IMPROVING THE WORKFLOW OF SEMANTIC WEB PORTALS USING M/R IN CLOUD PLATFORM

Seokchan Yun, Mina Song, Hyun Namgung, Sungkwon Yang, Harshit Kumar and Hong-Gee Kim
Biomedical Knowledge Engineering Lab, Seoul National University, Seoul, Korea

Keywords: Semantic Web Portal, RDF Data Processing, Cloud Computing, M/R.

Abstract: Semantic Web Portals (SWPs) provide web services supporting searching, sharing and exchanging of information using semantic web techniques. The pre-existing SWP construction workflow based on current RDF store has limited scalabilities for processing the large volumes of semantic data. In this paper, we propose M/R (M/R) based modules usable in each step (e.g., data storing, reasoning, and accessing) of the workflow to reduce overall processing time and cost. The proposed modules lesson burdens of each step by exploiting an M/R cluster which is easily enlargeable with use of a cloud computing platform.

1 INTRODUCTION

Recently, expanding use of semantic web derives an evolution of Semantic Web Portals (SWPs) (Lei, 2006). The SWPs are yet in its infancy, and, there could be several different definitions (Lausen, 2005) (Staab, 2000). We concentrate on some features of the SWPs that are based on semantic web technologies, and support searching, sharing and exchanging of information (Hartmann, 2003). Because of the limited scalability of currently developed semantic web data management systems, accomplishing the efficient workflow for building a SWP with a large volume of semantic dataset has some difficulties (Guo, 2005). There are some recent research for solving scalability issues that are embodied with an employment of massive computation environments like M/R (Dean, 2008), supercomputers, and parallel computing. But, such approaches are committed to a specific task only like a reasoning (Urbani, 2009) (Weaver, 2009).

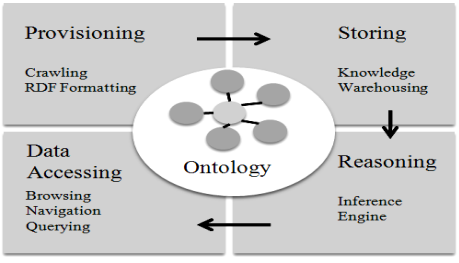

Figure 1: SWP Construction Workflow.

We focus on improving a general workflow of the SWPs with employment M/R execution environment which is easily usable with use of a commercial cloud computing service. Our ongoing project is devoted to provide improved methods to archive each step of the workflow. We assume this work will help creations of SWPs by lessening the burdens to build them.

2 THE WORKFLOW OF SWPS

Firstly let us depict a general picture of SWPs. Although there could several different points of view, we might find a general workflow of them based on the definition of a SWP as information providing service based on semantic web technology.

Figure 1 shows usual steps for a SWP construction. The SWP are equipped with a set of conceptual basis given through in form of ontology that provides the background knowledge and supports the presentation of information with semantics with in formalism. The workflow for the construction of SWP has often limited to the size of a data collection to be serviced. We can assume a large size of data set that hardly be handled in a single semantic data store with a real-time.

Between Providing and Storing: During provided data set (written in RDF/OWL formats) is stored into data storage, the storage usually creates indexing structure for each single data unit, e.g.,

Yun S., Song M., Namgoong H., Yang S., Kumar H. and Kim H..
IMPROVING THE WORKFLOW OF SEMANTIC WEB PORTALS USING M/R IN CLOUD PLATFORM.
DOI: 10.5220/0003142104850488
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2010), pages 485-488
ISBN: 978-989-8425-29-4

RDF triples. The indexing techniques improve the system's data storing and retrieving capabilities, but it also increase storing time of the data. Some benchmarks show that the data storing of several million triples takes several hours to even days by a type of RDF storages (Bizer, 2009).

Between Storing and Reasoning: The main burden of here will be large amount of time spending for reasoning from the stored data, and, that could be partially solved by M/R based Reasoning approaches (Urbani, 2009). The other point we are focused here is duplication of the same resource which have different identifier. Large size of data from distributed sources deteriorates the situation enlarging reputation rate. The reputation increases time and complexity of the Reasoning work.

