Accepted Manuscript

Impact analysis of data placement strategies on query efforts in distributed RDF stores

Daniel Janke, Steffen Staab, Matthias Thimm

PII: S1570-8268(18)30009-X
DOI: https://doi.org/10.1016/j.websem.2018.02.002
Reference: WEBSEM 456

To appear in: *Web Semantics: Science, Services and Agents on the World Wide Web*

Received date : 13 June 2017
Revised date : 18 December 2017
Accepted date : 15 February 2018

Please cite this article as: D. Janke, S. Staab, M. Thimm, Impact analysis of data placement strategies on query efforts in distributed RDF stores, *Web Semantics: Science, Services and Agents on the World Wide Web* (2018), https://doi.org/10.1016/j.websem.2018.02.002

Impact Analysis of Data Placement Strategies on Query Efforts in Distributed RDF Stores☆

Daniel Janke[a,*], Steffen Staab[a,b,*], Matthias Thimm[a,*]

[a] *Universität Koblenz-Landau, Institute for Web Science and Technologies, Universitätsstr. 1, 56070 Koblenz, Germany*
[b] *University of Southampton, Web and Internet Science Group, Building 32, Highfield Campus, SO17 1BJ Southampton, United Kingdom*

Abstract

In the last years, scalable RDF stores in the cloud have been developed, where graph data is distributed over compute and storage nodes for scaling efforts of query processing and memory needs. One main challenge in these RDF stores is the data placement strategy that can be formalized in terms of graph covers. These graph covers determine whether (a) the triples distribution is well-balanced over all storage nodes (storage balance) (b) different query results may be computed on several compute nodes in parallel (vertical parallelization) and (c) individual query results can be produced only from triples assigned to few — ideally one — storage node (horizontal containment). We analyse the impact of three most commonly used graph cover strategies in these terms and found out that balancing query workload reduces the query execution time more than reducing data transfer over network. To this end, we present our novel benchmark and open source evaluation platform Koral.

Keywords: Distributed RDF stores, graph partitioning, benchmark

1. Introduction

In the last years, the requirement for RDF stores that can cope with several trillions of triples has emerged. For instance, the number of Schema.org-based facts that are extracted out of the Web have reached the size of three trillions [2]. Another example is the European Bioinformatics Institute (EMBL-EBI) that would like to convert its datasets into RDF resulting in a graph consisting of several trillions of triples. To date no such scalable RDF store exists and the current EBI RDF Platform can handle only 10 billion triples [3].

We pursue the development of a scalable RDF store in the cloud, where graph data is distributed over compute and storage nodes for scaling efforts of query processing and memory needs. The main challenges to be investigated for such development are: (i) strategies for data placement over compute and storage nodes, (ii) strategies for distributed query processing, and (iii) strategies for handling failure of compute and storage nodes. In this paper, we focus on comparing the performance of data placement strategies.

Strategies for data placement may be formalized in terms of graph covers. Each compute and storage node hosts a graph chunk. Each triple is assigned to (at least) one graph chunk and the union of all graph chunks define a (possibly redundant) graph cover. When a query is requested to an RDF store in the cloud, the query is distributed over the different compute and storage nodes. Each node applies the query operators assigned to it on its local data. If the query requires the combination of data from different chunks, the required information has to be transferred between compute nodes.

One graph cover strategy commonly used is the hash cover that assigns triples to compute and storage nodes according to the hash value of, e. g., their subject (e. g., used by Virtuoso Clustered Edition [4], YARS2 [5, 6], Clustered TDB [7] and Trinity.RDF [8]). In order to reduce the number of transferred intermediate results, hierarchical hash has been proposed as an extension of the hash cover strategy that computes the hash only on IRI prefixes [9]. Another commonly used graph cover strategy is the minimal edge-cut cover that assigns vertices to similarly-sized partitions in a way that the number of edges connecting vertices assigned to different partitions is minimised (e. g., used by [10–12]). Fur-

☆This paper extends the 6 page workshop paper [1].

*Corresponding author
Email addresses: danijank@uni-koblenz.de (Daniel Janke), staab@uni-koblenz.de (Steffen Staab), thimm@uni-koblenz.de (Matthias Thimm)