A Linked Data Reasoner in the Cloud

Jules Chevalier

LT2C, Télécom Saint-Etienne, Université Jean Monnet,
10 rue Tréfilerie, F-4200 France
jules.chevalier@univ-st-etienne.fr

Abstract. Over the last decade, the paradigm of Linked Data has gained momentum. It is possible to leverage implicit knowledge from these data using a reasoner. Nevertheless, current methods for reasoning over linked data are well suited for small to medium datasets, and they fail at reaching the scale of the Web of Data. In this PhD thesis, we are interested in how distributed computing in the Cloud can help a linked data reasoner to scale. We present in this paper the early state of this thesis.

Keywords: Linked Data, Reasoning, Web of Data, Cloud Computing.

1 Research Questions

As Weiser predicted, computers have weaved themselves into the fabric of everyday life so that they are now indistinguishable from it. From personal computers to cars and televisions, all of these objects are now powerful computers generating more and more information. In many applications, the Semantic Web helps in changing this information into knowledge and linking it with other pieces of knowledge on the Web. Apart from their explicit knowledge, linked data contain implicit knowledge that can be leveraged using a reasoner. Reasoning is a complex process, and current solutions aim at reasoning at the scale of the Web of Data. That is why we need more powerful reasoners, scalable enough to make inference over very large datasets. So far, distributing and parallelizing this process over a cluster of computers seems the most adapted solution. Cloud Computing appears like an interesting environment for parallel inferencing. Elasticity is a primary characteristic of the Cloud, as it is composed of more or less heterogeneous clusters of commodity servers. Actually, Cloud providers APIs make it possible to scale up and down the number of dedicated Virtual Machines (VMs) that an application needs. A large-scale reasoner is an application presenting a profile that fits Cloud Computing. It would have computation bursts when new linked data arrive, depending on the amount of data and the number of new derived triples. Once triples are derived, they could be materialized, and then the reasoner no longer needs a large amount of VMs. The research question is therefore to propose a Cloud-ready linked data reasoner, whose architecture makes it possible to reason over a large scale corpus in a distributed way, and where scalability increases (resp. decreases) dynamically as the reasoning process is running (resp. no longer running). This research question also includes bandwidth optimization, which is part of the costs to run a service in the Cloud. In the following, we present previous work about distributed and parallelized reasoning and confront them to Cloud-hosted environments.

2 State of the Art

Until now, three works held our attention. These works are representatives of current solutions for distributed inferencing.

2.1 WebPie

In WebPie[8], the distribution is done thanks to the MapReduce paradigm[1], under the Hadoop framework. MapReduce is a very efficient paradigm for batch processing. In Webpie, each inference rule is a job. Jobs are executed one after each other, but this execution is distributed over a cluster. WebPie works with two logic fragments: RDFS[1] and OWL Horst[3]. The results show that quickly, over four cores, the gain of a new core is massively decreasing, against a logarithmic curve. [7] fixes some issues that optimises the reasoner implementation, improving its performance and completeness. But despite these upgrades, the results still suffer from the same issue. [4] critics this points in details. In short, while MapReduce is a handful paradigm which allows to set a distributed system implementing only two functions and that was popularized by its Hadoop-related eco-system, it is however not very adapted for reasoning. Actually, splitting the data in hermetic cores generates duplicates and therefore introduces unnecessary loops between jobs. This implies a higher bandwidth payload to exchange more batches of triples than necessary. Convergence is longer to reach, in a non linear way, as the number of triples increases. Although no theoretical evidence are provided, in practice a threshold close to four VMs limits the scalability.

2.2 MapResolve

[5] highlights the main drawback of the MapReduce paradigm : each *worker* must wait every other's end. This obviously slows down the computation speed, and decreases the project performance. Inspired by WebPie and other MapReduce works, they propose a reasoning solution over more expressive logic fragments. Despite their extensions, this follow-up proposal fails to provide significant improvements over WebPie in terms of performance.

2.3 Parallel Inferencing for OWL Knowledge Bases

For partitioning inference, we have two solutions: split the rules (that is what WebPie and MapResolve do), or split data. [6] proposes three methods to split data: graph partitioning, hash partitioning, or domain-specific partitioning, and

[1] A final recommendation from the W3C, `http://www.w3.org/TR/rdf-schema/`

a last technique to split rules depending on the rule-dependency graph. The authors also propose a parallelized reasoning algorithm based on existing reasoners. Unlike MapReduce-based solutions, data are not randomly splitted, with the aim to avoid duplicates and core communications. Unfortunately, these optimisations are not sufficient. Data are partitioned into hermetic cores, which still generates loops and duplicates.

2.4 Analysis

Among the three approaches we studied, approaches to build concurrent reasoners are divided into two categories :

Distributed : Both WebPie and MapResolve are based on MapReduce, which is a framework for distributed computing. Data partitioning in [6] is a distributed approach.

Parallel : In the case of rule-partitioning, [6] proposes a parallel approach.

In distributed computing each computing unit has its own private memory whereas in parallel computing all computing units access a shared memory. Due to the very own nature of the reasoning process, where rules can be interdependant, i.e. a directed graph, data cannot be splitted to be processed independently, which is a requirement of the MapReduce paradigm. To circumvent this issue, authors of the three approaches try to split data in order to minimize the overhead that will be implied by reprocessing data after a graph update (which occurs at each inference). This introduces loops and an overhead of bandwidth consumption that prevent scalability with more than half a dozen nodes. Surely the momentum gained by MapReduce for a few years, and the ease of implementation have oriented the authors towards this approach. The state of the art solutions do allow to handle more linked data than a single node could have done before. However, we believe that parallel processing could be an interesting paradigm to foster large scale linked data reasoning.

