SPARQLGX : Une Solution Distribuée pour RDF Traduisant SPARQL vers Spark

Damien Graux
Inria, France
damien.graux@inria.fr

Louis Jachiet
Inria, France
louis.jachiet@inria.fr

Pierre Genevès
Cnrs, France
pierre.geneves@cnrs.fr

Nabil Layaïda
Inria, France
nabil.layaida@inria.fr

ABSTRACT

SPARQL est un langage de requête standardisé par le W3C permettant d'interroger des données exprimées au format RDF (*Resource Description Framework*). Avec l'augmentation des volumes de données RDF disponibles, de nombreux efforts de recherche ont été faits pour permettre l'évaluation distribuée et efficace de requêtes SPARQL. Dans ce contexte, nous proposons et partageons SPARQLGX : notre solution de stockage RDF distribuée utilisant Apache Spark pour évaluer des requêtes SPARQL et stockant les données via des infrastructures Hadoop (HDFS). SPARQLGX repose sur un traducteur de requêtes SPARQL vers une séquence d'instructions exécutables par Spark en adoptant des stratégies d'évaluation selon (1) le schéma de stockage des données utilisé et (2) des statistiques sur les données. Nous montrons que SPARQLGX permet l'évaluation de requêtes SPARQL sur plusieurs milliards de triplets RDF répartis sur plusieurs nœuds. Nous comparons aussi SPARQLGX à d'autres solutions issues de l'état-de-l'art. Nous démontrons ainsi les performances obtenues en permettant aux participants de reproduire par eux-mêmes les résultats présentés grâce à différents scénarios mettant directement en compétition plusieurs solutions de l'état-de-l'art. Nous montrons dans ce travail que tout en ayant une architecture relativement simple, SPARQLGX représente une alternative intéressante dans de nombreux cas d'utilisation.

1. INTRODUCTION

SPARQL is the standard query language for retrieving and manipulating data represented in Resource Description Framework (RDF) [7]. SPARQL constitutes one key technology of the semantic web and has become very popular since it became an official W3C recommendation, first in 2008 [1] and then in 2013 as an improved version [2]. In this study, we investigate the problem of evaluating the conjunctive SPARQL queries *i.e.* the Basic Graph Pattern fragment.

The construction of efficient SPARQL query evaluators faces several challenges. First, RDF datasets are increasingly large, with some already containing more than a billion triples. To handle efficiently this growing amount of data, we need systems to be distributed and to scale. Furthermore, semantic data often have the characteristic of being dynamic (frequently updated). Thus being able to answer quickly after a change in the input data constitutes a very desirable property for a SPARQL evaluator.

In this context, we propose SPARQLGX: an engine designed to evaluate SPARQL queries based on Spark [11]: a MapReduce-like data-parallel engine. SPARQLGX relies on a compiler of SPARQL conjunctive queries which generates Scala code that is executed by the Spark infrastructure.

2. SPARQLGX: ARCHITECTURE

Data Storage Model. In order to process RDF datasets, we adopt the vertical partitioning approach introduced by Abadi *et al.* in [3] 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.

Compilation of SPARQL BGPs. Conjunctions of triple patterns (TPs) 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.

SPARQL Fragment Extension. Moreover, we support the selection of distinguished variables in addition of the "SELECT *" form. We also support SPARQL solution modifiers *e.g.* removing duplicates with DISTINCT, sorting with ORDER BY, returning only few answers with LIMIT.

Furthermore, we also easily translate two additional SPARQL keywords: UNION and OPTIONAL, provided they are located at top-level in the WHERE clauses. Indeed, Spark allows to aggregate sets having similar structures with `union` and is also able to add data if possible with `leftOuterJoin`. Thus SPARQLGX natively supports a slight extension (unions and optionals both at top level) of the extensively studied SPARQL fragment made of conjunctions of TPs.