Between Reasoning and Accessing: The answering time of current data storages is also quite insufficient, especially for large size of data set. Almost every storages sometime fails to deliver an answer for a quite complex query within a stipulated response time like few seconds. The late response time makes the SWPs are barely responsible to user's demand within guaranteed stability. The benchmark also shows that the query time is very long by the type of RDF storages (Bizer, 2009)

3 FILLING GAPS WITH M/R

In this section, we describe solutions for filling above gaps through M/R programming of which nature is proper to pre-processing work on very huge size of data in the SWP construction work. Also each proposed modules are developed to solving problems of data loading, duplication of resource and late query answering for supporting a real-time portal service that is equipped with RDF storages.

3.1 Improving Data Storing

When semantic data set are provided, a store takes them into a pre-processing phrase. Then, it parses and splits incoming data written as a set of RDF/OWL statements into atomic data units, RDF triples which are collections of subject, verb and object sometimes with the source of the triple (namely a graph or domain). Data storage stores the triples with employed index systems, e.g., B tree and Bitmap, for providing fast accesses on each RDF triple data.

Providing the data set cannot guarantee the pro-processed for the storing phrase. Then, randomized sequences of triple incomings will increase a time for index construction. Parsing the RDF/OWL statement is also a reason of late data storing. Therefore we can be noticed that the time spends of storing work can be reduced by incoming of previously parsed and sorted triples in advance.

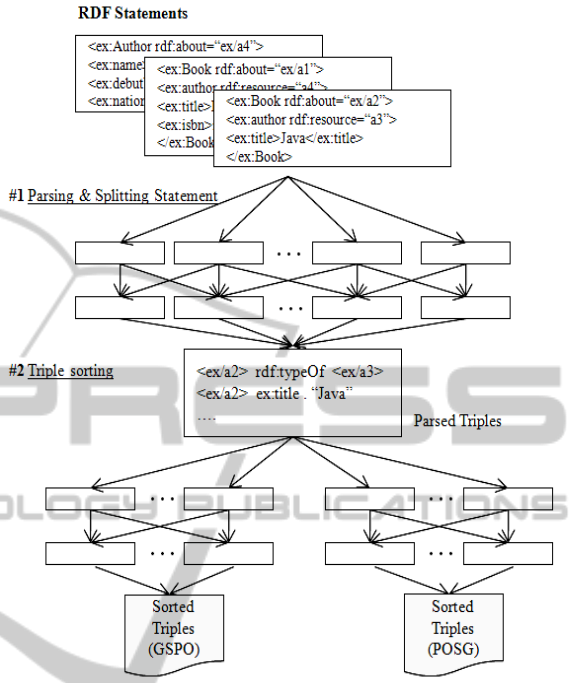

Figure 2: The M/R module for 'Before Storing'.

Therefore, our first modules will deliver a set of sorted triple assertions before Storing work. Figure 2 depicts M/R program which fulfils the requirement. When a set of RDF Statement are provided, the first M/R job parses each triples and split it into triple assertions written in a simple expression schema like N-triple. Then, each Mapper of the second M/R work takes the spited triples as values of Key-Value pair. The Key of each triple is decided by the required sequence of the triples set, for example, GSPO (Graph-Subject-Predicate-Object). The nature of the M/R framework makes the Key selection can deliver the sorted triples in corresponding to the sequence at the end of execution.

3.2 Improving Data Reasoning

For the Reasoning work, duplications of the same data resource are increasing an overload of the work. Various data sources of the service can promises large service usability and plentiful information, but, they also accompany with reputations of the same data (Resource) even if they share the same ontology. Resource reputations which mean entity

semantically same with different identifiers are frequently occurring among the information resources. Information integration removing the resource reputations reduces a reasoning overhead by relieving reputations of data and complexity of data graph.

Therefore, the second module of our work supports the information integration based on M/R. Table 1 includes an example which shows how the information integration is possible through M/R. Before execution of M/R, bold-style values are designated as a key whereas the statements become values of Key-Value pairs of M/R. Each key for each statement is marked to support a simple characteristic based integration strategy (Naumann, 2006). The same data resource holding the same key which consists of Class and some key features are sent to the same reducer to be treated as a resource.

