Optimization of Multiple Correlated Queries by Detecting Similar Data Source with Hadoop/Hive

Puneet Kansal*, Safdar Tanweer and Kavita Bhatnagar

Jamia Hamdard (Hamdard University), Hamdard Nagar, New Delhi – 110 062, India; Puneetkansal.iaf@gmail.com, safdartanweer@yahoo.com, bhatnagarkavita@yahoo.in

Abstract

Objectives: Generated new single Hive Query (HiveQL) by finding the similar type of operation and common data source from two or more input query and compare the total execution time of both queries. Methods/Statistical Analysis: Map Reduce concept of Hadoop Hive is utilized in this paper, a new single query is generated from two or more input queries and 3 sample of data generated with size of 2, 5 and 10 GB using free database generation tool DBGEN. TPC-H queries are executed on this data and total execution time of both the queries is compared to see the performance. Findings: As Hive executes single query at a time, and in this research, multiple queries are provided to hive by converting them into single query. This approach results in reduction of operation while executing the query, which further reduce the execution time and improve the performance of Hive. Hive process the structured data of data warehouse system, so by using this approach, the structured data can be process and analyzed in easily and convenient manner. Structured data is used for processing OLAP (Online Analytical Processing) queries so Hive also helps to process OLAP queries. Hive works in conjunction with Hadoop and it process or execute query on data which is stored on Hadoop. So firstly, Hadoop should be running on the system to use Hive query. This research requires huge amount of data for testing, for this sample data is generated using free data generation tool provided by TPC (Transaction Performance Council), DBGEN. TPC also provide the different types of queries for testing the performance query execution tool, so in this research TPC-H queries are utilized. Application/Improvements: By using the concept which is shown in this research, the total execution time of Hive queries can be reduced drastically and performance of Hive can be increased.

Keywords: DBGEN, Hive Query Language (HiveQL), Online Analytical Processing (OLAP), Transaction Processing Performance Council (TPC), TPC-H Queries

1. Introduction

Now a days everybody is generating or collecting data from various sources in different forms like pdf, mp3, movie, document files, spreadsheet file, images etc. and storing it at unprecedented rates. Due to this huge amount of data, the traditional database management system or currently available processing tools are unable to process it or takes a very long time to produce meaningful output. According to Gartner the Big Data term comprise of 3 V’s that is Volume, Velocity and Variety. Additionally, there are more V’s in the concept Visualization, Veracity (Reliability), Variability and Value. This huge amount of data or big data can be collected, stored, processed and analyzed using various approaches and tools, which are available in the market. Hadoop is an open source software which is based on java programming framework. It uses Map-reduce algorithm for processing of large data sets. Initially, Hadoop was developed by Yahoo in the year of 2006. It has the capability to process huge amount of data sets in the range of Petabyte or Zettabyte or more, which is stored on the cluster of commodity hardware (computers), connected to each other. The large data sets, which requires distributed processing, is processed by the Hadoop software library framework. It is a highly scalable storage platform, which is designed to process very large data sets across thousands of computing nodes that operate in parallel. It provides a cost
effective storage solution for large data volumes with no formal requirements and uses MapReduce processing technique. The MapReduce algorithm performs two important steps for processing of large data sets viz. Map and Reduce. In the first step (Map), it splits the large data set into smaller parts and convert into (key, value) pair, in the second step, the output of first steps became the input of this step (Reduce) and performs reduce operation. These systems are extensively used for analytics and data warehousing, either directly or through the use of a high level query language that is compiled down to a parallel dataflow graph for execution. Data warehousing is a centralized repository of integrated data from various sources. The data which is stored in data warehousing is historical in nature and further used to create analytical reports for the enterprise. Data warehousing is used to process OLAP queries and accompanied with batch oriented query workload that is relatively static. Data warehousing applications are non-interactive in nature, which provides freedom to reorder and optimize queries to improve overall performance. Map-Reduce task on Hadoop uses three functions viz. mapping, shuffling and reducing. Input Data format which Hadoop accept is in the form of (key, value) pair, which is provided by Hadoop Distributed File System (HDFS). As Hadoop uses MapReduce framework, so the input is assigned to mapper in the form of (key, value) pair. Mapper then process the input and produce a set of (key, value) pair as output. The mappers output (key, value) become the input for next step, which is reduce. Before reduce operation the set of (key, value) pair goes through the shuffle phase. In shuffle phase the same k1 pair is assigned to the k1 reducer, same way k2 pair is assigned to k2 reducer and so on as shown in Figure 1 with Finally reducer does aggregation operation by creating the pair of similar k1 value, k2 values etc., and produce output. Hadoop methodology shows that, it does not perform any optimization or global analysis to share work between similar jobs because, Map-Reduce operations are expressed as user-defined functions. Optimization in Hadoop can be achieved by using Hive or Pig as declarative query language. These both perform OLAP operation. The difference Hive and Pig is that Pig supports structured as well as unstructured data and Hive can process only structured data. Structured data is generally used in data warehouse. Pig cannot be used in handling data warehouse because it can handle both structured and unstructured data which is most suitable for streaming data. Data warehouses are not updated frequently and fast response time in processing of queries is not required. In this paper, The Hive framework is chosen for experiment due to the following reason

