SPARQLGX in Action: Efficient Distributed Evaluation of SPARQL with Apache Spark

Damien Graux, Louis Jachiet, Pierre Genevès, and Nabil Layaïda

Inria, Cnrs, lig and Univ. Grenoble Alpes, France
{damien.graux,louis.jachiet,nabil.layaida}@inria.fr
pierre.geneves@cnrs.fr

Abstract. We demonstrate sparqlgx: our implementation of a distributed sparql evaluator. We show that sparqlgx makes it possible to evaluate sparql queries on billions of triples distributed across multiple nodes, while providing attractive performance figures.

1 Introduction

We demonstrate the sparqlgx system introduced in [5] which is designed to evaluate sparql queries efficiently in a distributed manner on top of the Apache Spark framework[1]. sparql [1] is the standard query language for retrieving and manipulating data represented in rdf [7]. The core of the sparql query language is the Basic Graph Pattern fragment (bgp) composed of conjunctions of triple patterns (tps) which express conditions on rdf triples. sparqlgx supports the bgp fragment of sparql extended with union and optional operators at top level and with solution modifiers. sparqlgx implements specific optimizations aimed at optimizing the evaluation of bgps. For example, the following query taken from the WatDiv benchmark [3] (c3) involving one bgp composed of six triple patterns returns all the matching subjects from the dataset.

```
SELECT ?v0 WHERE {
  ?v0 wsdbm:likes ?v1 .       ?v0 wsdbm:friendOf ?v2 .
  ?v0 terms:Location ?v3 .    ?v0 foaf:age ?v4 .
  ?v0 wsdbm:gender ?v5 .      ?v0 foaf:givenName ?v6 . }
```

2 SPARQLGX Architecture and Principles

Data Storage Model. In order to process rdf datasets, we adopt the vertical partitioning approach introduced by Abadi *et al.* in [2] which stores a triple $(s\ p\ o)$ in a file named p whose contents keeps only s and o entries. Converting rdf data into a vertically partitioned dataset is straightforward while (1) tending to minimize the memory footprint and the datasets size on disks and (2) reducing response time when queries have bounded predicates since searches are limited to the relevant files.

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

Dataset	Number of Triples	HDFS File Size (with 2 replications)
WatDiv1k	109 million	46.8 Go
Lubm1k	134 million	72.0 Go
Lubm10k	1.38 billion	747 Go

Table 1: General Information about Used Datasets.

Compilation of SPARQL BGP*s.* Conjunctions of triple patterns are translated in terms of primitives of the Apache Spark framework expressed in Scala code. Each BGP is first translated in terms of a list of filters, which are then joined based on common variables.

Optimized Joins With Statistics. To speed up the evaluation of queries, particular attention is paid to the ordering of joins in the translation process. We compute statistics on data (*i.e.* we count all the distinct subjects, predicates and objects to obtain a notion of triple pattern selectivity based on occurrence numbers). Then we rewrite queries in order to minimize the sizes of intermediate results. Triple patterns are joined in decreasing order of their selectivities (*i.e.* triple patterns that return the smallest number of results are joined first).

Direct Evaluation. In certain situations, queried data might be subject to updates; in others users might only need to evaluate a single query once (for data cleaning purposes for instance). In such cases, it is interesting to limit as much as possible both the preprocessing time and the query evaluation time. For this purpose, SPARQLGX provides a specific tool, called SDE, capable of directly evaluating SPARQL queries without preprocessing.

3 Demonstration Details

We report on our experimental comparisons of SPARQLGX against other open source HDFS-based distributed RDF systems such as PigSPARQL [9], RYA [8], CliqueSquare [4], S2RDF [10] and RDFHive [5]. We deploy them on the same 10-node cluster having the HDFS installed with default settings which imply a resiliency to the loss of two nodes. Furthermore, we consider several datasets (presented in Table 1) coming from two popular BGP benchmarks: LUBM [6] and WatDiv [3]. These benchmarks are respectively composed of 14 and 20 SPARQL queries.

We present in Figure 1 the response times obtained with WatDiv1k. This illustrates that, for this dataset: (1) SDE always outperforms other tested "direct evaluators" (*e.g.* PigSPARQL and RDFHive); (2) SPARQLGX is able to answer all the queries unlike RYA and CliqueSquare; (3) it shares with CliqueSquare the same order of magnitude for queries L[1–5]; (4) it outperforms its competitors on queries C1, C2 and C3 (shown in the introduction).

