Distributed RDFS Knowledge-based System Update in Case of Deletions

Hamid Oliaei, Mahmoud Naghibzadeh *Member*, IEEE
Faculty of engineering
Ferdowsi university of Mashhad
Mashhad, Iran
oliaei.hamid@stu-mail.um.ac.ir, naghibzadeh@um.ac.ir

Abstract— **This article presents a method for maintaining the materialization of a RDFS Knowledge-Base in case of fact deletions in a distributed way. MapReduce framework is used to implement and test the method. Current MapReduce framework-based distributed reasoning is used for static RDFS knowledge bases. However, any changes in KB have been followed by rebuilding the closure which is very time consuming. The alternative is to update it. The method preserves soundness and completeness which are necessary conditions to any materialization process. The steps involved in fact deletion is recognized and a set of RDFS rules is devised accordingly to preserve correctness properties. Finally, we analyze the time and space consumed by the presented method.**

Keywords- Distributed RDFS; Knowledge base deletion maintenance; Soundness and completeness

I. INTRODUCTION

Scalable reasoning is a huge challenge in semantic web. With current large amount of data and its fast growth, even supercomputers maybe can't work efficient for reasoning on this scale. So a distributed way must be considered. For huge amount of data, computing the closure under a semantic needs distributed methods too.

Another challenge is maintaining materialization in case of changes. They can be deletion, insertion or updating some facts or rules in knowledge base. Of course, we can handle an update with a deletion and an insertion.

Between handling deletion and insertion, deletion is more important. After deletion of some facts or rules, it's completely possible that other facts or rules become incorrect. So if we can't use proper methods in case of deletions from KB, soundness may be lost.

After ontology materialization, most current distributed systems don't have any solution for changes facts or rules. In current state, after a deletion from a closure, for keeping its soundness, the closure must be rebuilt that is time consuming. In this paper we focus on fact deletion from a distributed RDFS knowledge base and found a solution for maintaining that in case of fact deletions using MapReduce framework. We use method introduced in [10] for rewriting rules for maintaining KB. We will show later that finding a way for doing the overestimation step as distributed is enough for maintaining the KB. We use MapReduce framework explained in [1] for processing the RDFS rules. We rewrites the procedures for deletion rules and find a procedure for doing the jobs correctly so after finishing the jobs we can be sure that our KB is still sound and degree of its completeness is equal to [1].

Our method's properties are:

- Using current maintenance method in a distributed way;
- Using MapReduce framework for distributed processing
- Skipping rederivation step to optimize consuming time & hard disk space

This paper as structured as follows: section 2 discuss about related works. Section 3 is about the whole Idea. Finally conclusion and future work will be discussed.

II. RELATED WORK

In scalable systems scope, Urbani et al. [1] found a way for building a closure Based on RDFS. From distributed aspect our work uses their ideas and algorithms. They extended their work to OWL/Horst KBs in [2] and add incrementally maintenance of materialization to the system in [3]. Oren et al. [4] use a data partitioning technique based on triple common terms. Their idea's problem is high overhead and redundancy. Weaver et al. [5] use a parallel algorithm for materializing the complete RDFS closure. They consider RDFS characteristics. They define classes of RDFS rules and use ABox Partitioning. Su et al. [6] have demonstrated a logic for large scale data with combination of OWL with Horn clauses called β-PSML. Verstichel et al. [7] break concepts in two parts including contents and services to present an OWL-based meta-model for distributed reasoning using data partitioning. Fensel et al. [8] in a project called LarKC, achieved scalability through parallelization and giving up completeness. They reason on part of data and then decide what to do. Schlicht et al. [9] presented a sound and complete distributed method for ontologies using ordered resolution. They implemented their work on $\mathcal{ALC}$ models.

In maintenance of materialization scope, Volz et al [10] use a technique for management of dynamic update using rewritten rules and maintenance programs. We borrow our deletion rules notations from them. Stuckenschmidt et al. [11] presented a notion called distributed description logics for modular ontologies based on logic *SHIQ*. They presented a method for handling simple changes and update the ontology. DeGiacomo et al. [12] presented an algorithm for