastructure. Executing RBSs on cloud infrastructure will allow for greater use of cost-effective commodity hardware and generic software. With the availability of the telco cloud, we expect that an RBS in future mobile infrastructure generations will only consist of a radio interface. The management of the RAN, individual radio channels, signalling, services, and signal processing, will be all virtualised and executed in a Proximal DC.

## 3 SIMULATION CHALLENGES

Simulation of a telco cloud is motivated by several factors, e.g., lack of existing infrastructure and appropriate control plane standards. The desired simulator has to fulfil many requirements, such as, to be able to simulate hundreds of thousands of various entities at fine grained time granularity (milliseconds) for long periods of time (hours). We here motivate and describe identified requirements and challenges in simulation of the telco cloud. Addressing the challenges and fulfilling the requirements is crucial while designing a meta-model and implementing a simulator.

Telco cloud stakeholders will benefit from being able to investigate the consequences of possible infrastructure configurations. For example, IPs responsible for building and maintaining the infrastructure may use the simulator for planning placement and capacity for new DCs, as well as modifying capacity and connectivity of existing DCs. Service providers that use infrastructures to host services are interested in comparing different strategies for placement of application components. Moreover, developers that implement mobile applications utilising a telco cloud, need to determine what application components could benefit from offloading. To answer these and similar questions, tests with various infrastructure configurations need to be performed and results compared. There are two options for performing these tests: by simulation or by running them in real test beds.

Currently, there is no existing infrastructure that can be used for testing the telco cloud. Creating a physical test bed for large-scale testing of a telco cloud in different configurations is economically infeasible and small-scale test beds will not be able to capture phenomena occurring in reality, e.g., user mo-

<sup>1</sup>What is considered a traditional rural/urban cell, constituted by a high power RBS, mounted on a tower.

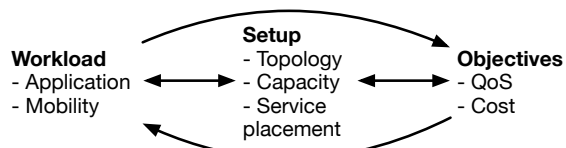


Figure 2: Dependencies between elements of a telco cloud.

bility patterns or latency issues.

For these reasons, we believe that simulation is the most feasible option to evaluate the telco cloud. However, we have identified several requirements that make simulation of telco clouds challenging. First of all, the scale of simulation is large in terms of number and types of entities. The simulation of a telco cloud has to concurrently cover hundreds of thousands of MDs moving around a simulated area, each generating requests; hundreds of RBSs providing an access to the core network; and tens of DCs, running services that process requests. Another challenge is the ratio between time precision and length of simulation. We are interested in a very fine-grained latency simulation, that requires precision of at least milliseconds. However, to capture the daily patterns of MD movement (e.g., moving between home, work, and shops) caused by migrations of users carrying them, the whole system needs to simulate several run-time hours. Moreover, simulation of application statefulness, and of transferring those states when MDs are moving, have not yet been described in the literature and requires new models to be developed.

## 4 TELCO CLOUD DYNAMICS

Before constructing a meta-model of the telco cloud we first need to understand fundamental telco cloud dynamics, in terms of the relations between system input, configuration and output. Later, we will use the knowledge of these dynamics to build the intended simulation meta-model.

Figure 2 visualizes the dynamics inside the telco cloud. The *workload* is an input to the system that IPs have no influence over. It includes: applications, with a request generation model (rate and size), a resource requirements model (the amount of resources needed to process a request), and application statefulness (the overhead of transferring user's state between DC); as well as, the mobility of users carrying MDs.

Next, *objectives* describe required output characteristics of the system. We have identified two fundamental objectives. Firstly, QoS, which imposes performance requirements, e.g., latency or throughput through Service Level Objectives (SLO). Secondly,

the monetary cost associated with energy consumed for computation and maintenance of an infrastructure. The objectives can be used when constructing an optimisation problem with QoS as conditions and cost as the function that should be minimised.

Finally, *setup* is that part of the system that can be adjusted by designers or operators to achieve desired objectives when processing existing workloads. Setup includes topology, location, capacity of DCs and the network that interconnects MDs and RBSs with DCs, as well as resource management policies that control placement and migration of services.

The above mentioned elements are all dependent on each other. The setup of a telco cloud is related to the existing workload. The capacity of a DC is defined by the resource demands of the services, e.g. how memory-, CPU-, network-, and disk-intensive the services are. The locations and topology of DCs are defined by the geographic and demographic scope of the services, the number of MDs that reside in that domain, and the capacity of the associated telecommunication infrastructure.

In addition, workload influences objectives. For example, user mobility is inducing delays during service migrations and potentially causes QoS penalties. Moreover, application statefulness introduces additional costs of storing data in a DC and transmitting data between Proximal DCs. It may also increase latency due to an additional time necessary to send the state data before processing of the request can begin.

Finally, objectives depend on the setup: QoS is proportional to the proximity and capacity of DC – a smaller DC catchment (the geographic area the DC serves) translates to greater locality and reduced propagation latency, while higher capacity allows hosting of more services. Moreover, the capacity and catchment of DCs determine telco cloud costs. Costs are proportional to the dispersion of computing capacity. There is an overhead of each DC, regardless of its capacity, e.g., building, cooling infrastructure, and connection to energy or network. In addition, dispersion increases costs of maintenance, e.g., technicians have to travel between locations. Therefore, costs are proportionally higher in smaller, dispersed DCs because of high initial costs and proportionally lower in larger, centralized DCs due to the economy of scale (Armbrust et al., 2010).

## 5 EXISTING MODELS

To support creation of a meta-model that incorporates workload, setup, and objectives of the telco cloud described in the previous section, we here survey exist-

ing models in the following categories: application request generation and resource requirements, MD mobility, networks, DCs, and infrastructure costs.

