An Adaptive Framework for RDF Stream Reasoning

Qiong Li[1,3], Xiaowang Zhang[1,3,*], Zhiyong Feng[2,3], and Guohui Xiao[4]

[1] School of Computer Science and Technology, Tianjin University, Tianjin 300350, P. R. China,
[2] School of Computer Software,Tianjin University, Tianjin 300350, P. R. China
[3] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350, P.R. China
[4] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano I-39100, Italy
[*] Corresponding author.

Abstract. In this paper, we propose an adaptive framework for RDF stream reasoning (PRSPR) in order to obtain more meaningful and valuable information, which is an extension of our previous work. Moreover, our work is a kind of plug-in framework which makes it more adaptive and flexible. Within this framework, not only can we apply all kinds of SPARQL query engines to process RDF streams, but also simultaneously support various inference engines for RDFS/OWL for stream reasoning. Finally, we experimentally evaluate the performance of PRSPR on YABench. The experiments show that PRSPR can still maintain the high performance with SPARQL query engines in RDF stream reasoning although there are some slight differences among them.

1 Introduction

RDF stream, as a new type of dataset, can model real-time and continuous information in a wide range of applications, e.g., environmental monitoring and smart city [1]. In [3] we already presented an adaptive framework PRSP to process RDF streams by exploiting various SPARQL query engines in a brief way. However, if we want to obtain more detailed and abounding information, it is necessary to address the problem of performing reasoning for very dynamic inputs. There are many approaches about RDFS reasoning over static RDF graphs, but rare over RDF streams. Liu et al. [5] proposed a method over static data to enhance the performance of rule-based OWL reasoning on Spark by exploiting a locally optimal executable strategy. Chang et al. [4] present a approach to perform stream reasoning on RDF data using the GPU computing architecture, but it reasons streams after processing streams.

In this paper, we provide an adaptive framework for RDF stream reasoning named PRSPR which is an extension of our PRSP. In our work, we further optimize PRSP by improving the performance. On the other hand, we apply the locally optimal executable strategy to implement stream reasoning. Therefore, PRSPR not only makes it possible to use the high-performance SPARQL query engines to process large-volume RDF streams, but also makes it easier for stream reasoning by applying all kinds of RDF/RDFS and OWL entailment rules in a convenient way.

2 Preliminaries

RDF stream An RDF stream S is defined as ordered sequences of pairs, made of an RDF triple and a timestamp τ: $(\langle s, p, o\rangle, \tau)$.
Continuous Query Formally, a continuous SPARQL query Q can be taken as a 5-tuple of the form:

$$Q = [\mathrm{Reg}, S, \mathrm{w}, \mathrm{s}, \rho(Q)]$$

where

- Reg: the registration;
- S: the RDF stream registered;
- w: RANGE, i.e., the window size;
- s: STEP, i.e., the updating time of windows;
- $\rho(Q)$: a SPARQL query.

RDF Schema RDF Schema (RDFS) is a set of classes with certain properties using the RDF extensible knowledge representation data model, providing basic elements for the description of ontologies, otherwise called RDF vocabularies, intended to structure RDF resources.

3 A framework for RDF stream reasoning

PRSPR is a framework for querying and reasoning both RDF graphs and RDF streams shown in Figure 1. Both continuous query and RDF streams as the input of PRSPR, they continuously reason and process by exploiting the following four modules: Query Preparation, Data Transformer, RDFS Reasoner and Query Execution. We give a detailed description about the workflow below.

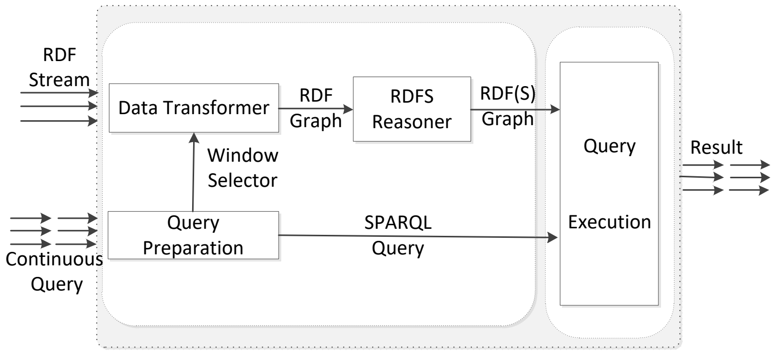


Fig. 1. The framework of PRSPR

Query Preparation Continuous queries, as the input of Query Preparation mode, generate two types of queries, namely, SPARQL query and Window Selector, which can be addressed in the Query Execution and Data Transformer module respectively.

