Towards a Rule Based Distributed OWL Reasoning Framework

Raghava Mutharaju(✉), Prabhaker Mateti, and Pascal Hitzler

Wright State University, Dayton, OH, USA
{mutharaju.2,prabhaker.mateti,pascal.hitzler}@wright.edu

Abstract. The amount of data exposed in the form of RDF and OWL continues to increase exponentially. Some approaches have already been proposed for the scalable reasoning over several language profiles such as RDFS, OWL Horst, OWL 2 EL, OWL 2 RL etc. But all those approaches are limited to the particular ruleset that the reasoner supports. In this work, we propose the idea for a rule-based distributed reasoning framework that can support any given ruleset and highlight some of the challenges that needs to be solved in order to implement such a framework.

1 Introduction

The W3C recommendations RDF and OWL, are primarily used to represent data in the Semantic Web. Large amount of data in these formats is now available and it only continues to grow. Several billions of RDF triples are available as Linked Open Data (close to 90 billion[1]). Automated generation of OWL axioms from streaming data [9] and text [10] can result in very large knowledge bases. Reasoning is one of the most important operations that can be performed over OWL and RDF knowledge bases. It is required to infer logical consequences and to check the consistency of the knowledge base. Reasoning is memory and compute intensive. So reasoning over large knowledge bases needs a scalable approach. Currently, all the popular off-the-shelf reasoners work only on a single machine, possibly with multiple cores. It is not possible for a single machine to keep up with the growth rate of data. Also, for some reasoning tasks the output is several times larger than the input. Distributed memory reasoning provides a viable alternative.

There are some existing approaches for scalable reasoning over each individual Semantic Web language profile such as RDFS, OWL Horst, OWL 2 EL, OWL 2 RL (see Sect. 4). Reasoning over ontologies in each of these profiles is performed using a set of rules that vary with each profile (there is some overlap among the different rulesets). Generally, the existing solutions are tuned towards a particular ruleset and are not adaptable to other rulesets. This poses a problem for users who work with multiple rulesets and also in cases where users need a scalable solution for a ruleset which does not have a customized approach. In this paper, we propose the idea for a unified distributed reasoning framework that

[1] http://stats.lod2.eu/.

V. Tamma et al. (Eds.): OWLED 2015, LNCS 9557, pp. 87–92, 2016.
DOI: 10.1007/978-3-319-33245-1_9

Table 1. RDFS closure rules

1:	$s\ p\ o$ (if o is literal)	$\Rightarrow$ _:n rdf:type rdfs:Literal
2:	p rdfs:domain x & $s\ p\ o$	$\Rightarrow$ s rdf:type x
3:	p rdfs:range x & $s\ p\ o$	$\Rightarrow$ o rdf:type x
4a:	$s\ p\ o$	$\Rightarrow$ s rdf:type rdfs:Resource
4b:	$s\ p\ o$	$\Rightarrow$ o rdf:type rdfs:Resource
5:	p rdfs:subPropertyOf q & q rdfs:subPropertyOf r	$\Rightarrow$ p rdfs:subPropertyOf r
6:	p rdf:type rdf:Property	$\Rightarrow$ p rdfs:subPropertyOf p
7:	$s\ p\ o$ & p rdfs:subPropertyOf q	$\Rightarrow$ $s\ q\ o$
8:	s rdf:type rdfs:Class	$\Rightarrow$ s rdfs:subClassOf rdfs:Resource
9:	s rdf:type x & x rdfs:subClassOf y	$\Rightarrow$ s rdf:type y
10:	s rdf:type rdfs:Class	$\Rightarrow$ s rdfs:subClassOf s
11:	x rdfs:subClassOf y & y rdfs:subClassof z	$\Rightarrow$ x rdfs:subClassOf z
12:	p rdf:type rdfs:ContainerMembershipProperty	$\Rightarrow$ p rdfs:subPropertyOf rdfs:member
13:	o rdf:type rdfs:Datatype	$\Rightarrow$ o rdfs:subClassOf rdfs:Literal

can work on any given ruleset. This framework can, not only handle the aforementioned language profiles but also avoids the need to develop a customized scalable approach for any new ruleset.

2 Challenges

Some of the challenges in the design and implementation of a rule-based distributed reasoning framework are discussed here.

2.1 Rule Dependency Analysis

If the input to a rule R_i, depends on the output of another rule R_j, then the rule R_i is dependent on R_j. The more independent the rules are, better can be the rule distribution among the nodes in the cluster.

A rule dependency graph can be constructed in order to determine the inter-dependency among the rules. Each vertex represents a rule and an outgoing edge between vertex v_i and v_j represents the dependency of vertex v_i on v_j. Isolated vertices are independent of each other and can be executed in parallel. For each vertex v_i, all the vertices that are reachable from it are dependent on each other.