Most of the models and simulators are assigned to only one of the above mentioned categories, however capabilities of four surveyed simulators extend to many categories, so we summarise them in Table 1.

Table 1: Overview of surveyed simulators.

Framework	RG	RR	M	N	DC
NS-3	✓		✓	✓	
OMNeT++	✓		✓	✓	
CloudSim		✓			✓
GreenCloud		✓			✓

RG – Request Generation, RR – Resource Requirements, M – Mobility, N – Network, DC – Data Centre.

## 5.1 Workload Models

Applications running in the telco cloud consist of a mobile client and a server processing offloaded computations. Therefore, they should be modelled from two perspectives: *request generation* that describes how requests are created and sent to the DCs; and *resource requirements* that describes how much computational resources are needed to process the requests.

### Request Generation

Request generation models capture the user's behaviour by primarily representing interaction times or the timing clicks through a stochastic process, often Poissonian in nature. A user behaviour model can be further refined by introducing a stochastic model for how long time a user consumes a certain type of content. Additionally, the transition between types of content is often modelled as a Markov process.

Furthermore, the request generation characteristics are commonly modelled with multiple stochastic processes, encompassing the number of packets in a session, and the size of each packet. Request generation models are either closed or open looped. In an open loop model, the generation of each new session is typically a Poisson process independent of the resulting DC action. Conversely, in a closed loop model, the generation of new sessions is dependent on timing of the response from the DC and thus the properties of the previously generated session.

In the packet-level event driven network simulators, NS-3 (Riley and Henderson, 2010) and OMNeT++ (Varga et al., 2001), a node can act as either a client or server, by the mechanism of either sending packets provided by a stochastic model, at a given

rate, within a certain time period, or at a certain interval; or processing received packets from a buffer, at a given rate. Both server and client models can be augmented with a more complex system of queues to such an extent that they can represent an abstract DC that hosts multiple applications.

### Resource Requirements

CloudSim (Calheiros et al., 2011), a simulator of cloud infrastructure, provides an application model that describes computational requirements – the amount of resources that needs to be available (e.g. number of cores, memory and storage); and communicational requirements – the amount of data that needs to be transferred. GreenCloud (Kliazovich et al., 2012), a packet level simulator based on NS-2, apart from computational and communicational requirements, describes also QoS requirements, expressed by an execution deadline. The application model may also include the size of the code that has to be offloaded and dependencies on other services, e.g., in terms of amount of data that has to be sent or received (Kovachev, 2012).

### Mobility

The NS-3 and OMNeT++ nodes described above can be set into motion given a certain stochastic mobility model. They can for example traverse the space as pedestrians, or in auto mobiles, with corresponding velocity and rate of change. The spatial relationship between nodes and RBS affects the channel properties and RBS-to-node associations. Node mobility will also result in handover between RBSs, which in turn will alter the paths of the node-generated workload in the network.

## 5.2 Setup Models

Below, we describe existing models and simulators of networks and DCs, which can be used to configure the setup of the telco cloud meta-model.

### Network

There are several well-established event-driven frameworks that are capable of modelling computer networks, mobile networks, applications, packet-level network traffic, infrastructure, and independent users. The two primary examples are, as mentioned in the previous section, NS-3 and OMNeT++. These two are commonly deployed in academic network research and provide detailed results on network utilisation, throughput, congestion, and latency.



Both NS-3 and OMNeT++ are comprehensive packet-level network simulation frameworks that include wired and wireless standards, and are able to simulate communication channel conditions. Furthermore, both frameworks have detailed models for channel definition, such as propagation delay, interference, data rate, and medium access schemes. In addition, both NS-3 and OMNeT++ support control plane signalling for a number of wireless standards and complex network topologies.

Both frameworks have support for modelling different types of network nodes, ranging from computers to routers and switches. Each edge and node pair has a defined communication and medium access standard, such as TCP/IP and Ethernet. Each packet that is sent over the network is treated in accordance with the prevailing network's transport protocols and routing standard. In both, the events of arrival and departure of packets drive the simulation clock.

Furthermore, they require detailed configuration of all communication modes and node behaviour, making it very time-consuming to implement and verify systems with different levels of abstractions, and are thus cumbersome to model abstract systems.

As the telco cloud topology is yet to be defined with unspecified control planes, it would be counter-intuitive and time consuming to implement telco cloud topologies in either NS-3 or OMNeT++. In some instances, some modules would have to be completely redesigned, and others would have to be specified to a much greater detail than the telco cloud can offer at this stage.

## Data Centre

The purpose of this section is to survey the DC models that are the most suitable for inclusion in the telco cloud meta-model. An extensive list of mathematical models, simulation approaches, and test beds can be found in (Sakellari and Loukas, 2013), while (Ahmed and Sabyasachi, 2014) provides a survey of twelve cloud simulators. After careful examination, we chose the ones that best suit our goals.

We compare DC models and simulators based on descriptions provided by the authors of the simulators. For each model we describe: *Resource Provisioning* – what resources are included and how they are modelled; *QoS* – what performance indicators are measured; *Costs* of computation in the DC; *Performance* of simulator – an estimation of the time needed to perform a simulation.

**CloudSim** is an event-based simulator implemented in Java, for simulation of cloud computing system and application provisioning environments.

*Resource Provisioning.* The CloudSim simulation layer offers dedicated management interfaces for CPU, memory, storage and bandwidth allocation, as well as, defining policies in allocating hosts to Virtual Machines (VM) – VM provisioning. Hosts are described by processing capabilities (in MIPS) and a core provisioning policy, together with an amount of available memory and storage. A model supports time-sharing and space-sharing core provisioning policies on both host and VM levels.

*Latency (QoS).* The latency model is based on conceptual networking abstraction, where the communication delays between each pair of entity type (e.g. host, storage, end-user) are described in a latency matrix as a constant value expressed in simulation time units (e.g. milliseconds).

