3-D Cloud Monitoring: Enabling Effective Cloud Infrastructure and Application Management

Clarissa Cassales Marquezan*, Dario Bruneo†, Francesco Longo†, Florian Wessling*,
Andreas Metzger*, Antonio Puliafito†
* Paluno, University of Duisburg-Essen, Essen, Germany
{clarissa.marquezan, florian.wessling, andreas.metzger}@paluno.uni-due.de
†Dipartimento di Ingegneria DICIEAMA, Università degli Studi di Messina, Messina, Italia
{dbruneo,flongo,apuliafito}@unime.it

***Abstract*—A cloud environment is a complex environment composed of many different entities and layers. Each of these cloud entities may be furnished with mechanisms offering various management actions. For any given situation, different management actions may be applicable and often simultaneously. Enforcing isolated management actions or combining contradictory management actions may negatively affect cloud application quality and cloud infrastructure performance. This means that correctly selecting and effectively combining these management actions for a given situation becomes an important challenge in cloud computing. In this paper, we address the problem of identifying situations where more than one management action can be performed. The key contributions of our paper are: (1) a three dimensional (3-D) monitoring model for analyzing cloud monitoring information; (2) the concept and formalization of Context of Interest (CoI) that specifies how to retrieve meaningful information from the 3-D model to support the coordination of management actions between cloud infrastructure and application. We conducted experiments in a real testbed using Openstack and the WordPress Web site application. Our results show that analyzing cloud monitoring information using the 3-D model and the CoI can support a more effective identification of management actions to be taken.**

I. Introduction

Appropriate cloud management is essential. Popular management actions to handle dynamic changes inside the cloud environment are: migration and elasticity of virtual resources [1][2][3][4], multi-cloud resource management [5][6], load balancing [7], and infrastructure resource management [8]. However, a cloud environment is complex and composed of many different entities and layers [9]. These entities include, to name a few, physical resources, virtual resources deployed on top of physical resources, Web application servers deployed inside virtual machines, as well as software code running inside a Web application server. These entities typically offer dedicated management actions that can be used to enable dynamic adaptation at runtime. As a consequence, various management actions can be applied simultaneously and this might negatively affect both cloud application quality of service and cloud infrastructure performance [9].

Thus, the decision on which management action(s) are appropriate for a given situation depends on monitoring and analysis support able to consider the complex dependencies between cloud entities and their management mechanisms. Few solutions in cloud monitoring consider the collection of information from different layers [10]. Other works focus on building the support for collecting information from the cloud environment [11] or automating the configuration of the monitoring solutions, such as Chef and Puppet frameworks. However, these solutions do not provide a way to analyze this data in order to identify multiple management actions for a given situation. In addition, current management solutions for cloud applications [3][4] and cloud infrastructures [12][13] are still limited in analyzing the information and adaptation options from the different layers of cloud environment.

In this paper, we address the problem of identifying situations where various management actions can be performed. Instead of looking at the cloud monitoring information as isolated streams of data, we propose to analyze this information as a three-dimensional data space. Inside this space there are combinations of points that indicate situations in which one or more management actions could be applied in different layers of the cloud. We assume that the Monitoring as a Service model is used by all the stakeholders of a cloud environment that are willing to coordinate their management actions. This allows information to be collected from the different layers of the cloud environment. The key contributions of our paper are: (1) a three dimensional (3-D) monitoring model for analyzing cloud monitoring information as volumes; and (2) the formalization of Context of Interest (CoI) that specifies how to retrieve meaningful information from the 3-D monitoring model to support coordinated infrastructure and application management actions and handle dynamic changes in the cloud environment. We applied our formulation to a testbed using Openstack and the WordPress application. Experiments conducted in the testbed show that with the 3-D monitoring model and Contexts of Interest it is possible to identify multiple management actions and support a more efficient decision-making on which action should be taken.

The remainder of this paper is organized as follows. Section II describes the related work. Section III presents the background on cloud layers and introduces the 3-D monitoring model. The Context of Interest is formalized in Section IV. Section V describes the experiments. Finally, Section VI contains the conclusions and future work.

II. Related Work

In this section, we discuss two classes of related work. First, the ones devoted to cloud monitoring. Second, we analyze which kind of monitoring information is currently used for performing management actions in cloud environments.

Montes et al. [10] introduced the idea of levels of monitoring in a cloud environment. They defined the monitoring in the virtual system level (composed by the sub-levels application, platform, and infrastructure) and the physical system level. The focus of this work is the collection of the monitoring information and not how the information is processed and associated with different layers of monitoring to be correlated.

Popular tools such as Chef and Puppet are indeed frameworks for the automatic configuration of infrastructures. They also provide support for the most common available resource usage visualization tools. Puppet provides some more functionalities for inspection of events grouped together. However, the combination of more sophisticated information in a cloud environment is still a task to be done by human administrators.

In addition, monitoring tools and APIs are also available on cloud commercial and open source solutions such as Google App Engine, Amazon Web Services, Ceilometer, Ganglia, Nagios, and Zabbix. The Google App Engine provides support for monitoring a fixed set of performance metrics. Amazon Web Services (AWS) provide a set of tools and APIs for the developers, such as AWS OpsWorks, CloudWatch, and EC2. CloudWatch provides a large set of metrics and possibilities for the configuration of new metrics and it also allows to capture some metrics associated with EC2 (related to auto scaling). Ceilometer provides all basic information about the OpenStack components, such as Nova-Compute, Swift, Cinder, etc. It provides support for the generation of events. The traditional monitoring frameworks such as Ganglia, Nagios, and Zabbix also support extensions for monitoring the cloud environment. However, the aforementioned solutions do not provide the sophisticated logic for relating monitoring information since their purpose is the collection of monitoring information and handling event notifications.