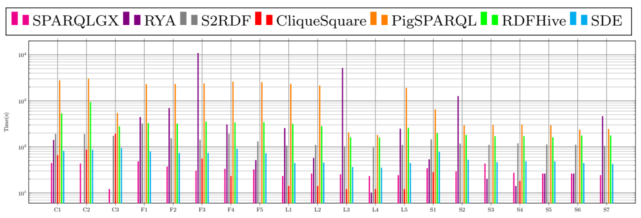

Figure 1: Query Response Times with WatDiv1k.

Optimized Joins With Statistics. The evaluation process (using Spark) first evaluates TPs and then joins these subsets according to their common variables; thus, minimizing the intermediate set sizes involved in the join process reduces evaluation time (since communication between workers is then faster). Thereby, statistics on data and information on intermediate results sizes provide useful information that we exploit for optimisation purposes.

Thereby, to rank each TP, we compute statistics on datasets counting all the distinct subjects, predicates and objects. This is implemented in a compile-time module that sorts TPs in ascending order of their selectivities before they are translated.

Finally, we also want to avoid cartesian products. Given an ordered list l of TPs we compute a new list l' by repeating the following procedure: remove from l and append to l' the first TP that shares a variable with a TP of l'. If no such TP exists, we take the first.

S-DE: SPARQLGX as a Direct Evaluator. 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 [5], S2RDF [10] and RDFHive. Furthermore, we consider several datasets coming from two popular BGP benchmarks: LUBM [6] and WatDiv [4].

We present in Figure 1 the response times obtained with WatDiv1k (about 100 million triples). This illustrates that, for this dataset: (1) SDE always outperforms other tested "direct evaluators" (*i.e.* 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 linear queries; (4) it outperforms its competitors on complex queries. To further illustrate the performance of SPARQLGX, we provide an interactive GUI. Attendees can interact directly with our engines (*i.e.* SPARQLGX and SDE) and test them against state-of-the-art solutions.

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`

Related Publications

- **SPARQLGX: Efficient Distributed Evaluation of SPARQL with Apache Spark.**
  D. Graux, L. Jachiet, P. Genevès, N. Layaïda. *The 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II, pages 80,87.*

- **SPARQLGX in action: Efficient Distributed Evaluation of SPARQL with Apache Spark.**
  D. Graux, L. Jachiet, P. Genevès, N. Layaïda. *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference, Kobe, Japan, October 19, 2016.*

5. REFERENCES

[1] SPARQL query language for RDF, January 2008. www.w3.org/TR/rdf-sparql-query/.

[2] SPARQL 1.1 overview, March 2013. http://www.w3.org/TR/sparql11-overview/.

[3] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases*, pages 411–422. VLDB Endowment, 2007.

[4] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of RDF data management systems. In *ISWC*, pages 197–212. Springer, 2014.

[5] F. Goasdoué, Z. Kaoudi, I. Manolescu, J.-A. Quiané-Ruiz, and S. Zampetakis. Cliquesquare: Flat plans for massively parallel RDF queries. In *ICDE*, pages 771–782. IEEE, 2015.

[6] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):158–182, 2005.

[7] P. Hayes and B. McBride. RDF semantics. *W3C Rec.*, 2004.

[8] R. Punnoose, A. Crainiceanu, and D. Rapp. Rya: a scalable RDF triple store for the clouds. In *International Workshop on Cloud Intelligence*, page 4. ACM, 2012.

[9] A. Schätzle, M. Przyjaciel-Zablocki, and G. Lausen. PigSPARQL: Mapping SPARQL to pig latin. In *Proceedings of the International Workshop on Semantic Web Information Management*, page 4. ACM, 2011.

[10] A. Schätzle, M. Przyjaciel-Zablocki, S. Skilevic, and G. Lausen. S2RDF: RDF querying with SPARQL on spark. *VLDB*, pages 804–815, 2016.

[11] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 2–2. USENIX Association, 2012.