Efficient Processing of RDF Data Using Hadoop

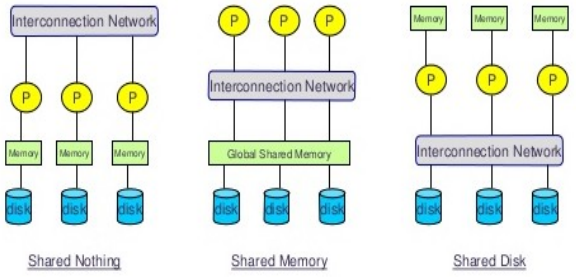
Tigran Shahinyan
Institute for Informatics and Automation Problems
Yerevan, Armenia
e-mail: tigran.shahinyan@gmail.com

ABSTRACT
Semantic Web concepts and technologies are getting wide acceptance among different groups interested in interoperable knowledge representation and reasoning. With emergence of cloud technologies for distributed storage and processing of huge amounts of data reasoning of large volumes of RDF data is becoming possible. We are using the MapReduce paradigm with Jena framework for representing and processing distributed semantic data in RDF format. In this paper we represent techniques of using Elephas library and its extensions and Hadoop framework to support efficient processing of RDF data stored in distributed storage.

Keywords
Distributed computing, mapreduce, semantic web.

1. INTRODUCTION
One of the challenges brought by the development of information technologies is the processing of huge amounts of data. The necessity for their efficient storage processing using the computing power of the Internet, computer grids and cloud system caused the emergence of non-relational data processing paradigms and technologies. MapReduce is a paradigm introduced by Google and implemented by several vendors [1]. One of the most widespread implementations of the MapReduce paradigm is the open source Apache Hadoop [2].
The Semantic Web is an extension of the current one, in which information is given a well-defined meaning, better enabling computers and people to work in cooperation. The most important technologies for making Semantic Web possible are Resource Description Framework(RDF), in which meaning is expressed in sets of subject-predicate-object triples, Universal Resource Identifier (URI) is used for identifying resources, and XML is used as one of the possible ways of document serialization and exchange[3].
Our goal is the efficient usage of both technologies and approaches to achieve knowledge storage and processing in modern heterogeneous computing environments.

2. DISTRIBUTED DATA PROCESSING
There are two main approaches for distributed data storage and processing. Parallel relational database systems seek to improve performance through parallelization of various operations, such as loading data, building indices and evaluating queries (involving selection, projection, aggregation and join operations).
Parallel databases are based on one of the following architectures[4]:



Shared nothing architecture is now most often chosen due to availability of cheaper and easier to maintain hardware of computer clusters, grids and clouds.
Parallel databases improve processing and input/output speeds by using multiple CPUs and disks in parallel. In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially. Parallel databases are mainly focused on high performance query processing, including database queries and transactions that make use of parallelism techniques applied to an underlying parallel computing platform in order to achieve high performance [5].
There are two measures of parallel database performance:

$$\text{Speed up} = \frac{\text{elapsed time on a single processor}}{\text{elapsed time on multiple processors}},$$

$$\text{Scale up} = \frac{\text{volume of processed data in a time interval on a single processor}}{\text{volume of processed data in the same time interval on multiple processors}}$$

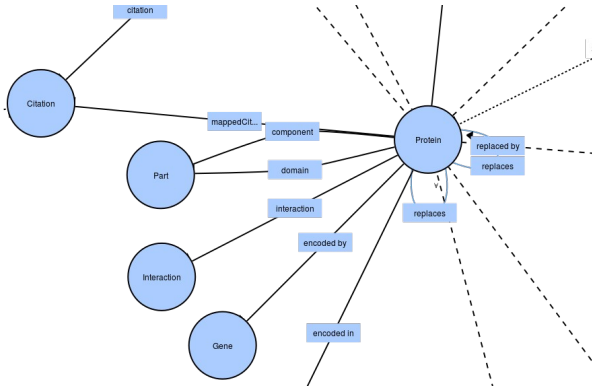
The volume of processed data may be either the number of transactions or the number of records in the database.
Scalability issues have lead to non-relational approaches of data modeling.
MapReduce is a paradigm introduced by Google for processing huge datasets on certain kinds of distributable problems using a large number of computers (nodes), a cluster or a grid. It is based on two functions: Mapper and Reducer, which are used as arguments by higher-order functions Map and Reduce [1]. On the "Map" step the master node takes the input, partitions it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes that smaller problem, and passes the answer back to its master node. On the "Reduce" step the master node then takes the answers to all the sub-problems and combines them in some way to get the output – the answer to the problem it was originally trying to solve. The advantage of MapReduce is that it allows for distributed processing of the map and reduction operations. Provided each mapping operation is independent of the

