HadoopRDF : A Scalable RDF Data Analysis System

Yuan Tian[1], Jinhang DU[1], Haofen Wang[1], Yuan Ni[2], and Yong Yu[1]

[1] Shanghai Jiao Tong University, Shanghai, China
{tian,dujh,whfcarter}@apex.sjtu.edu.cn
[2] IBM China Research Lab
niyuan@cn.ibm.com

Abstract. As data on the public web has been expanded rapidly and more data is represented in the form of RDF as for its advantages, the scalability problem while dealing with it becomes important. To store and retrieve the RDF data, triple stores such as Sesame, 4store, have been researched. However, these tools are designed on a single computer and the ability to handle the scalability is limited. Hadoop is an open source platform for distributed computing and has been widely used for large-scale data analysis.
In this paper, we propose a system, HadoopRDF, to combine the two using both advantages. HadoopRDF is built on a hadoop cluster with many computers, and echo node in the cluster has a sesame server to supply the service of storing and retrieving the RDF. Well, data partition while loading the data, query optimization while being executed, and its optimization evaluation model are concerned in the system. With the computing ability of the cluster, we can do complex analytical tasks in large-scale data set. Several special tasks are designed to express the capability of the system.

1 Introduction

Nowadays, many data has been expressed in the form of RDF, for the reason of its advantages. RDF is a standard format defined by W3C, which makes data represented in a structural way and easily accessed by computers. Well, many triple stores have been designed to store the kind of data and supply the service of retrieving. The most famous are Sesame, Jena, 4store, and so on. Well, these triple store are all designed on the single computer, and the ability to handle the scalability is limited. So we need to expend the tools.

Hadoop is an open-source software for reliable scalable, distributed computing. Many large-scale data analysis tasks have been done on this platform. Using a cluster with lots of computers, Hadoop provides a powerful computing ability to handle the scalability well. Hadoop consists of two layers, the data storage layer or the Hadoop Distributed File System (HDFS) and the data processing layer or the MapReduce Framework.

HDFS is a block-structured file system. Files are cut into blocks and stored in the cluster nodes which are called datanode. And there is a namenode which contains metadata about the size and the location of the blocks.

MapReduce is a framework for the computing in Hadoop, which follows master-slave architecture. Each job is broken into map jabs and reduce jobs, which are executed in the slave nodes.

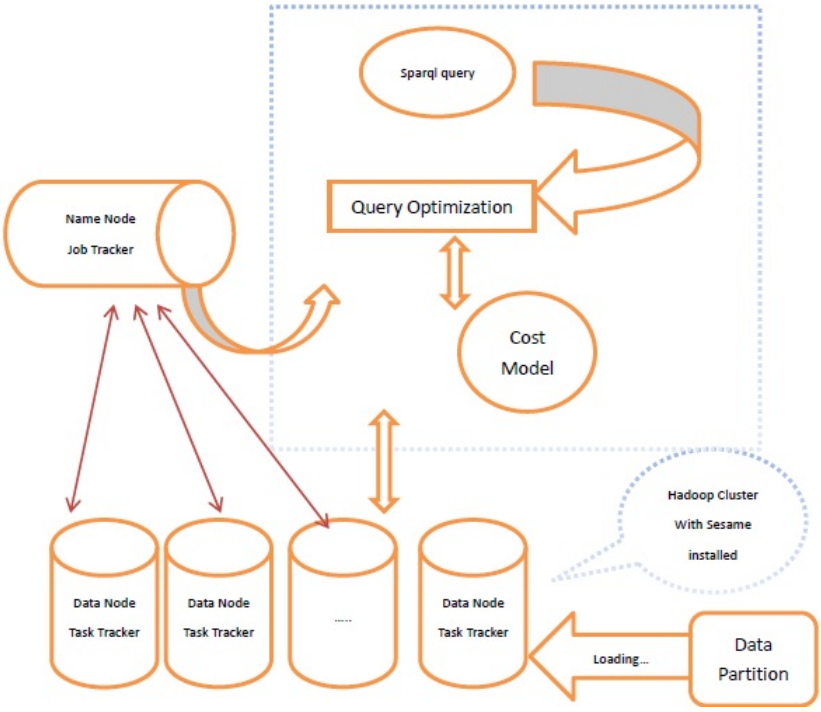
HadoopRDF combines both of the systems. Hadoop is used as the base architecture. A cluster is arranged by hadoop which is considered as the communication and controlling tool. All the jobs are ordered and arranged by the internal rules in hadoop cluster. RDF triple store is placed in the slave nodes in the cluster which supplies the basic storage and retrieves needs.

