Scaling Out Sound and Complete Reasoning for Conjunctive Queries on OWL Knowledge Bases

Sambhawa Priya

Department of Computer Science and Engineering, Lehigh University
19 Memorial Drive West, Bethlehem, PA 18015, USA
sps210@lehigh.edu

Abstract. One of the challenges the Semantic Web community is facing today is the issue of scalable reasoning that can generate responsive results to complicated queries over large-scale OWL knowledge bases. Current large-scale semantic web systems scale to billions of triples but many such systems perform no reasoning or rely on materialization. On the other hand, most state-of-the-art, sound and complete DL reasoners are main memory-based and fail when given ontologies that include enormous data graphs in addition to expressive axioms. Thus, until now, reasoning has been restricted to either limited expressivity or limited size of the data. The focus of this thesis is to develop a scalable framework to perform sound and complete reasoning on large and expressive data graphs for answering conjunctive queries over a cluster of commodity machines. In order to achieve our goal, we outline our approach to address the following challenges: partitioning large and expressive datasets across the cluster for distributed reasoning, and allocating reasoning and query-execution tasks involved in processing conjunctive queries to nodes of the cluster. We include evaluation results for our preliminary framework.

Keywords: distributed reasoning, partitioning, conjunctive queries, scalable framework

1 Problem Statement

Objective of the Research: This thesis focuses on how to scale out sound and complete reasoning for answering conjunctive queries over real-world ontologies with increasingly large data graphs over commodity clusters. Our goal is to design a framework that can scale to clusters of commodity machines for answering complex queries over large-scale knowledge bases characterized by small but expressive TBox and very large ABox. In order to reach this objective, we plan to address the following research questions: How to partition expressive datasets across the cluster such that traditional DL reasoners can answer subproblems independently at each partition? How to recombine these results to get sound and complete answers? How to co-locate partitions to reduce communication overheads? How to allocate reasoning and query-execution tasks involved in processing conjunctive queries to different nodes of the cluster? Note, we restrict

our application to handle queries about instances (i.e. ABox queries) and not the classes and properties (i.e. TBox queries) in the ontology.
Motivation and Relevance: A framework that allows reasoning for answering queries over large linked datasets can be a useful tool for finding insights from large graphs of data coming from diverse sources such as medicine and health-care, finance, social networks, government and the Internet of Things. Our proposed system can allow users to quickly answer questions of interest without creating application specific code or performing domain specific graph analyses.
Challenges and Opportunities: The main challenges we want to be able to address are as follows:

a) Partitioning large, highly networked and expressive datasets across the cluster for distributed reasoning to answer conjunctive queries: The partitioning approach [1] we used in our preliminary implementation [2] works only for $\mathcal{SHIF}$ DL. This approach might be limited by power law and, in case of highly networked data, might generate very large partitions. The challenge is to extend the expressivity of the approach and generalize it to handle very expressive, highly networked and complex datasets.
b) Assigning the data-partitions and query-execution tasks to the nodes in the cluster such that the reasoning and query-execution tasks involved in processing the conjunctive queries are efficiently performed: Most existing works [3, 6] on scalable and distributed processing of conjunctive queries do not take reasoning into account or rely on materialization. On the other hand, existing works on parallel implementation of backward-chaining reasoning [7] do not address the challenge of computing conjunctive queries that involve DL reasoning. We want to address a combination of both of these issues in a scalable manner.

2 Proposed Approach

In order to address the challenges listed above, our proposed approach is as follows: Adopting a divide-and-conquer approach, we plan to explore partitioning of large and expressive datasets so that reasoning can be performed on subsets of data in a distributed environment where CPUs and memory of many commodity machines are harnessed. We plan to devise strategies to select partitions relevant to a query which, not only takes into account the property or the constant terms appearing in the query, but also incorporates the reasoning involved in answering that query. We plan to explore techniques to allocate the data partitions to nodes such that the load is evenly balanced across the cluster and the cost of communicating intermediate query results between the nodes is minimized. Time taken to perform expressive reasoning over semantic web datasets can be vulnerable to the order in which the query clauses are evaluated. In order to address this, we propose to implement and evaluate techniques for reasoning-aware query planning and develop strategies for allocating subqueries to compute nodes to perform distributed reasoning for answering conjunctive queries efficiently.

