The Tale of Sansa Spark

Ivan Ermilov[3], Jens Lehmann[1,2], Gezim Sejdiu[1], Lorenz Bühmann[3], Patrick Westphal[3], Claus Stadler[3], Simon Bin[3], Nilesh Chakraborty[1], Henning Petzka[2], Muhammad Saleem[3,4], Axel-Cyrille Ngonga Ngomo[3,4], and Hajira Jabeen[1]

[1] University of Bonn, Germany
{jens.lehmann, sejdiu, chakrabo, jabeen}@cs.uni-bonn.de
[2] Fraunhofer IAIS, Bonn, Germany
{jens.lehmann, henning.petzka}@iais.fraunhofer.de
[3] Institute for Applied Informatics (InfAI), University of Leipzig, Germany
{buehmann,patrick.westphal,cstadler,iermilov,sbin,saleem}@informatik.uni-leipzig.de
[4] Paderborn University, Data Science Group, Paderborn, Germany
axel.ngonga@uni-paderborn.de

Abstract. We demonstrate the open-source Semantic Analytics Stack (SANSA), which can perform scalable analysis of large-scale knowledge graphs to facilitate applications such as link prediction, knowledge base completion and reasoning. The motivation behind this work lies in the lack of scalable methods for analytics which exploit expressive structures underlying semantically structured knowledge bases. The demonstration is based on the BigDataEurope technical platform, which utilizes Docker technology. We present various examples of using SANSA in the form of interactive Spark notebooks, which are executed with Apache Zeppelin. The technical platform and the notebooks are available on SANSA Github and can be deployed on any Docker-enabled host, locally or in a Docker Swarm cluster.

1 Introduction

SANSA[5] is an open-source[6] *structured data processing engine* for performing distributed computation over large-scale RDF datasets [1]. It provides data distribution, scalability and fault tolerance for (1) manipulating large RDF datasets, and (2) applying analytics on the data at scale by making use of cluster-based big data processing engines. In this demonstration paper, we describe a web-based prototype for interacting with SANSA via a web interface.[7] SANSA comes with: (i) specialised serialisation mechanisms and partitioning schemata for RDF, using vertical partitioning strategies, (ii) a scalable query engine for large RDF datasets and different distributed representation formats for RDF, (iii) an adaptive reasoning engine which derives an efficient execution and evaluation plan from a given set of inference rules, (iv) several distributed structured machine learning algorithms that can be applied on large-scale RDF data, and (v) a framework with a unified API that aims to combine distributed in-memory computation technology with semantic technologies. To achieve the goal of storing and manipulating

[5] `http://sansa-stack.net/`
[6] `https://github.com/SANSA-Stack`
[7] Please note that any similarities of the paper title to popular TV series are purely coincidental.

large RDF datasets, SANSA leverages existing big data frameworks like Apache Spark and Apache Flink,[8] which have matured over the years and offer a reliable method for general-purpose processing of large-scale data.

In this demonstration, we will present and describe our implementation of interactive Spark Notebooks for SANSA.[9] These notebooks are a collaborative environment implemented as an interactive web editor. They allow access to the SANSA layers and hence provide data scientists, data engineers and students with means to easily use and execute the functionality of SANSA to explore, analyze and learn from large-scale RDF datasets.

2 The SANSA Stack

Research efforts in the areas of distributed analytics and semantic technologies have been mostly isolated until now. We aim to proceed one step further by using the semantic modelling standard as a basis for machine learning and data analytics. The layered architecture of SANSA is a direct consequence of this integrated vision and is depicted at the top of Figure 1. For a detailed description of each layer, we refer to [1].

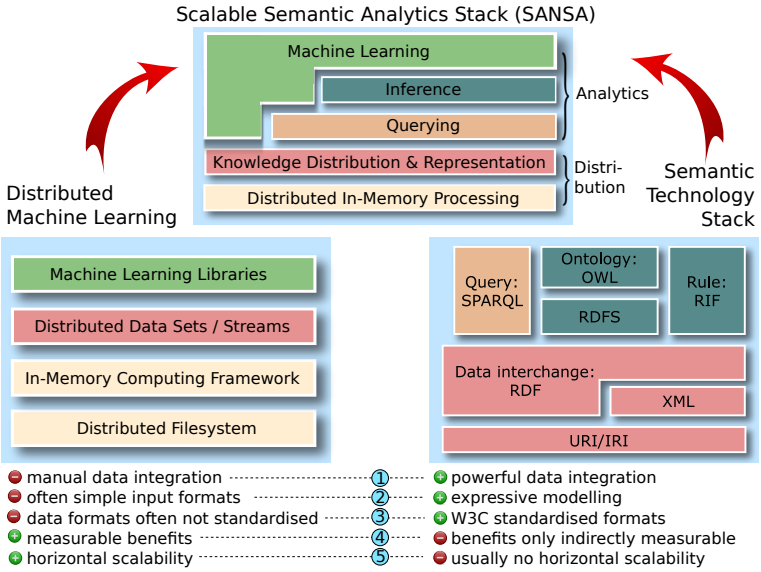

Fig. 1. The SANSA framework combines distributed analytics (left) and semantic technologies (right) into a scalable semantic analytics stack (top). The colours encode how the original stacks influence the SANSA stack. A main vision of SANSA is the belief that the the characteristics of each technology stack (bottom) can be combined and retain the respective advantages (figure taken from [1]).