Leitner et al. [14] introduce a framework where application developers can define which application metrics should be monitored. The framework collect the desired application information from single hosts (i.e., single VMs), pools of hosts (i.e., pools of VMs) and correlate this information. In their case, correlation means filtering and aggregating data to reduce the flood of information. No analysis specifically target to activating management actions is proposed in this work. Differently from this work, for us, set of pieces of monitored information are modeled to indicate one or more management actions in one or more layers of the cloud environment. Miglierina et al. [6] and Dutta et al. [3] both proposed solutions for auto-scaling cloud applications. Their solutions consider monitored information about the architecture of the application in addition to the ones from the cloud virtualization layer. He et al. [4] provide a framework for the deployment, migration, and elasticity of applications based on Elastic Application Containers. The monitored information used by the them is related to physical and virtual machines, such as CPU, I/O operations, and network traffic. In contrast to these approaches, we focus on using different sources of monitoring information not only for auto-scaling but also for triggering different types of management mechanisms inside a cloud environment.

In addition to the work on the adaptation of cloud applications, there is a large amount of research related to management of cloud infrastructures. Fischer et al. [8] present a large survey about solutions for embedding virtual resources into physical infrastructures. The main information taken into account are IaaS measurements, such as network usage, CPU, and memory. Recent works proposed by Wuhib et al. [12] and Esteves et al. [13] also take into account the structure of the application as part of the problem formulation of cloud resource management. However, no other application information is considered for taking the management decisions for allocating the resources.

As discussed above, there is still a gap when it comes to better combining and analyzing the information from the cloud environment in order to enable a more coordinated execution of management actions in a cloud environment. The advances in monitoring information collection and more sophisticated management solutions for cloud management discussed in this section serve as basis for the next step proposed in this paper.

III. New Approach for Cloud Monitoring

For adaptations to take place, it is first necessary to monitor the information that is relevant to detect a situation in which changes are required, i.e., management actions should be activated. In a previous work [9], we defined the layers and entities inside the cloud environment that can initiate or suffer adaptation actions, and the influences that adaptation actions can cause among the various entities of the cloud environment. In this paper, we focus on defining the necessary monitoring model and initial support to enable the identification of situations inside the cloud environment that can lead to activation of distinct management actions in different entities of the cloud layers. The long term goal of our research is to automatically analyze the monitored situations and their multiple management actions, aiming at enabling a better coordination of these actions to reduce their influences in the entities of cloud environment. In this section, we focus on the presentation of the reference cloud scenario and its layers. Then, we propose a high level three dimensional cloud monitoring data model, called 3-D monitoring model. The purpose of this model is to support the identification of monitoring information inside the cloud environment that will reveal situations leading to multiple and eventually conflicting management actions.

A. Cloud Layers

Fig. 1 depicts a simplified version of the four layers and entities composing our reference model of a cloud environment [9]. The *Physical Layer* (PL) is in the base of Fig. 1. It includes the data center physical resources (e.g., Physical Machines and routers) and all management entities controlling those

physical resources. The *Virtualization Layer* (VL) is composed of virtual resources and the respective management support. The *Application Architecture Layer* (AAL) encloses the set of entities, such as servers and software components, necessary to support the architecture of an application. For example, in the case of Web applications, the components associated with this layer are Web Servers and Database servers. Examples of monitoring metrics in this layer are: number of transactions per second and response time of the server requests. Finally, the *Application Business Logic Layer* (ABLL) involves the entities directly related to the application business logic implementation. In the case of an Web application, an example of monitoring information in this layer is the response time of method associated with the purchase order.

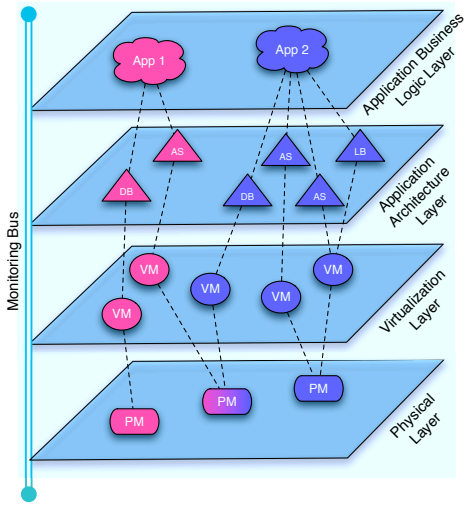


Fig. 1. Cloud layers, entities, and relationships.

As illustrated in Fig. 1, the different entities populating the cloud layers are related to each other. For instance, each Physical Machine (`PM`) can host multiple Virtual Machines (`VMs`). Each of these `VMs` can run different servers, e.g., Web servers, database servers, application servers that host components of the application. For instance, a server like Apache Tomcat is able to host different `servlets`, which are entities of the *Application Business Logic Layer*.

In order to make the cloud infrastructure ready for identification and later coordination of the multiple management actions, the entities in the layers and their relationships need to be accurately monitored. For instance, the approach proposed by Montes et al. [10] already foresees a mechanism for data collection such as the one represented in Fig. 1. However, just the collection of data is not enough. It is still necessary the definition of a monitoring model that can lead to a successful data analysis and easy runtime detection of situations where multiple management actions could be applied. The next section introduces such a monitoring model.

B. 3-D Cloud Monitoring Model

We propose using three dimensions to define a cloud monitoring model able to support an elaborated analysis of the measured information. This model is based on the following assumptions:

- We consider that for any type of applications executing inside the cloud environment there will be entities and relationships in different layers of this environment that can change and affect each other, independently of who is selling or buying cloud services. These entities can be monitored and their measurements can be correlated.
- We assume the existence of mappings among the entities across layers, as illustrated in Fig. 1. This means that virtual entities are mapped to physical ones, application architecture entities are mapped to virtual ones, and application business logic entities are mapped to application architecture ones. OpenStack, for example, has support for mapping the physical and virtual entities, but the mapping for the other layers is not supported yet.
- There must always exist one application associated with the entities in the *Application Architecture*, *Virtualization*, and *Physical* layers. It is out of the scope of this paper to discuss which stakeholder in the cloud environment would be responsible for keeping this information. We assume that the information is available and is delivered using a Monitoring as a Service model.

