Reasoning with Large Scale OWL 2 EL Ontologies based on MapReduce

Zhangquan Zhou[1], Guilin Qi[2],
Chang Liu[3], Raghava Mutharaju[4], and Pascal Hitzler[5]

[1,2]Southeast University, China, {quanzz,gqi}@seu.edu.cn
[3]Carnegie Mellon University, United States, liuchang2005acm@gmail.com
[4,5]Wright State University, United States,
{pascal.hitzler,mutharaju.2}@wright.edu

Abstract. OWL 2 EL, which is underpinned by the description logic $\mathcal{EL}$, has been used to build terminological ontologies in real applications, like biomedicine, multimedia and transportation. On the other hand, there have been techniques that allow developers and users acquiring large scale ontologies by automatically extracting data from different sources or integrating different domain ontologies. Thus the issue of handling large scale ontologies has to be tackled. In this short paper, we report our work on classification of OWL 2 EL ontologies using MapReduce, which is a distributed computing model for data processing. We discuss the main problems when we use MapReduce to handle OWL 2 EL classification and how we address these problems. We implement the algorithm using Hadoop, and evaluate it on a cluster of machines. The experimental results show that our prototype system achieves a linear scalability on large scale ontologies.

Keywords: OWL 2 EL, description logics, classification, MapReduce

1 Introduction

Among different profiles of OWL 2[1], OWL 2 EL (EL for short), which is based on description logics $\mathcal{EL}$ family, stands out for its positive complexity results and sufficient expressivity. EL is mainly used in biomedicine. One of the most popular biomedicine ontology expressed in EL, SNOMED CT [5], is supposed to be a large scale ontology, i.e., nearly six hundred thousand axioms (or statements) are involved. Thus, to make EL play a better role in real applications, it is necessary to give efficient solutions for handling large scale ontologies. One of the most important reasoning services in EL, called *classification* [2], is the task of computing a subsumption hierarchy between concept descriptions.

In order to handle large ontologies efficiently, several works employ parallel or distributed computing techniques. ELK [3] is the first reasoner that exploits multi-core techniques to enhance the efficiency of classification in EL. Although

[1] www.w3.org/TR/owl2-overview/

the experimental results show that these in-memory reasoners can be scalable to some extent, they are restricted to the main memories of the utilized machines. On the other hand, a distributed approach based on Redis is also proposed in [4] for EL classification. This work also verifies that distributed techniques can handle classification on large scale EL ontologies.

In this paper, we report our work on classification of EL ontologies using MapReduce, which is conducted parallelly with the work [4]. We briefly discuss the main problems when we use MapReduce to handle OWL 2 EL classification and how we address these problems. We implement a prototype system and evaluate it on a Hadoop cluster. The experimental results show that our system has a linear scalability on large real ontologies. The details of this work can be found in our technical report which is available at this address[2].

2 The Problems of Performing EL Classification on MapReduce and Our Solutions

Due to the page limit, we refer the readers to our technical report for the formalism of EL and the work mechanism of MapReduce. In this part, we briefly discuss the main problems when we use MapReduce to handle OWL 2 EL classification and how we address these problems.

• **Translating ontologies and rules to MapReduce languages.** Since MapReduce programs can only handle key/value pairs. We should first translate EL axioms to key/value pairs. However this is not a trivial task when considering the issue of performance. We consider representing EL axioms using relational tables. For example, for axioms of the form $A \sqsubseteq \exists r.B$, we introduce a relational table of the form $R(A, r, B)$ where R is the name of this table, (A, r, B) is the table schema. In this way, applying classification rules can be transformed to joins of relational tables. Finally, it is relatively easy to map operations of relational tables to MapReduce programs.

• **Reducing the number of jobs.** If we use MapReduce programs to perform EL classification, several jobs are needed to apply different rules until a fix-point is reached. This delays computation due to the inherent overheads of platforms. Thus reducing the number of jobs can significantly improve running performance. We apply some optimizations to reach this goal. For example, we carefully decide rule application order. We also combine some MapReduce jobs into one.

• **Handling multi-way joins.** It is easy for MapReduce to handle a two-way join using one job. For a multi-way join, we can partition it into several two-way joins to adapt to MapReduce programs. In the case of EL classification, there are three rules which have multiple joints in their preconditions. However, it is challenging to partition it into several two-way joins and, the performance of the whole computation can also be guaranteed. We analyze and compare different partition methods. The details can be found in our technical report.

[2] https://drive.google.com/drive/folders/0ByjKIQyCPHldSDNrWnlZN3pRdTg

3 Evaluation

We implement a prototype system based on Hadoop[3], which is an open-source Java implementation of MaReduce. We conduct all the experiments in a cluster that consists of 14 nodes, and each node has a 16 Gigabyte RAM and two quad-core processors. We use two famous medical ontologies: Galen and SNOMED CT. In order to validate the scalability of our system, we evaluate it on different copies of Galen and SNOMED CT. The Galen copies are renamed by n-Galen, where n is the number of copies in n-Galen. SNOMED CT is processed similarly.