*Costs.* CloudSim provides a two-layered cost model, where the first layer relates to Infrastructure as a Service (IaaS), with costs per unit of resources, while the second one relates to Software as a Service (SaaS), with costs per task units (application requests). This model allows calculation of the costs of using the cloud from the end-user perspective or the revenue from the IP perspective.

*Performance.* CloudSim is able to perform large-scale simulations, e.g., it can instantiate an experiment with 1 million hosts in 12 seconds. Moreover, memory usage grows linearly with the host number and even with 1 million hosts it does not exceed 320 MB.

**CloudAnalyst** (Wickremasinghe et al., 2010) is a simulator of geographically distributed large-scale cloud applications, that is developed in Java and utilises CloudSim and SimJava.

*Resource Provisioning.* CloudAnalyst uses the resource provisioning model of CloudSim.

*Latency (QoS).* A latency model allows configuration of network delays, available bandwidth between regions, and current traffic levels. CloudAnalyst facilitates experiments with latency by producing the following statistical metrics: average, minimum, and maximum response time of all user requests; and response time grouped by time of the day, location, and DC.

*Costs.* CloudAnalyst supports calculation of costs for using cloud resources, such as cost per VM per hour and cost per Gigabit of data transfer.

*Performance.* To improve performance of simulation entities are grouped at three levels: clusters of users, cluster of requests generated by users, and clusters of requests processed by VM.

**GreenCloud** is a packet level simulator based on NS-2, for simulation of energy-aware clouds.

*Resource Provisioning.* Servers are modelled as single core nodes with defined processing power limit (in

MIPS or FLOPS), size of memory and storage, and implementing different task scheduling mechanisms. *Latency (QoS)*. Full support for the TCP/IP protocol reference model is provided and the simulator calculates communication latency with high accuracy.

*Costs*. GreenCloud allows detailed modelling of energy consumption by implementing energy models for every DC element.

*Performance*. Given that GreenCloud has to simulate the full stack of Internet protocols, each simulation may take even tens of minutes for a DC with a few thousands of nodes.

### 5.3 Costs Models

The above mentioned DC models focus mostly on the costs of running applications in DCs from the end-user perspective. Since we want to model costs of DCs from the IP point of view, additional models are needed for capital expenditures (CAPEX) and operating expenditures (OPEX).

**CAPEX** includes costs of infrastructure that needs to be built and servers that have to be bought.

*Infrastructure Costs*. Costs of building, power distribution, (and cooling can be estimated using a following equation:  $\$200M \cdot (1 + c_m) / a_i$ , where  $c_m$  is the cost of money<sup>2</sup>, and  $a_i$  is the time of infrastructure amortisation [in years] (Greenberg et al., 2008).

*Server Costs*. Costs of servers can be modelled as  $n_s \cdot p_s \cdot (1 + c_m) / a_s$ , where  $n_s$  is the number of servers,  $p_s$  is the price of one server [in \$],  $c_m$  is the cost of money, and  $a_s$  is the time of server amortisation [in years] (Greenberg et al., 2008).

**OPEX** consists of power and personnel costs.

*Power Costs*. To estimate costs of power, the following equation can be used,  $n_s \cdot pc_s / 1000 \cdot PUE \cdot p_{kWh} \cdot 24 \cdot 365$ , where  $n_s$  is the number of servers,  $pc_s$  is the power consumption of one server [in W],  $PUE$  is Power Usage Efficiency, and  $p_{kWh}$  is the price of electricity [in \$ per kWh] (Greenberg et al., 2008).

*Personnel Costs*. Costs of personnel can be calculated using  $M_1 \cdot C_1 + M_2 \cdot C_2 + M_3 \cdot C_3$ , where  $M_1$  is the number of IT personnel per rack,  $M_2$  is the number of facility personnel per rack,  $M_3$  is the number of administrative personnel per rack, and  $C_1, C_2, C_3$  are the average costs per person for each of the above mentioned categories (Patel and Shah, 2005).

From another perspective, Muñoz et al., provide an evaluation of telecom, storage and computing costs for cloud infrastructure (Muñoz et al., 2011)

<sup>2</sup>Cost of money is the rate of interest or dividend payment on borrowed capital.

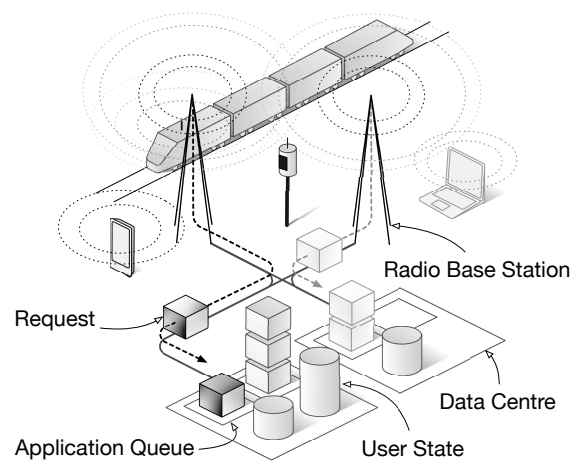


Figure 3: Visualisation of telco cloud meta-model.

## 6 TELCO CLOUD META-MODEL

We here detail how we have composed the above surveyed models into a telco cloud meta-model. Figure 3 depicts the visualisation of the meta-model. MDs, such as cell phones or laptops, are carried by end-users, who are in motion. The MDs generate requests which are sent over the network to a DC. It is also possible that requests are generated by sensors that may be static (e.g. traffic cameras) or mobile (e.g. trains). The requests are processed in the DC and the response is sent back to the MD or sensor. Processing requests, in case of stateful applications, generates a user state that has to be migrated with the end-user if he moves to the catchment of another DC.

The primary objective of the meta-model is to capture the interactions between application workload, MD mobility, network topology, and DC characteristics, and their influence on QoS and costs of telco cloud. The parameters that define the meta-model are presented in Table 2 and described in detail below.

