WebPIE: a Web-scale Parallel Inference Engine

Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Niels Drost, Frank Seinstra,
Frank van Harmelen, Henri Bal
Department of Computer Science, Vrije Universiteit Amsterdam
{j.urbani, kot}@few.vu.nl, {jason, ndrost, fjseins, frankh, bal}@cs.vu.nl

1 Introduction and problem statement

The Semantic Web [1] extends the World Wide Web by providing well-defined semantics to information and services. Through these semantics machines can "understand" the Web, making it possible to query and reason over Web information, treating the Web as if it were a giant semi-structured database.

Over the recent years, large volumes of data have been published in a Semantic Web format, constituting a valuable resource for researchers across several fields: in medicine, there are dozens of datasets that comprise protein sequence and functional information, biomedical article citations, gene information and more[1]. The US and UK governments are putting major effort in making public information more accessible by providing it in Semantic Web formats [2]. General knowledge extracted from Wikipedia [3] and geographical knowledge [4] is also available.

Semantic Web data is expressed in statements, also known as *triples*. Available data is quickly outgrowing the computational capacity of single machines and of standard indexing techniques. In March 2009, around 4 billion statements were available. In the following 9 months this number had tripled to 13 billion statements and the growth continues.

A statement consists of a sequence of three terms: *subject*, *predicate* and *object*. An example is:

```
<http://www.vu.nl> <rdf:type> <http://dbpedia.org/University>
```

This statement states that the concept *http://www.vu.nl* is of type *http://dbpedia.org/University*. The reason of using URIs to identify concepts is that URIs are unique on the Web while a classical name (for example "Vrije Universiteit") could introduce ambiguity.

Machines can reason using these triples, inferring new statements from the existing statements. This is usually accomplished by applying a set of rules. The most commonly used rulesets are the RDFS ruleset [3] and the OWL-horst ruleset [4], with the second being more powerful and more difficult to implement. Rules are typically applied recursively, adding the inferred statements to the input and stopping only when *the closure* is reached (i.e., no further conclusions can be derived). An example rule is:

[1] `http://www.linkedlifedata.com/` (2.7GTriples) and `http://esw.w3.org/topic/HCLSIG/LODD`
[2] `http://www.data.gov/(5.07GTriples)` and `http://www.data.gov.uk/`
[3] `http://www.dbpedia.org(450MTriples)`
[4] `http://www.geonames.org(77MTriples)`

```
if (<?p isA transitiveProperty>, <?a ?p ?b>, <?b ?p ?c>) then <?a ?p ?c>
```

Unbound variables are denoted with a question mark. To apply this rule, we must perform a three-way join between all the statements that match the patterns on the left side of the rule. For all bindings, a statement of the form given at the right side will be generated. Both the RDF and the OWL-Horst rulesets contain a dozen of these rules. In the worst case, reasoning with these rules has an exponential complexity

The current state of the art of building inference engines with these rulesets mainly consists of programs that run on a single machine. However, since the volume of Semantic Web data vastly outgrows the capacity of such centralized solutions, a parallel approach is needed. This is far from trivial, as (i) performing joins in parallel is known to be a challenging problem, (ii) it is difficult to split the data into independent partitions, (iii) the term "popularity" follows a very skewed distribution, which raises load balancing issues, and (iv) rules must be applied recursively. In the next section we we briefly explain how we solve these problems.

In this paper, we present WebPIE, a parallel reasoner based on MapReduce [2] which aims at reasoning on the scale of the Web. In section 2 we highlight some of the main features of WebPIE while in section 3 we show that WebPIE vastly outperforms the best published centralised approaches, being able to reason over one-order-of-magnitude larger datasets and achieving up to 60 times better throughput, compared to the best published approaches.

2 WebPIE: A Web-scale Parallel Inference Engine

We have developed WebPIE (Web-scale Parallel Inference Engine). WebPIE encodes Semantic Web reasoning as a set of Map and Reduce operations, tapping on the power of the MapReduce programming model [2] for performance, fault-tolerance and simplified administration. It aims at high scalability in terms of the number of processing nodes and data size. This is achieved by optimizing the execution of joins required to apply the RDFS and OWL-horst rules. More information can be found in the relevant publications [6][5]. In this paper, we will summarize some of the main features of our approach.

A MapReduce implementation of Semantic Web reasoning is complicated by the inability of MapReduce to execute data joins efficiently. In order to match a rule, the antecedents must be grouped together based on a term and processed by a single reduce function running on a single machine. The term distribution of Semantic Web statements is very skewed, therefore grouping on single terms will lead to severe load balancing problems.

In addition, the same statement may be derived on the basis of multiple inputs and by multiple rules. This leads to an excessive number of *duplicate inferences*, which has a detrimental effect on load balancing and wastes resources, rendering the system practically unusable.

Since reasoning is a recursive problem, the output has to be joined with the input and processed again, until no new statements are derived (*fixpoint iteration*).

We have addressed the above issues with the following key optimisations:

- We have improved *load balancing* by applying some rules in separate MapReduce jobs and filtering out statements that cannot fire any rules for the current job. Furthermore, observing that many rules include schema (i.e., ruleset) statements, which are limited in number, we load the schema statements in memory. In this way, it was possible to iterate over the statements in one part of the join, exploiting the streaming nature of MapReduce operations.