Based on the above assumptions, we propose that a specific metric within the space of the collected data can be identified by three different characteristics: (1) the `PM` on top of which the metric was collected, (2) the application to which the metric is related; and (3) the kind of metric and its corresponding cloud layer. These characteristics can be expressed as coordinates in a 3-dimensional representation of data collection space in a cloud environment, thus leading to the proposed 3-D monitoring model as depicted in Fig. 2.

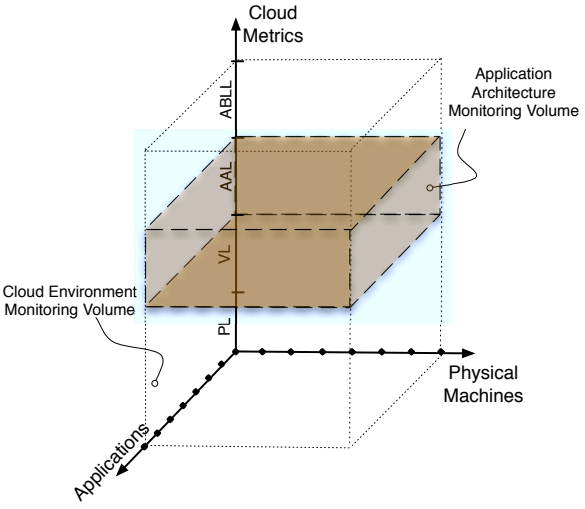


Fig. 2. Example of 3-D monitoring volume for the data collected in a Cloud Environment for the Application Architecture Layer.

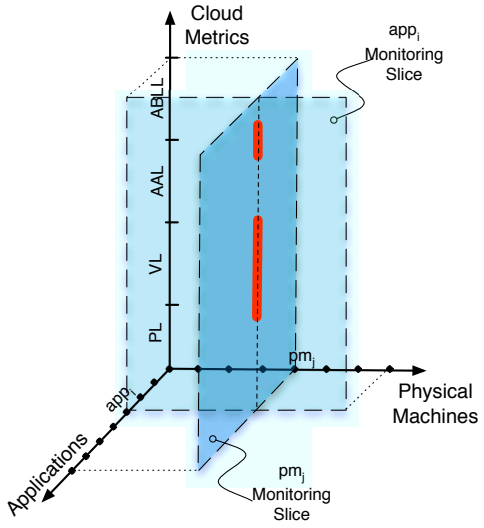


Fig. 3. Monitoring slices in the 3-D monitoring model.

The transparent volume with dotted lines illustrated in Fig. 2 represents the entire space of data that can be monitored in a cloud environment, and we named it: *Cloud Environment Monitoring Volume* (or monitoring volume). The monitoring volume can be sub-divided and reduced into monitoring sub-volumes. For instance, Fig. 2 shows the Application Architecture Monitoring Volume which is a sub-volume with monitoring metrics from the *Application Architecture Layer* related to all applications deployed in the cloud environment under consideration. In fact, the space of the collected metrics can be cut through with the use of planes to identify subsets of the space that are of interest for a given situation. For example, Fig. 3 shows the *monitoring slices* obtained by cutting the space through a couple of planes. The application app_i monitoring slice groups all the metrics in all of the layers that are collected for application app_i, on top of all the PMs that host virtual resources associated with app_i. On the other hand, the pm_j monitoring slice involves all the metrics in all the layers that are related to pm_j. The intersection of the two slices (highlighted in red) represents the set of the metrics in the different cloud layers that are related to application app_i and measured in pm_j. We use the intersection among monitoring slices from different planes to detect situations that will eventually lead to the activation of multiple management actions throughout the different cloud entities. However, not all the monitoring data in an intersection is relevant for the identification of these situations. Therefore, in the next section, we introduce the concept of *Context of Interest* (CoI), a formalization that allows dealing with the complexity of the monitoring volumes, slices, and intersections.

IV. Context of Interest

A *Context of Interest* (*CoI*) is the simultaneous observation of metrics in the intersection of monitoring slices that are relevant for taking management actions in different layers of the cloud environment in response to dynamic changes inside such an environment. The *Context of Interest* necessarily implies that for the observed information a distinct and maybe conflicting set of cloud management actions could be triggered. Thus, the definition and identification of *CoIs* enable the mapping of different management actions in different layers of the cloud environment. The *CoI* concept builds upon a sequence of definitions and characterizes the 3-D monitoring scope and the monitoring slices, defining which cuts on the monitoring slices are relevant to determine monitoring information that characterize situations leading to multiple management actions. This concept does not determine how information is collected in the cloud environment. It rather establishes how such information will be analyzed. In the following section, we provide a mathematical formulation of the *Context of Interest* concept in order to enable the automatic analysis of the collected monitoring information.

A. Formalization

Definition 1 (Application) An application is a set of components (e.g., systems, sub-systems) and it has necessarily a unique identifier in the cloud environment (app_i). The components of the application might be mapped to different VMs along the cloud infrastructure. However, they will be associated with the same application identifier.

Definition 2 (Application Set) The application set $\mathscr{A}$ refers to the set of all apps deployed inside the cloud environment. $\mathscr{A} = \{app_i\}$ where $1 \leq i \leq I$ and I is the total number of applications deployed in the cloud environment.

Definition 3 (Physical Machine Set) The physical machine set $\mathscr{P}$ refers to the set of all PMs in the cloud environment. $\mathscr{P} = \{pm_j\}$ where $1 \leq j \leq J$ and J is the total number of PMs in the cloud environment.

Definition 4 (Physical Layer Metric Set) The physical layer metric set $\mathscr{M}_{PL}$ refers to the set of all physical metrics in the cloud environment. $\mathscr{M}_{PL} = \{plm_p\}$ where $1 \leq p \leq P$ and P is the total number of physical layer metrics measured.