Related Work: A few authors [4, 5] have proposed approach for partitioning large OWL TBox based on the structure of class hierarchy and dependencies between the TBox elements. However, our focus is on such knowledge bases where the TBox is small enough to be replicated on each compute node of the cluster but the ABox is very large and requires partitioning across the cluster for scalable reasoning. Most of the previous work on parallel and distributed reasoning on semantic web datasets has been limited to forward chaining for less expressive logic such as RDFS, while our focus is on performing backward-chaining reasoning on rich description logics, beginning with $SHIF$ and pushing towards achieving a distributed system for $SHROIQ$. Oren et al. [8] combine parallel hardware with distributed algorithms to implement a system called MARVIN for scalable RDFS reasoning. Weaver et al. [9] derive an 'embarrassingly parallel' algorithm for materializing complete RDFS closure using C/MPI. WebPIE [10] has been used to compute the transitive closure of up to 100 billion triples, using the OWL Horst fragment (which is less expressive than $SHIF$). AllegroGraph reports that a prerelease has been tested on up to 1 trillion triples [12], but the reasoning is limited to little more than RDFS (subsumption, domain, range) plus inverses, sameAs, and transitivity. One of the few known systems to perform backward chaining (in combination with materialization) is QueryPIE [7], a parallel engine for OWL Horst reasoning that has scaled to 1 billion triples using an 8 machine cluster. However, they show query evaluation for only single pattern queries, not conjunctive queries. Triple stores [13, 14] build specialized indices and apply many join optimization techniques to improve processing of conjunctive queries expressed in SPARQL. Gurajada et. al. [6] developed a distributed triple store with novel join-ahead pruning technique for the distributed processing of SPARQL queries. However, none of these triple stores perform any reasoning. Mutharaju et. al. [11] present an approach for distributed reasoning for OWL 2 EL ontologies where the main reasoning task is classification where as our focus is on distributing DL reasoning for conjunctive queries where finding all the answers to a conjunctive query is the primary reasoning task.

3 Implementation of the Proposed Approach

Partitioning of large and expressive datasets: In our preliminary work [2], we utilize a partitioning technique for OWL Lite datasets proposed by Guo and Heflin [1]. Since this technique is restricted to $SHIF$ DL, we plan to explore how this partitioning technique can be extended to $SHOIN$ and $SHROIQ$. One idea is to use theory approximation [15] to create a SHIF approximation of a more expressive set of axioms. In particular, if this approximation is a lower-bound on the models of the original theory, then any logical consequences of the original theory will also be logical consequences of the approximation. Introducing such approximations may lead to unsoundness resulting in partitions that will include triples that do not need to be grouped together, resulting in larger than needed partitions. However, since sound and complete reasoners are used to perform reasoning over each partition, the inferred knowledge will still be sound and complete. We will evaluate using datasets of different sizes and expressivity to determine the limitations of the approximation approach. The approach described in [1] may be limited by power law which is characteristic of real-world, large and networked data. We plan to analyze the partitionability of such datasets, taken from Linked Data and synthetic data sources to investigate the limitations of the technique and adapt it to handle such arbitrary data graphs. In [1], after the ontology axioms have been analyzed, assertional triples that have common subjects or objects and interrelated predicates are placed in a common partition. This implies that it is possible to scale-out partitioning by creating a MapReduce version of the algorithm where the key is generated from the triple's subject and/or object and predicate.

Distributing data partitions across the cluster: The data-partitions can be distributed across the cluster during the query execution time or prior to all queries. For query-time distribution, we can select partitions relevant to a given query and distribute them among the compute nodes that results in even load balancing and minimized cost of transferring intermediate join-results. We can sequentially load partitions on compute nodes using multiple threads to even out the disk-access cost. Query-time distribution can be inefficient when dealing with multiple concurrent queries. However, in our first implementation, we will implement a framework to handle only one query at a time, and later adapt it to support multiple concurrent queries. We can also distribute data partitions across the cluster prior to all queries. The fundamental question here is how to determine the best data allocation for a mix of unknown queries. We want to explore whether we can can use heuristics to create a reasonable allocation for a query mix that fits certain basic assumptions. For example, we know that partitions containing the same predicate are more likely to be relevant to the same query triple pattern and, hence, should be spread across nodes in order to maximize their utilization. When processing a query, we can identify if the load is unbalanced and depending upon the query-processing task allocation on the nodes, we can determine if shuffling a subset of partitions between certain nodes can achieve a better load balance.

Reasoning-aware query planning for distributed query execution: The order of query clause evaluation is critical to the query response time. Processing selective query triple patterns and joins early can reduce the volume of intermediate results and can reduce the query processing time. Typical query optimization algorithms use statistics about the data such as the number of triples matching a given predicate, the number of distinct subjects/objects for each predicate, the distribution of these values using histograms, and statistics on the joined triple patterns [16]. However traditional SPARQL query optimization does not consider that reasoning can produce results with significantly different cardinalities than those estimated from the raw data. We plan to compute reasoning statistics as part of data partitioning process. In the long term, heuristics from previous queries can be cached and used for query planning when statistics fall short. For executing these query plans on the cluster, we need to allocate the query plan components to different nodes such that the reasoning and join-execution tasks