[5]the OWL-horst reasoning method is currently under review and the source code repository is located at https://launchpad.net/reasoning-hadoop

- We have addressed the problem of *duplicate inferences* by grouping statements by the inferred statements that they *may* produce. This makes elimination of duplicates straightforward and efficient. Furthermore, we have developed rule-specific optimisations for problematic rules. For example, to avoid generating duplicates for the rule mentioned in the introduction, we have used the graph distance to intelligently select the statements that should be included in each rule application.
- We have optimised the order of the rule application to reduce the number of required *iterations* over the dataset.

3 Performance and scalability

We have implemented a prototype using the Hadoop framework and conducted experiments to evaluate the performance of our system. Our experiments were run on 64 nodes of the DAS3 multi-cluster[6] using a gigabit ethernet interconnect. Each node was equipped with two dual-core Opteron processors, 4GB of main memory and a 250GB SATA hard drive.

In Table 1(a), we report the runtime of our prototype on five real-world datasets and a benchmark. The first three datasets are Web crawls. As such, they contain poor quality data and cannot be used for more powerful OWL-horst reasoning, as this would result in meaningless derivations. Instead, much simpler RDF reasoning is used. LDSR and Uniprot are curated datasets that have sufficient quality for OWL-horst reasoning, as does the LUBM benchmark dataset. Although more inferences are made on the data crawled from the Web, RDFS reasoning is much simpler than OWL-horst reasoning, resulting is significantly shorter runtimes.

We also notice that the execution time is not proportional on the input size. This behavior is expected because the computation complexity does not only depend on the input size, but also on how the statements are related to each other. In theory, both RDFS and OWL-horst reasoning have exponential complexity, and it would be straightforward to devise an artificial dataset that would render our optimisations ineffective. However, we have empirically found in realistic datasets that such artificial worst cases do not appear.

We have evaluated the scalability of our approach, in terms of data size, using the standard LUBM benchmark. The results are shown in Figure 1(b). For LUBM, the complexity of the generated dataset does not increase with size. The results show that our solution scales linearly up a very large input size (100 billion statements). In fact, the system performs *better* for datasets of larger size, since the platform overhead becomes lower relative to the total execution time.

The current state-of-the-art includes single machine reasoners [7] and distributed reasoners [5, 7].

Compared to the former, we have shown inference on a number of statements which is one order of magnitude *larger* than reported anywhere (100 billion triples against 12 billion triples). Furthermore, our inference is 60 times *faster* (10 billion triples in 4 hours against 12 billion triples in 290 hours for LUBM) against the best performing reasoner (BigOWLIM) on a large server. For UniProt, BigOWLIM 3.1 needs 21 hours to perform forward reasoning on 1.15 billion triples[8] (yielding a throughput of 15.2 Ktriples/sec) while our system needs only 6 hours for 1.5 billion triples (yielding a throughput of 68.3 Ktriples/sec).

There are two other distributed reasoning platforms that have shown RDFS inference in the order of hundreds of millions of triples. The MARVIN platform, which we have developed in previous work [5], and the system presented in [7]. WebPIE vastly outperforms both systems

[6] `http://www.cs.vu.nl/das3/`

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

[8] D5.5.2 at `http://www.larkc.eu/deliverables`

Dataset	Input (Triples)	Output (Triples)	Runtime (h:m:s)
RDFS			
Falcon	32.5M	863.7M	0:04:19
Swoogle	78.8M	1.50B	0:07:15
BigWeb	864.8M	30.0B	0:56:57
OWL-horst			
LDSR	862M	935M	3:31:00
Uniprot	1.5B	2.0B	6:05:49
LUBM	1.1B	496M	0:36:36

(a)

Input (Triples)	Output (Triples)	Runtime (h:m:s)	Throughput (Kt/sec)
1.07B	0.50B	0:36.36	455.2
2.14B	0.99B	0:59:40	602.6
10.71B	4.97B	4:03:37	684.6
102.50B	47.56B	45:46:12	606.8

(b)

Figure 1. (a) Reasoning performance over rulesets and datasets of varying complexity. (b) Reasoning performance over input size for the LUBM benchmark

in terms of data size and throughput. Furthermore, it supports OWL-horst inference, which is significantly more complex than the RDFS inference supported in these systems.

4 Conclusions

In this paper we have described the problem of very large scale reasoning, stressing its relevance to the Semantic Web. We have proposed WebPIE, a parallel approach based on MapReduce which vastly outperforms state-of-the-art approaches and scales linearly with input size.

We have demonstrated reasoning on Web scale up to 60 times faster and with 10 times more data than any other approach. A drawback of our system is that it does batch processing, i.e. every time we execute a rule all the input must be read. This can be improved by designing a specific data layer so that the MapReduce can read only the relevant statements. Further research is also required to evaluate how relevant the input complexity is in the calculation of the closure.

We believe that this work establishes that computing the closure of a very large dataset is no longer an important bottleneck, and that research efforts should switch to other modes of reasoning.

We can provide a live demonstration of our system as well as results on additional datasets.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

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

[3] P. Hayes, (ed.) *RDF Semantics*. W3C Recommendation, 2004.

[4] H. J. ter Horst. Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2–3):79–115, 2005.

[5] E. Oren, S. Kotoulas, G. Anadiotis, R. Siebes, *et al.* Marvin: distributed reasoning over large-scale semantic web data. *Journal of Web Semantics*, 2009.

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

[7] J. Weaver and J. Hendler. Parallel materialization of the finite rdfs closure for hundreds of millions of triples. In *8th International Semantic Web Conference (ISWC2009)*. 2009.