An efficient Join-Engine to the SQL query based on Hive with Hbase

KEYWORD: Hbase; Hive; multi-join; Join Engine

ABSTRACT. Hbase combine Hive application platform can analysis of huge amounts of data easily. And querying on multi-join is the bottleneck of restricting the performance. To solve this problem designing a Join Engine between Hbase and Hive. It can read Hbase data and complete multiple table joins and optimization in advance to queries. Using the Join Engine can reduce the time of mapreduce process to data sort and shuffle, effectively improve the efficiency of Hbase combine Hive queries. The experimental results show that the solution with Join Engine can obviously shorten the query time in multi-table join, support the Hbase quasi real-time application.

1 INSTRUCTION

With the development of Internet, the growth of the data presented crazy. Huge unstructured data has already far more than the traditional structured data. The traditional databases have been unable to meet the requirements. How to read and analysis the large amounts of unstructured data efficiently become the issues of common concern. Distributed column type database Hadoop Database (Hbase) [1] not only has the advantage of high expansibility and large capacity but also has hadoop platform support. So, Hbase become very popular. But it does not support the SQL query [2] [3]. Restrict the development of Hbase. At present, Hbase use Hive query mechanism to solve the problem.

1.1 *Relative work*

HBase community puts forward two kinds of composite architectural solutions to solve the problem of Hbase do not support SQL queries. One of them is the integration of the MySQL framework [4]. The second is integration framework with Hive. The first solutions add a MySQL layer which has an indirect links with HBase. When the data in HBase has updated, the scheme must restore the HBase data to MySQL database for SQL queries. The second scheme use HBaseIntegration to establish a directly connected between Hbase and Hive. Hive support real-time query under Hbase database updates. Hive is an open source data warehouse project on Hapdoop platform [5]. It can parse SQL sentence into graphs task execution and execute it with MapReduce task job. Hive is widely use in massive log analysis. The Hbase integrated with Hive scheme can make full use of graphs of parallel capability to provide users with easy operation of SQL query and analysis of data [6].So the scheme has been widely used in the era of big data web [7]. But the plan cannot avoid problem of multi-table join queries [8]. Many application choices change the sequence of the data before join. Although the performance has a little improved, but the scheme must know the SQL sentence before optimizing the order of join data [10].

In conclusion, Hbase integrated with Hive scheme is optimal solution to SQL query on Hbase. But the MapReduce job cost so much time that the scheme cannot use in real time or quasi real time application.

2 THE HBASE AND HIVE INTEGRATION FRAMEWORK

Hive establishes one-to-one relationship with the Hbase original table. Hive parse SQL queries and run the MapReduce task.

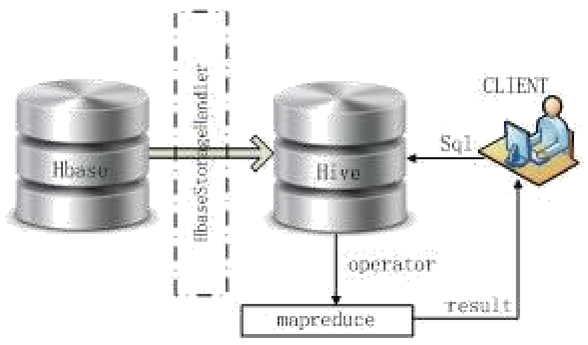

Figure 1 Hive and HBase integration

As shown in figure 1, The Hbase application programming interface HbaseStorageHandler support the one-to-one relationship with Hive. Hive has the responsibility to parse SQL sentence and execute the query task. HBase provide the storage capacity.

The basic process Hbase integration Hive SQL query is first of all establish an association between Hbase and Hive, then parsing and decomposed SQL statement to execution plan by Hive, at last excute Mapreduce task, get the result return to Hbase and clients. Detail of the process is as follows:

1) Hive and Hbase corresponding mapping relationship must be established before the query. Hive's usually in contact with the client side to establish the appearance of establishing to Hbase physical storage. After mapping, Hive can read and write data which store in Hbase.