To make the methods effective, several special parts should be considered during the design of the system.

2 Our Approach

In order to address the problem of offering a scalable data intensive computing environment, we propose the HadoopRDF, which combines both advantages of high fault tolerance and high throughput of the MapReduce distributed framework and the sophisticate indexing and query answering mechanism. HadoopRDF is based on a modification to the Hadoop open source project. We connect the single RDF data stores under the MapReduce framework. We rely on the logic of Hadoop scheduling and tracking jobs to attain high fault tolerance. The idea here is to substitute the rudimentary Hadoop Distributed File System (HDFS), i.e. without indexes and a query execution engine, with more elaborated RDF stores.

2.1 Architecture



HadoopRDF has a architecture similar to Hadoop that theoretically scales up to thousands of nodes. The core of HadoopRDF is the Hadoop framework.

Hadoop is built on top of HDFS. HDFS is a replicated key-value store under the control of a central NameNode. Files in HDFS are broken into chunks fixed size and the replica of these chunks are distributed across a group of DataNodes. The NameNode keeps track of the size and location of each replica.

Hadoop is a MapReduce framework for computational propose in data intensive applications. It is a batch processing system conceptualizing an operation unit as a job. A job consists of two steps - the map task and the reduce task. These two tasks are distributed and run on a group of TraskTrackers. A JobTracker is designated to schedule the jobs and maintain an optimal throughput.

As stated, we aim at incorporating the consummate index and the powerful computational ability of current RDF stores into the MapReduce framework. HadoopRDF extends the Hadoop framework in the following perspectives.

Communication with RDF store The RDF store connector is the interface between the underlying RDF store and the TaskTrackers. The communication between the RDF store and the task processor is in SPARQL. The task processor feeds the RDF store with SPARQL queries and the RDF store returns results in the form of key-value pairs. Hence, RDF stores act as the HDFS providing a key-value store for RDF data.

Metainformation The meta-data of the chunks are kept in the NameNode. The function of the metainformation is two-fold : 1) parameters for connection between TaskTrackers and RDF store; 2) the information, including how the data is partitioned and located, of the replicas in each DataNode. This info is retrieved by the JobTracker and TaskTrackers to make up schedules and queries.

2.2 Data partitioning

When HadoopRDF loads an RDF graph, it partitions and distributes the data onto DataNodes. The hasher reads in the raw RDF graph stored in the HDFS and repartitions them according to the characteristics of the RDF stores.

The hashing function is specifically designed to adapt to the RDF data. It ensures the optimal load balancing when executing the MapReduce job.

2.3 Query Optimizer

HadoopRDF provides a SPARQL query interface.

The query planner extends the Hive query planner. The Hive query planner transforms HiveQL, a variant of SQL, into MapReduce jobs. The generated MapReduce jobs correspond to the relational operators such as SELECT, JOIN, TABLE SCAN and AGGREGATION.

The HadoopRDF query optimizer makes use of the relational operators in Hive and generates a query plan according to the optimal performance of current resources. The input SPARQL query will be first parsed into an abstract syntax tree. The query optimizer retrieves the metadata for each subgraphs stored in the NameNode. The plans are then analyzed and evaluated under a cost model. Finally, the optimal MapReduce job will be selected as the plan. The job will then be assigned to the TaskTrackers.

3 Application

The Billion Triple Challenge 2010 contains 1,441,499,719 triples which is commensurate to the 2009 dataset.

3.1 Mining Job

We make use of the SPARQL to make statistics on the current RDF graph.

4 Appendix

4.1 Minimal Requirements

HadoopRDF loads the Billion Triple Challenge 2010 dataset and provides users with a SPARQL query interface. We give various sample queries that utilize

the global information of the whole RDF graph. In these queries, we show the capability of HadoopRDF with high throughput and high fault tolerance.

The main motivation of our application is to offer the community with a solution building a distributed RDF query engine. The query engine theoretically scales up to hundreds of nodes.

4.2 Additional Desirable Features

HadoopRDF scales up to hundreds of nodes with the Hadoop architecture.