Assignment: 3

AIM: Write an application using HiveQL for flight information system

PROBLEM STATEMENT /DEFINITION

Write an application using HiveQL for flight information system which will include

a. Creating, Dropping, and altering Database tables.
b. Creating an external Hive table.
c. Load table with data, insert new values and field in the table, Join tables with Hive
d. Create index on Flight Information Table
e. Find the average departure delay per day in 2008.

OBJECTIVE

- To understand various NOSQL database
- To understand the integration of NOSQL database with Hadoop.
- To analyze the performance of distributed processing with NOSQL.

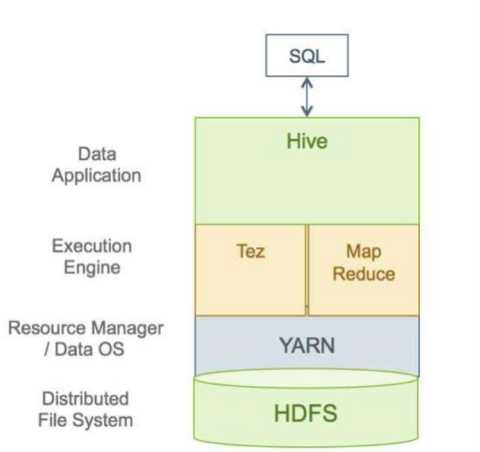
THEORY:

SQL queries are submitted to Hive and they are executed as follows:

1. Hive compiles the query.
2. An execution engine, such as Tez or MapReduce, executes the compiled query.
3. The resource manager, YARN, allocates resources for applications across the cluster.
4. The data that the query acts upon resides in HDFS (Hadoop Distributed File System). Supported data formats are ORC, AVRO, Parquet, and text.
5. Query results are then returned over a JDBC/ODBC connection.

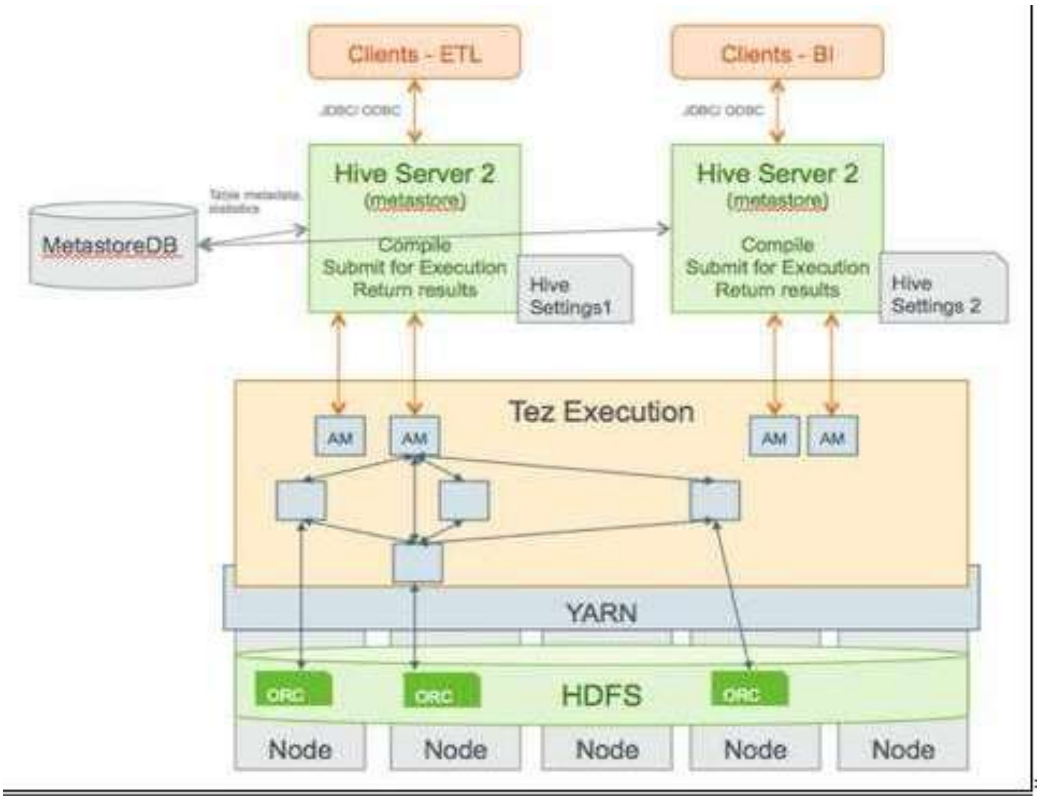
A simplified view of this process is shown in the following figure.

Figure 1.1. SQL Query Execution Process



Detailed Query Execution Architecture
The following diagram shows a detailed view of the HDP query execution architecture:

Figure 1.2. SQL Query Execution Process



The following sections explain major parts of the query execution architecture.

Hive Clients

You can connect to Hive using a JDBC/ODBC driver with a BI tool, such as Microstrategy, Tableau, BusinessObjects, and others, or from another type of application that can access Hive over a JDBC/ODBC connection. In addition, you can also use a command-line tool, such as Beeline, that uses JDBC to connect to Hive. The Hive command-line interface (CLI) can also be used, but it has been deprecated in the current release and Hortonworks does not recommend that you use it for security reasons.

SQL in Hive

Hive supports a large number of standard SQL dialects. In a future release, when SQL:2011 is adopted, Hive will support ANSI-standard SQL.