[8] http://spark.apache.org and http://flink.apache.org

[9] https://github.com/SANSA-Stack/SANSA-Notebooks

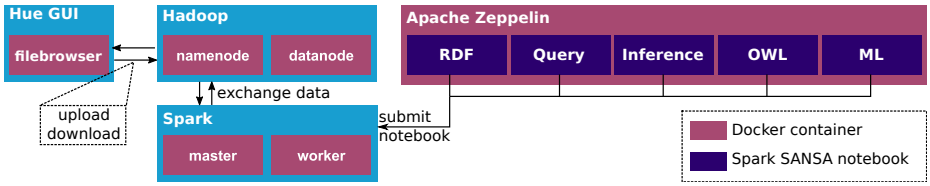

Fig. 2. SANSA-Notebooks architecture.

3 SANSA Notebooks

SANSA provides Notebooks for an easy local deployment for development and demonstration purposes. SANSA-Notebooks is an interactive toolkit on top of Hadoop-Spark-Workbench[10] with Apache Zeppelin,[11] which allows the copying of files from/to HDFS and an interactive Spark code execution via a web GUI. The architecture of SANSA-Notebooks is depicted in Figure 2. The authors utilize SANSA-Notebooks (see Figure 3) in Big Data labs and courses as they alleviate the complicated Hadoop/Spark setup and allow the students to focus on developing distributed algorithms on top of SANSA. Cluster deployment of the examples is also possible through Docker images (see SANSA-Examples Github repository[12]). Additionally, SANSA is readily available from the Maven Central Repository. Thus it is straightforward to include it in other projects using Maven or SBT – the most popular build managers for Scala – for both Spark- and Flink-based setups.

During the demonstration, we will present the example notebooks.[13] These examples give a quick overview of the SANSA APIs. SANSA is build on the concepts of distributed datasets (i.e RDD, DataFrame, DataSet). A dataset is inferred from the external data, then parallel operations e.g. *transformations* and *actions* are applied which trigger a job execution on a cluster. Depending on the network connection, the demonstration will be performed on a local single node cluster or a remote multi node cluster. In the following, we provide a concise description for the examples grouped by the SANSA layers.

1. ***RDF***.
   (a) Reading and writing triple files from HDFS or file system and some basic triple operations.
   (b) A distributed evaluation of numerous RDF Dataset Statistics dubbed RDF-Stats (see Figure 3), for example, property distribution, class distribution, distinct subjects/objects/entities as well as statistics summary.
   (c) Assigning weights to a given entity based on the Spark GraphX PageRank algorithm after triples have been transformed to a graph representation (i.e. PageRank for resources).

[10] `https://github.com/big-data-europe/docker-hadoop-spark-workbench`
[11] `https://zeppelin.apache.org/`
[12] `https://github.com/SANSA-Stack/SANSA-Examples`
[13] The source code for all of them is provided at `https://github.com/SANSA-Stack/SANSA-Examples`. We will present a tale / storyline using different examples across the SANSA layers for booth visitors and adapt them interactively (with new parameters, other datasets etc.) in the web browser.

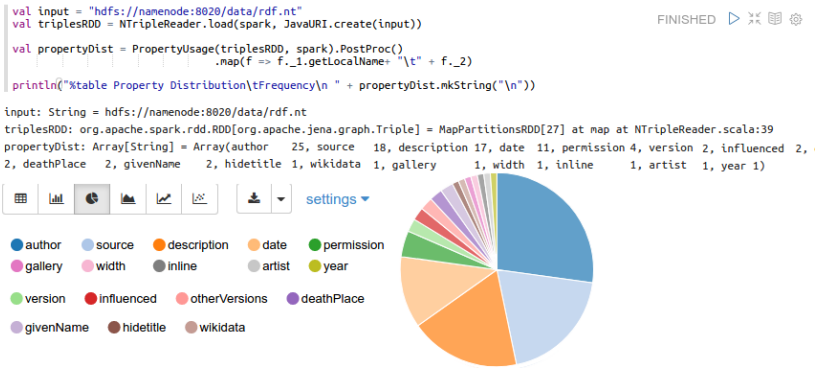

Fig. 3. RDF-Stats Spark application running in SANSA-Notebooks with statistics visualization.

2. ***Query***. The example applies Sparqlify,[14] which is a SPARQL-to-SQL rewriter, for data partitioning and schema extraction. The queries are executed using the SparkSQL engine.
3. ***RDF inference***. The examples apply a reasoning profile (RDFS Full, RDFS Simple, OWL Horst, Transitive) on a given input file with an optimised execution plan.
4. ***OWL***. The examples provided for the OWL layer demonstrate the process of loading an OWL file into Spark RDD, a Spark Dataset, or a Flink DataSet.
5. ***Machine Learning***.
   (a) Clustering algorithms. Three examples for different clustering algorithms are provided, namely power iteration clustering, BorderFlow and modularity clustering. They all take an RDF graph as input and return the list of triples for each of the different clusters.
   (b) Rule mining. This example applies association rule mining on a given RDF knowledge base. The output is the set of closed Horn rules that satisfy a support-confidence threshold.

One of the powerful features of the SANSA Notebooks is that you can view the result set of the previous session within the Spark framework and, in case you have found some insight for your data and would like to share, you can easily create a report and either print or send it.

References

1. J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. Ngonga Ngomo, and H. Jabeen. Distributed Semantic Analytics using the SANSA Stack. In *Proceedings of 16th International Semantic Web Conference - Resources Track (ISWC'2017)*, 2017.

[14] http://aksw.org/Projects/Sparqlify.html