Definition 5 (Virtualization Layer Metric Set) The virtualization layer metric set $\mathscr{M}_{VL}$ refers to the set of all virtualization metrics in the cloud environment. $\mathscr{M}_{VL} = \{vlm_q\}$ where $1 \leq q \leq Q$ and Q is the total number of virtualization layer metrics that can be measured.

Definition 6 (Application Architecture Layer Metric Set) The application architecture layer metric set $\mathscr{M}_{AAL}$ refers to the set of all metrics related to the application architecture layer in the cloud environment. $\mathscr{M}_{AAL} = \{aalm_r\}$ where $1 \leq r \leq R$ and R is the total number of application architecture layer metrics that can be measured.

Definition 7 (Application Business Logic Layer Metric Set) The application business logic layer metric set $\mathscr{M}_{ABLL}$ refers to the set of all metrics related to application business logic layer in the cloud environment. $\mathscr{M}_{ABLL} = \{abllm_s\}$ where $1 \leq s \leq S$ and S is the total number of application business logic layer metrics that can be measured.

Definition 8 (Set of Metrics) The set of all metrics $\mathscr{M}$ is composed by all the metrics that can be measured in the cloud environment. $\mathscr{M} = \{m_k\}$ where $m_k \in (\mathscr{M}_{PL} \cup \mathscr{M}_{VL} \cup \mathscr{M}_{AAL} \cup \mathscr{M}_{ABLL})$, $1 \leq k \leq K$, and $K = P+Q+R+S$.

Definition 9 (Cloud Monitoring Volume) The cloud monitoring volume $\mathscr{V} = \{(app_i, pm_j, m_k)\}$ (with $|\mathscr{V}| = |\mathscr{A}| \times |\mathscr{P}| \times |\mathscr{M}|$) is the Cartesian product of sets $\mathscr{A}$, $\mathscr{P}$, and $\mathscr{M}$ indicating the space of information that can be collected in a cloud environment.

Fig. 2 graphically depicts $\mathscr{V}$, also highlighting the subset corresponding to the Application Architecture Layer. Let us consider a function $f : \mathscr{V} \times T \rightarrow \mathbb{R} \cup \{null\}$ where $T \subseteq \mathbb{R}^+$ is the set of time instants. Then, $f(i,j,k,t)$ is the value associated with the metric m_k related to the components of application app_i deployed in the PM pm_j, measured at time instant t. If the application app_i does not have any component actually deployed in pm_j or the measure m_k does not apply to the specific application, the *null* value is returned.

Definition 10 (Application Monitoring Slice) An application monitoring slice $\mathscr{S}_A$ is a plane $\{(\overline{app}, pm_j, m_k)\}$ in the cloud monitoring volume $\mathscr{V}$ obtained by considering all the points in the volume associated to a specific $\overline{app} \in \mathscr{A}$.

Definition 11 (Physical Machine Monitoring Slice) A physical machine monitoring slice $\mathscr{S}_{PM}$ is a plane $\{(app_i, \overline{pm}, m_k)\}$ in the cloud monitoring volume $\mathscr{V}$ obtained by considering all the points in the volume associated to a specific $\overline{pm} \in \mathscr{P}$.

Fig. 3 graphically depicts two monitoring slices. The intersection between the application monitoring slice associated to app_i and the physical machine monitoring slice associated to pm_j is represented by a dashed line.

Definition 12 (Observing Data Set) An observing data set $\mathscr{O}$ over a subset of metrics $\mathscr{M}_o \subseteq \mathscr{M}$ is a subset $\{(\overline{app}, \overline{pm}, m_o)\}$ (with $m_o \in \mathscr{M}_o$) of the monitoring volume $\mathscr{V}$ obtained by properly selecting the points in the intersection between a physical machine monitoring slice $\mathscr{S}_{PM}$ (associated to a physical machine $\overline{pm}$) and an application monitoring slice $\mathscr{S}_A$ (associated to an application $\overline{app}$) that correspond to metrics in $\mathscr{M}_o$.

From a practical point of view, an observing data set can be used to select a certain number of metrics of interest for the components of a specific application deployed on a certain PM. In Fig. 3, an example of an observing data set is highlighted in red.

Definition 13 (Application Cut Set) An application cut set $\mathscr{C}_A$ is a set of application monitoring slices (with $1 \leq |\mathscr{C}_A| \leq |\mathscr{A}|$).

Definition 14 (Physical Machine Cut Set) A physical machine cut set $\mathscr{C}_{PM}$ is a set of physical machine monitoring slices (with $1 \leq |\mathscr{C}_{PM}| \leq |\mathscr{P}|$).

Definition 15 (Application Context of Interest) An application context of interest CoI_A is a set of observing data sets $\{\mathscr{O}_z\}$ obtained from the intersection between the application monitoring slice $\mathscr{S}_A$ related to application $\overline{app}$ and a physical machine cut set $\mathscr{C}_{PM}$, with $1 \leq z \leq |\mathscr{C}_{PM}|$ satisfying the following constraints:

$$\exists\ c' = (\overline{app}, pm', m'),\ c'' = (\overline{app}, pm'', m'') \in CoI_A :$$
$$c' \in \mathscr{O}_{z'}, c'' \in \mathscr{O}_{z''},\ z' \neq z''\ \ OR$$
$$layer(m') \neq layer(m'')$$

where the function *layer* returns the metric set corresponding to a metric m. The constraints guarantee that there will exist at least 2 metrics associated with the same application that belong either to different PMs or layers of the cloud environment. This is necessary to characterize situations where management actions for the same application can be coordinated. Being these actions either in entities hosted in different physical machines or in different layers of the cloud environment but all associated with the same application.