HiveServer2

Clients communicate with HiveServer2 over a JDBC/ODBC connection, which can handle multiple user sessions, each with a different thread. HiveServer2 can also handle long-running sessions with asynchronous threads. An embedded metastore, which is different from the MetastoreDB, also runs in HiveServer2. This metastore performs the following tasks:

- Get statistics and schema from the MetastoreDB
- Compile queries
- Generate query execution plans
- Submit query execution plans
- Return query results to the client

Multiple HiveServer2 Instances for Different Workload

Multiple HiveServer2 instances can be used for:

- Load-balancing and high availability using ZooKeeper
- Running multiple applications with different setting

Because HiveServer2 uses its own settings file, using one for ETL operations and another for interactive queries is a common practice. All HiveServer2 instances can share the same MetastoreDB. Consequently, setting up multiple HiveServer2 instances that have embedded metastores is a simple operation.

Tez Execution

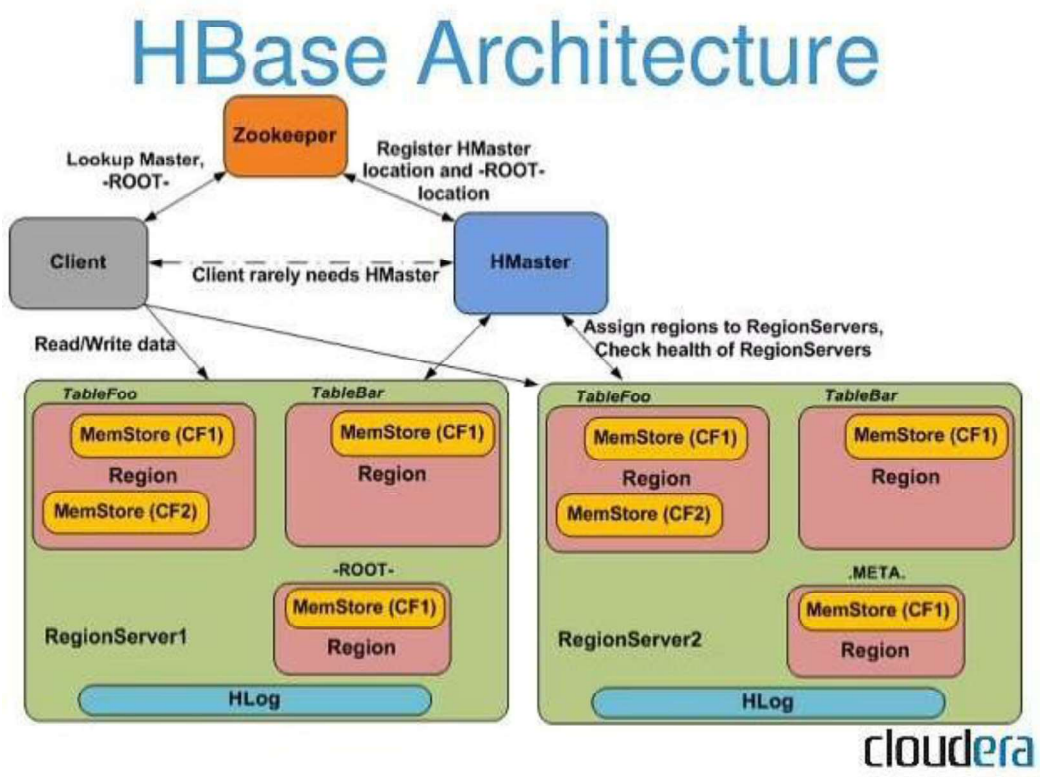
After query compilation, HiveServer2 generates a Tez graph that is submitted to YARN. A Tez Application Master (AM) monitors the query while it is running.

Security

HiveServer2 performs standard SQL security checks when a query is submitted, including connection authentication. After the connection authentication check, the server runs authorization checks to make sure that the user who submits the query has permission to access the databases, tables, columns, views, and other resources required by the query. Hortonworks recommends that you use SQLStdAuth or Ranger to implement security. Storage-based access controls, which is suitable for ETL workloads only, is also available.

File Formats

Hive supports many file formats. You can write your own SerDes (Serializers, Deserializers) interface to support new file formats



Components of Apache HBase Architecture

HBase architecture has 3 important components- HMaster, Region Server and ZooKeeper.

i. HMaster

HBase HMaster is a lightweight process that assigns regions to region servers in the Hadoop cluster forload balancing. Responsibilities of HMaster –

- Manages and Monitors the Hadoop Cluster
- Performs Administration (Interface for creating, updating and deleting tables.)
- Controlling the failover
- DDL operations are handled by the HMaster
- Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.
- Region Server

These are the worker nodes which handle read, write, update, and delete requests from clients. Region Server process, runs on every node in the hadoop cluster. Region Server runs on HDFS DataNode andconsists of the following components –

- Block Cache – This is the read cache. Most frequently read data is stored in the read cache and whenever the block cache is full, recently used data is evicted.
- MemStore- This is the write cache and stores new data that is not yet written to the disk. Every column family in a region has a MemStore.
- Write Ahead Log (WAL) is a file that stores new data that is not persisted to permanent storage.
- HFile is the actual storage file that stores the rows as sorted key values on a disk.

Zookeeper

HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization. Whenever a client wants to communicate with regions, they have to approach Zookeeper first. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquoram will trigger error messages and start repairing failed nodes.

ZooKeeper service keeps track of all the region servers that are there in an HBase cluster- tracking information about how many region servers are there and which region servers are holding which DataNode. HMaster contacts ZooKeeper to get the details of region servers.

Various services thatZookeeper provides include –

- Establishing client communication with region servers.
- Tracking server failure and network partitions.
- Maintain Configuration Information
- Provides ephemeral nodes, which represent different region servers.

CONCLUSION:

Understand to use various aggregate functions using map reduce.