Tajo: A Distributed Warehouse System for Large Datasets

Sagar Vora[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*

May 3, 2017

There has been an increase in the volume of relational data generated today through a large number of sources. The large volume of data forces us to find solutions which can cope with them. Recently several hybrid approaches like HadoopDB, Hive, etc have been introduced to handle this large data. Although these have been successful in handling large data, but their architecture makes them inefficient to handle suboptimal execution strategies. Therefore, in order to solve the above problem, Apache has developed Tajo, a relational, distributed data warehouse system on large clusters. It uses Hadoop Distributed File System (HDFS) for storing data and has its own query execution engine instead of the MapReduce framework.

Keywords: HadoopDB, Hive, Distributed, MapReduce, Cluster, Query processing

https://github.com/cloudmesh/sp17-i524/raw/master/paper1/S17-IR-2041/report.pdf

1. INTRODUCTION

In this Big Data era, Hadoop [1] MapReduce [2], has been used for processing large-scale data sets. To handle a large amount of data, several hybrid approaches have been integrated with Hadoop and parallel databases. However, these cannot avoid the choice of suboptimal execution strategies because of their architecture, which led to the development of Tajo. Tajo [3] is a relational, distributed data warehouse system which runs on shared-nothing clusters. It uses Hadoop Distributed File System (HDFS) as the storage layer and has its own query execution engine instead of the MapReduce framework. A Tajo cluster consists of one master node and a number of workers. The master is responsible for the query planning and coordination among the workers.

The internals of Tajo has three main steps:-

1. Each worker has a local query engine that executes a directed acyclic graph (DAG) of physical operators. A DAG of operators can accommodate multiple sources of input and can be pipelined within the local query engine. Each worker generates query execution plan that can employ the existing query evaluation technique [4] [5] which lie in the database community.

2. Tajo can make use of various repartition methods specialized for specific queries. Consider joining two relations which are already sorted on the join key, Tajo needs to repartition only one relation to workers in which the corresponding part of another relation resides.

3. In Tajo, a physical plan that a worker executes is generated at runtime according to the available resources like memory, processing capability, etc of the workers. Workers can simultaneously execute different physical plans in the same phase which allows Tajo to maximize the utilization of the resources.

2. ADVANTAGES OF TAJO OVER MAPREDUCE HIVE

The limitations [6] of Hadoop MapReduce-Hive [7] technology due to its architecture led to the development of Tajo. These limitations are as follows:-

1. Single Source Input: The join operation in relational data warehouses integrates heterogeneous data sets from multiple sources, but MapReduce supports only a single input source, the join operation is performed by dividing input relations through one map reduce job only which doesn't produce optimal results.

2. Fixed Data Flow: The 3 phases of MapReduce which are the map, shuffle, and sort and reduce. The join operation and aggregation are performed in shuffle and sort phase. So it always follows a fixed execution flow leaving no room for any optimization needed in the processing. Sometimes processing huge datasets need to follow some hybrid map reduce flow which is not applicable to the Hadoop MapReduce framework.

3. Separate Storage: Some hybrid approaches using the database layer require separate data storages for data distribution and processing. The partitioned data must be loaded into the database layer for performance benefits. In Hadoop, it takes a long time for data loads causing overhead other running workloads in the cluster because of a separate data storage layer on the HDFS.

Tajo's system architecture has been designed in such a way to overcome the above shortcomings.

3. SYSTEM ARCHITECTURE

Figure 1 shows the system architecture of Tajo.

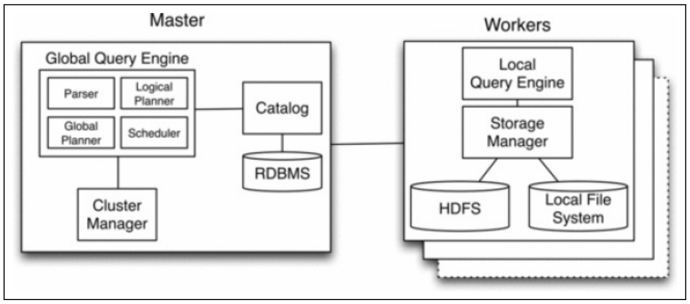

Fig. 1. [6] System Architecture of Tajo

A Tajo cluster consists of one master node and a number of workers. They are connected to one another via high-speed network. Tajo uses HDFS as the storage layer. HDFS consists of one name node (master) and a number of data nodes. A Tajo worker and HDFS data node can run on the same physical machine.