### 6.1 Workload Model

The first group of parameters in Table 2 describes the mobility of end-users carrying MD and the characteristics of requests generated by these MDs.

#### Request Generation

Many services may run in the telco cloud at the same time and their number is defined by  $N_{ser}$ . We model a service application as a stateful web service. Each session is separated in time with a Poisson process  $\lambda_{ses}$  (Reyes-Lecuona et al., 1999). Each session produces  $N_{req}$  requests, sampled from an inverse Gaus-

Table 2: Fundamental meta-model parameters.

Type	Parameters	Unit	Description
WORKLOAD			
Request Generation	$N_{ser}$		Total number of services
	$\lambda_{ses}^i$ , where $i = 1, 2, \dots, N_{ser}$	s	Session arrival rate to DC
	$N_{req}^i$ , where $i = 1, 2, \dots, N_{ser}$		Number of requests per session
	$S_{req}^i$ , where $i = 1, 2, \dots, N_{ser}$	KB	Size of requests
	$D_{req}^i$ , where $i = 1, 2, \dots, N_{ser}$	s	Inter-request time
Resource Requirements	$CPU_{idle}^i, CPU_{req}^i$ , where $i = 1, 2, \dots, N_{ser}$	MI	CPU cycles used by service
	$mem_{idle}^i, mem_{req}^i$ , where $i = 1, 2, \dots, N_{ser}$	MB	Size of memory used by service
	$disk_{idle}^i, disk_{req}^i$ , where $i = 1, 2, \dots, N_{ser}$	MB	Size of storage used by service
	$state^i$ , where $i = 1, 2, \dots, N_{ser}$	MB	Size of user's state per request
Mobility	$N_{MD}$		Number of Mobile Devices
	$s_t^i, a_t^i, \theta_t^i, \omega_t^i$ , where $i = 1, 2, \dots, N_{MD}$		Movements of Mobile Devices
SETUP			
Network	$N_{RBS}$		Number of Radio Base Stations
	$d_{RBS}$	m	Dimensions of an RBS cell
	$D_{net}$	s	Cumulative network delay
Data Centre	$N_{DC}$		Number of Data Centres
	$N_S^i$ , where $i = 1, 2, \dots, N_{DC}$		Number of servers in Data Centre
	$N_{CPU}^j$ , where $j = 1, 2, \dots, N_S^i$		Number of CPUs per server
	$s_{CPU}^j$ , where $j = 1, 2, \dots, N_S^i$	MIPS	CPU's speed
	$memory^j$ , where $j = 1, 2, \dots, N_S^i$	MB	Amount of memory per server
	$storage^j$ , where $j = 1, 2, \dots, N_S^i$	GB	Amount of storage per server
	$network_{bw}^i$ , where $i = 1, 2, \dots, N_{DC}$	Mb/s	Network bandwidth
	$t_{init}, t_{idle}, t_{term}$	s	Times of VM transitions
Service Placement	$placement = \{every, n-closests, \dots\}$		Service placement policy
OBJECTIVES			
Quality of Service	$RT^i$ , where $i = 1, 2, \dots, N_{ser}$	s	Application response time
	$TP^i$ , where $i = 1, 2, \dots, N_{ser}$	req/s	Application throughput
Costs	$Cost$	\$	Total costs of infrastructure

sian distribution, where each request is separated in time by Log-Normal distributed delay  $D_{req}$  in seconds. The size of each request is given by  $S_{req}$  KB and is drawn from a Pareto distribution.

### Resource Requirements

To model application resource requirements we propose a linear model specifying the needed amount of resources, both for an idle service and per processing each request. An idle service uses  $CPU_{idle}$  CPU operations,  $mem_{idle}$  amount of memory, and  $disk_{idle}$  amount of storage. Additionally for each processed request, the service uses  $CPU_{req}$  CPU operations,  $mem_{req}$  amount of memory, and  $disk_{req}$  amount of storage. The amount of user's state data created by each request is defined by  $state$  and expressed in absolute value or percentage of request size  $S_{req}$ .

### Mobility

The network is populated by  $N_{MD}$  MDs, each subscribing to a subset of the  $N_{ser}$  available services. The 2-dimensional, multi modal, mobility model detailed in (Bettstetter, 2001) provides us with an on-average uniform distribution of users, with movement proportional to the duration of a session and the scale of the mobile network. The aforementioned model defines the properties of an MD's movement. A MD's momentary movement is defined by its velocity constituted by the current speed  $s$  and current direction  $\theta$ . Changes in mobility are defined by multiple stochastic processes that describe the duration of its state. An entity's speed  $s$  is independent of direction  $\theta$  and is maintained for  $T_s$  seconds, after which acceleration  $a$  between  $a_{min}$  and  $a_{max}$  is applied for time  $T_a$ , until it reaches  $s_{min}$  or  $s_{max}$ . Furthermore, direction  $\theta$  is maintained for time  $T_\theta$  until the next change-event where

the direction  $\theta$  is altered for  $T_\omega$  seconds with at the rate of  $\omega$  radians per second.  $T_s$ ,  $T_a$ ,  $T_\theta$ , and  $T_\omega$ , describing the timing of each change-event, are set for each mobility mode, and are each defined by a probability distribution bounded by a maxima and minima.

## 6.2 Setup Model

The second group of parameters in Table 2 characterises the network and DCs.

### Network

In our model, the core network introduces a cumulative propagation, switching, and routing delay and it is modelled with a Weibull delay  $D_{net}$  in multiples of the number of network nodes between the source and the destination (Papagiannaki et al., 2003). The network distance between RBSs is equal to the cell dimension  $d_{RBS}$ . The associated RBSs are equidistant to their common DC, and are for the sake of simplicity assumed to be separated by one network edge.

Furthermore, forthcoming cell planing practices aim to increase area energy efficiency by favouring smaller cells in urban areas (Shahab et al., 2013; Fehske et al., 2009). Our model employs a small homogeneous mobile network composed of  $N_{RBS}$  equidistantly distributed RBSs.