Definition 16 (Physical Machine Context of Interest) A physical machine context of interest CoI_{PM} is a set of observing data sets $\{\mathscr{O}_z\}$ obtained from the intersection between the physical machine monitoring slice $\mathscr{S}_{PM}$ related to the physical machine $\overline{pm}$ and an application cut set $\mathscr{C}_A$, with $1 \leq z \leq |\mathscr{C}_A|$ satisfying the following constraints:

$$\exists\ c' = (app', \overline{pm}, m'),\ c'' = (app', \overline{pm}, m'') \in CoI_{PM} :$$
$$c' \in \mathscr{O}_{z'}, c'' \in \mathscr{O}_{z''},\ z' \neq z''\ \ OR$$
$$layer(m') \neq layer(m'').$$

The constraints, in this case, guarantee that there will exist at least 2 metrics associated with the same PM that belong either to different applications or layers of the cloud environment. The objective of these constraints is to characterize situations to coordinate management actions inside the same physical machine that affect either different applications or entities in different layers of the cloud for the specific physical machine.

Definition 17 (Context of Interest) A context of interest *CoI* is either an application context of interest or a physical machine context of interest. Multiple *CoIs* can be defined for the same cloud monitoring volume. In addition, the same type of metric can appear in one or more *CoIs*.

In a cloud environment, the main role played by distinct *CoIs* is to support the detecting of specific situations leading to multiple management actions. In the next section, we use the WordPress application to provide concrete examples of how *CoIs* and management actions are related to each other.

B. Context of Interest and Management Actions

The goal of using the *CoIs* is to define the set of metrics that expose situations in which multiple management actions would be applicable, but not all of them should be actually enforced. This is because either one of the management actions is not the most appropriate for the situation or because all of them applied together will only reinforce the problem experienced by the application. In this section, we describe how *CoIs* can

TABLE I
EXAMPLE OF A *CoI* FOR THE WORDPRESS APPLICATION.

Layer	Tuple	Description
$\mathscr{M}_{PL}$	(app_1, pm_2, plm_cpu)	Physical CPU Usage
$\mathscr{M}_{PL}$	(app_1, pm_2, plm_mem)	Physical Memory Usage
$\mathscr{M}_{VL}$	(app_1, pm_2, vlm_cpu)	Virtual CPU Usage
$\mathscr{M}_{VL}$	(app_1, pm_2, vlm_mem)	Virtual Memory Usage
$\mathscr{M}_{AAL}$	$(app_1, pm_2, aalm\_mysql\_conn)$	MySQL Num. of Connections
$\mathscr{M}_{AAL}$	$(app_1, pm_2, aalm\_wb\_pages)$	Apache Pages per Second
$\mathscr{M}_{ABLL}$	$(app_1, pm_2, abllm\_wb\_perrors)$	WordPress Pages Errors
$\mathscr{M}_{ABLL}$	$(app_1, pm_2, abllm\_response)$	WordPress Response Time

be used for the WordPress application in a cloud environment. WordPress is a CMS (Content Management System) for personal blogs and Web sites. It allows a simple creation of posts and pages together with a simple comment management. The WordPress system architecture is composed of the Apache2 Web server, the MySQL DBMS, the Memchached tool, and the code implementing the functionalities of the CMS itself. Different cloud deployment configurations can be used for this application. We assume the simplest deployment alternative where all components of the WordPress application are deployed inside the same VM. Using the model proposed in [9], we first identify the possible changes that can be suffered by the entities in the cloud environment hosting this specific deployment of the WordPress Application (as illustrated in Fig 5): (1) increase virtual resources; (2) request a new VM and move some components of the application to the new VM; (3) change the Apache Web server configuration; (4) change the MySQL DBMS configuration.

The above listed management actions can be activated independently of each other and eventually simultaneously if no mechanism is used to coordinate their activation. In this WordPress example, suppose we want to identify the management actions to be taken before appealing to auto-scaling, which typically implies some kind of cost either in monetary or availability terms. Then, we define a *CoI* for this application as illustrated in Table I. The *CoI* in this table is the physical machine CoI_{PM} for the machine depicted in Fig. 5. The goal of this example is to identify which management actions can be activated in respect to application app_1 in different entities of the cloud layers related to this application when these metrics are observed together. Then, the next step is to define rules upon the metrics of a *CoI* that determine when each of the identified management actions can be applied, as illustrated in Fig. 4. The joint observation of the metrics in the *CoI* leads to the identification of different management actions. For now, the analysis of the actions is a simple hierarchy of management actions denoted by `if-then-else`. However, these actions could be ranked according to the cost of their enforcement, and this is planned to be developed as future work.

The *CoI* and the management actions presented in this paper still require the human intervention for their configuration. However we intent to create automatic configuration tools based on information models [9]; and investigate the use of correlation techniques to automatically devise *CoI*.

```
if ((abllm_response is high)
    AND (aalm_mysql_conn is high)
    AND (abllm_wb_perrors is high)
    AND (aalm_wb_pages is low)
    AND (vlm_cpu is high)
    AND (vlm_mem is high))
    then
      Apply management action (4)
...
if ((aalm_mysql_conn is high)
    AND (abllm_wb_perrors is low)
    AND (vlm_cpu is high)
    AND (vlm_mem is high))
    then
        Apply management action (1)
```

Fig. 4. Using *CoI* for specification of management action rules.

V. EXPERIMENTS

In this section, we present the experimental results on analyzing cloud monitoring information using the 3-D monitoring model and the *CoI* concept introduced in this paper. The aim is to have a simple testbed where we show the data complexity and the necessity to correctly interpret information monitored from all the cloud layers. Our goal is to demonstrate that by using *CoIs* it is possible to identify and choose more effective management actions. To achieve this goal, we make use of the Wordpress application case study described in the previous section. We introduce three experiments for the same case study. In the first no management action is activated. The second one considers the activation of auto-scaling policies. The third one uses the *CoI* depicted in Table I and the management rules in Fig. 4. The experiments show how these actions behave for the given scenario and demonstrate the effectiveness of our proposed approach. In Section V-A, we describe a real testbed and how it emulates the deployment of the WordPress application in a cloud environment. Section V-B presents the details about the synthetic load generated to stress the WordPress application. Section V-C discusses the numerical results observed during the experiments.