2) Hive parse SQL statements to execution plan: When Hive client receiving the user's SQL statement Hive parser convert the statement into ASTNode types of syntax tree, and then generate operator tree / DAG, make some optimization operation, then take the tree / DAG generate sever-al MapReduce tasks.

3) Compile the task information generated serialization stage to plan.xml, then start map-reduce, read from Hbase corresponding data input to map function. When configure deserialization plan.xml.

4) After the map function and reduce function returns a result set, save the results output as Collector class, set the file type to save the results by Record Writer. HiveBaseResultSet class will then update the results to HBase data and display to the client.

3 HBASE HIVE QUERY OPTIMIZATION

Hbase Hive query mechanism is mainly composed of Hbase and Hive connection Sql parsing and Mapreduce jobs three modules. Mapreduce job is the most time- consuming process. Because the table join operation will completely processes by mapjoin function in MapReduce tasks, it takes a lot of time.

3.1 *Multi-table query problem analysis*

A multi -table join query is very effective data analysis methods, but Hbase does not support mul-tiple table queries. Although with Hbase Hive fusion can solve this problem, but the multi-table query efficiency has become a new challenge. At present most of the commercial application and research only adjust the join order to improve the query efficiency. Some low efficiency SQL statements even cause the process collapse. The problem is now the multi-table connection in MapReduce tasks needs a lot of process time and more task jobs.

For instance three tables join in a query will generate at least two MR jobs. SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key2) Three tables connection with two keys. How doses MapReduce task connect the three tables? In first job table A and table B based on first key connect. Results of last job and table c based on the second key connect in second map join task. The SQL query needs to wait for two MR jobs for tables' connection. This became the most time-consuming process in Hbase Hive integration query.

3.2 *Hbase and Hive integration optimization*

Due to Hbase column structure, each Hbase table column of the cluster is given priority to one or two. If the column clusters too much, Hbase will be affected read and write speed. In column type database the read and write unit is one column. So it is not necessary to store too much columns in a single table. Hbase compared with the traditional database can store more redundant data. Duplication of the column data in a multi-table join queries occur frequently equijoins. Utilizing the principle of pre-processing and cache this article designs a Join- Engine which pre-establish a connection on multi-table and sort the data after optimization. When a client queries arrive, you can quickly get the results directly in the Join-Engine, thus greatly improving the efficiency of client queries. In practical application is unable to determine whether the tables should read into the Join-Engine. If Join-Engine read some tables based some rules, but the user never requested, then the Join-Engine is not worth. Because queries have memory characteristic that query often repeat on the same system. It is best to read the related tables which are appear in the query history.

3.3 *Join-Engine*

Because of the separation of multi-table joining with query operations and utilizing the principle of locality and log analysis, the article change the traditional Hbase and Hive integration framework add the Join-Engine as shown in Figure 2.

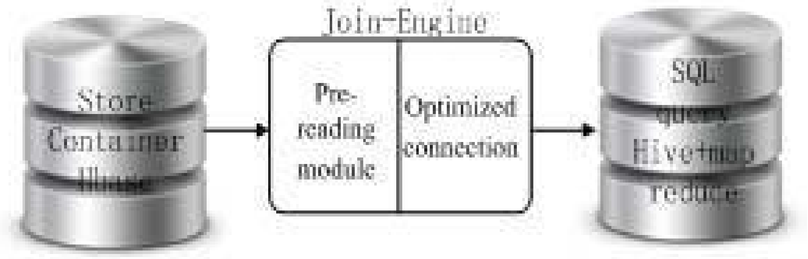

Figure 2 Join-Engine

Join-Engine consists of two modules, Pre-reading mechanisms: Read the relevant data from Hbase based on the query history in Hive; Optimized connection: Complete multi-table join operations, and sort optimize data before Mapreduce upgrade query efficiency.

