RDF Big Data Management on top of Spark

Olivier Curé[1], Hubert Naacke[2], Tendrie Randriamalala[2], Mohamed-Amine Baazizi[2], Bernd Amann[2]

[1] Université Paris-Est, LIGM, CNRS UMR 8049, France
olivier.cure@u-pem.fr
[2] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris,
CNRS, UMR 7606, LIP6, F-75005, Paris, France
{firstname.lastname}@u-pem.fr

1 Big Data et RDF

Modern information and knowledge-centric applications produce, store, integrate, query, analyze and visualize rapidly growing data sets. Traditional data processing technologies (data management and warehousing systems) are inadequate for processing this data and for responding to the new Big Data challenges (the Four+ V's). A first important challenge related to data volume and velocity concerns data and processing scalability which is achieved through generic massively distributed parallel data processing architectures like Hadoop [3], Spark [4] and Flink [5]. A second challenge related to data variability concerns the definition and implementation of flexible semi-structured / semantic data models and languages for facilitating complex semantic data representation, integration and access as a complement to traditional structured SQL database systems (noSQL). The Resource Description Framework (RDF) is one of the first data models for modeling and processing this kind of data. Some relevant application examples are the Semantic Web's Linked Open Data (LOD) cloud which contains over 60 billions RDF triples or the data originating from Schema.org annotations in Web documents.

2 RDF data storage and processing

RDF data takes the form of triples consisting of a (subject, predicate, object) tuple. Intuitively, a subject is related to an object via a given predicate. Such a set of triples forms a directed, labeled graph. This graph category can be considered as a superset of the property graph supported in NoSQL graph stores such as Neo4J[6] and Titan[7]. As a logical data model, RDF does not impose anything regarding a physical storage approach. Hence, many kinds of RDF stores [1] have emerged, either designed on top of a database management system, e.g., relational or NoSQL, or implemented from scratch by taking benefits of graphical modeling and processing aspect. Most of the existing RDF stores make an intensive use of indexes for processing SPARQL queries, e.g., RDF-3X [4] defines 15 indexes of triple elements. While being efficient at generating optimized query plans, they also present some limitations. In the first hand, the cost of creating and maintaining indexes over data sets of billions of RDF triples is prohibitive. On the other hand, even in the cloud computing era, the storage space is not infinite and has a certain cost. This is particularly relevant in our setting since we aim to store most of the data in main-memory. In the HAQWA platform (Hash-based And Query Workload-Aware distributed RDF store), we claim that an **index-free RDF Store** (and graph store in general) is viable.

In such a context, the only data access method is data scanning. The four main contributions of HAQWA that make a scan-oriented query execution plans possible, are:

[3] http://hadoop.apache.org/
[4] http://spark.apache.org/
[5] http://flink.apache.org/
[6] http://neo4j.com/
[7] http://thinkaurelius.github.io/titan/

– less data scanning: we provide a compact data encoding approach that ensures query answering completeness in the face of common inferences. This encoding approach supports a light materialization approach that can be considered as a trade-off between materialization and query reformulation.
– fast data reading: the encoded data is mostly in main-memory and does not require any serialization or disk access.
– fast data processing: we propose fast matching operations which are particularly adapted to SPARQL processing and associated reasoning services.
– few scan operations: we propose an efficient scan operator that ensures that a single scan is sufficient for star-shaped queries. In chain queries, the resulting star sub-queries are efficiently joined using existing parallel hash join operators.

Our HAQWA prototype is implemented over the Apache Spark [5]. Currently considered as the state-of-the-art in clustering computing framework, it ensures high availability, fault-tolerance and scalability and provides high performance by making an intensive use of the main-memory.

3 Talk Outline

In this talk, we will present an **overview of HAQWA's architecture** which has been implemented and validated by experimentation conducted over large RDF datasets of several hundred millions and billions of triplets. In particular, we will provide details about our platform's main features:

– a novel RDF triples hash-based distribution and allocation, supporting workload aware replication, strategies [2].
– a compressed semantic-enabled dictionary encoding that proposes a trade-off between triple materialization and query reformulation [3].
– a parallel SPARQL query engine that outperforms existing Spark query processing based on DataFrame. This is achieved by optimizing star-shaped query graph patterns.

We will emphasize on **different real-world RDF data processing use cases and ongoing research projects**, e.g., capturing and querying over IoT sensor data, reasoning over RDF streams. Finally, we will highlight some **important challenges** such as reasoning over distributed unbounded data, query aware replication for RDF federation, interactions between query processing, reasoning and visualization, e.g., inferring over navigation traces.

References

1. O. Curé and B. Guillaume, editors. *RDF Database Systems: Triples Storage and SPARQL Query Processing.* Morgan Kaufmann, Boston, MA, USA, 1st edition, 2015.
2. O. Curé, H. Naacke, M. A. Baazizi, and B. Amann. On the evaluation of RDF distribution algorithms implemented over apache spark. In *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015.*, pages 16–31, 2015.
3. O. Curé, H. Naacke, T. Randriamalala, and B. Amann. Litemat: a scalable, cost-efficient inference encoding scheme for large RDF graphs. *CoRR*, abs/1510.03409, 2015.
4. T. Neumann and G. Weikum. The rdf-3x engine for scalable management of rdf data. *VLDB J.*, 19(1):91–113, 2010.
5. M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, 2010.