A. Testbed arrangement

Fig. 5 shows the testbed arrangement. We exploited three PMs. Two of them (PM1 and PM3) are IBM LS21 blade servers, while the third one (PM2) is an IBM X3630 M3 storage servers. The corresponding hardware configurations are shown in Fig. 5. On all the PMs, we installed the Scientific Linux 6.5 distribution. On PM1, we installed an OpenStack controller node with Keystone, Glance, Nova (but without Nova-Compute), and Horizon services. On PM2, the Nova-Compute service was installed in order to be able to instantiate VMs on top of it. We deployed a single VM (VM1) on PM2 where we host all the components of the WordPress application. In VM1, we installed the Apache2 Web server, the MySQL DBMS, the Memchached tool, and WordPress. In commercial clouds, a database service is usually purchased with a limited number of supported concurrent connections. For example, Amazon Relational Database Service (Amazon

RDS) provides predefined database instances with different maximum number of concurrent connections, e.g., t1.micro (with 34 connections at most), or m1.small (with 150 connections at most). Therefore, we initially configured the MySQL DBMS to support a maximum of 50 concurrent connections. On PM3, we installed the Apache Jmeter tool that emulates clients of the WordPress application as discussed in the next section. Software configurations and versions are also depicted in Fig. 5. Finally, we installed a set of monitoring tools. We exploited: Nagios for retrieving data from the Physical layer (CPU and memory). Havana OpenStack Ceilometer to collect information from the virtualization layer (vitual CPU and memory consumption). AWStats and MyCheckPoint extract data from the Application Architecture layer, respectively, collecting the number of pages per second provided by the Apache Web server and the number of concurrent connections at the MySQL DBMS. Finally, at the Application Business Logic layer, Apache Jmeter analysis plug-ins collects the percentage of errors experienced by users while accessing the WordPress Web site and the response time of each user action.

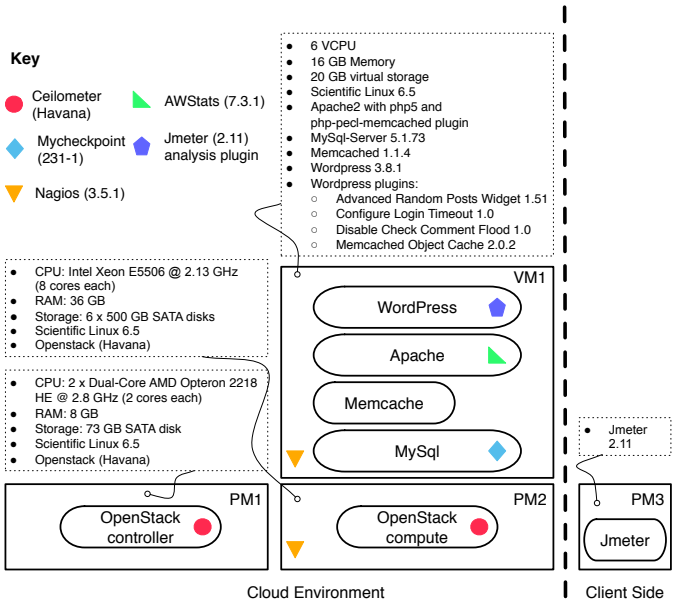


Fig. 5. Testbed arrangement with PMs, VMs, hardware/software configuration, and monitoring tools.

B. Synthetic traffic generation

During the experimentation, the load on the WordPress Web site is synthetically generated using the Apache Jmeter tool. It allows to record the actions executed by users when interacting with the Web site. These actions are saved in terms of the HTTP requests the users performed and then Jmeter reproduces such requests according to a certain traffic pattern. To reproduce such a pattern, we generated the synthetic traffic according to Table II. We consider five categories of users, who perform a different set of operations in a specific order. The users continuously perform their set of actions waiting for a random time between one set and the next one. We defined three different scenarios (*low*, *medium*, and *high*) in order to stress the system. For each scenario, we change the number of users in each category that are concurrently performing a set of operations in the WordPress Web site.

TABLE II
CATEGORIES OF USERS AND POPULATION CHARACTERIZATION FOR EACH OF THE THREE SCENARIOS DESCRIBED IN EACH COLUMN.

