Scalable and parallel reasoning in the Semantic Web

Jacopo Urbani

Department of Computer Science, Vrije Universiteit Amsterdam,
j.urbani@few.vu.nl

Abstract. The current state of the art regarding scalable reasoning consists of programs that run on a single machine. When the amount of data is too large, or the logic is too complex, the computational resources of a single machine are not enough. We propose a distributed approach that overcomes these limitations and we sketch a research methodology. A distributed approach is challenging because of the skew in data distribution and the difficulty in partitioning Semantic Web data. We present initial results which are promising and suggest that the approach may be successful.

1 Problem statement

Most of the current reasoners are programs that are executed on a single machine. The scalability of these approaches is limited by the physical resources of the single machine. The size of the Semantic Web has grown to the point where this limitation notably affects the performance of the reasoners. Therefore, in order to realize the vision of Semantic Web, a scalable and efficient way to reason over an ever-growing amount of data is crucial.

The scalability is typically evaluated on two aspects: computational *complexity* (i.e. the ability to perform more complex tasks) and input *size* (i.e. the ability to process a larger input). It is essential that the reasoning process is *scalable* regarding both aspects. The research question we pose is:

> We aim to research algorithms which allow complex and scalable reasoning over a large amount of data (billions of statements).

The reasoning results should be accessed with interactive queries. In this context we identify two subtasks: reasoning and querying. Depending on the type of reasoning, these two tasks can be either independent from each others or strongly interlinked. When possible, we intend to put more emphasis on the task of reasoning than on the task of querying, but both tasks are important because if the results of the reasoning are unavailable to the end user then the entire process becomes meaningless.

2 State of the art

The scientific community has already made notable efforts for reasoning in a large scale. An exhaustive list of all the relevant work goes beyond the scope of this paper. Here, we will only report some of the work that shows our intended goal.

Currently, several single-machine Semantic Web stores support reasoning and can load up to ten billion triples[1]. However, loading the data at this scale can take up to one week and it can only scale with more powerful hardware.

Hogan et al. [3] compute the closure of an RDF graph doing two passes over the data on a single machine. They have implemented only a fragment of the OWL Horst semantics, in order to prevent ontology hijacking.

Several distributed systems were proposed to calculate the closure and querying. Mika and Tummarello [4] use MapReduce to answer SPARQL queries over large RDF graphs, but details and results are not reported.

Soma and Prasanna [7] present a technique for parallel OWL inferencing through data partitioning. The experiments were conducted only on small datasets (1M triples) with a good speedup but the runtime is not reported.

Marvin [5] presents a technique which partitions the data in a peer-to-peer network but results with very large datasets have not been presented.

In Weaver and Hendler [12] incomplete RDFS reasoning is implemented on a cluster replicating the schema on all the nodes. This approach is embarrassingly parallel and it cannot be extended to more complicated logic like OWL.

Schlicht and Stuckenschmidt [6] present a promising technique to parallelize DL reasoning with a good speedup but the performance was evaluated on a small input.

Some of the approaches here presented have good scalability but on a weak logic ([12]), while others implement a complex logic like OWL but do not appear to scale to a very large size ([7],[3]). The approach we propose in the next section aims to research the best tradeoff between scalability and the complexity of richer logics so that we are able to perform complex but feasible reasoning over billions of statements (web-scale).

3 Proposed approach

Our purpose is to reason over a very large amount of data which cannot be handled by a single machine. To overcome this problem, we propose a distributed approach where the reasoning task is executed simultaneously on several independent machines. We assume that we have complete control of the machines in which the data is processed.

We will consider only monotonic rule-based reasoning. The motivation behind this choice lies on several considerations:

– in the Web, the data is distributed and we cannot retract existing facts;

[1] `http://esw.w3.org/topic/LargeTripleStores`

– there are already some standardized rule sets (RDFS, OWL Horst, OWL 2 RL) that are widely used and which allow us to compare our work with already existing one;
– currently there is no distributed rule-based reasoner which implements a complex logic on a very large scale.

A distributed approach is potentially more scalable than a single machine approach because it can scale on two dimensions: the hardware and the number of the machines. However, it is more challenging to design because:

– In the Semantic Web, the data is not uniformly distributed, but instead there is an high data skew which generates load balancing problems;
– In rule-based reasoning the data must be joined together many times in order to find a possible derivation. Doing joins over distributed data is one of the most challenging tasks in a distributed environment.

Our approach aims to limit the exchange of the data (which is expensive) and try instead to move the computation (which is cheap) because rule based reasoning is mainly a data intensive task. There are several distributed programming models, like MapReduce [1] or Satin [11] which reduce the data communication and efficiently distribute the computation.

In some cases, there are additional technical problems introduced by a distributed approach. For example, the nodes must communicate to each other using an efficient protocol, otherwise the performance will suffer. For our work, we intend to use the Ibis framework [10] to ease the development of our approach. The Ibis framework offers many tools like IPL [2] or JavaGAT [9] which handles many technical aspects of the communication between the nodes and the heterogeneity of the system.

