Hive Queries Optimization Based on Acemap

Shiyuan Zhan

Database Group

Table of Contents

1 Introduction

1.1 background

Since nowadays the amount of the data on the internet grows much larger than before, it is necessary to use some corresponding platform to computer and process such big data. Hadoop is such a platform which consists of hdfs and mapreduce framework. Similarly, our acemap also need to handle big data, therefore our search engine group began to construct a Hadoop ecosystem and we will use some new query methods to handle these big data such as hive. I am responsible for the construction of hive and UI design for the database.

1.2 Goals

In the project, I mainly accomplished these goals: 1. familiar with Hadoop cluster. 2. Optimize queries with Hive. 3. Compare the queries with Mysql. 4.Do some extra work about matching learning.

2 Environment

The following is the environment on which I worked for this project. Hadoop

Started	Tue Apr 18 17:29:30 CST 2017
Version	2.7.3
Configured Capacity	15.06 TB
Live Nodes	3

is an open source distributed data processing platform, it consists of hdfs and mapreduce framework

2.1 HDFS

HDFS is the primary distributed storage used by Hadoop applications. A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data. The HDFS Architecture Guide describes HDFS in detail.Followings are some features of HDFS:

1. Streaming Data Access
2. Simple Coherency Model
3. Large Data Sets
4. "Moving Computation is Cheaper than Moving Data"
5. Portability Across Heterogeneous Hardware and Software Platforms

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

2.2 Mapreduce

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and

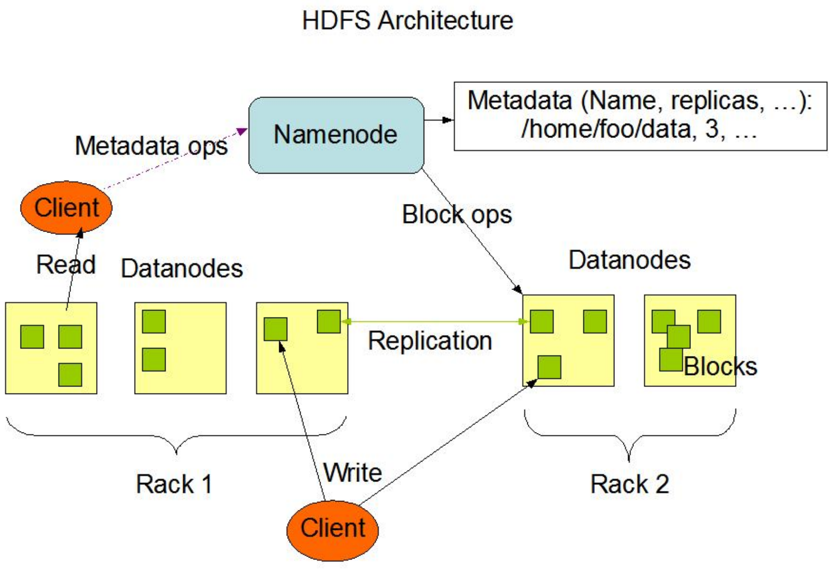

Fig. 1. NameNode and DataNodes

re-executes the failed tasks. We can look at MapReduce is as a 5-step parallel and distributed computation:

1. Prepare the Map() input
2. Run the user-provided Map() code
3. "Shuffle" the Map output to the Reduce processors
4. Run the user-provided Reduce() code
5. Produce the final output

3 Implementation

As we know, Acemap used to use Mysql to do database queries. Mysql is not suitable for big data such as academic database. So we decide to turn to other methods of queries. I am in charge of HIVE optimization. I will introduce HIVE in this part first.

3.1 Features

The folowing are the features of HIVE

1. Tools to enable easy access to data via SQL, thus enabling data warehousing tasks such as extract/transform/load (ETL), reporting, and data analysis.
2. A mechanism to impose structure on a variety of data formats

3. Access to files stored either directly in Apache HDFS^{TM} or in other data storage systems such as Apache HBase^{TM}
4. Query execution via Apache Tez^{TM}, Apache Spark^{TM}, or MapReduce
5. Procedural language with HPL-SQL
6. Sub-second query retrieval via Hive LLAP, Apache YARN and Apache Slider.

3.2 Architecture

The components of Architecture can be diveded into 3 parts:Metastore,Driver and UI.
Metastore stores metadata for each of the tables such as their schema and location. It also includes the partition metadata which helps the driver to track the progress of various data sets distributed over the cluster. The data is stored in a traditional RDBMS format. The metadata helps the driver to keep a track of the data and it is highly crucial. Hence, a backup server regularly replicates the data which can be retrieved in case of data loss.
Driver acts like a controller which receives the HiveQL statements. It starts the execution of statement by creating sessions and monitors the life cycle and progress of the execution. It stores the necessary metadata generated during the execution of an HiveQL statement. The driver also acts as a collection point of data or query result obtained after the Reduce operation.
CLI, UI, and Thrift Server are command Line Interface and UI (User Interface) allow an external user to interact with Hive by submitting queries, instructions and monitoring the process status. Thrift server allows external clients to interact with Hive just like how JDBC/ODBC servers do.