Pre-reading module select the table in Hbase by query history record. In practical systems can also manually configure achieve more accurate results. Optimized connection module sort and merge tables in units of data. After adding Join- Engine, (1) Mapjoin operation which is most time-consuming is complete before the SQL queries come. Conversely without Join-Engine, First you need to generate a small table Hash Table file and distributed to Cache. Then MapJoinOperator load small table (Hash Table) from the Distributed Cache execute map Join operations. So the connection operation for Mapreduce process is very complex and time -consuming work. (2) Join-Engine carry on sort optimization process when connect operation is in progress. Then the map input split process can be achieved optimal effect. Similar data are assigned to the same slice task statistics, which can speed up the cleaning work after the map statistics. Conversely without Join-Engine, Then fragments will be random load into the map function. Prior to Reduce function get the data, the data need to go through the process of data cleansing – shuffle. Shuffle process is constantly read the data into the RAM carry out sorting and merging operations. So join the pre-optimized processing can effectively reduce the cleaning process of sorting and merging. Query time can be shortened.

3.4 *Optimization Hbase and Hive SQL query*

The SQL query process of Hbase and Hive integration added Join-Engine as shown in Figure 3，

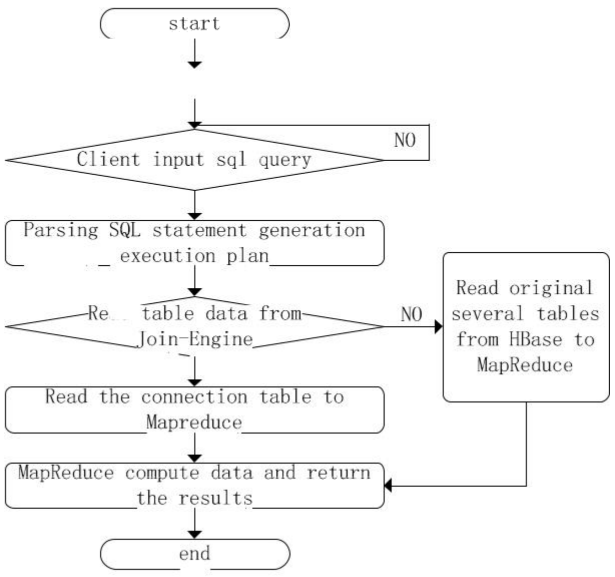
start
Client input sql query
NO
Parsing SQL statement generation execution plan
Re table data from Join-Engine
NO
Read original several tables from HBase to MapReduce
Read the connection table to Mapreduce
MapReduce compute data and return the results
end

Figure 3 SQL query process

There are two ways to process the SQL query. The one is MapReduce job directly read data from the Join-Engine. Another is read data from Hbase original tables.

4 THE DESIGN OF EXPERIMENT

4.1 *The data source of experiment*

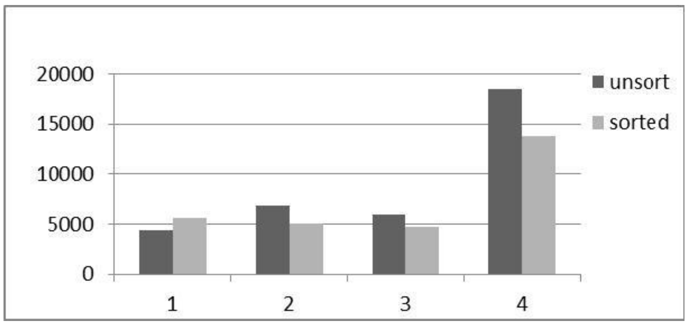

Figure 4 SQL query time

The data source of experiment comes from two custom data sets A and B. They have the same structure: table a<column family: name, column family id>.table b <column family id, column family motive>.table c <column family name, column family tag>. At sets A there are a lot of the same data between two tables in the column of id or name. Sets B have scarcely duplicate data.