In our context, rule based reasoning can be applied either in a forward way or in a backward way. In forward reasoning the algorithm first materializes all the closure and then the data is queried in a database fashion. This approach is ideal when we have a dataset which does not change frequently and we need to query it extensively. It can be problematic if the closure is very large or if the data changes too frequently.

In backward reasoning the derivation is calculated on the fly when the data is queried. This approach is generally more complex and it makes the queries much slower, however it has the advantage that it works if the data changes often or if the complete closure is too large to materialize.

Our purpose is to apply the current programming models to the different types of reasoning (forward, backward, or a combination of the two) and to research what model is the most efficient and under which conditions. In case none of the existing programming model suits well for our purpose, our research will try to design a new programming model which overcomes the limitations of the existing ones.

4 Methodology

Our research will be carried out in several phases. In the first phase, we will study the existing literature and research some programming models which fit our need.

The second phase consists of implementing a reasoning algorithm using these specific programming models. We will first pick one programming model and we will research an efficient algorithm which implements a weak logic (i.e. RDFS) using that model.

In the third phase we will evaluate the performance of the algorithm (the tests will be conducted mainly on the cluster DAS-4[2]). If the algorithm has good performance then we will increase the complexity of the logic, otherwise we will focus on the problems of the algorithm. If the problems are in the nature of the programming model, then we will try another model which could solve these problems (this means return to phase 1). If all programming models do not suit our problem, then the research will aim in finding a new programming model which fits our requirements and solve the problems that came up previously.

It is fundamental to clarify what we mean with "good" performance. Keeping in mind that our goal is to query the reasoning results in an interactive way, the evaluation will consider the total runtime necessary to reason and query the data. Another very important aspect in the evaluation is the scalability. The algorithm must be able to work with different input sizes and different number of nodes and it should implement a relatively complex logic (at least OWL-Horst or higher).

5 Results

Research was conducted to implement forward reasoning using the MapReduce programming model. At first the work focused on the RDFS semantics and several optimizations were introduced to speed up the computation. The performance showed linear scalability and an high throughput, but the work could not be extended to more complex logic because the optimizations exploited some specific characteristics of RDFS [8].

Next, the research aimed to see whether MapReduce could be used also to implement the more complex OWL reasoning. Other optimizations were introduced and the results of this work are currently under submission for the conference ESWC 2010.

The main limitation of the work done so far is that the reasoning cannot be applied only on a subset of the input. In a realistic scenario where the dataset is gradually extended with new data, computing the closure every time over all the input is not efficient.

The next step in our research consists of finding a way to query the derivation obtained by the reasoner. In order to do so, the results can be loaded on a database or an RDF store and queried with the standard techniques.

[2] `http://www.cs.vu.nl/das/`

6 Conclusions and future work

In this paper, we have described the problem of scalable reasoning and proposed a distributed approach to solve it. The results we have obtained with forward reasoning and MapReduce showed that indeed this approach may be successful but further research is necessary.

In our approach, we made some assumptions to narrow down the scope of the research to distribution and performance. We do not consider other important aspects of reasoning on a large scale like, for example, the quality of the data. Some work about it is presented in [3] and it could be integrated in our reasoning algorithms to make them more robust. Another assumption we make is that we have control of the machines in which the data is processed. If the data comes from the Web it must be first fetched and copied locally. Some extra work could extend the reasoning process to work directly on the distributed data.

References

[1] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the USENIX Symposium on Operating Systems Design & Implementation (OSDI)*, pages 137–147, 2004.

[2] N. Drost et al. Resource tracking in parallel and distributed applications. In *Proc. of 17th IEEE International Symposium on High-Performance Distributed Computing (HPDC)*, 2008.

[3] A. Hogan, A. Harth, and A. Polleres. Scalable authoritative OWL reasoning for the web. *International Journal on Semantic Web and Information Systems*, 5(2), 2009.

[4] P. Mika and G. Tummarello. Web semantics in the clouds. *IEEE Intelligent Systems*, 23(5):82–87, 2008.

[5] E. Oren et al. Marvin: A platform for large-scale analysis of Semantic Web data. In *Proceedings of the International Web Science conference*, 2009.

[6] A. Schlicht and H. Stuckenschmidt. Distributed Resolution for Expressive Ontology Networks. In *Proceedings of the Web Reasoning and Rule Systems 2009*, page 87. Springer, 2009.

[7] R. Soma and V. Prasanna. Parallel inferencing for OWL knowledge bases. In *International Conference on Parallel Processing*, pages 75–82, 2008.

[8] J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen. Scalable distributed reasoning using mapreduce. In *Proceedings of the ISWC '09*, 2009.

[9] R. V. van Nieuwpoort, T. Kielmann, and H. E. Bal. User-friendly and reliable grid computing based on imperfect middleware. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, 2007.

[10] R. V. van Nieuwpoort et al. Ibis: a flexible and efficient Java based grid programming environment. *Concurrency and Computation: Practice and Experience*, 17(7-8):1079–1107, June 2005.

[11] R. V. van Nieuwpoort et al. Satin: Simple and efficient Java-based grid programming. *Scalable Computing: Practice and Experience*, 6(3), 2005.

[12] 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.