can be efficiently performed over the cluster. We plan to implement a master-slave architecture where the master will create a reasoning-aware logical query plan and a corresponding physical query plan to map the data and query tasks to different slaves; and slaves nodes will process their respective physical query plans concurrently, interleaving sound and complete reasoning for triple patterns with distributed join processing using a message-passing protocol. We plan to study the impact of our distributed query-answering framework on query performance using different query patterns involving complex reasoning on large scale real-world and synthetic datasets.
Current Implementation: We developed a preliminary framework [2] for parallel reasoning on partitioned dataset where we first partition the knowledge base using the strategy developed by Guo and Heflin [1] and then the execute reasoning tasks on data-partitions in parallel on independent machines. We implemented a master-slave architecture that distributes an input query (expressed in SPARQL query language) to the slave processes on different machines. All slaves run in parallel, each performing sound and complete reasoning using a tableau-based reasoner (Pellet) to execute each subgoal of the given conjunctive query on its assigned set of partitions. As a final step, the master joins the results computed by the slaves. We use an off-the-shelf database to store the results of query subgoals and to perform the final join on the results of the conjuncts. However, we have identified a few drawbacks in our preliminary framework: each compute node performs reasoning on every partition assigned to it, irrespective of the partition's relevance to the subgoal; and the relational database becomes a bottleneck while inserting and joining large intermediate results. As described in the previous section, we plan to implement an improved architecture, which, for a given query, generates reasoning-aware query plan and distributes relevant data partitions and query processing tasks across the cluster. We plan to implement our own distributed join-processing framework that will utilize a message passing protocol.

4 Empirical Evaluation Methodology

General Strategy: We plan to evaluate both a) the partitioning strategies for large and expressive datasets with different connectivity patterns, and b) the performance of our distributed reasoning framework. We will vary the size of the data, the size of the query (i.e., the number of query triple patterns), the number of compute nodes, the partitioning approach, and query optimization strategies for distributed query processing.

Hypotheses:

1. Applying theory approximation to more expressive set of axioms, as discussed in Section 3, will result in coarser ABox partitions but will preserve independence of resulting partitions.
2. Loosely connected datasets with weak axioms will produce a large number of smaller partitions. Vice versa for highly connected datasets with rich axioms.

3. It is possible to scale-out the partitioning technique by exploiting the MapReduce version of the algorithm where the key of each triple is generated from it's predicate, subject and/or object.
4. There exist data distribution strategies that can improve average query performance on an unknown mixes of queries while making minimal assumptions about those query.
5. Selectivity statistics about query triple patterns and their joins can be computed during the partitioning process without materialization of all triples and these statistics can be utilized to construct more efficient query plans.
6. Our distributed reasoning framework will perform better with rich queries involving complex reasoning and more number of join triple-patterns than the queries involving weak axioms and fewer triple patterns.

In table 1, we list the metrics and the datsets that we plan to use in our evaluation. Since there is a dearth of synthetic and real-world datasets that have large-scale data graph with very expressive DL axioms and realistic queries that can be used for testing our framework, we would like to explore the creation of synthetic datasets with tunable expressivity and queries with wide range of properties with respect to number of conjuncts, selectivity and diameter of the query graph. We would also like to augment some real world datasets with handcrafted DL axioms to test our framework.

Table 1. Metrics and Datasets

Metrics
With respect to partitioning: partitioning time, the total number of partitions, the size of resulting partitions (minimum, median, mean, maximum), and the standard deviation of partition size. *With respect to distributed reasoning*: query response time, utilization of compute nodes, amount of communication.
Datasets
Synthetic Datasets: Lehigh University Benchmark (LUBM)[17], University Benchmark (UOBM) [18] *Real-world datasets and ontologies*: DBPedia, Yago, Barton, DBLP, LinkedMDB, and Linked Life Data, Billion Triple Challenge datasets.

Preliminary Evaluation Results and Lessons Learned: In our preliminary framework [2], we conducted the experiments on LUBM data (LUBM-50, LUBM-100 and LUBM-200) to evaluate performance of the partitioner and found that the partitioning system scales well. It was possible to create very small partitions (for LUBM-200 with 27.6M triples, the largest partition had fewer than 6800 triples). Our experiments on the preliminary parallel reasoning framework using up to 32 nodes demonstrates significant parallelism, with 32 nodes being 3.5x faster than 8 nodes (see figure 1). The speedups fall short of embarrassing parallelism, mostly due to the setup cost (time spent distributing the query to the slaves) and performing join to get the final answers. More details on this evaluation can be found in [2].

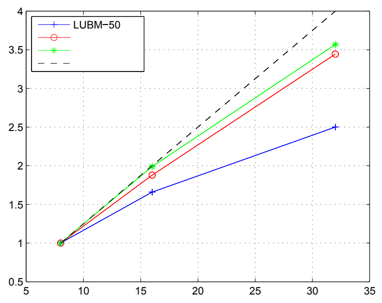

Fig. 1. Speed up for LUBM-50, LUBM-100 and LUBM-200 on 8, 16 and 32 processes.