- Hive process the structured data which is the base of a data warehouse system.
- The applications which are available in a data warehouse system developed using SQL based framework like RDBMS. Which is also supported by the Hive

So based on above reason, it is very easy for user to migrate data warehouse application into Hadoop application. Hive execute each query independently means it support the “query-at-a-time” model. Due to which it is possible to optimize the execution of query by applying the optimization technique. Two or more queries can be converted into single query by detecting the correlation and similar data source, and execution of this single query will drastically reduce the total execution time of query.

![MapReduce tasks](image1)

**Figure 1.** MapReduce tasks.

### 1.1 HIVE

Hive is an open source SQL like interface, which works on distributed data warehouse system. Hive resides on top of the Hadoop framework and provides execution of schema based structured data queries are shown in Figure 2. Hive cannot run independently, it requires Hadoop to be run first. Hive stores data in the form of Metadata (table and partitions) which is stored in database like Derby, MySQL Oracle etc. It facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. It provides data summarization, query, and analysis. It provides a command line tool and JDBC driver to connect users to Hive console Refer Figure 2. Facebook having a huge amount of data in Exabyte, Petabyte on its server, for managing such huge
amount of data Facebook developed Hive, but after some time Apache Software Foundation adopted it. After adoption Apache made the Hive as open source and declared it as Apache Hive. As Hive is an open source so many other companies also developing the Hive like Netflix, Amazon etc. Hive convert SQL query into MapReduce and uses HiveQL for querying the data, which is stored in Hadoop cluster. In terms of storage Hive can use any file system supported by Hadoop, although HDFS is by far the most common. Hive provides its own query language HiveQL, which is similar to SQL, for querying data on a Hadoop cluster. It can manage data in HDFS and run jobs in MapReduce without translating the queries into Java. While executing MapReduce jobs, Hive doesn’t generate Java MapReduce programs, whereas, it uses built-in, generic Mapper and Reducer modules that are driven by an XML file representing the “job plan”. Hive Queries are translated to a graph of Hadoop MapReduce jobs that get executed on the Hadoop grid. Hive Query Language (HQL) is based on SQL, and there are many of the familiar constructs similar to SQL. The concept of storing Database, Tables and data is similar to a RDBMS in HIVE. As like SQL, Hive stores “tables” in a container called “Database”. The data stored in a table according to the “schema” defined by the user. The schema is defined in the form of row and column. Hive is not suitable for executing real time queries because it process OLAP (Online Analyzing Processing) queries.

