Querying Large-scale RDF Datasets Using the SANSA Framework

Claus Stadler[1], Gezim Sejdiu[2], Damien Graux[3,4], and Jens Lehmann[2,3]

[1] Institute for Applied Informatics (InfAI), University of Leipzig, Germany
[2] Smart Data Analytics, University of Bonn, Germany
[3] Enterprise Information Systems, Fraunhofer IAIS, Germany
[4] ADAPT Centre, Trinity College of Dublin, Ireland
cstadler@informatik.uni-leipzig.de
{sejdiu,jens.lehmann}@cs.uni-bonn.de
{damien.graux|jens.lehmann}@iais.fraunhofer.de

Abstract. In this paper we present Sparklify: a scalable software component for efficient evaluation of SPARQL queries over distributed RDF datasets. In particular, we demonstrate a W3C SPARQL endpoint powered by our SANSA framework's RDF partitioning system and Apache Spark for querying the DBpedia knowledge base. This work is motivated by the lack of Big Data SPARQL systems that are capable of exposing large-scale heterogeneous RDF datasets via a Web SPARQL endpoint.

1 Introduction

SANSA [4] is an open-source framework which allows performing distributed computing over large-scale RDF datasets. At its core, it provides high-level APIs for reading, querying using SPARQL, performing inference as well as analytics over large-scale RDF datasets. In the recent years, RDF data has grown rapidly. And, querying the data is essential to browse, search and explore the structured information. SPARQL, the RDF query language, is very expressive which allows extracting complex relationships. It takes the description in the form of a query and returns the information in the form of a set of bindings or a derived RDF graph. Querying such large amount of data becomes challenging as the size of the data grows.

To overcome this, many systems have been proposed and developed ([5], [2]) using the Apache Spark[1] framework. Apache Spark is a generic-purpose cluster computing framework which allows running applications in a distributed manner. Its core abstraction data structure are Resilient Distributed Datasets (RDD) [7] which are immutable collections of records that Spark uses to distribute the workload across the cluster. Besides RDDs, Spark has a rich set of high-level APIs. SparkSQL [1] for SQL-like and structured data processing which allows querying structured data on Spark programs.

[1] http://spark.apache.org/

One of such systems, built on the concepts of RDDs and SparkSQL, is Sparklify [6] – a scalable software component for efficient evaluation of SPARQL queries over distributed RDF datasets, already integrated into the SANSA framework. It evaluates SPARQL queries while transforming them into a lower-level of Spark programs. Our Sparklify query processor interfaces with Apache Spark and can be used programmatically within Big Data workflow construction and in SPARQL server mode. In contrast, most of the existing systems only feature cli-oriented prototypes.

In this demonstration, we present and describe our implementation of the SANSA SPARQL endpoint. This interface allows access to the RDF data using the common SPARQL endpoint which provides an easy-to-use method to run SPARQL queries via the Web[2].

This is an accompanying poster paper for Sparklify [6], which was accepted at the ISWC resource track. The addition made in this demo is the SPARQL endpoint Web interface for Sparklify, which enables executing SPARQL queries on-the-fly using the Web interface i.e., without engineering efforts and command line interfaces.

2 Querying large-scale RDF data: DBpedia as a use case

DBpedia [3] is among the largest and most well-known sources of structured information on the Web. The public DBpedia SPARQL endpoint[3] at present offers access to 438.336.350 triples representing information in a variety of domains. In addition, with public downloads of auxiliary information the data size exceeds a billion triples. Due to its size and variety, DBpedia is often used as a test-bed for novel Linked Data technology in order to analyze strengths and weaknesses. For these reasons, we choose to run Sparklify over the DBpedia knowledge base. The whole pipeline is described below.

Figure 1 depicts the Sparklify architecture. The data (e.g. DBpedia knowledge graph) first has to be loaded on a distributed file system (*Step 1*). In our case, we use Hadoop Distributed File System (HDFS) for storing the RDF graph that SANSA can read efficiently. Afterwards, data ingestion is performed (*Step 2*). SANSA reads RDF data into an initial RDD of triples. Sparklify applies data partition in parallel over the initial RDD (*Step 3, part one*). This partitioning facilitates fast querying by giving both the SPARQL-to-SQL rewriter and the Spark processor additional opportunities for pruning during static query analysis. These partitioned data are then queried, based on the input SPARQL query (*Step 1*) using the Spar*q*lify system – a SPARQL to SQL rewriter. This leverage the potential of the optimizers of both the rewriter as well as those of the underlying frameworks for SQL. The raw output of a query execution is a pair comprised of the resulting RDD together with a mapping for construction of the corresponding SPARQL result set, i.e. a set of bindings. The mapping associates each of the requested SPARQL query's result variables with an expression over