5 Open Issues and Future Directions

Most existing scalable systems for processing conjunctive queries do not take reasoning into account. On the other hand, existing works on scalable reasoning are limited to forward chaining or reasoning on less expressive logic and do not handle conjunctive queries involving DL reasoning. In this thesis, we address both the issues by proposing a scalable and distributed framework for performing sound and complete reasoning for answering conjunctive queries over increasingly large data graphs involving expressive DL axioms. We plan to address the following core open issues in the future: a) partitioning large, highly networked and expressive datasets across the cluster for distributed reasoning, b) determining the best data allocation strategy for a mix of unknown queries such that load is evenly balanced and communication cost is minimized across the cluster, and c) assigning reasoning-aware query-plan components to the nodes for efficient processing of the reasoning and query-execution tasks involved in processing the conjunctive queries.

Stage of doctoral work: Middle.

Acknowledgement: I would like to thank Prof. Jeff Heflin (adviser) and Prof. Michael Spear (co-adviser) for their valuable comments on this paper.

References

1. Y. Guo and J. Heflin . A Scalable Approach for Partitioning OWL Knowledge Bases. In Proc. of the 2nd International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2006). Athens, Georgia. 2006.

2. S. Priya, Y. Guo, M. Spear, and J. Heflin. Partitioning OWL Knowledge Bases for Parallel Reasoning. In Eighth IEEE International Conference on Semantic Computing (ICSC 2014), Newport Beach, CA, 2014.
3. J. Huang, Daniel J. Abadi, and Kun Ren. Scalable SPARQL Querying of Large RDF Graphs. In Proceedings of Conference on VLDB. 4(11):1123-1134. 2011.
4. H. Stuckenschmidt and M. Klein. Structure-based partitioning of large concept hierarchies. In Proc. 3rd International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004.
5. A. Schlicht and H. Stuckenschmidt. A Flexible Partitioning Tool for Large Ontologies. In Proc. of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (Sydney, Australia, December 9-12, 2008). IEEE Computer Society 482-488.
6. S. Gurajada, S. Seufert, I. Miliaraki and M. Theobald: TriAD: A Distributed Shared-Nothing RDF Engine based on Asynchronous Message Passing, Proceedings of the 2014 ACM International Conference on Management of Data (SIGMOD 2014), Snowbird, UT, USA, 2014.
7. J. Urbani, F. van Harmelen, S. Schlobach, H. E. Bal: QueryPIE: Backward Reasoning for OWL Horst over Very Large Knowledge Bases. International Semantic Web Conference (1) 2011: 730-745
8. E. Oren, S. Kotoulas, G. Anadiotis, R. Siebes, A. Teije, and F. van Harmelen. MARVIN: A platform for large scale analysis of Semantic Web data, In: Proceedings of the WebSci'09: Society On-Line, 18-20 March 2009, Athens, Greece.
9. J. Weaver and J. Hendler. Parallel materialization of the finite RDFS closure for hundreds of millions of triples, In Proceedings of the ISWC '09, 2009.
10. J. Urbani, S. Kotoulas, J. Maassen, F. Van Harmelen and Henri Bal. OWL reasoning with WebPIE: calculating the closure of 100 billion triples, Journal of Web Semantics, Vol 10, 2012.
11. R. Mutharaju, P. Hitzler, P. Mateti, and F. Lcu. Distributed and Scalable OWL EL Reasoning. In Proceedings of the 12th Extended Semantic Web Conference, Portoroz, Slovenia, To Appear, 2015.
12. Franz Inc. AllegroGraph RDFStore Benchmark Results, 2014. http://franz.com/agraph/allegrograph/agraph benchmarks.lhtml.
13. T. Neumann and G. Weikum. Scalable Join Processing on Very Large RDF Graphs. In Ugur C etintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul, editors, SIGMOD Conference, pages 627640. ACM, 2009.
14. A. Harth, J. Umbrich, A. Hogan, and S. Decker. YARS2: A Federated Repository for Querying Graph Structured Data from the Web. In ISWC/ASWC '07, Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference, LNCS, volume 4825, pages 211224. Springer, 2007.
15. B. Selman and H. Kautz. Knowledge Compilation Using Horn Approximations. In Proceedings of Ninth National Conference on Artificial Intelligence (AAAI 1991), 1991, 904-909.
16. M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds. SPARQL basic graph pattern optimization using selectivity estimation. In Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008, pages 595604. ACM, 2008.
17. Y.Guo and Z.Pan and J.Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics, 3(2), 158-182, 2005.
18. L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan and S. Liu. Towards a complete OWL ontology benchmark. In Proceedings of the 3rd European conference on The Semantic Web (ESWC'06), pages 125-139, Springer, 2006.