In the absence of a specific mobile generation standard, an MD is handed over between RBSs at the geographic point where they cross the cell boundary distinguishing two independent RBSs defined by the width of the rectangular cells  $d_{RBS}$ .

### Data Centre

The DC model captures the influence that its capacity has on performance and costs of computation, as described in Section 4.

To capture the influence on performance, quantity and quality of each DC resource is described. DC consists of  $N_S$  servers, that can differ in specification. Server contains  $N_{CPU}$  CPUs capable of executing  $s_{CPU}$  operations in every second. Values of *memory* and *storage* specify the total amount of available memory and storage, respectively. The network bandwidth is specified with  $network_{bw}$ . The DC model includes also a provisioning model, that describes how available resources are shared among several applications, e.g., time-sharing or space-sharing.

A DC hosts services in VMs. A service can be distributed over multiple VMs. Incoming workload is load-balanced by either a method of round-robin, random selection, or placed in the VM with the lowest load. However, a user's requests are always for-

Table 3: States of Virtual Machine.

Name	Description
INACTIVE	VM is turned off.
INITIATING	VM is booting up.
PROCESSING	VM is serving requests.
IDLE	VM is waiting for requests.
MIGRATING	VM is transmitting data.
TERMINATING	VM is shutting down.

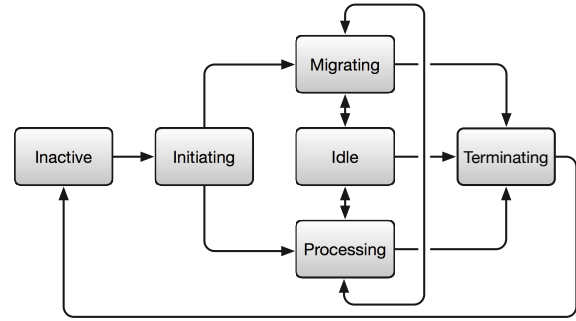


Figure 4: Transitions between Virtual Machine states.

warded to the VM that served his first request. A service can specify a minimum and maximum number of VMs it requires. The DC scales the application within these bounds based on the load-balancing outcome.

To emulate the life-cycle of a VM we have defined six VM states as described in Table 3. The transitions between the states are presented in Figure 4. At the beginning all VMs are in INACTIVE state. A VM is initiated when the first request arrives to a DC. It takes  $t_{init}$  seconds before VM is ready to start processing requests or receiving migrated requests and user state from other DC. We assume that a VM is not able to process requests and handle migrations at the same time, so it changes state between PROCESSING and MIGRATION over the time. Moreover, we give migrations a higher priority than processing, so processing is paused if there are any migrations to perform. When there are no requests to process and no migrations to handle a VM goes into IDLE state. A VM is terminated if IDLE state lasts for longer than  $t_{idle}$  seconds, and the VM termination takes  $t_{term}$  seconds.

### Service Placement

Service placement policies define in what DC(s) a service should be hosted, what number of replicas should be running, and when a service should be migrated between DCs. These decisions depend on the mobility of users, the size of users' state that has to be migrated, and QoS requirements. For example, a service can be hosted in  $n$  Proximal DCs closest to the majority of its users ( $n$ -closests), or in the case of la-

tency sensitive services in every Proximal DC that is needed to provide acceptable QoS (*every*).

### 6.3 Objectives Model

The third group of parameters in Table 2 describes QoS and costs of the telco cloud.

#### Quality of Service

Combining the resource requirements model, which describes the amount of resources an application needs, with a DC model, allows to simulate how co-location of different services in a DC influences their response times  $RT^i$  and throughputs  $TP^i$ .

#### Costs

In our opinion the cost models available in the literature and described in Section 5.3 are very "country dependent", because of the inclusion of variable parameters such as salaries, costs of energy or costs of property. They are also not taking into account parameters important from the perspective of the telco cloud, such as the size of DC. Therefore, we model the costs of the telco cloud using a basic heuristic based on observation that dispersion of infrastructure causes additional costs, e.g.: increase of administrator travel time between locations, and higher unit costs of computation in proximal DCs because of smaller scale and high initial costs.

$$\text{Cost} \propto \frac{N_{\text{DC}}}{\sum^{\text{DC}} N_S} \quad (1)$$

As shown in Equation 1, the total cost of a telco cloud is directly proportional to the number of DCs and inversely proportional to the total number of servers in all DCs. It means that distributing the same number of servers among many DCs is more expensive than placing them in a single DC.

### 6.4 Limitations

The proposed meta-model has several limitations. The application model assumes that all requests generated by one application are homogeneous and that each of them consumes the same amount of resources. The mobile access network model does not take into account the physical layer, channel provisioning, and cell load balancing. Additionally, the radio access network functions as a mechanism to associate MDs with DCs propagation, and system processing delays are thus not modelled.

## 7 SIMULATION SHOWCASE

We have implemented a coarse grained simulator using SimJava (Howell and McNab, 1998) as the underlying event-driven simulation framework. All modules are implemented from scratch but are based on the meta-model presented in Section 6. The simulator fully implements the proposed request generation and network models, but has implemented more abstract mobility, resource requirements, DC, and service placement models.

To demonstrate the scope of the telco cloud meta-model and the simulator we introduce an elementary showcase scenario below. The scenario is designed to reveal the basic relationship between workload – MD mobility, setup – Proximal DC catchment, and objectives – the aggregate utilisation of a telco cloud.

### 7.1 Experiments

For the sake of clarity we present a simplified scenario. Only one service is considered and the size of the simulation is reduced when compared to the desired scale. The VM scalability and placement models are included as proof-of-concepts. The goal is to obtain conclusions about the relation between MD mobility and DC catchment, and avoid the interference of other elements. The scenario is described in detail below.