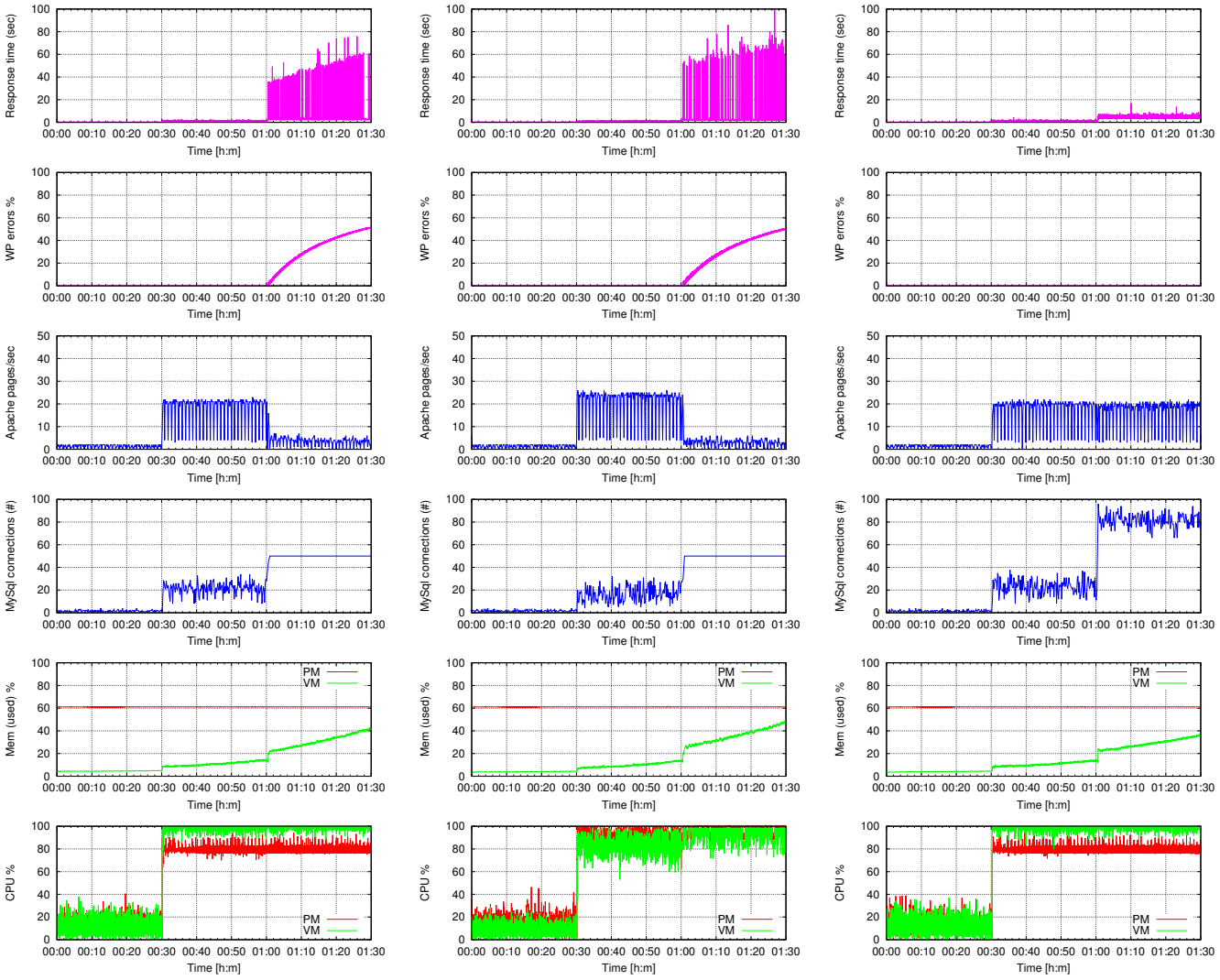
User Type and Description	*Low Load*	*Medium Load*	*High Load*
Admin creating new post	1	1	1
Guest reading latest post	1	15	30
Guest reading random post	1	15	30
Guest reading latest post and leaving a comment	1	15	30
Guest reading random post and leaving a comment	1	15	30

C. Experimentation

We conducted the experimentation by running each scenario for 30 minutes on the testbed (Fig. 5) and collected from the monitoring tools all the metrics specified in the *CoI* for the WordPress application, as illustrated in Table I. In this experiment we are interested in the Physical Machine Monitoring Slice for PM2. There is one main reason for this choice: all the components for the WordPress application and their respective monitoring information are deployed only on this machine. In a more complex scenario, where the application is using multiple VM, we could focus, for instance, on the Application Monitoring Slice for the WordPress application. This would give us all the monitoring metrics associated with this application throughout all the machines where it is hosted. Figs. 6(a), 6(b), and 6(c) show the monitored information observed for the WordPress application *CoI*. In all these figures, the first 30 minutes of experimentation from 00:00 to 00:30 represent the low load scenario; the interval from 00:30 to 01:00 depicts the medium load scenario; and the high load scenario corresponds to the interval from 01:00 to 01:30.

Fig. 6(a) corresponds to the initial deployment of the WordPress application: 6 CPUs are assigned to the VM1, and the maximum number of concurrent connections in the MySQL DBMS is 50. In this case, there is no *CoI* and management rules configured. In Fig. 6(a), we observe that during the *low* and *medium* load scenarios the users experience a good quality of service (with response time below 500 ms) and no errors are present. However, during the *high* load scenario, the system is overloaded and both the number of errors and the response time increase considerably. Without using the models proposed in this paper, the administrator of the WordPress Web site would have to look in different monitoring graphics (not necessarily clustered together) and manually relate the information to determine the best management action.

A natural reaction to the situation illustrated Fig. 6(a) would be the activation of traditional auto-scaling policies. This means that typically the metrics from Physical and Virtualization layers would be observed in order to activate the



(a) Without Management Actions (b) Applying Scaling Policies (c) Applying *CoI* Support

Fig. 6. Monitored metrics in the case of (a) 6 virtual CPUs and a maximum of 50 MySQL connections, (b) 8 virtual CPUs and a maximum of 50 MySQL connections, and (c) 6 virtual CPUs and a maximum of 100 MySQL connections.

scaling of resources. Using these metrics, one could conclude that a good management action to solve the overload situation would be to increase the number of virtual CPUs associated with VM1. This conclusion could be reached because of three factors: First, the memory occupation for both PM2 and VM1 is always below 60%, and this indicates that memory is not the problem in this situation. Second, the VM1 reaches 100% of CPU usage during the *medium* load scenario and remains overloaded also during the *high* load scenario. Third, the PM2 CPU load stays below 80% during both the *medium* and *high* load scenarios. This suggests the possibility of applying a management action related to increasing the number of virtual CPUs associated to VM1 to solve the overload situation. Of course, adding more virtual CPUs to VM1 could affect the contention on top of PM1 if multiple VMs were deployed on top of it influencing the QoS of the co-hosted VMs.

To demonstrate the effect of auto-scaling policies, we conducted a second phase of experimentation for the same WordPress application, increasing the number of virtual CPUs associated with VM1 to 8 without changing any other parameter. Fig. 6(b) shows the monitoring information collected in this case. We observe that the VM1 CPU load with the new configuration is no more at 100%. However, there are no significant improvements in terms of quality of service (i.e., response time) and the number of errors experienced by the users. As depicted in Fig. 6(b), these numbers remain high, despite the fact that in the *medium* load scenario the Apache Web Server is able to deal with more pages per second. Thus, this experiment demonstrates that the management action consisting of increasing the number of virtual CPUs associated with VM1 does not solve the situation.