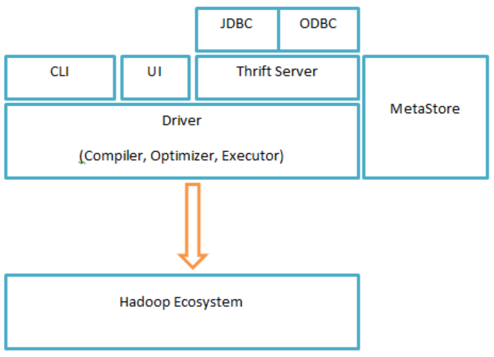

Fig. 2. NameNode and DataNodes

3.3 Installment

And then I will introduce how to install HIVE on an HDFS and run your first code on HIVE. First we need to reach these requirements:

1. Java 1.7 Note: Hive versions 1.2 onward require Java 1.7 or newer. Hive versions 0.14 to 1.1 work with Java 1.6 as well. Users are strongly advised to start moving to Java 1.8. Java in our cluster is 1.8.0.
2. Hadoop 2.x (preferred), 1.x (not supported by Hive 2.0.0 onward).Hive versions up to 0.13 also supported Hadoop 0.20.x, 0.23.x. Our Hadoop version is 2.7.3
3. "Shuffle" the Map output to the Reduce processors
4. Run the user-provided Reduce() code
5. Produce the final output

Then download the recent stable release of Hive 2.1.1 and do the following operation:

```
tar -xzvf hive-2.1.1.tar.gz
cd hive
export HIVE_HOME={{pwd}}
export PATH=$HIVE_HOME/bin:$PATH
cd conf
vim hive-site.xml
<configuration>
        <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://hadoop-master:3306/hive?
createDatabaseIfNotExist=true</value>
        <description>JDBC connect string for a JDBC
metastore</description>
        </property>
        <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
        <description>Driver class name for a JDBC
metastore</description>
        </property>

        <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>hive<value>
        <description>username to use against metastore
database</description>
        </property>
        <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>hive</value>
        <description>password to use against metastore
database</description>
        </property>
</configuration>
```

Then we can use HIVE

4 Result

Based on installed HIVE, I have finished some comparision between old and new queres. I have made some conclusions according to the speed and throughput of queries.

4.1 Evaluation of HIVE

First, I run our queries on HIVE and mysql and I made some comparisions between different queries based on the difficulties of query on HIVE. We can see the data in P. I chose one query which is diffficult and the other one which is easier.

The first query is in the following. The purpose of the query is to find papers which quote the author and how many times the author is quoted.

```
select PaperReferences.PaperID,count(*)
    from PaperReferences inner join
    (select * from PaperAuthorAffiliations
    where AuthorID='0C7733DB') as TB
    on PaperReferences.PaperReferenceID = TB.PaperID
    group by PaperReferences.PaperID
```

The second query is in the follings.The purpose of this query is find how many times the author is quoted by SCI.

```
SELECT  count(*),SUM(SCICitation) as sum from
    PaperSciReferencesCount CROSS JOIN
    (select PaperID
    from PaperAuthorAffiliations
    where AuthorID = '0000194E' ) AS TB1 on
    PaperSciReferencesCount.PaperReferenceID = TB1.PaperID
```

We can see that Hive functions better on the first query from the figure for the reason that HDFS is designed for big data which is complicated for an easy query needed to be transformed to map-reduce from Mysql. So. if we use HIVE on Acemap, there may not be a sharp promotion on speed but we can use HIVE to do some research about academic data.

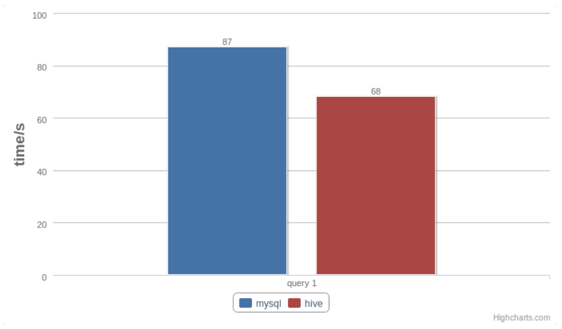

Fig. 3. query1

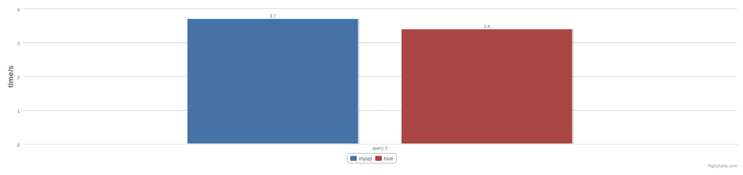

Fig. 4. query2

4.2 Discussion about Cache

Cache means a collection of data duplicating original values stored elsewhere on a computer, usually for easier access. In database, it means if you do a query twice, yuo will find the second time be faster. I still use the first query to discuss about Cache.

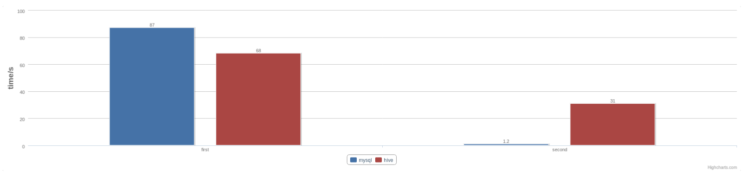

Fig. 5. query2

We can conclude from the figure that Cache in mysql has done a better work than HIVE.

5 Conclusion

Hive is an great database for its speed to query. Also Hive is not adapt well with our Acemap system because the query in our website is not so complicated to use Hive. We can use Hive to do some data mining work.