PigSPARQL: A SPARQL Query Processing Baseline for Big Data

Alexander Schätzle, Martin Przyjaciel-Zablocki,
Thomas Hornung, and Georg Lausen

Department of Computer Science, University of Freiburg
Georges-Köhler-Allee 051, 79110 Freiburg, Germany
schaetzle|zablocki|hornungt|lausen@informatik.uni-freiburg.de

Abstract. In this paper we discuss PigSPARQL, a competitive yet easy to use SPARQL query processing system on MapReduce that allows ad-hoc SPARQL query processing on large RDF graphs out of the box. Instead of a direct mapping, PigSPARQL uses the query language of Pig, a data analysis platform on top of Hadoop MapReduce, as an intermediate layer between SPARQL and MapReduce. This additional level of abstraction makes our approach independent of the actual Hadoop version and thus ensures the compatibility to future changes of the Hadoop framework as they will be covered by the underlying Pig layer. We revisit PigSPARQL and demonstrate the performance improvement when simply switching the underlying version of Pig from 0.5.0 to 0.11.0 without any changes to PigSPARQL itself. Because of this sustainability, PigSPARQL is an attractive long-term baseline for comparing various MapReduce based SPARQL implementations which is also underpinned by its competitiveness with existing systems, e.g. HadoopRDF.

1 Introduction

Today, *MapReduce* has been widely adopted in manifold application fields, especially in the broad area of Big Data, with *Hadoop* being the most prominent open source implementation. Though node efficiency is known to be rather poor, its success is mainly attributed to the inherent high degree of parallelism, robustness, reliability and excellent scalability properties while running on cheap and heterogeneous commodity hardware. Furthermore, new nodes can be added to the system on demand seamlessly at runtime.

Driven by the *Semantic Web* and *Linked Open Data*, new challenges with regard to SPARQL query evaluation arise and scalability becomes an issue as RDF datasets continuously grow in size, exceeding the capabilities of state of the art non-distributed RDF triple stores [2]. The wide spread adoption of MapReduce makes it an interesting candidate for distributed SPARQL processing on large RDF graphs, especially for rather costly queries involving several joins that cannot be executed in real-time at web-scale and hence need to be processed offline. However, existing approaches in this direction are often accompanied by proof-of-concept implementations that are hard to deploy or not compatible

with newer versions of Hadoop, do not run out of the box or they are even not available for download at all. Moreover, they often support only a small subset of SPARQL, usually basic graph patterns. All this hampers the comparison of different approaches as evaluation results are hard to reproduce and a comprehensive evaluation becomes very cumbersome and time consuming.

In this paper we first revisit *PigSPARQL*[1], a mapping from SPARQL to the query language of *Pig* [4], that was originally presented in [6]. PigSPARQL is easy to use without complicated deployment, installation or configuration. By using Pig Latin as an intermediate layer of abstraction between SPARQL and MapReduce, the mapping is automatically compatible to future versions of Hadoop (including major changes like the new YARN framework) while it benefits from further developments and optimizations of Pig without having to change a single line of code since the query language of Pig is kept backward compatible. This is confirmed by experiments that are presented in short in this paper (cf. Section 3). Switching the version of Pig from 0.5.0 to 0.11.0 improved the query execution times by up to one order of magnitude, while no adaptations of PigSPARQL were required. Because of this feature of sustainability, PigSPARQL is an attractive long-term baseline for comparing various MapReduce based SPARQL implementations. This is also underpinned by PigSPARQL's competitiveness with existing systems like HadoopRDF and others (cf. Section 3).

2 PigSPARQL Architecture

Pig is a data analysis platform on top of Hadoop with a fully nested data model, complemented by a comprehensive imperative query language (Pig Latin) that gives us a simple level of abstraction from the procedural model of MapReduce by providing relational style operators like filters and joins which are not available in MapReduce out of the box. We represent an RDF triple in the data model of Pig as a tuple of three atomic fields with schema (`s`, `p`, `o`). As we do not require any preprocessing and RDF triples are converted into the data model of Pig on the fly, PigSPARQL is particularly suited for ad-hoc query processing, e.g. for ETL like scenarios where we do not want to build up a costly index structure in advance. In a typical SPARQL query the predicate of a triple pattern is usually bounded. Hence, PigSPARQL also supports optional vertical partitioning of the dataset as an additional preprocessing step.

Our mapping of SPARQL to Pig Latin follows a common design principle based on an algebraic representation of SPARQL expressions (cf. Figure 1). First, a SPARQL query is parsed to generate an abstract syntax tree which is then translated into a SPARQL algebra tree. Next, we apply several optimizations on the algebra level like the early execution of filters and a rearrangement of triple patterns by selectivity. Finally, we traverse the optimized algebra tree bottom up and generate for every SPARQL algebra operator an equivalent sequence of Pig Latin expressions. At runtime, Pig automatically maps the resulting Pig Latin script into a sequence of MapReduce iterations. More details are given in [6].

[1] See `http://dbis.informatik.uni-freiburg.de/PigSPARQL` for download.

