Knowledge Tags Repository

Karol RÁSTOČNÝ*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
rastocny@fiit.stuba.sk

Semantics in the Web is necessary for computer systems, e.g. for recommending. For this reason, they process documents on the Web and assign knowledge tags, short metadata (e.g. keywords) to documents' parts. By sharing of these knowledge tags, new layer of the lightweight semantics over the Web can be created collaboratively. But systems currently store these metadata to their private repositories in a form which is understandable only for them, so other systems could not use these metadata and they have to do this work redundantly. This inspired us to propose knowledge tags maintenance approach (Figure 1), which allows software tools to share their metadata.

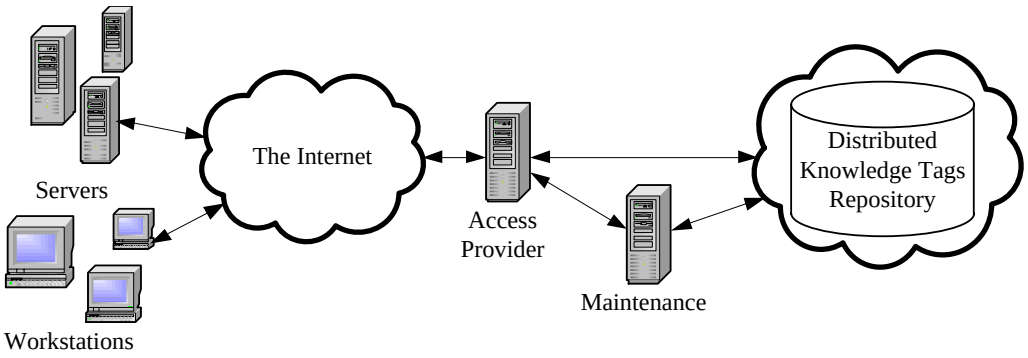

Figure 1. Architecture of proposed knowledge tags maintenance approach.

Knowledge tags repository is one of core parts of proposed maintenance. This repository has to store large amount of knowledge tags in flexible open format which has to be understandable for computer systems and the repository has to still provide fast parallel access for a numbers of systems. As knowledge tags model we utilizes existing Open Annotation (OA) model[1]. We decided for this model because of it is

* Supervisor: Mária Bieliková, Institute of Informatics and Software Engineering
[1] http://www.openannotation.org/spec/beta

already accepted by wide range of systems and knowledge tags and annotations have common characteristics. Both of them are anchored to specific parts of documents and they contain small information on these documents' parts. This model is based on RDF and it is highly recommended to implement it by RDF triple databases and support at least basic access (e.g. querying by SPARQL) with RDF serialization output format.

To provide effective and powerful data access, we analyzed standard use cases of annotation repositories and itemize list of requirements which respect these use cases and specific requirements of OA model and maintenance part of proposed method. We have found up, that manipulation with whole knowledge tags is important for almost all use cases. But this is in disagreement with RDF triple databases, which have good deduction possibilities but they have serious issue with obtaining complete information about an object, when several simple queries have to be processed and each query can take several seconds in large datasets [3]. Document databases are in correlation with our need of access to whole knowledge tags, while they store documents (in general objects) as one item and not sparse over several tables or collections. This allows fast access to whole objects without necessity of time expensive joins.

We concretely decided for MongoDB, which matches requirements: it provides efficient data access (loading and updating) and supports distributed data processing via MapReduce [1]. But it does not provide support for SPARQL query processing. This is implementable via MapReduce, but existing approaches are proposed for Apache Hadoop [2], which has some differences in processing of Map and Reduce fazes and works with RDF triples databases. Our algorithm for distributed SPARQL query processing firstly builds optimal joining tree, to minimalize count of necessary join operations. MapReduce uses ordered list of join attributes with their values as a key and list of deduction objects which consists of ordered list of joined pattern ids and ordered list of attributes from patterns with their values as a result. MapReduce phase is executed iteratively over remaining layers of optimal joining tree. Map function of iteration emits for each result from previous iteration new result with a key from current joining attributes and same value. Reduce function creates result as Cartesian product of results with same keys. The last finalize function removes from results deduction objects, which do not have complete list of patterns mapped to join keys.

Acknowledgement. This work was partially supported by the Slovak Research and Development Agency under the contract No. APVV-0208-10.

References

[1] Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. *Commun. of the ACM*. (2008), vol. 50, no. 1, pp. 107-113.

[2] Kim, H.S., Ravindra, P., Anyanwu, K.: From SPARQL to MapReduce: The Journey Using a Nested TripleGroup Algebra. In: *Proc. of the VLDB Endowment* 4. VLDB Endowment, Inc., (2011), pp. 1426-1429.

[3] Rohloff, K. et al.: An Evaluation of Triple-Store Technologies for Large Data Stores, In: *Proc. of the 2007 OTM Confed. Int. Conf. on the Move to Meaningful Internet Systems, LNCS 4806*. Springer, Berlin, (2007), pp. 1105–1114.