Table 1: Key Selection for Information Integration.

Map (Key: ex:name+ ex:debutYear)

```
<ex:Author rdf:about="ex/a3">
 <ex:name>Harshit</ex:name>
 <ex:debutYear>1988</ex:debutYear >
  <ex:nationality>UK</ex:nationality>
</ex:Author >
<ex:Author rdf:about="ex/a4">
  <ex:name>Harshit</ex:name>
  <ex:debutYear>1988</ex:debutYear>
  <ex:publish rdf:resource="ex/a6" />
</ex:Author >
```

Reduce

```
<ex:Author rdf:about="ex/a4">
  <ex:name>Harshit</ex:name>
<ex:debutYear>1988</ex:debutYear>
<ex:nationality>UK</ex:nationality>
 <ex:publish rdf:resource="ex/a6" />
 </ex:Author >
<ex:Author rdf:about="ex/a4">
 <owl:sameAs rdf:resource="ex/a3" />
</ex:Author >
```

3.3 Improving Data Accessing

The final gap filling we are dealing with are formed as query pre-processing. If we consider the query answering time of current semantic data management system, actual user accessing need be directed to an index, or cache server not the data store. Therefore, this module provides a very fast SPARQL answer preparation supporting a quick build of the index server.

The preparation of SPARQL can be invoked by a seed query which shows reflex a user interface of SWP. The pre-processing phase takes as input a queries and generates a collection of answer set which will serve as an index for seed queries and also for a set of queries semantically similar to seed queries, termed as extended query. The execution of M/R will deliver answer sets for those seed and extended queries which cover semantic extension of possible query in a SWP user interface.

We can make the workflow for the answer set derivation with four M/R executions and a simple seed query as following:

```
Select ?title WHERE {
?b rdf:typeof  ex:Book
?b ex:title  ?title}. )
```

Each RDF file goes to one mapper; output from the 1st M/R is a set of triples in turtle format. Let the set of triples be termed as (1).

$$T = \{t_1, t_2, \ldots . t_n\} \quad (1)$$

Each triple t_i has turtle format (s,p,o). The output from 1st M/R goes as input to 2nd M/R which generates C(n,k) number of triples, called as tuples. The tuple file further goes to 3rd M/R that filters and produces a reduced set of tuples, called as reduced tuples. The last and 4th M/R partition the reduced set of tuples and generate partitioned answered sets.

The number of tuples from 2nd M/R depends on the number of triple patters in the WHERE clause of seed query. If there are k triple patterns in the WHERE clause of seed query, each t_i in T is concatenated with other (n-1) triples with no repetition in order, producing a tuple of length k. This set is termed as (2) and * is a concatenation operator. This step was carried out to compute join of tuples.

$$T_{ktuple} = <ti * tj * \ldots * tk\text{-}1> \quad (2)$$

Furthermore, 3rd M/R eliminates unrelated tuples from T_{ktuple} producing a set of proper tuples, let this set be R_{ktuple}. A proper tuple is a one in which subject (t_1) == subject (t_2)…==subject (t_k) or object (t_1) = subject (t_2)… and so on. We call the set R_{ktuple} as reduced set because set R_{ktuple} has less number of tuples compared to T_{ktuple}.

When an example seed query has 2 triple patterns in the WHERE clause, the 2nd M/R will create C(n,2)=n*(n-1)/2 tuples from T. The example in Figure 2 has n=13, cardinality of T_{2tuple} will be 78. The tuples in set T_{2tuple} are further filtered by 3rd M/R using the following condition: if subject (t_1) == subject (t_2), which will result in x number of tuples in R_{2tuple}.

Now that we have a reduced tuple set R_{2tuple}, 4th M/R partitions it and indexes each partitioned answer set based on attributes in the WHERE clause

of SPARQL query and semantically similar attributes. Semantically similar attributes are generated by the M/R module. Each partitioned answered set is stored in a separate file with unique key value for direct access.