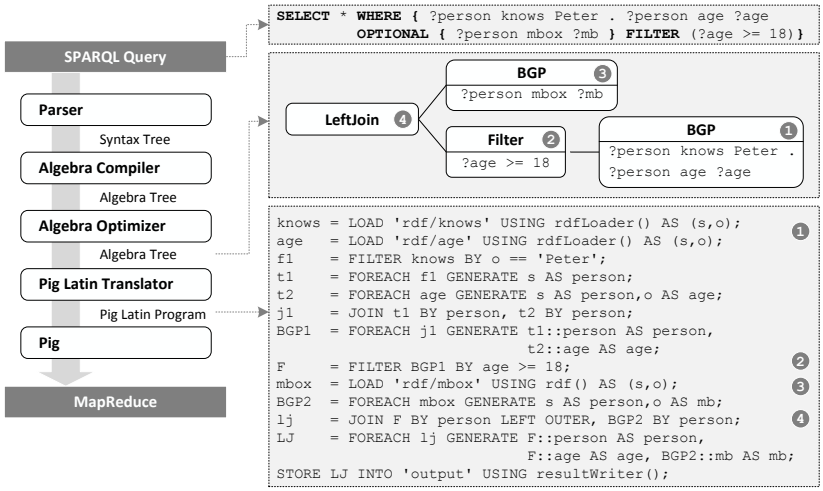

Fig. 1. PigSPARQL workflow from SPARQL to MapReduce

3 Experiments

Most of the published MapReduce based approaches are proof-of-concept implementations, which are neither well documented nor running out of the box, nor are available for public. Evaluation of such systems is time consuming and easily leads to inexplicable results. This hampers the comparability of different proposed solutions which is a key driver for further development. To introduce a stable basis for comparison, we suggest PigSPARQL as an easy to use baseline for SPARQL query processing with MapReduce because of the following reasons:

1. PigSPARQL is a reliable and stable system as it uses Pig as an intermediate layer which is widely-used and maintained by Yahoo! Research. Pig's processing framework is fairly competitive and continuously optimized and enhanced with new features. This is confirmed by Figure 2.a that shows exemplary the runtime improvement of PigSPARQL for SP^{2}Bench Query 2 between Pig 0.5.0 and Pig 0.11.0 where we can observe a speed up by an order of magnitude without changing a single line of code - other queries exhibit a similar behavior as can be expected because of PigSPARQL's architecture.
2. For a comprehensive evaluation of different systems, they should be installable and usable within a reasonable effort, without the need of tricky configurations. In the context of such evaluations, PigSPARQL is very attractive. The LUBM evaluation of HadoopRDF [3], for example, took us several weeks including an exhaustive troubleshooting whereas the same evaluation with PigSPARQL was done in only one day.
3. We evaluated the competitiveness of PigSPARQL with respect to three other SPARQL engines based on MapReduce by using LUBM, as some of these systems only support basic graph patterns: (1) *HadoopRDF* [3] is an advanced

SPARQL engine that utilizes a cost-based execution plan for reduce-side joins. (2) *MAPSIN* [5] is a map-side index nested loop join based on HBase. (3) *Merge Join* [1] is a MapReduce adoption of merge joins for SPARQL basic graph patterns. Figure 2.b illustrates the execution times for LUBM Query 4 distinguishing between n-way and 2-way join execution, if supported. PigSPARQL shows a competitive runtime performance while scaling smoothly when increasing the size of the dataset. MAPSIN performs a bit faster, however it uses a sophisticated storage schema based on HBase that works well for selective queries but decreases significantly in performance for less selective ones [5]. All approaches need a rather time consuming initial preprocessing of up to several hours compared to the vertical partitioning of PigSPARQL, which took less than 14 minutes for 1.6 billion triples.

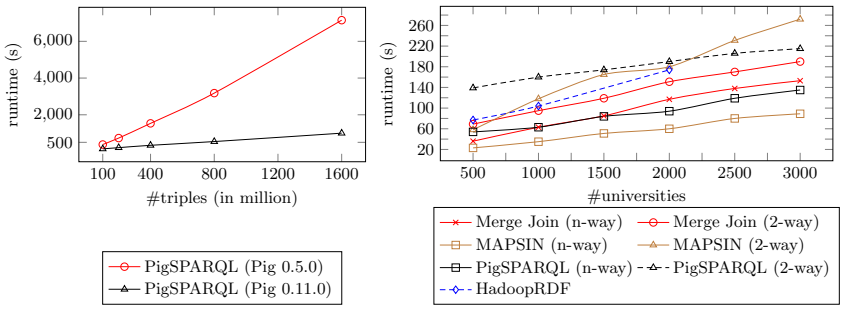

Fig. 2. (a) SP^{2}Bench Query 2. (b) LUBM Query 4.

Conclusion. PigSPARQL is an easy to use and competitive baseline for the comparison of MapReduce based SPARQL processing. With the support of SPARQL 1.0, it already exceeds the functionalities of most existing research prototypes. For future work, we plan to add support for additional SPARQL 1.1 features.

References

1. (2013), `http://dbis.informatik.uni-freiburg.de/forschung/projekte/DiPoS/`
2. Huang, J., Abadi, D.J., Ren, K.: Scalable SPARQL Querying of Large RDF Graphs. PVLDB 4(11), 1123–1134 (2011)
3. Husain, M.F., et al.: Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing. IEEE TKDE 23(9) (2011)
4. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A Not-So-Foreign Language for Data Processing. In: SIGMOD. pp. 1099–1110 (2008)
5. Schätzle, A., Przyjaciel-Zablocki, M., et al.: Cascading Map-Side Joins over HBase for Scalable Join Processing. In: SSWS+HPCSW. p. 59 (2012)
6. Schätzle, A., Przyjaciel-Zablocki, M., Lausen, G.: PigSPARQL: Mapping SPARQL to Pig Latin. In: Proc. SWIM. pp. 4:1–4:8 (2011)