4.2 *The results and analysis of experiment*

Experimental Methods: Contrast SQL query time between added Join-Engine and without Join-Engine. Take 10 groups SQL query comparative experiments, each set of experiments carried out ten times the mean value. The 1-3 group are three multi-table queries experiments, use sets A; 4-6 group experiment is corresponds with 1-3 group use the same SQL queries, but using the dataset B, 7-10 group SQL experiments using data sets A, non-multi-table join queries. As shown in Figure 4

As shown, the 1-3 group experiments have very large difference between after added Join-Engine and old solution. But while using the same SQL query statements the 4-6 groups have little difference on SQL query time. The only difference is the data sets. The data set A has a lot of duplicate data so that it will produce more results at mapJoin process. The data set B will generate much less data at mapJoin because it has less correlation data. Because of each 100 000 associated data of table input will return billions of data need so much memory and CPU compute time. The value associated with the data set B does not exist, in the multi-table joins returns the result is zero, consuming very little. So the efficiency of Join-Engine is closely related to the data set. When the data set has more associated column values, The Join-Engine has more efficiency on multi-table SQL query. The 7-10 groups' experiments shown the Join-Engine has less influence on single table SQL query.

Experiment 2:， Sorted data has impact on Mapreduce query efficiency. Use two contains the same 100 000 data tables, one is sorted, and another is an unordered table. Contrast their sql query time. The results shown in Figure 5：

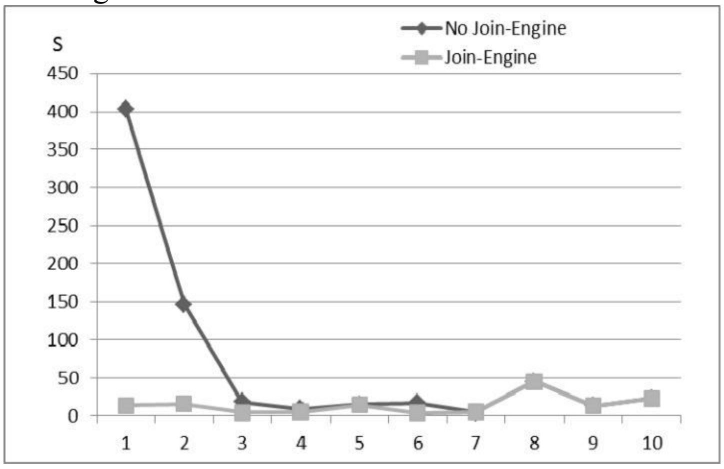

Figure5 sort and unsort data query time comparison

As shown in Figure 2-4 experiments displays the sorted data source has better performance in the implementation of Mapreduce. Because the 2-4 SQL queries has some conditions the result of SQL queries are sorted. SO the reduce function has to sort the data. So the sorted source data can effectively reduce Mapreduce work of sorting data. The first group of SQL test is statement as select *, then reduce function does not need to sort the results, so Mapreduce processing performance has no connection with source data.

The result of experiment shows that， Add Join-Engine can effectively save the mapjoin time from Mapreduce operation. In the case of highly relevant data sets Join-Engine can effectively reduce the time multi-table join queries Because the higher correlation datasets the more Join-Engine works preprocess. That means add Join-Engine will be able to better support the quasi-realtime SQL query applications, the more obvious the effect of Preprocessing mechanism; Pre-sorting process can be reduced when the conditions are in the process of inquiry Mapreduce sort operation, effectively reducing Mapreduce time. Therefore, Join-Engine can improve query efficiency from two aspects, reducing Mapreduce time of mapjoin operations and sorting operations.

5 CONCLUSION AND FUTURE WORK

In the development of the integration technology of Hbase and Hive， Multi-table query performance problems become the key factor to restrict the development of this technology. This article provides a Hbase and Hive integration optimization which added a Join-Engine pre-process Multi-table joining and sorting. Experimental results show that, Added Join-Engine hive higher efficient on SQL query especially on multi- table join query. The next step will continue to optimize the design of the Join-Engine, allowing to the original data analysis capabilities can have an adaptive work on any Hbase database.