3.1. Storage Layer

Tajo uses Hadoop Distributed File System (HDFS) as the basic storage layer. The data in HDFS is distributed automatically among the cluster nodes. A local query engine of each worker scans data sets on HDFS. Tajo takes input data from HDFS and later outputs the results back into HDFS. Tajo even has a local file system and its storage manager provides interfaces to the operating system. The physical operators can process data sets on either HDFS or on local file system.

3.2. Master

The master is responsible for planning queries and coordinating the activities of workers. The master includes four components, cluster manager, catalog, global query engine, and history manager. All cluster nodes report their resource information like the number of available processors, memory usages, and remaining disk spaces periodically to the cluster manager. The catalog maintains various metadata, such as tables, schemas, partitions, functions, indices, and statistics. Since the metadata are frequently accessed by the global query engine, they are stored in a conventional RDBMS [8]. The global query engine builds a global plan based on the metadata of tables and cluster information which are provided from the catalog and the cluster manager respectively. Finally, the history manager records the metadata of the executed queries, including query statements, statistics, and logical plans.

3.3. Worker

A worker has a query execution engine that performs assigned query. A query unit contains a logical plan and fragments. A fragment is chunk information of an input relation. During execution, a worker sends periodically the reports of the running queries and the resource status to the master which aids in failure. The local query engine communicates with the storage manager which has its data sets stored on HDFS and has a local file system storage as well.

4. QUERY PROCESSING

Tajo has an SQL-like query language, called Tajo Query Language (TQL) [9] which supports most of the SQL commands. In addition to this, TQL also supports two kinds of variables to indicate a scala value and a temporary table respectively.

4.1. Query Planning

Figure 2 shows the steps involved in query transformation.

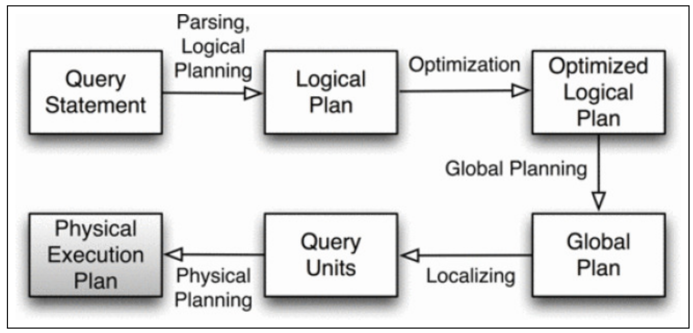

Fig. 2. [6] Query Transformation

Tajo has multiple steps to transform a query statement to physical execution plans. When a user submits a query, the global query engine parses the query into an abstract syntax tree and compiles it into a logical plan. The query optimizer finds the best logical plan which is similar to the original logical plan. This is the optimized logical plan which is transformed into a global query plan. In this step, some logical operators like group-by, sort, and join are transformed into two phases with appropriate repartition methods. Usually, the first phase computes local data on each node, and the intermediate data are range-partitioned or hash-partitioned on the specified keys (e.g., sort keys, grouping keys). The second phase computes the partitioned data on each node. As a result, a global query plan forms of a directed acyclic graph (DAG) of subqueries, which represents a data flow.Based on the physical information of the table, a subquery is localized into a number of query units, each of which is a basic unit of a query executed by a worker. Then, the global query engine schedules the query units along with the DAG of the global plan.

When a worker receives a query unit, it transforms the logical plan of the query unit into a physical execution plan according to its own computing resources. As a result, the query units of the same subquery can lead to different physical execution plans on different workers.

4.2. Query Execution

For input and output of data, Tajo has scanners and appenders. A scanner reads input data from HDFS or local file system, whereas an appender writes output data to either of them. In the current implementation, the scanners/appenders of Tajo support CSV and row-based binary file formats. Since a worker executes a DAG of physical operators and Tajo can use various repartition methods, it can do more optimized and efficient query processing.