1.1.1 HiveQL

The Query language for execution of queries in Hive is known as HiveQL, is used to organize and query the data stored in Hive. The current version of HiveQL provides the feature to CREATE and DROP tables and partitions. In HIVE, a table split into multiple parts on a specified partition key, as well as query them with SELECT statements. HiveQL does not support UPDATE and DELETE

---

**Figure 2.** Hive external architecture.
The following are the most important functionalities that are supported using the `SELECT` statements in HiveQL are:

- There is the possibility to join tables on a common key,
- Data can be filtered using row selection techniques,
- And columns can be projected

The above mentioned functionalities are similar to functionality which is provided to the user in a RDBMS (relational database management system). The following example shows how a `SELECT` statement looks in HiveQL:

```sql
SELECT o1_orderkey, o1_custkey, c1_custkey
FROM customer c1 JOIN
Orders o1 ON c1.c1_custkey = o1.o1_custkey JOIN
Line item l1 ON o1.o1_orderkey = l1.l1_orderkey;
```

In this example, a simple join between the table customer, order and line item is initiated on their respective join keys, which are custkey and order key. The above query also shows the projections in first part of the statement which is pushed down to the table scan level by the framework. Multiple HiveQL queries can also be execute during one single operation in Hive. This is known as parallelizing the execution in Hive. The above example also shows that HiveQL statements with multiple tables require specific columns on which those tables are joined. Hive does not support cross products, which is commonly supported in database management systems such as DB2 or Oracle.

### 1.1.2 Hive Architecture

Various components of Hive’s internal architecture are depicted in the Figure 3 and their introduction in appended blow:

**User Interface:** This is a first layer of Hive data warehouse infrastructure. The purpose of user interface for users to submit queries and other operations to the system. User interface provides the facility to user to submit queries and other operations to the system. Hive provides different mechanism for user interaction Viz.

- Web UI
- Hive Command Line and
- HD insight

**Metastore:** It is an object store, which have a database. The purpose of this database is to store schema or metadata for tables. It also store information about databases, column, and their type and HDFS mapping.

There are two ways by which Metastore can be configured:-

- Remote
- Embedded

**HiveQL Process Engine:** HiveQL is a query language, which is similar to SQL. The syntax in HiveQL is very similar to SQL for creating, editing, deletion and query on table. Customized MapReduce script can also be embedded by the user in HiveQL.

**Execution Engine:** It works in conjunction with HiveQL process Engine and MapReduce. The purpose of execution engine is to execute the plan created by compiler which is in the form of DAG stages. It provides connectivity between Hive and Hadoop for execution of query. There is a similarity in the result which is produced by MapReduce and execution engine.

**Figure 3.** Hive internal architecture.

### 2. TPC (Transaction Processing Performance Council)

It is a nonprofit organization, which was founded in 1988. It provides different types of benchmark like TPC-H, TPC-C, TPC-E etc. are shown in Figure 4 to define the performance of transaction processing. The performance can be measured on the basis speed and accuracy of result. The speed is measured in terms of number of queries executed per second or minute and accuracy results are expressed in percentage.

#### 2.1 TPC-H

It consists of various business oriented queries and data modification is also permitted the user. Data can be populated by user from 2 GB to TBs. The more data gives the good performance result. Minimum 100 GB of data...
is considered a good database for running TPC queries. In this paper, TPC-H queries are merged based on their correlation and similar data source and, executed on different data size. The execution time of queries on different database size is compared to check the performance of Hive query execution. TPC-H support the performance metric, which is called TPC-H Composite Query-per-Hour Performance Metric (QphH@size). The TPC-H Price/Performance metric is expressed as $/QphH@size.

For multiple query optimization, different optimization techniques has been experimentally shown\(^\text{16}\), in this author proposed “the concept of SharedHive for improving the performance of execution of queries”\(^\text{16}\). It is also experimentally shown in this paper that using the “SharedHive concept the total execution time is significantly reduced for query execution”. In\(^\text{12}\) user experimentally shown that the execution time of query can be drastically reduced if the correlated query can be detected before execution. So, in this manuscript, it is experimentally shown, that, if the correlated query that has similar data source can be combined separately, and new query is created then executed. So, the execution will be drastically reduced. This time will depend on the number of correlated queries, who has similar data source, more number of queries, more reduction in time, less number of queries, less reduction in time.