RDFS rules from [21] are shown in Table 1 and its dependency graph is shown in Fig. 1. There are several rules that are independent and can be given to separate nodes in the cluster. Rules 5, 6 and 7 are dependent on each other and can be grouped together i.e., all the three rules can be executed by one node. Same holds for the group of rules 9, 2, 3, 10, 11.

On the other hand, rules for $\mathcal{EL}^{++}$ [1] are highly inter-dependent and are difficult to parallelize.

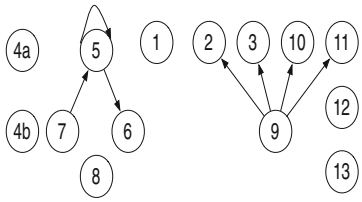

Fig. 1. RDFS rule dependency graph

2.2 Data Distribution

Rule dependency analysis can provide the basis for data distribution. Some rules are applicable only to a specific type of data and in these cases, it would be easy to partition the data based on the rule distribution. Cases such as rules 4a, 4b and 6 in Table 1 should be handled differently since they are applicable to the entire dataset. These rules should be run on all the nodes of the cluster so as to avoid data overloading. Heuristics such as number of variables in a rule in proportion to the constants could be used to determine such type of rules.

2.3 Rule Implementation

Interpreting different rulesets will be very difficult for the framework. Instead, rulesets should be converted to a common domain specific language (DSL) that is supported by the framework. This DSL should be able to define the vocabulary, syntax and semantics of the language to be used.

For the choice of DSL, there are some options. (1) general purpose rule languages such as RETE, Datalog and Prolog. (2) or a custom DSL for the rules supported by the framework. DSL should support the declaration of variables and constants in the rules.

3 Evaluation Plan

The rule-based distributed reasoning framework can be evaluated along the lines of adaptation and extension.

- There are several existing specialized and scalable reasoners for rulesets such as RDFS (WebPIE, Cichlid), OWL Horst (QueryPIE) and OWL 2 EL (DistEL). The framework should be able to handle these rulesets. The performance of the general purpose framework in comparison to the specialized ones remains to be seen.
- The framework should be able to take in a new ruleset and provide sound and complete inference over the given data.

4 Related Work

There are several language profiles in the Semantic Web that support rule-based reasoning. Other reasoning approaches such as tableau algorithms are not considered here. Scalable reasoning approaches such as parallel shared memory reasoning to distributed shared-nothing reasoning exist.

RDFS has around 34 inference (entailment) rules including simple, extensional and datatype entailment rules [3]. Almost all of the existing work on scalable RDFS entailment (closure) computation considers only a subset of these rules. Marvin computes the closure of RDF triples using a peer-to-peer model [16]. In [22], triples are distributed across the cluster by making a distinction between the schema and instance triples. Finding closure becomes an embarrassingly parallel computation. Several other scalable approaches exist for RDFS closure computation [4,6,21].

OWL Horst (also known as pD*) [5] extends RDFS entailment rules to include reasoning with datatypes from a given datatype map D. Rule partitioning and data partitioning strategies are explored in [19] for computing closure over OWL Horst knowledge base. QueryPIE [20] is a backward chaining distributed reasoner that supports OWL Horst reasoning over large knowledge bases. A recent Apache Spark implementation of RDFS and OWL Horst rules named Cichlid, is 10 times faster than state-of-the-art implementations [2].

OWL 2 RL [11] is further extension of pD* and has several entailment rules. A scalable OWL 2 RL inference engine has been implemented inside a relational database system (Oracle) in [8]. QueryPIE has partial support for OWL 2 RL.

The description logic underlying OWL 2 EL is $\mathcal{EL}^{++}$ and there are 11 completion rules ([1]) for classification (computing the subsumption hierarchy of all concepts). Three approaches to distributed $\mathcal{EL}^{++}$ reasoning were discussed in [13] including MapReduce [15]. Among them, the most efficient system named DistEL, follows a peer-to-peer model that uses rule partitioning based on the axiom types [14]. Though not distributed, parallelization of OWL 2 EL classification has been studied in [7,18].

There are some existing generic rule-based scalable reasoning approaches. RETE implementation on GPUs for RDFS and OWL Horst rulesets is shown in [17]. Another alternative is to convert different rulesets into datalog rules. A parallel implementation of datalog programs with application to RDFS rules is shown in [12]. A distributed approach for either of these two has not been developed yet.

5 Conclusion

There are different rulesets for different language profiles and there are scalable approaches for many of these rulesets. However, such a specialized scalable reasoner does not work on other rulesets. A general purpose rule-based distributed reasoning framework is proposed here to fill that gap. This framework provides more flexibility in terms of rulesets but there could be a possible loss

in performance when compared to specialized scalable reasoners. A proper evaluation of the framework is required in order to determine its flexibility and performance.