4 USE CASE - SEMANTIC MUSIC SERVICE

This section describes the example of SWP construction with our proposed M/R execution modules. We developed a semantic music service and the data navigator browsing a music metadata with a FLEX based user interface. When user types a keyword, it can be accessed in musical entities such as artist, album and song and some metadata including artist's birth day and release date of album or song as Fig.3.

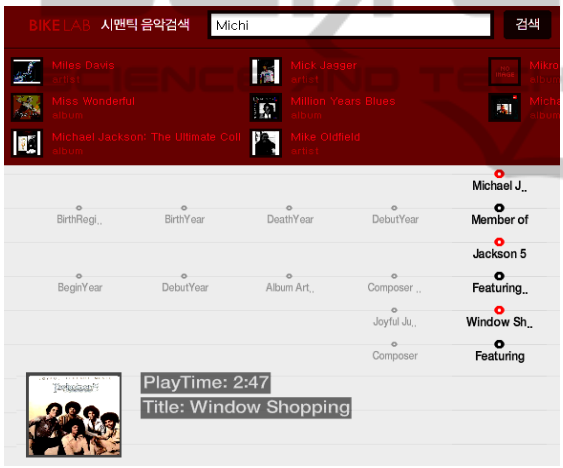

Figure 3: Semantic music service.

In this implementation, we applied only improved data storing process to reduce processing time of making RDF triple indexes. Firstly, we gathered a music database supplied by KBS (Korea Broadcasting System), an open API crawling from madiadb.com database and RDF formatted Musicbrainz dataset. We covered 8 million songs from MusicBrainz and 1 millon from KBS and ManiaDB, 0.5 million artist profiles and 1 million albums including each relationships data.

From these dataset, we made over 10 million RDF files based on the simple music ontology (http://wiki.musicontology.com) and generated over 200 million triples using ten Hadoop instances of the iCube cloud(https://www.icubecloud.com) platform made by NEXR in South Korea.

In the future, we will implement proposed modules for helping reasoning and data accessing steps and evaluate our workflow compared with pre-existing methods. Also, the other important issues involving massive computation like pre-raking of answer set will be dealt with in this project.

ACKNOWLEDGEMENTS

This work was supported in part by MKE & KEIT through the Development of Independent Component based Service-Oriented Peta-Scale Computing Platform Project.

REFERENCES

Lei, Y., Uren, V., Motta, E., 2006. SemSearch: A Search Engine for the Semantic Web, *In proceeding of European Semantic Conference 2006.*

Hartmann, J.; Sure, Y.; Volz, R.; Studer, R., 2003. Extended OntoWeb.org Portal, *OntoWeb Deliverable 6.4*

Lausen, H., Ding, Y., Stollberg, M., Fensel, D., Hernández, R., Han, S., 2005. Semantic web portals: state-of-the-art survey Export, *Journal of Knowledge Management*, Vol. 9, No. 5, 2005.

Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H., Studer, R., Sure, Y., 2000. Semantic Community Web Portals. *The International Journal of Computer and Telecommunications Networking archive*. Volume 33, Issue 1-6, 2000.

Bizer, C., Schultz, A., Berlin SPARQL Benchmark, 2009 (BSBM), *http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark*

Dean, J., Ghemawat, S., 2008. MapReduce: Simplified data processing on large clusters, *Communications of the ACM*, v.51 n.1, January 2008.

Guo, Y., Pan, Z., Heflin., J., 2005. LUBM: A Benchmark for OWL Knowledge Base Systems, *Journal of Web Semantics 3*(2), 2005.

Urbani, J., Kotoulas, S., Oren, E., Harmelen, F., 2009. Scalable distributed reasoning using MapReduce. *In proceeding of ISWC 2009.*

Weaver, J., Hendler, J., 2009. Parallel materialization of the finite RDFS Closure for Hundreds of Millions of Triples, *International Semantic Web Conference 2009.*

Naumann, F., Bilke, A., Bleiholder, J., Weis, M., 2006. Data fusion in three steps:Resolving schema, tuple, and value inconsistencies, *IEEE Data Eng.* Bull., 2006.