others, all maps can be performed in parallel – though in practice it is limited by the data source and/or the number of CPUs near that data. Similarly, a set of 'reducers' can perform the reduction phase.. While this process can often appear inefficient compared to algorithms that are more sequential, MapReduce can be applied to significantly larger datasets than "commodity" servers can handle. The key feature of MapReduce is Scalability. That's why we've chosen it as an alternative to relational database approach. The most important drawbacks of the MapReduce paradigm is the lack of indexes and query optimization. Parallel relational database systems show high performance especially in homogeneous environments [6], whereas the most important advantages of MapReduce are fault tolerance and the ability to operate in heterogeneous environments [7]. We've chosen Hadoop since it is one of the most widely used and supported open-source implementations of MapReduce[2].

3. DATA REPRESENTATION IN RDF

The Resource Description Format (RDF) and OWL (Web Ontology Language) are used to represent information resources modeled as directed labeled graphs, where edges represent the named link between two resources, represented by the graph nodes [8]. Resources and their properties are identified with URI references.

One of the widely used solutions for processing RDF data is Apache Jena. A SQL-like declarative language is used to query RDF graphs called SPARQL query language [9]. Semantic web is penetrating many areas of information processing and exchange activity. An example is UniProt, an effort to create a comprehensive catalog of protein data in RDF [10].The image below displays the graphical interpretation of a part of the protein ontology:



With SPARQL queries the UniProt knowledge base allows to retrieve any information about proteins, for example the following query will return the preferred gene name and disease annotation of all human entries that are known to be involved in a disease:

```
SELECT ?name ?text
WHERE{
        ?protein a up:Protein.
        ?protein up:organism taxon:9606.
        ?protein up:encodedBy ?gene.
        ?protein up:annotation ?annotation.
        ?gene skos:prefLabel ?name.
        ?annotation a up:Disease_Annotation .
        ?annotation rdfs:comment ?text}
```

4. PROCESSING OF DISTRIBUTED RDF DATA

We have tried to process a distributed RDF graph from UniProt catalog by using Hadoop as an implementation of MapReduce. The data was stored in HDFS distributed file system in line-based NTriples. For processing the data we have used Jena with its Elephas library, which provides Hadoop InputFormat and OutputFormat implementations for RDF[11]. It covers all RDF serializations that Jena supports and extensions by custom formats.

Elephas splits and parallelizes processing of input where the RDF serialization allows it.

We have used and extended various reusable basic Mapper and Reducer implementations covering the following common tasks: counting, filtering, grouping, splitting, transformation.

For example, filtering tasks allow rewriting SPARQL queries similar to the one above to filter resources satisfying given criteria by sequentially applying filters on different predicates of resources, to retrieve needed protein records. Using Elephas on large distributed networks show significant improvement of processing efficiency.

The future work will be the development of automatic converter of SPARQL queries into MapReduce tasks to be executed on Jena Elephas.

REFERENCES

[1] Jeffrey Dean, Sanjay Ghemawat, MapReduce: A Flexible Data Processing Tool, Communications of the ACM, Volume 53 Issue 1, January 2010

[2] Apache Hadoop, https://hadoop.apache.org/

[3] The Semantic Web, Berners-Lee, Tim; James Hendler; Ora Lassila Scientific American Magazine, May , 2001

[4] David DeWitt, Jim Gray, Parallel database systems: the future of high performance database systems, Communications of the ACM, Volume 35 Issue 6, June 1992

[5] David Taniar, Clement H.C. Leung, Wenny Rahayu, Sushant Goel, High Performance Parallel Database Processing and Grid Databases, 2008, John Wiley & Sons

[6] A. Pavlo, A. Rasin, S. Madden, M. Stonebraker, D. DeWitt, E. Paulson, L. Shrinivas, and D. J. Abadi. A Comparison of Approaches to Large Scale Data Analysis. In Proc. Of SIGMOD, 2009.

[7] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi,A. Silberschatz, A. Rasin, HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads, VLDB 2009

[8] W3C Semantic Web Activity, http://www.w3.org/2001/sw/

[9] Michael Grobe, RDF, Jena, SparQL and the “Semantic Web” SIGUCCS’09, October 11–14, 2009, St. Louis, Missouri, USA.
[10] Universal Protein Resource. http://www.uniprot.org/.
[11] Apache Jena Elephas Documentation Page,https://jena.apache.org/documentation/hadoop/