The telecommunication infrastructure is composed of 16 RBSs, in a 4x4 layout. The cells are tangent but not overlapping and are dimensioned as a typical LTE micro-cell at 750 m, as detailed in (Shahab et al., 2013). The number of DCs varies between the experiments and thus so, also the DC catchment defined as the ratio between DCs and RBSs, changes between (1:1) and (1:16). In abstract terms, the (1:1) catchment represents a setup with one Proximal DCs per RBS. In contrast, the (1:16) catchment approaches a more traditional case of a Remote DC serving all users in the domain.

To reveal the effects of DC catchment, all DCs are of the same capacity. The number of VMs in each DC is scaled proportionally to the number of users they serve. The DC in the (1:16) catchment scenario has 16 VMs, while the DC in the (1:1) scenario has just one VM. The workload is balanced among available VMs, and new sessions are forwarded to the least loaded VM. To reveal the full extent of the effect of user mobility, user states and requests are strictly migrated to the geographically nearest DC.

We use a request generation model with a session arrival rate of  $\lambda_{ses}$  described by a Log-Normal distribution with the parameters  $\mu = 3$  and  $\sigma = 1.1$ .

The number of requests per session  $N_{req}$  is taken from an inverse Gaussian distribution with the parameters  $\lambda = 5$  and  $\mu = 3$ . Inter-request time is  $D_{req}$  seconds and is modelled with an exponential distribution with  $\lambda = 0.1$ . The simulation domain is populated by 480 MDs, all subscribing to the same service. Due to the size and simplicity of the network topology in the scenario, we are deploying a Markov-based mobility model. The mobility model is based on a car and is as specified in Section 6.1, with parameters from (Bettstetter, 2001). To allow the mobility and workload models to jointly reach a steady state, the simulation is run for 8 simulated hours. This results in an average processing load of 30%, which gives enough time for migrations to complete successfully.

The user state is proportional to the aggregate size of that user's sessions with the application it subscribes to and is defined by a 5<sup>th</sup> order AR-process with linearly decaying parameters. In our simulation, initialisation of a VM takes  $t_{init} = 81s$ , similarly as for m1.small VM type in Amazon EC2 (Ostermann et al., 2010). A VM is terminated if it remains in the IDLE state longer than  $t_{idle}$  which is equal to the mean inter-session time. It takes  $t_{term} = 21s$  to terminate a VM.

To investigate the influence of Proximal DC catchment on the performance of the telco cloud we observe the life cycle of the VMs by recording the amount of time they spend in each state. We run two sets of experiments. In the first set, end users are static. The second set introduces mobility.

## 7.2 Results

Figure 5(a) shows the breakdown of the mean time spent in each VM state in the system per DC catchment. With a (1:1) DC catchment the utilisation suffers from the proportion of time spent in IDLE state due to the relatively low request arrival rate generated by one sixteenth of all users. The inefficiency is caused by the time the system spends in the IDLE, INITIATING, and TERMINATING states. The composition of time spent in these states changes with DC catchment, and is a reflection of the number of VMs in a DC and load-balancing effort. Reducing the time spent on starting and terminating VMs would free up more resources and perhaps also make the system more reactive to sudden workload changes. The management of VM scalability and placement in telco clouds is clearly something that requires optimisation.

Figure 5(b) reveals the overhead of user mobility and the migration effort it incurs. Depending on the DC catchment, different migration dynamics come into play. As migrations are more frequent in the (1:1) case than in the (1:8) case, user states do not have the

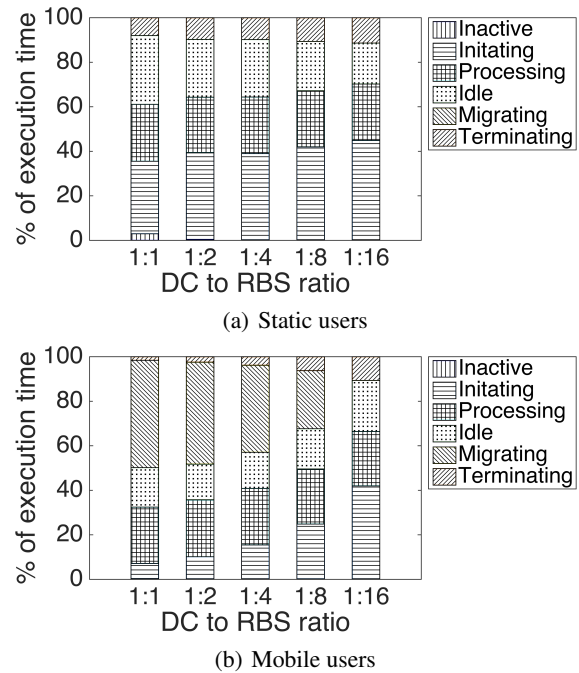


Figure 5: DC catchment vs. time spent in each VM state.

time to grow as much between migrations in the former case. The migration effort is therefore not a factor eight lower in the (1:8) case versus the (1:1) case, but rather, they spend 26% and 47% of their time in the MIGRATING state, respectively. The system dynamics revealed by Figure 5(b), where at worst, 47% of the execution time is spent migrating users, points to the need to find scaling mechanisms for the telco cloud that take into account mobility and inactivity, so that resources can be freed dynamically for other active applications. A policy of strictly migrating user states and requests to the geographically closest DC, irregardless of DC catchment, in order to obtain minimal propagation and communication latency, is sub-optimal.

## 8 CONCLUSIONS

In this paper we present a way to combine existing models of user mobility, mobile and core networks, and DCs into a meta-model capable of capturing dynamics of the telco cloud. We also implement a prototype simulator based on a simplified meta-model.

The meta-model can be used by telecommunication operators as well as equipment developers to model existing infrastructures and to plan future changes. Researchers can test algorithms for resource management, e.g., migration of services between geo-distributed DCs. Also developers can benefit from us-

ing the simulator to observe how their mobile applications behave in telco cloud environments.

