Query Optimization in Big Data Using Hive

Mrs. Hemalata Chavan
MCA (Eng.)
Pratibha College of Commerce & Computer Studies, Chinchwad, Pune India
m_hemalata2000@yahoo.com

Mrs. Aparna Joshi
M.Sc. (CS)
Pratibha College of Commerce & Computer Studies, Chinchwad, Pune India
aajoshi2@gmail.com

Abstract - The size of data sets being collected and analyzed in the industry for business intelligence is growing rapidly. This makes traditional warehousing solutions expensive. Hadoop is a popular open-source map-reduce implementation which is being used in companies like Yahoo, Facebook etc. to store and process extremely large data sets on commodity hardware. But, the map-reduce programming model is very low level and requires developers to write custom programs which are hard to maintain and reuse. Modern database technology uses Hive for analysis of large dataset. Hive is a framework for querying unstructured data as if it were structure. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy. Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. The traditional SQL queries must be implemented in the MapReduce Java API to execute SQL applications and queries over a distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like Queries (HiveQL) into the underlying Java API without the need to implement queries in the low-level Java API. Since most of the data warehousing application work with SQL based querying language, Hive supports easy portability of SQL-based application to Hadoop.

Keywords: Big Data, Query Optimization, Hive, HiveQL.

I. INTRODUCTION

The size of data sets being collected and analyzed in the industry for business intelligence is growing rapidly. This is making traditional warehousing solutions very expensive. Hadoop is a popular open-source map-reduce implementation which is being used as an alternative to store and process extremely large data sets on commodity hardware. However, the map-reduce programming model is very low level and requires developers to write custom programs which are hard to maintain and reuse. Hive is an open-source data warehousing solution built on top of Hadoop. Hive supports queries expressed in a SQL-like declarative language - HiveQL, which are compiled into map-reduce jobs executed on Hadoop. HiveQL supports custom map-reduce scripts to be plugged into queries. The language includes a type system with support for tables containing primitive types, collections like arrays and maps, and nested compositions of the same. The underlying IO libraries can be extended to query data in custom formats. Hive also includes a system catalog, Hive-Metastore, containing schemas and statistics, which is useful in data exploration and query optimization.

II. HIVE DATABASE

In Hive data is organized into Tables, Partitions, and Buckets.

a. Tables: In Hive, the basic structure in the database is a table, analogous to the tables in relational database. All the tables in hive database have an associated HDFS directory. Tables which are external to Hive and lie in a different file system like NFS or local directories are also supported.
b. Partitions: Each table can have one or more partitions in Hive. The data is distributed within the Hadoop File System under sub directories.
c. Buckets: The data in partitions is further distributed as buckets. The division in buckets is based on the hash of column in a table. Each bucket is stored as a file in the partition sub-directories.

HIVEQL:

HiveQL is a query language similar to SQL. It is used to organize and query the data stored in Hive. In the current version, HiveQL makes it possible to CREATE and DROP tables and partitions, a table split into multiple parts on a specified partition key, as well as query them with SELECT statements. The most important functionalities that are supported through the SELECT statements in HiveQL are

- the possibility to join tables on a common key,
- to alter data using row selection techniques
- to project columns.

These functionalities are analogous to functionality provided to the user in a relational database system. A typical SELECT statement in HiveQL would for example look like this:

SELECT o_orderkey, o_custkey, c_custkey
FROM customer c JOIN
orders o **ON** c.c_custkey = o.o_custkey **JOIN**
lineitem l **ON** o.o_orderkey = o.o_orderkey;

In this specific example, a simple join between the tables customer, order and lineitem is initiated on their respective join keys, which are custkey and orderkey. The projections in the first part of the statement are pushed down to the table scan level by the framework. Hive also supports the execution of multiple HiveQL statements during one operation, parallelizing as many tasks as possible. The example also shows that HiveQL statements with multiple tables require specific columns on which those tables are joined. In general, there are no cross products possible in Hive as commonly supported in database management systems such as DB2 or Oracle.