[2] Demo. resources: `https://github.com/SANSA-Stack/SANSA-Examples`

[3] `https://dbpedia.org/sparql`

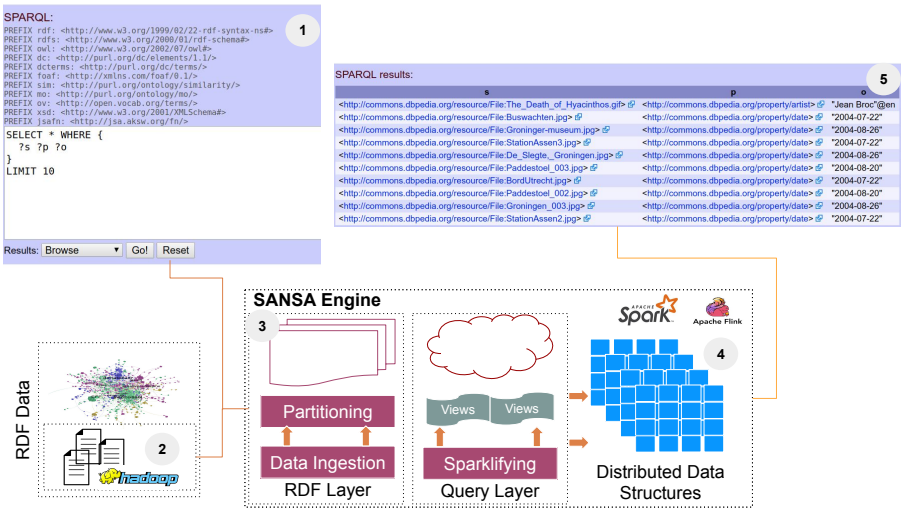

Fig. 1. Sparklify SPARQL endpoint.

the RDD schema, such that from each RDD row a corresponding binding can be computed. Hence, the output is again a distributed data structure which can further processed in a distributed fashion (*Step 4*) or visualized or as a result set of records (*Step 5*). Figure 2 demonstrates reuse of a third-party SPARQL-based Wikidata visualization app[4] for DBpedia data served with SANSA.

3 Conclusion

Processing and querying RDF data becomes challenging when the size of the data increases. To solve it, many systems have been proposed and try to solve it using the distributed computing framework. Most of the existing systems mostly are cli-oriented prototypes. This brings many confusion and a lot of engineering effort in order to run SPARQL queries over it. Most of the users, who are familiar with the semantic web technologies, are capable of writing SPARQL queries. But, when it requires cluster configuration, and running prototypes from the source; it is considered as a dead-end. Therefore, we wanted to bring Sparklify – a scalable software framework for efficient evaluation of SPARQL queries over distributed RDF datasets. It contains a user interface as a SPARQL endpoint for easy to write and query RDF data distributed across the Spark cluster.

Acknowledgment

This work was partly supported by the EU Horizon2020 projects BigDataOcean (GA no. 732310), Boost4.0 (GA no. 780732), SLIPO (GA no. 731581) and QROWD (GA no. 723088).

[4] `https://github.com/stevenliuyi/wikidata-visualization`

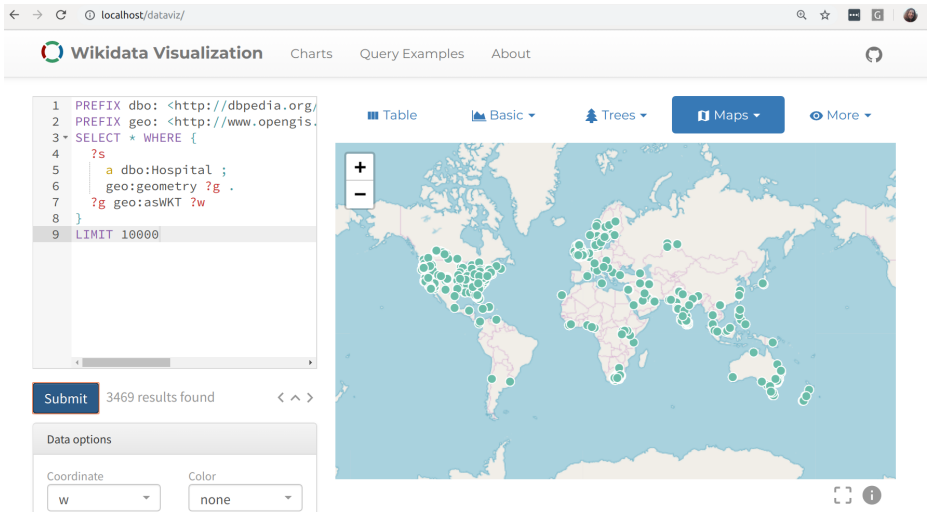

Fig. 2. Dataviz on SANSA's Spark-powered SPARQL endpoint.

References

1. Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., Zaharia, M.: Spark SQL: Relational Data Processing in Spark. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 1383–1394. SIGMOD '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2723372.2742797, `http://doi.acm.org/10.1145/2723372.2742797`
2. Graux, D., Jachiet, L., Genevès, P., Layaïda, N.: SPARQLGX: Efficient Distributed Evaluation of SPARQL with Apache Spark. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) The Semantic Web – ISWC 2016. pp. 80–87. Springer International Publishing, Cham (2016)
3. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal **6**(2), 167–195 (2015), `http://jens-lehmann.org/files/2014/swj_dbpedia.pdf`
4. Lehmann, J., Sejdiu, G., Bühmann, L., Westphal, P., Stadler, C., Ermilov, I., Bin, S., Chakraborty, N., Saleem, M., Ngonga Ngomo, A.C., Jabeen, H.: Distributed semantic analytics using the SANSA stack. In: ISWC Resources Track (2017)
5. Schätzle, A., Przyjaciel-Zablocki, M., Skilevic, S., Lausen, G.: S2RDF: RDF querying with SPARQL on Spark. Proc. VLDB Endow. **9**(10), 804–815 (Jun 2016)
6. Stadler, C., Sejdiu, G., Graux, D., Lehmann, J.: Sparklify: A Scalable Software Component for Efficient Evaluation of SPARQL Queries over Distributed RDF Datasets. In: Proceedings of 18th International Semantic Web Conference (2019)
7. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. pp. 2–2. USENIX Association (2012)