Future work will be focused on enhancing the functionality of the simulator to incorporate other parameters from the presented meta-model. Then, using the simulator we would like to explore the following telco cloud challenges: minimising the trade-offs between costs and performance of telco cloud depending on the DC placement and capacity, and optimal placement and migration of services between DCs.

## ACKNOWLEDGEMENTS

This work is funded by the Swedish Research Council (VR) project Cloud Control and the European Union's Seventh Framework Programme under grant agreement 610711 (CACTOS). Maria Kihl and William Tärneberg are members of the Lund Center for Control of Complex Engineering Systems (LCCC) funded by the Swedish Research Council. Also, they are members of the Excellence Center Linköping – Lund in Information Technology (ELLIIT).

The authors thank Johan Eker (Ericsson Research) who contributed with feedback during several discussions and Tania Lorido-Botran (University of the Basque Country) for her critical comments on early drafts of the paper.

## REFERENCES

- Ahmed, A. and Sabyasachi, A. S. (2014). Cloud computing simulators: A detailed survey and future direction. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 866–872. IEEE.
- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.
- Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Comp. net.*, 54(15):2787–2805.
- Barker, S. K. and Shenoy, P. (2010). Empirical evaluation of latency-sensitive application performance in the cloud. In *Proc. of the 1st annual ACM SIGMM conference on Multimedia systems*, pages 35–46. ACM.
- Baroncelli, F., Martini, B., and Castoldi, P. (2010). Network virtualization for cloud computing. *Annals of telecommunications*, 65(11-12):713–721.
- Bettstetter, C. (2001). Smooth is better than sharp: A random mobility model for simulation of wireless networks. In *Proc. of the 4th ACM Int. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '01, pages 19–27. ACM.
- Bosch, P., Duminuco, A., Pianese, F., and Wood, T. L. (2011). Telco clouds and virtual telco: Consolidation, convergence, and beyond. In *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 982–988. IEEE.
- Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C., and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.
- Fehske, A., Richter, F., and Fettweis, G. (2009). Energy efficiency improvements through micro sites in cellular mobile radio networks. In *GLOBECOM Workshops, 2009 IEEE*, pages 1–5.
- Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. (2008). The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73.
- Howell, F. and McNab, R. (1998). Simjava: A discrete event simulation library for java. *Simulation Series*, 30:51–56.
- Kliazovich, D., Bouvry, P., and Khan, S. (2012). Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283.
- Kovachev, D. (2012). Framework for computation offloading in mobile cloud computing. *Int. J. of Interactive Multimedia and Artificial Intelligence*, 1(7):6–15.
- Muñoz, V. M., Kaci, M., Gadea, A., and Salt, J. (2011). On the Economics of Huge Requirements of the Mass Storage-A Case Study of the AGATA Project. In *CLOSER*, pages 507–511.
- Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D. (2010). A performance analysis of EC2 cloud computing services for scientific computing. pages 115–131. Springer.
- Papagiannaki, K., Moon, S., Fraleigh, C., Thiran, P., and Diot, C. (2003). Measurement and analysis of single-hop delay on an IP backbone network. *IEEE J. on Selected Areas in Communications*, 21(6):908–921.
- Patel, C. D. and Shah, A. J. (2005). Cost model for planning, development and operation of a data center.
- Reyes-Lecuona, A., González-Parada, E., Casilari, E., Casasola, J., and Diaz-Estrella, A. (1999). A page-oriented www traffic model for wireless system simulations. In *Proc. ITC*, volume 16, pages 1271–1280.
- Riley, G. F. and Henderson, T. R. (2010). The ns-3 network simulator. In *Modeling and Tools for Network Simulation*, pages 15–34. Springer.
- Sakellari, G. and Loukas, G. (2013). A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. *Simulation Modelling Practice and Theory*, 39(0):92 – 103.
- Shahab, S., Kiong, T., and Abdulkafi, A. (2013). A Framework for Energy Efficiency Evaluation of LTE Network in Urban, Suburban and Rural Areas. *Australian J. of Basic and Applied Sciences*, 7(7):404–413.
- Varga, A. et al. (2001). The OMNeT++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM'2001)*, volume 9, page 65.
- Wang, A., Iyer, M., Dutta, R., Rouskas, G. N., and Baldine, I. (2013). Network virtualization: Technologies, per-



spectives, and frontiers. *Journal of Lightwave Technology*, 31(4):523–537.

- Wickremasinghe, B., Calheiros, R., and Buyya, R. (2010). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 446–452.