To further illustrate the performance of SPARQLGX, we provide an interactive GUI. Attendees can interact directly with our engines (*i.e.* SPARQLGX and SDE)

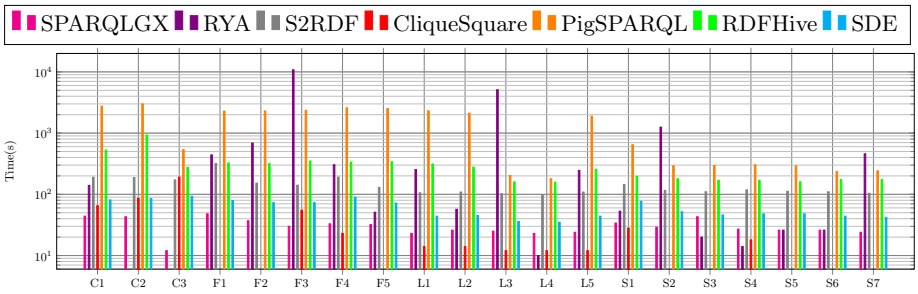
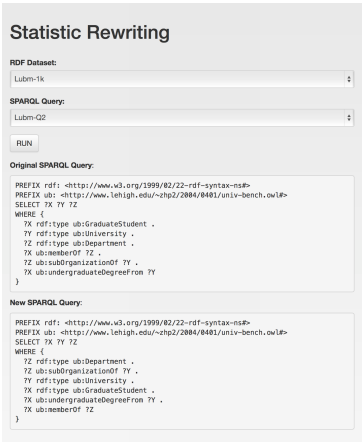
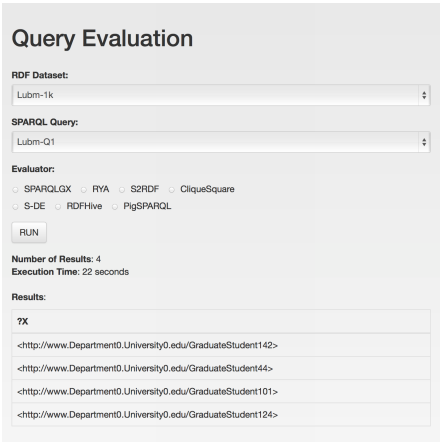


Fig. 1: Query Response Times with WatDiv1k.



(a) Statistic Module Screenshot.



(b) Query Evaluator Screenshot.

Fig. 2: Screenshots.

by evaluating several SPARQL queries on various RDF datasets: such as those presented in Table 1 and other smaller ones. Additionnally, we also deploy the other HDFS-based distributed systems (to have a common basis of comparison); thereby, participants can compare several systems running exactly on the same cluster. We demonstrate SPARQLGX and SDE with various interaction scenarios:

1. Loading datasets: participants can select a dataset among several predefined ones and a system to run its preprocessing phase. They can experience the preprocessing time of SPARQLGX, RYA, CliqueSquare and S2RDF. Since the loading and indexing cost can be high, we limit this feature to the smaller RDF datasets.
2. Query Evaluation (*e.g.* Figure 2b): attendees can evaluate predefined SPARQL queries which are extracted from LUBM and Watdiv. After choosing a dataset and a query, participants can select the evaluation system among SPARQLGX, SDE, RYA, CliqueSquare, S2RDF, RDFHive and PigSPARQL.

3. Query Rewriting (*e.g.* Figure 2a): participants can also experience the optimizer module which orders by selectivity the triple patterns of a SPARQL query according to a chosen RDF dataset. Attendees can see the generated Spark code.
4. System Comparison: finally, participants can observe how the different systems compare head-to-head.

4 Conclusion

SPARQLGX outperforms several related implementations in many cases, while implementing a simple architecture exclusively built on top of open source and publicly available technologies. The SPARQLGX implementation is available from:

`http://github.com/tyrex-team/sparqlgx`

References

1. SPARQL 1.1 overview (March 2013), http://www.w3.org/TR/sparql11-overview/
2. Abadi, Marcus, Madden, Hollenbach: Scalable semantic web data management using vertical partitioning. VLDB (2007)
3. Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified stress testing of RDF data management systems. In: ISWC. pp. 197–212. Springer (2014)
4. Goasdoué, F., Kaoudi, Z., Manolescu, I., Quiané-Ruiz, J.A., Zampetakis, S.: Cliquesquare: Flat plans for massively parallel RDF queries. In: ICDE. pp. 771–782. IEEE (2015)
5. Graux, D., Jachiet, L., Genevès, P., Layaïda, N.: SPARQLGX: A distributed RDF store mapping SPARQL to Spark. ISWC (2016)
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics (2005)
7. Hayes, P., McBride, B.: RDF semantics. W3C Rec. (2004)
8. Punnoose, R., Crainiceanu, A., Rapp, D.: Rya: a scalable RDF triple store for the clouds. In: Workshop on Cloud Intelligence. p. 4. ACM (2012)
9. Schätzle, A., Przyjaciel-Zablocki, M., Lausen, G.: PigSPARQL: Mapping SPARQL to pig latin. In: SemWeb Information Management. p. 4. ACM (2011)
10. Schätzle, A., Przyjaciel-Zablocki, M., Skilevic, S., Lausen, G.: S2RDF: RDF querying with SPARQL on spark. VLDB pp. 804–815 (2016)