4. Proposed Model/Solution

Here, brief information is provided about architecture of proposed Hive (Correlated Query Optimizer Hive) which is the modified version of Hadoop Hive with new Multiple Query Optimization (MQO) component shown in Figure 5.

Figure 4. Different benchmark of TPC.

3. DBGEN

DBGEN is used with TPC-H to generate the database\(^\text{12}\). It can generate database from Gbs to TBs. In data generation it takes from minutes to days or weeks. The more data generated using this takes more time. Generally minimum 100 GB of data is considered good for execution of TPC-h queries. This tool is written in ANSI ‘C’ for portability and provides support to various systems. This tool also provides the source code, which makes the process of building a compliant database population as simple as possible.
6. Methodology

Before execution of query, the proposed Hive architecture, scan the input queries for similar data sources, similar tasks and create a single query which is in the form of HiveQL. The number of operations in a newly generated single HiveQL is comparatively less than the two or more input queries.

**Input:** Group of input queries: \( Q = \{q_1, \ldots, q_n\} \).

**Output:** Generated new single HiveQL query \( Q = \{q_1, \ldots, q_m\} \), where \( m \leq n \) and \( m_{\text{ds}} = n_{\text{ds}} \).

The proposed system reduces the number of operation in query execution therefore reduce the time of total execution. The reduction in execution time is achieved by generating new single query from two or more input query. If \( t' \) is the execution time and \( m, n \) is the number of operation then execution time difference between new query and old can be shown like following:

\[
\sum_{i=1}^{m} (t'_{i}) \leq \sum_{i=1}^{n} (t_{i})
\]

The table structure consisting of tuples and columns for the new single query and input query will be same. So...
that the output of new single query will not be differ from the output of input query. The main difference between the output of old query and new query would be in number of tables. The input query will generate table for each query but the new merged single query will generate only single table for the output. There is also a difference between number of operation in the input query and new single query. The number of operation will be more in the input query and will be less in new single query. Because similar operation has been merged in the new query. As number of operation is less in the execution of single query so there will also be less number of MapReduce operations thus increase the performance of Hive query execution.

Resulting query after searching the similar operation, same data source from input queries (Q1 and Q6)

    DROP TABLE q1_price_smmry_rpt;
    DROP TABLE q6_rvnu_chng;

    CREATE TABLE q1_price_smmry_rpt
    ( l1_returnflag string,
      l1_linestatus string,
      sum_qty_psr int,
      sum_base_price_psr double,
      sum_disc_price_psr double,
      sum_charge_psr double,
      avg_qty_psr int,
      avg_price_psr double,
      avg_disc_psr double,
      count_order_psr int);

    CREATE TABLE q6_rvnu_chng (REVENUE double);

    FROM lineitem
    INSERT OVERWRITE TABLE q1_price_smmry_rpt
    select l1_returnflag,
      l1_linestatus string,
      sum(l1_quantity),
      sum(l1_extendedprice_psr),
      sum(l1_extendedprice_psr * (1 - l1_discount)),
      sum(l1_extendedprice_psr * (1 - l1_discount) * (1 + l1_tax)),
      avg(l1_quantity),
      avg(l1_extendedprice),
      avg(l1_discount),
      count(*)
    where
      l1_shipdate <= '1998-09-16'
      group by
      l1_returnflag,
      l1_linestatus
    order by
      l1_returnflag,
      l1_linestatus

    INSERT OVERWRITE TABLE q6_rvnu_chng
    select
      sum(l1_extendedprice * l1_discount) as revenue
    where
      l1_shipdate >= '1993-02-02'
      and l1_shipdate < '1994-01-02'
      and l1_discount between 0.06 - 0.01 and 0.06 + 0.01
      and l1_quantity < 26;

The underlying SQL-to-MapReduce translator of Hive uses one operation to one job model and opens a new job for each operation (table scan, join, group by, etc.) in a SQL statement. Significant performance increases can be obtained by reducing the number of MapReduce tasks of these jobs.