Table 1: The results of scalability tests

datasets	#input axioms(k)	#derived axioms(k)	time(min)	♯jobs
1-Galen	91	6,941	143.4	93
2-Galen	182	13,808	179.8	98
4-Galen	364	27,626	242.7	94
8-Galen	728	55,209	319.5	94
1-SCT[a]	1,151	18,980	479.5	94
2-SCT	2,302	38,795	976.8	93
3-SCT	3,453	60,715	1589.5	94

[a] abbr. of SNOMED CT.

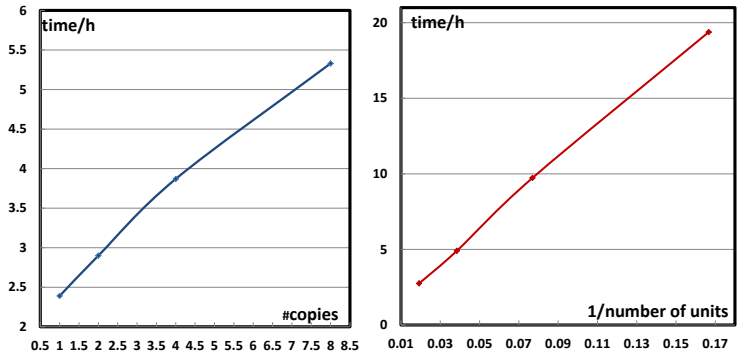

Fig. 1. Classification time on different copies and different units.

Scalability tests. We assign 52 units (a unit represents a logic core) of the cluster to perform classification on different ontology copies. The experimental results are collected in Table 1. We use Figure 1 to give a graphical representation of the relevant contents with respect to Galen in Table 1, where the curve in the left part (resp. right part) shows the relation between the classification time and the number of copies (resp. the inverse of the number of units we set to perform classification on 1-Galen). From Figure 1, we can see that the classification time is

[3] http://hadoop.apache.org/

approximately linear in the number of copies, and inversely linear in the number of units. The experiments on SNOMED CT have the similar results.

From Table 1, we can estimate the maximum speedup of the evaluated ontology based on the the view of Amdahl's Law[4] that indicates the speedup of a computation task only depends on the fraction of sequential computation part with *the assumption of infinite processors being allocated* (IPA). According to the Amdahl's Law, with IPA, the run-time of each job is $O(1)$ in theory, while the whole run-time depends on the number of jobs[5]. This helps estimate the maximum speedup to be $O(T_1/n)$, where T_1 is the run-time units of classification on one processor and n is the number of jobs (or the run-time units) on infinite processors. For example, the number of jobs for classifying 1-SNOMED ($d(\mathcal{O}_{1-SNOMED})$) is 94 and more than 18 million axioms are derived (see Table 1). Thus the maximum speedup of classification on 1-SNOMED can be estimated to be $max-speedup(\mathcal{O}_{1-SNOMED}) \geq \frac{|\mathcal{C}_\mathcal{O}|}{d(\mathcal{O}_{1-SNOMED})} (= 191,489)$. Similarly, the maximal speedup for classifying 1-Galen is greater than 70,000. It indicates that classifying these two ontologies has a high degree of parallelism. That is to say if we set more computing units to our system, it should have a higher performance.

4 Conclusions

In this paper, we reported our work on classification of EL using MapReduce. We discussed the main problems and gave our solutions. We implemented a prototype system using Hadoop. The experimental results showed that our system has a linear scalability on real medical ontologies and their copies.

References

1. G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proc. of AFIPS*, pages 483–485, 1967.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Proc of IJCAI*, pages 364–369, 2005.
3. Y. Kazakov, M. Krötzsch, and F. Simancik. Concurrent Classification of EL Ontologies. In *Proc. of ISWC*, pages 305–320, 2011.
4. R. Mutharaju, P. Hitzler, P. Mateti, and F. Lécué. Distributed and scalable OWL EL reasoning. In *Proc. of ESWC*, pages 88–103, 2015.
5. M. Q. Stearns, C. Price, K. A. Spackman, and A. Y. Wang. SNOMED Clinical Terms: Overview of the Development Process and Project Status. In *Proc. of AMIA Symposium*, pages 662–666, 2001.

[4] Amdahl argues in [1] that the speedup s of a computation task depends on the fraction of sequential computation part, i.e., $s \leq 1/(f + \frac{1-f}{p})$, where s is the speedup, f is the fraction of sequential computation part and p is the number of processors.

[5] This conclusion also holds with a polynomial number of processors, since the classification on EL ontologies is in P.