The next step is to choose the appropriate DSL by checking the advantages and disadvantages of RETE, Datalog and Prolog. After this, we plan to proceed with the implementation of rest of the pieces in the framework.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-2005, July 30–August 5 2005, Edinburgh, Scotland, UK, pp. 364–369. AAAI (2005)
2. Gu, R., Wang, S., Wang, F., Yuan, C., Huang, Y.: Cichlid: efficient large scale RDFS/OWL reasoning with spark. In: 2015 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2015, 25–29 May 2015, Hyderabad, India, pp. 700–709. IEEE Computer Society (2015)
3. Hayes, P., Patel-Schneider, P.F.: RDF Semantics (2014). http://www.w3.org/TR/rdf11-mt/
4. Heino, N., Pan, J.Z.: RDFS reasoning on massively parallel hardware. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 133–148. Springer, Heidelberg (2012)
5. ter Horst, H.J.: Combining RDF and part of OWL with rules: semantics, decidability, complexity. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 668–684. Springer, Heidelberg (2005)
6. Kaoudi, Z., Miliaraki, I., Koubarakis, M.: RDFS reasoning and query answering on top of DHTs. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 499–516. Springer, Heidelberg (2008)
7. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 305–320. Springer, Heidelberg (2011)
8. Kolovski, V., Wu, Z., Eadon, G.: Optimizing enterprise-scale OWL 2 RL reasoning in a relational database system. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 436–452. Springer, Heidelberg (2010)
9. Lécué, F., Tucker, R., Bicer, V., Tommasi, P., Tallevi-Diotallevi, S., Sbodio, M.: Predicting severity of road traffic congestion using semantic web technologies. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 611–627. Springer, Heidelberg (2014)
10. Lehmann, J.: DL-learner: learning concepts in description logics. J. Mach. Learn. Res. (JMLR) **10**, 2639–2642 (2009)
11. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language Profiles. In: W3C Recommendation (2012). http://www.w3.org/TR/owl2-profiles/
12. Motik, B., Nenov, Y., Piro, R., Horrocks, I.: Parallel materialisation of datalog programs in main-memory RDF databases. In: Brodley, C.E., Stone, P. (eds.) Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27–31 2014, Qébec City, Qébec, Canada. AAAI Press (2014)

13. Mutharaju, R., Hitzler, P., Mateti, P.: Distributed OWL EL reasoning: the story so far. In: Liebig, T., Fokoue, A. (eds.) Proceedings of the 10th International Workshop on Scalable Semantic Web Knowledge Base Systems, Riva Del Garda, Italy. CEUR Workshop Proceedings, vol. 1261, pp. 61–76. CEUR-WS.org (2014)
14. Mutharaju, R., Hitzler, P., Mateti, P., Lécué, F.: Distributed and Scalable OWL EL Reasoning. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 88–103. Springer, Heidelberg (2015)
15. Mutharaju, R., Maier, F., Hitzler, P.: A MapReduce algorithm for EL+. In: Haarslev, V., Toman, D., Weddell, G.E. (eds.) Proceedings of the 23rd International Workshop on Description Logics (DL 2010), 4–7 May 2010, Waterloo, Ontario, Canada. CEUR Workshop Proceedings, vol. 573. CEUR-WS.org (2010)
16. Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., van Harmelen, F.: Marvin: distributed reasoning over large-scale semantic web data. Web Seman. Sci. Serv. Agents World Wide Web **7**(4), 305–316 (2009)
17. Peters, M., Sachweh, S., Zündorf, A.: Large scale rule-based reasoning using a laptop. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 104–118. Springer, Heidelberg (2015)
18. Ren, Y., Pan, J.Z., Lee, K.: Parallel ABox reasoning of $\mathcal{EL}$ ontologies. In: Pan, J.Z., Chen, H., Kim, H.-G., Li, J., Horrocks, I., Mizoguchi, R., Wu, Z., Wu, Z. (eds.) JIST 2011. LNCS, vol. 7185, pp. 17–32. Springer, Heidelberg (2012)
19. Soma, R., Prasanna, V.K.: Parallel inferencing for OWL knowledge bases. In: 2008 International Conference on Parallel Processing, ICPP 2008, 8–12 September 2008, Portland, Oregon, USA, pp. 75–82. IEEE Computer Society (2008)
20. Urbani, J., van Harmelen, F., Schlobach, S., Bal, H.: QueryPIE: backward reasoning for OWL horst over very large knowledge bases. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 730–745. Springer, Heidelberg (2011)
21. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)
22. Weaver, J., Hendler, J.A.: Parallel materialization of the finite RDFS closure for hundreds of millions of triples. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 682–697. Springer, Heidelberg (2009)