7. Summary and Conclusion

In this paper, it is experimentally shown that how the performance of conventional Hive can be enhanced. As hive does not support the execution of multiple queries, so if we can find those query from given query set who have similar data source and correlation operation, and create a single query based on these parameters. Then these all queries are executed in Hive and the execution time is compared (Graph 1). It is clear from the graph that the query who has similar data source takes very less time in execution compare to individual query. For detection of correlated and similar data source queries, new layer “Multiple Query Optimization” introduced. The job of this layer is to scan the existing query and generate the new query based on correlation and similar data source. For testing of this approach, huge data set is required, for which TPC-H query set is chosen. For comparison of execution time the queries executed on three variation of database size viz. 2 GB, 5 GB and 10 GB. The performance of generated new query is completely depends on the correlation and similar data source, more correlation more enhancement in performance, less correlation less enhancement in performance. As Hive works on struc-
tured data so as future work the same approach can be applied using Pig because Pig uses both structured and unstructured type of data, and result of both framework (Hive and Pig) can be compared.

8. References

1. Gartner IT Glossary. Available from: http://www.gartner.com/it-glossary/big-data/
2. The 7 V’s of Big Data. Available from: https://www.impactradius.com/blog/7-vs-big-data/
3. What is Apache Hadoop. Available from: http://hadoop.apache.org/
4. Dean J, Ghemawat S. MapReduce, simplified data processing on large clusters. Commun ACM. 2008 Jan; 51(1):107–13.
5. Stonebraker M, Abadi D, Madden S, Paulson E, Pavlo A, Rasin A. MapReduce and parallel DBMSs: friends or foes? Commun ACM. 2010 Jan; 53(1):64–71.
6. DeWitt D, Stonebraker M. MapReduce, A major step backwards. 2008 Jan. p. 1-9.
7. He Y, Lee R, Huay I, Shao Z, Jain N, Zhang X, Xu Z. RCFILE, A fast and space-efficient data placement structure in MapReduce-based warehouse systems. International Conference on Data Engineering; 2011 Apr. p. 1199–208.
8. Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig latin, a not-so-foreign language for data processing. Proceedings of the ACM SIGMOD International Conference on Management of data (SIGMOD’08); 2008. p. 1099-110.
9. Robert J, Stewart, Phil W, Trinder, Loidl HW. Comparing high level MapReduce query language. Conference Paper in International Workshop on Advance Parallel Processing Technologies; 2011. p. 58-72.
10. Timos K, Sellis. Multiple-query optimization. ACM Transactions on Database Systems. 1988; 13(1):23-52.
11. Bayir MA, Toroslu IH, Cosar A. Genetic algorithm for the multiple query optimization problems. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews. 2007; 37(1):147-53.
12. Hive Foundation project home page. Available from: http://hive.apache.org
13. Design - Hive Architecture. Available from: https://cwiki.apache.org/confluence/display/Hive/Design
14. TPC-H query benchmark reference. Available from: http://www.tpc.org/tpch/default.asp.
15. DBGEN – query generator tool reference. Available from: https://github.com/electrum/tpch-dbgen
16. Dokeroglu1 T, Ozal1 S, Bayir MA, Cinar MS, Cosar A. Journal of Cloud Computing. 2014;3-12.
17. Garg V. Optimization of multiple queries for big data with apache hadoop/hive. International Conference on Computational Intelligence and Communication Networks; 2015. p. 938-41.
18. Giannikis G, Alonso G, Kossmann D. SharedDB: Killing one thousand queries with one stone. Prof VLDB. 2010; 5(6):526–37.