## AUTHOR INDEX

Adkinson-Orellana, L. ....	514	Dubois, E. ....	565
Åhlund, C. ....	145	Elmroth, E. ....	597
Ahmed, N. ....	163	Feinbube, F. ....	427
Akaichi, J. ....	310	Felhi, F. ....	310
Akhbar, F. ....	113	Ferme, V. ....	241
Alansari, M. ....	451	Ferrucci, F. ....	521
Almeida, A. ....	451	Flliopoulou, E. ....	499
Andrikopoulos, V. ....	352	Fortes, R. ....	331
Angelis, F. ....	119, 155	Gantikow, H. ....	543
Anseeuw, J. ....	304	Garg, R. ....	87
Ayadi, M. ....	310	Ghandorh, H. ....	251
Barán, B. ....	439	Ghédira, K. ....	199
Barbhuiya, S. ....	343	Ghodous, P. ....	276
Barker, A. ....	373	Ghose, A. ....	233
Barros, B. ....	214	Gillam, L. ....	60
Bauer, H. ....	297	González-Castaño, F. ....	514
Bencomo, N. ....	451	Gouvas, P. ....	206
Benhammadi, F. ....	105	Greiner, T. ....	127
Bey, K. ....	105	Grolinger, K. ....	186
Bhargava, B. ....	163	Gupta, S. ....	194
Bhiri, S. ....	397	Gusev, M. ....	71
Biennier, F. ....	276	Hadji, M. ....	17
Binz, T. ....	487	Hahn, M. ....	352
Blažič, B. ....	365	Halima, Y. ....	199
Blödorn, L. ....	297	Hans, R. ....	221
Blomer, R. ....	268	Haque, R. ....	260
Bordbar, B. ....	451	Harper, R. ....	559
Breitenbücher, U. ....	475, 487	Hasham, K. ....	49
Buckley, K. ....	172	Healy, P. ....	95
Bulander, R. ....	127	Herbert, J. ....	79
Capretz, M. ....	186	Holzschuher, F. ....	535
Carvalho, H. ....	406	Hübsch, G. ....	206
Carvalho, N. ....	406	Hunt, G. ....	95
Carvalho, T. ....	214	Ippoliti, F. ....	155
Casola, V. ....	551	Iturbe, E. ....	551
Cavallo, M. ....	414	Iwaya, L. ....	214
Chaki, N. ....	233	Jaatun, M. ....	30
Chenal, B. ....	565	Janusz, D. ....	427
Cigoj, P. ....	365	Jiménez, L. ....	145
Corcoran, D. ....	95	Jlassi, A. ....	178
Corradini, F. ....	119, 155	Johansen, D. ....	586
Cruzes, D. ....	30	Karastoyanova, D. ....	352
Deb, D. ....	233	Kaviani, N. ....	381
Defude, B. ....	397	Kechadi, M. ....	521
Donevski, A. ....	71	Kemmler, B. ....	135

## AUTHOR INDEX (CONT.)

Khadraoui, D. ....	565	Nikolaidou, M. ....	499
Khemakhem, M. ....	506	Nikolopoulos, D. ....	343
Kihl, M. ....	597	Oikawa, S. ....	529
Kilpatrick, P. ....	343	O'Loughlin, J. ....	60
Klingberg, S. ....	543	Orue-Echevarria, L. ....	551
Klobučar, T. ....	365	Östberg, P. ....	597
Knoblauch, J. ....	127	Ouedraogo, M. ....	565
Kopp, O. ....	284, 487	Ouedraogo, W. ....	276
Kostoska, M. ....	71	Ovatman, T. ....	113
Kranzlmüller, D. ....	135	Pakdel, R. ....	79
Kristiansson, J. ....	145	Papazachos, Z. ....	343
Krzywda, J. ....	597	Paraskakis, I. ....	206
Kübler, E. ....	389	Park, W. ....	5
Kvalnes, Å. ....	586	Pautasso, C. ....	241
Lampe, U. ....	221	Pedrero-López, B. ....	514
Lawkobkit, M. ....	268	Peinl, R. ....	535
Leymann, F. ....	241, 284, 352, 475, 487	Pettersen, R. ....	586
Liang, Y. ....	40	Pires, F. ....	439
Lucrédio, D. ....	331	Poggi, S. ....	565
Lutfiyya, H. ....	251	Polini, A. ....	119
Lynn, T. ....	95	Polito, C. ....	414
Malik, Z. ....	577	Polze, A. ....	427
Marcantoni, F. ....	155	Pulls, T. ....	321
Marco, L. ....	521	Rak, M. ....	551
Martineau, P. ....	178	Reich, C. ....	321, 543
Mataoui, M. ....	105	Rezgui, A. ....	577
Maximilien, M. ....	381	Richerzhagen, B. ....	221
McClatchey, R. ....	49	Rios, E. ....	551
Méhes, A. ....	214	Ristov, S. ....	71
Meskini, A. ....	260	Rodríguez-Silva, D. ....	514
messaoud, W. ....	199	Roller, D. ....	241
Michalakelis, C. ....	499	Rouvellou, I. ....	381
Michalas, A. ....	206	Ruebsamen, T. ....	321
Minor, M. ....	389	Sabbatini, S. ....	119
Misra, S. ....	194	Sáez, S. ....	352
Mitropoulou, P. ....	499	Salapura, V. ....	559
Moalla, S. ....	506	Samet, K. ....	506
Modica, G. ....	414	Schelén, O. ....	145
Mohamed, M. ....	186	Schiefer, G. ....	206
Molina-Jimenez, C. ....	463	Schwarz, C. ....	297
Moreira, A. ....	331	Sebbak, F. ....	105
Morrison, J. ....	95	Seghbroeck, G. ....	304
Munir, K. ....	49	Sellami, R. ....	397
Näslund, M. ....	214	Serrão, C. ....	406
Neuhaus, C. ....	427	Sfyarakis, I. ....	463

## AUTHOR INDEX (CONT.)

Silva, C. ....	276
Silva, E. ....	331
Silva-Lepe, I. ....	381
Simplicio Jr., M. ....	214
Simón, M. ....	145
Skouradaki, M. ....	241, 352
Slimani, Y. ....	260
Solaiman, E. ....	463
Steffen, D. ....	221
Steinmetz, R. ....	221
Stiller, B. ....	87
Synnes, K. ....	145
Taher, Y. ....	260
Tärneberg, W. ....	597
Thai, L. ....	373
Tkindt, V. ....	178
Tomarchio, O. ....	414
Tsaroucha, S. ....	499
Turck, F. ....	304
Valvåg, S. ....	586
Varghese, B. ....	373
Vedrine, M. ....	397
Verginadis, Y. ....	206
Volckaert, B. ....	304
Vukojevic-Haupt, K. ....	352
Wagner, S. ....	284
Wettinger, J. ....	475, 487
Wood, K. ....	172
Yamany, H. ....	186
Yang, C. ....	5
Zinser, E. ....	297

**Proceedings of CLOSER 2015**

5<sup>th</sup> International Conference on Cloud Computing and Services Science

[www.closer.scitevents.org](http://www.closer.scitevents.org)

IN COOPERATION WITH:



**ISBN:** 978-989-758-104-5

Copyright © 2015 **SCITEPRESS** - Science and Technology Publications - All Rights Reserved