However, when we apply our approach, as illustrated in Fig. 6(c), management actions are taken considering information also at the Application Architecture and Application

Business Logic layers. In Fig. 6(c), we present the monitoring information for the WordPress application when deployed using 6 CPUs and this time a number of 100 maximal concurrent connections to the MySQL DBMS, as an effect of the management action activated according to Fig 4. The components of applications and the cloud infrastructure related to each other and the *CoI* as well as the management rules help express these relationships. In the case of the WordPress application illustrated in Fig. 6(a), the actual problem is related to the maximum number of concurrent connections at the MySQL DBMS. In the *high* load scenario this number is reached and the MySQL DBMS starts to reject connections, and this generates the Web Page errors. This also limits the number of pages per second that the Apache Web server is able to provide to the users. In Fig. 6(c), we show the results when the situation is analzyed according to the *CoI* and the management rules discussed Section IV-B. As it can be observed, the quality of service experienced by the users has been greatly improved and the errors have been eliminated.

The knowledge of information coming from different layers allows for a different management action to be considered as a possible solution to the overload situation. The management action taken in Fig. 6(c) was to keep the same amount of virtual resources but increase from 50 to 100 the maximal number of concurrent connections. Such a management action has the additional advantage of not influencing other applications in other VMs on top of the same PM, contrary to the case where the number of virtual CPUs is increased. Therefore, the analysis of the monitoring information using *CoI* and management rules shows that it is possible to achieve more effective management actions. These experiments demonstrate the practical necessity to relate and model information at all layers of the cloud environment in order to act and solve the overloaded situation in a more informed way.

VI. Conclusion

Usually, several management mechanisms are available in cloud environments, allowing dynamic adaptations to be performed at different cloud layers. However, the complexity of the scenario may cause such multiple adaptation actions to interfere with each other, canceling their benefits and negatively affecting both the infrastructure and applications QoS. In this paper, we deal with this challenge by proposing a new 3-D monitoring model and by formalizing the innovative concept of context of interest (*CoI*). Such tools allow to analyze the huge quantity of cloud monitoring information that is usually available in a cloud environment and to identify situations in which multiple adaptations are possible, coordinating them and selecting the most appropriate. A set of experiments performed on a real OpenStack cloud testbed and taking into consideration a WordPress application have been performed. The obtained results demonstrate that when information at the higher layers of the cloud environment is taken into consideration adaptation actions can be chosen in a more reasoned way, thus minimizing their interferences and optimizing cloud resources utilization and the overall cloud and applications QoS. Future work will deal with: (i) the analysis and experimentation of the framework supporting the proposed model, (ii) automation of the deployment of the monitoring tools based on the context of interests (*CoIs*), (iii) formalization of the adaptation actions and their relationship with the 3-D monitoring model and with the concept of *CoI*, (iv) ranking of management actions according to QoS and cost principles, (v) implementation of more complex scenarios with multiple VMs, PMs, and applications in order to demonstrate the advantages of our approach in a more realistic context.

Acknowledgements

Thanks to Christina Bellinghoven, Nicola Peditto, Carmelo Romeo, and Fabio Verboso for the support on the experimentation. This research received funding from the European Community's Seventh FrameworkProgramme (FP7/2007-2013) under grant agreement Nr. 610802 (CloudWave).

References

[1] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *Communications Magazine, IEEE*, vol. 50, no. 9, pp. 34–40, 2012.

[2] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *Network and Service Management (CNSM), 2010 International Conference on*, 2010, pp. 9–16.

[3] S. Dutta, S. Gera, A. Verma, and B. Viswanathan, "Smartscale: Automatic application scaling in enterprise clouds," in *IEEE 5th Int. Conf. on Cloud Computing (CLOUD), 2012*, June 2012, pp. 221–228.

[4] S. He, L. Guo, Y. Guo, C. Wu, M. Ghanem, and R. Han, "Elastic application container: A lightweight approach for cloud resource provisioning," in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, March 2012, pp. 15–22.

[5] G. Foster, G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "The right tool for the job: Switching data centre management strategies at runtime," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 2013, pp. 151–159.

[6] M. Miglierina, G. Gibilisco, D. Ardagna, and E. Di Nitto, "Model based control for multi-cloud applications," in *5th Int. Workshop on Modeling in Software Engineering (MiSE), 2013 in conjunction with ICSE 2013*, 2013, pp. 37–43.

[7] S. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 1887–1895.

[8] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[9] C. C. Marquezan, F. Wessling, A. Metzger, K. Pohl, C. Woods, and K. Wallbom, "Towards exploiting the full adaptation potential of cloud applications," in *In Proceedings of 6th International Workshop on Principles of Engineering Service-Oriented and Cloud System (PESOS 2014) in conjunction with IEEE and ACM ICSE 2014*, 2014, p. 10.

[10] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, and G. Antoniu, "Gmone: A complete approach to cloud monitoring," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2026 – 2040, 2013.

[11] M. Carvalho, R. Esteves, G. Rodrigues, C. C. Marquezan, L. Z. Granville, and L. M. R. Tarouco, "Efficient configuration of monitoring slices for cloud platform administrators," in *In Proceedings of 19th IEEE Symposium on Computers and Communications (ISCC 2014)*, 2014, p. 7.

[12] F. Wuhib, R. Yanggratoke, and R. Stadler, "Allocating compute and network resources under management objectives in large-scale clouds," *Journal of Network and Systems Management*, pp. 1–26, 2013.

[13] R. Esteves, L. Z. Granville, M. F. Zhani, and R. Boutaba, "Evaluating allocation paradigms for multi-objective adaptive provisioning in virtualized networks," in *In Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS 2014)*, 2014, p. 9.

[14] P. Leitner, C. Inzinger, W. Hummer, B. Satzger, and S. Dustdar, "Application-level performance monitoring of cloud services based on the complex event processing paradigm," in *Service-Oriented Computing and Applications (SOCA), 2012 5th IEEE Int. Conf. on*, 2012, pp. 1–8.