3 Proposed Approach

After studying existing solutions, we have initiated some features of our own solution for the case of a Cloud-hosted linked data reasoner.

1. Shared memory for a full parallel solution
2. Sort axioms by relevance instead of existing fragments
3. Stream compliant reasoner

3.1 Parallel Processing : Shared Memory

The main difficulty to design a parallel reasoner over the cloud is to efficiently implement a shared memory among numerous VMs. This problem, of the utmost practical interest, has been tackled by several approaches, especially for the Java language. Solutions such as Jelastic, Terracotta, HazelCast, Coherence present features that could be suitable to implement a concurrent reasoner.

3.2 Axioms Sorted by Relevance for the Web of Data

All axioms of description logic are not used with the same frequency. Figure 1 presents a rank of logic axioms as actually used in practise on the Web of Data. Using this histogram, instead of reasoning over defined fragments of description logic, we first reason over the most used axioms, to fit the use made by the Web of Data. This method favours the most-used concepts instead of grouping them by fragments.

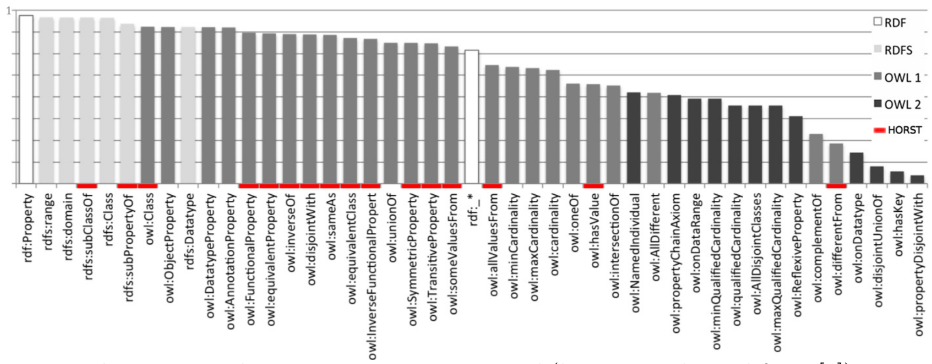

Fig. 1. Logic description axioms PageRank(histogram derived from [2])

A parameter **n** determines how many fragments are taken into account. We would be able to optimize **n** with respect to the time of reasoning that is acceptable for the application that requires the reasoning.

3.3 Stream-Based Architecture

Our last targeted feature is the ability to fire new triples to other nodes as soon as they are created by a rule. This prevents the use of the MapReduce paradigm to split the load. Instead of waiting the entire process of a MapReduce batch of data, each newly created axiom would be sent to all nodes that is expected to leverage new knowledge from this triple. This configuration tends to be as close as possible to pseudo real time reasoning. It could also be more compatible with incremental reasoning strategies of stream-oriented applications to be in semantic sensors networks. It implies to be able to connect the node in the architecture with respect to the dependency graph if rules in the considered linked data logic fragment. This depends on **n**.

4 Planned Research Methodology and Schedule

Deploy WebPie and Reproduce Results. The first point will be to deploy a WebPie version in our private Cloud. Thanks to this, we will be able to reproduce [8] experiments on different datasets, from the smallest to the biggest. This will let us have a baseline to compare our own experimentation results.

Propose and Implement Our Stream Reasoner. The second step will be the proposal of our reasoner. It is for now an archetype which needs to be completed. We must precise its core strategies (rule and data partition especially) and to implement it.

Compete against WebPie Results. When a first implementation will be finished, we would deploy it on our private Cloud, and run tests with the same datasets as for WebPie.

– **May 2013 -** State of the art internal report.
– **October 2013 -** WebPie deployment and tests
– **February 2014 -** Proposal of our Cloud-hosted linked data reasoner.
– **May 2014 -** First implementation of our reasoner.
– **November 2014 -** 'Stable' version deployed on our private Cloud.
– **January 2015 -** Evaluation campaign and interpretation of results.
– **April 2015 -** Writing the PhD thesis.

Acknowledgement. Frédérique Laforest, Christophe Gravier and Julien Subercaze, in charge of this thesis.

OpenCloudware, funded by the French Fonds national pour la Société Numérique (FSN), and is supported by Pôles Minalogic, Systematic and SCS.

References

1. Dean, J.: MapReduce: Simplified data processing on large clusters. Communications of the ACM, 1–13 (2008)
2. Glimm, B., Hogan, A., Krötzsch, M., Polleres, A.: OWL: Yet to arrive on the Web of Data? CoRR, abs/1202.0 (2012)
3. Ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3), 79–115 (2005)
4. Patel-Schneider, P.: Comments on webpie. Web Semantics: Science, Services and Agents on the World Wide Web 15(3) (2012)
5. Schlicht, A.: Mapresolve. Web Reasoning and Rule Systems, 1–6 (2011)
6. Soma, R., Prasanna, V.K.: Parallel Inferencing for OWL Knowledge Bases. In: 2008 37th International Conference on Parallel Processing, pp. 75–82 (September 2008)
7. Urbani, J., Kotoulas, S., Maassen, J.: WebPIE: A Web-scale parallel inference engine using MapReduce. Web Semantics: Science, 59–75 (2012)
8. Urbani, J., Oren, E.: RDFS/OWL reasoning using the MapReduce framework. Science, 1–87 (2009)