III. HIVE ARCHITECTURE

Hive is an open-source data warehousing solution that is built on top of the Hadoop MapReduce framework. Hive consists of different components that interact in order to build a data warehouse on top of the MapReduce framework.

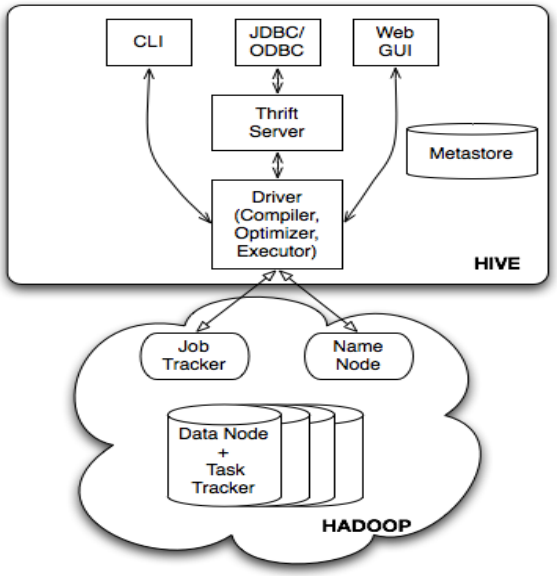

Figure 1:Hive Architecture

The most important components of the Hive architecture are:

- **User Interface:** The user interface for users to submit queries and other operations to the system. Currently the system has a command line interface where the user can enter HiveQL queries.
- **Driver:** This component receives the queries and retrieves the session specific information for the other components.
- **Compiler:** This component parses the query into different query blocks and expressions. It also generates the query plan and optimizes it to generate an execution plan.
- **Metastore:** This component stores the metadata on the different tables and partitions.
- **Execution Engine:** This component executes the plan generated by the compiler. The plan is a DAG of stages. The execution engine manages the dependencies between these different stages of the plan and executes on the appropriate system components.

Basically, the user connects to the user interface and executes a HiveQL command, which is sent to the driver. The driver then creates a session and sends the query to the compiler which extracts metadata from the Metastore and generates an execution plan. This logical plan is then optimized by the query optimization component of Hive and translated into an executable query plan consisting of multiple map and reduce phases. The plan is then executed by the MapReduce execution engine consisting of one job tracker and possibly several task trackers per map and reduce phase. Hive stores the tables created by the user either as textfile or it accesses the data through interfaces with external systems. Examples for interface systems are PostgreSQL and MySQL. It is also possible to use files as input via an external table functionality, which allows users to avoid time-intensive data imports. The Metastore of Hive then stores information about the tables like the number of partitions or the size of a table which can be used for query optimization.

IV. CONCLUSION

All the big data analytical tools are open source. Hadoop provides parallel query executing system. It can handle fault efficiently. Its attractive feature is capacity of handling heterogeneous data and dealing large amount of data for query optimization. Hive system fits the low level interface requirement of Hadoop perfectly. It supports external tables which make it possible to process data without actually storing in HDFS. It has a rule based optimizer for optimizing logical plans. Hive supports partitioning of data at the level of tables to improve performance. In Hive, Metastore or Metadata store is a big plus in the architecture which makes the lookup easy. There are certain disadvantages of hive such as No support for update and delete. It does not support for singleton inserts. Data is required to be loaded from a file using LOAD command. No access control implementation. In hive Correlated sub queries are not supported.

REFERENCES

1. The Apache Software Foundation. Hadoop MapReduce. http://hadoop.apache.org/.
2. V. P. Mahatme, Yogeshwary Sarode, Shital Radke, *Big Data Analytics Tools: A Review* International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
3. The Apache Software Foundation. Hive Wikipedia.http://wiki.apache.org/Hadoop/Hive/.
4. Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy, Hive – *A Petabyte Scale Data Warehouse Using Hadoop*.
5. The Apache Software Foundation. Hive. http://hive.apache.org/.