5. EXPERIMENT

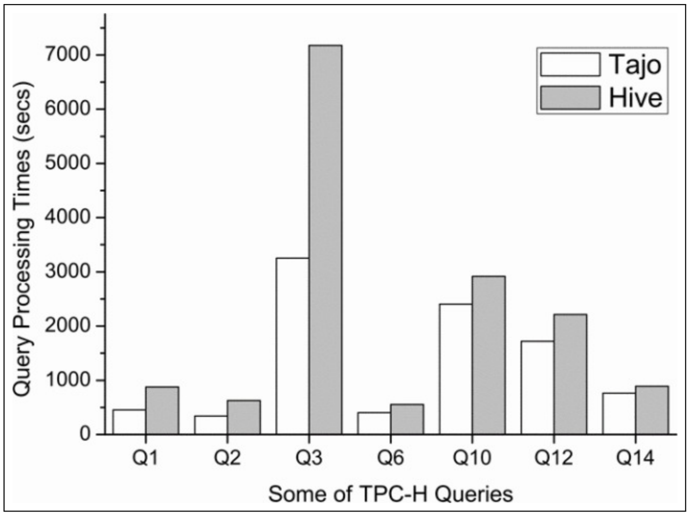

Fig. 3. [6] Performance Evaluation using TPC-H

In [6], the performance of Tajo and Hive has been compared by using 1TB TPC-H benchmark [10] set. Figure 3 shows the experimental results. For this experiment, an in-house cluster of 32 nodes, each of which is equipped with 16GB RAM, 4TB HDD, and an Intel i5 quad core CPU. The x-axis means the TPC-H queries and the y-axis indicates the processing times. The results show that the time take by Tajo to execute SQL queries is less than Apache Hive on the top of MapReduce.

6. TAJO SHELL

Tajo provides a shell utility named Tsql [9]. It is a command-line interface where users can create or drop tables, inspect schema and query tables and can execute other sql commands.

For example: bin/tsql [options] [database name]
If a database name is given, tsql connects to the database at startup time else connects to default database.

7. USE CASES OF TAJO

7.1. Data warehousing and analysis

Korea's SK Telecom firm [11] ran Tajo against 1.7 terabytes worth of data and found it could complete queries with greater speed than either Hive or Impala [12].

7.2. Data discovery

The Korean music streaming [11] service Melon uses Tajo for analytical processing. Tajo executes ETL (extract-transform-load process) jobs 1.5 to 10 times faster than Hive.

8. CONCLUSION

So, with the use of Tajo, efficient data processing and analysis is possible.It processes data faster than the MapReduce Hive framework and provides distributed data warehouse capabilities. By supporting SQL standards and advanced database techniques, Tajo allows direct control of distributed execution and data flow across a variety of query evaluation strategies and optimization opportunities.Lastly, being compatible with ANSI/ISO SQL standard, JDBC [13] driver and various file formats such as CSV, JSON, etc extends its capabilities further.

ACKNOWLEDGEMENTS

I would like to thank my professor Gregor von Laszewski and all the associate instructors for their constant technical support.

REFERENCES

[1] "Apache hadoop," Web Page, accessed: 2017-2-26. [Online]. Available: http://hadoop.apache.org/
[2] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, published as an Article. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492
[3] "Apache tajo," Web Page, accessed: 2017-2-26. [Online]. Available: https://tajo.apache.org
[4] G. Graefe, "Query evaluation techniques for large databases," *ACM Comput. Surv.*, vol. 25, no. 2, pp. 73–169, Jun. 1993, published as an Article. [Online]. Available: http://doi.acm.org/10.1145/152610.152611
[5] D. Kossmann, "The state of the art in distributed query processing," *ACM Comput. Surv.*, vol. 32, no. 4, pp. 422–469, Dec. 2000, published as an Article. [Online]. Available: http://doi.acm.org/10.1145/371578.371598
[6] H. Choi, J. Son, H. Yang, H. Ryu, B. Lim, S. Kim, and Y. D. Chung, "Tajo: A distributed data warehouse system on large clusters," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. San Francisco, CA: Institute of Electrical and Electronics Engineers (IEEE), Apr. 2013, pp. 1320–1323. [Online]. Available: http://ieeexplore.ieee.org/document/6544934/
[7] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using hadoop," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. San Francisco, CA: Institute of Electrical and Electronics Engineers (IEEE), Mar. 2010, pp. 996–1005. [Online]. Available: http://ieeexplore.ieee.org/document/5447738/
[8] "What is rdbms," Web Page, accessed: 2017-4-11. [Online]. Available: http://www.javatpoint.com/what-is-rdbms
[9] "Apache tajo," Web Page, accessed: 2017-2-26. [Online]. Available: http://tajo.apache.org/docs/0.8.0/cli.html
[10] "What is tpc-h," Web Page, accessed: 2017-4-11. [Online]. Available: http://www.tpc.org/tpch/
[11] "Apache tajo," Web Page, accessed: 2017-2-26. [Online]. Available: https://www.tutorialspoint.com/apache_tajo/apache_tajo_introduction.htm
[12] M. Kornacker and J. Erickson, "Cloudera impala: Real-time queries in apache hadoop, for real," Blog, Oct. 2012, accessed: 2017-4-11. [Online]. Available: http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/
[13] "What is jdbc," Web Page, accessed: 2017-4-11. [Online]. Available: http://www.oracle.com/technetwork/java/javase/jdbc/index.html