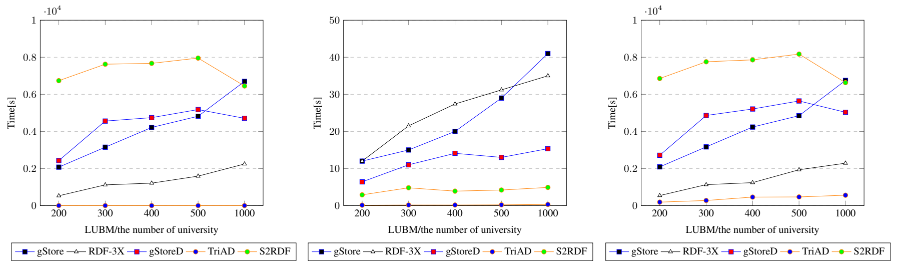
Data Transformer Data Transformer module processes RDF streams. It transforms RDF streams into RDF graphs based on the window size and step size set at Window Selector. And the output, RDF graphs are as the input of RDFS Reasoning module.
RDFS Reasoner RDFS Reasoner is responsible for reasoning RDF graphs based on the rule-based knowledge ontology on Spark. The reasoning computes the deductive closure of an ontology by applying RDF/RDFS entailment rules in order to make implicit knowledge explicit, i.e., obtaining RDF(S) graphs.
Query Execution PRSPR defines a unified interface for SPARQL query engines, which makes it possible and easy for SPARQL query engines to process RDF streams.

4 Experiments and Evaluations

All centralized experiments were carried out on a machine running Linux, which has 4 CPUs with 6 cores and 64GB memory, and 4 machines with the same performance for distributed experiments. The version of Spark is 1.5.2. We utilized YABench RSP benchmark[2], and registered LUBM as the RDF streams. The complexity of the scenarios was in the ascending order, from the least complex configuration (*LUBM200*) that loaded roughly 28 million triples to the most complex configuration (*LUBM1000*) that injected more than 130 million triples. In our experiments, we perform sliding windows with a 2400-seconds-window which slides every 2300 seconds, and chose the two queries over LUBM, i.e., Q6 and Q10. The reason is that the two queries can not return results over LUBM unless the data has been inferred by reasoners. We use the OWL-Horst rules[5] as the reasoner in our experiments.

Table 1. RDFS reasoning time

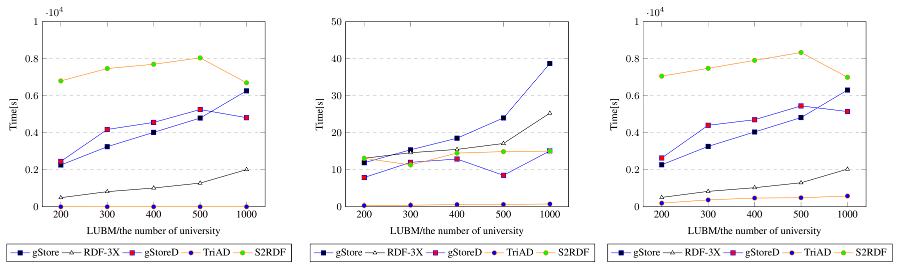
Dataset	lubm200	lubm300	lubm400	lubm500	lubm1000
Time(s)	3155	4039	5024	6145	9688



(a) Triples loading time (b) Query response time (c) Engine execution time

Fig. 2. RDF stream for processing time of Q6 within PRSPR

We compare the performance of the five different SPARQL query engines, including two centralized engines (RDF-3X and gStore) and three distributed systems (gStoreD, TriAD and S2RDF) in a unified way. The RDFS reasoning time (shown in table 1), is increased in varying degrees with the growth of dataset. The processing time of the two queries within PRSPR is shown in Fig 2 and Fig 3, respectively. gStoreD and S2RDF can not process the whole triples in the window at the specific time over LUBM1000, which results in some window triples lose and the results are incomplete. When the load ranges from LUBM200 to LUBM1000, the triples loading time, query response time and engine execution time are increasing. But we can also get that the distributed engines (TriAD and S2RDF) have the better performance than centralized engines.



(a) Triples loading time (b) Query response time (c) Engine execution time

Fig. 3. RDF stream for processing time of Q10 within PRSPR

5 Conclusions

In this paper, we present the PRSPR, as a plugin adaptable for various SPARQL query engines and reasoning machines, which makes the system more adaptive. Moreover, PRSPR is for both RDF streaming and stream reasoning, so that it can obtain more valuable information. Therefore, it can also process large-volume RDF streams by applying distributed SPARQL query engines.

Acknowledgments

We thank Bo Zhao for his assistance in the experiments of this paper. This work is supported by the programs of the National Natural Science Foundation of China (61672377), the National Key R&D Program of China (2016YFB1000603), and the Key Technology Research and Development Program of Tianjin (16YFZCGX00210).

References

1. Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., and Grossniklaus, M.: Querying RDF streams with C-SPARQL. SIGMOD REC. 39(1), 20–26 (2010)
2. Kolchin, M., Wetz, P., Kiesling, E., and Tjoa, A. M.: YABench: A comprehensive framework for RDF stream processor correctness and performance assessment. In: Proc. of ICWE '16, pp. 280–298 (2016)
3. Li, Q., Zhang, X., Feng, Z.: PRSP: A plugin-based framework for RDF stream processing. In: Proc. of WWW'17, poster, pp. 815–816 (2017)
4. Liu C, Urbani J, and Qi G.: Efficient RDF stream reasoning with graphics processingunits (GPUs). In: Proc. of WWW '14, pp. 343–344 (2014)
5. Liu, Z., Feng, Z., Zhang, X., Wang, X., and Rao, G.: RORS: Enhanced Rule-Based OWL Reasoning on Spark. In: Proc. of APWeb '16, pp. 444–448 (2016)