**ACD\_BDDOF\_Session\_2\_Assignment\_9\_Main**

**Problem Statement:**

* Explain Hive Architecture in Brief.
* Explain Hive Components in Brief.

**Hive:**

Apache Hive is a data warehouse system built on top of Hadoop and is used for analyzing structured and semi-structured data.

* Hive abstracts the complexity of Hadoop MapReduce. Basically, it provides a mechanism to project structure onto the data and perform queries written in HQL (Hive Query Language) that are similar to SQL statements.
* Internally, these queries or HQL gets converted to map reduce jobs by the Hive compiler. Apache Hive supports Data Definition Language (DDL), Data Manipulation Language (DML) and User Defined Functions (UDF).

Hive is not

* A relational database
* A design for OLTP
* A language for real-time queries and row-level updates

**Features of Hive:**

* It stores schema in a database and processed data into HDFS.
* It is designed for OLAP.
* It provides SQL type language for querying called HiveQL or HQL.

**Advantages of Hive:**

* Extensible and scalable to cope up with the growing volume and variety of data, without affecting performance of the system.
* It is as an efficient ETL (Extract, Transform, Load) tool.
* Hive supports any client application written in Java, PHP, Python, C++ or Ruby by exposing its Thrift server. (You can use these client – side languages embedded with SQL for accessing a database such as DB2, etc.).
* As the metadata information of Hive is stored in an RDBMS, it significantly reduces the time to perform semantic checks during query execution.

**Hive Architecture:**



This component diagram contains different units. The following table describes each unit:

|  |  |
| --- | --- |
| **Unit Name** | **Operation** |
| User Interface | Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server). |
| Meta Store | Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping. |
| HiveQL Process Engine | HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it. |
| Execution Engine | The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce. |
| HDFS or HBASE | Hadoop distributed file system or HBASE are the data storage techniques to store data into file system. |

**Working of Hive:**

The following diagram depicts the workflow between Hive and Hadoop.



The following table defines how Hive interacts with Hadoop framework:

|  |  |
| --- | --- |
| **Step No.** | **Operation** |
| 1 | **Execute Query**  The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute. |
| 2 | **Get Plan**  The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query. |
| 3 | **Get Metadata**  The compiler sends metadata request to Metastore (any database). |
| 4 | **Send Metadata**  Metastore sends metadata as a response to the compiler. |
| 5 | **Send Plan**  The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete. |
| 6 | **Execute Plan**  The driver sends the execute plan to the execution engine. |
| 7 | **Execute Job**  Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job. |
| 7.1 | **Metadata Ops**  Meanwhile in execution, the execution engine can execute metadata operations with Metastore. |
| 8 | **Fetch Result**  The execution engine receives the results from Data nodes. |
| 9 | **Send Results**  The execution engine sends those resultant values to the driver. |
| 10 | **Send Results**  The driver sends the results to Hive Interfaces. |

**Hive Components:**

**Hive Clients:**

Hive supports application written in many languages like Java, C++, Python etc. using JDBC, Thrift and ODBC drivers. Hence one can always write hive client application written in a language of their choice.

**Hive Services:**

Apache Hive provides various services like CLI, Web Interface etc. to perform queries. We will explore each one of them shortly in this Hive tutorial blog.

**Processing framework and Resource Management:**

Internally, Hive uses Hadoop MapReduce framework as de facto engine to execute the queries. Hadoop MapReduce framework is a separate topic in itself and therefore, is not discussed here.

**Distributed Storage:**

As Hive is installed on top of Hadoop, it uses the underlying HDFS for the distributed storage. You can refer to the HDFS blog to learn more about it.

**First two major components in the Hive Architecture:**

Hive clients & Hive Services

**1. Hive Clients:**

Apache Hive supports different types of client applications for performing queries on the Hive. These clients can be categorized into three types:

1. Thrift Clients
2. JDBC Clients
3. ODBC Clients

**Thrift Clients:** As Hive server is based on Apache Thrift, it can serve the request from all those programming language that supports Thrift.

**JDBC Clients:** Hive allows Java applications to connect to it using the JDBC driver which is defined in the class org.apache.hadoop.hive.jdbc.HiveDriver.

**ODBC Clients:** The Hive ODBC Driver allows applications that support the ODBC protocol to connect to Hive. (Like the JDBC driver, the ODBC driver uses Thrift to communicate with the Hive server.)

**2. Hive Services:**

Hive provides many services.

**Hive CLI (Command Line Interface):** This is the default shell provided by the Hive where you can execute your Hive queries and commands directly.

**Apache Hive Web Interfaces:** Apart from the command line interface, Hive also provides a web based GUI for executing Hive queries and commands.

**Hive Server:** Hive server is built on Apache Thrift and therefore, is also referred as Thrift Server that allows different clients to submit requests to Hive and retrieve the final result.

**Apache Hive Driver:** It is responsible for receiving the queries submitted through the CLI, the web UI, Thrift, ODBC or JDBC interfaces by a client. Then, the driver passes the query to the compiler where parsing, type checking and semantic analysis takes place with the help of schema present in the metastore. In the next step, an optimized logical plan is generated in the form of a DAG (Directed Acyclic Graph) of map-reduce tasks and HDFS tasks. Finally, the execution engine executes these tasks in the order of their dependencies, using Hadoop.

**Metastore: M**etastore is a central repository for storing all the Hive metadata information. Hive metadata includes various types of information like structure of tables and the partitions along with the column, column type, serializer and deserializer which is required for Read/Write operation on the data present in HDFS. The metastore comprises of two fundamental units:

* A service that provides metastore access to other Hive services.
* Disk storage for the metadata which is separate from HDFS storage.

**Metastore Configuration:**

Metastore stores the meta data information using RDBMS and an open source ORM (Object Relational Model) layer called Data Nucleus which converts the object representation into relational schema and vice versa. The reason for choosing RDBMS instead of HDFS is to achieve low latency.

We can implement metastore in following three configurations:

***1. Embedded Metastore:***

Both the metastore service and the Hive service runs in the same JVM by default using an embedded Derby Database instance where metadata is stored in the local disk. This is called embedded metastore configuration. In this case, only one user can connect to metastore database at a time. If you start a second instance of Hive driver, you will get an error

**2. Local Metastore:**

This configuration allows us to have multiple Hive sessions i.e. Multiple users can use the metastore database at the same time. This is achieved by using any JDBC compliant database like MySQL which runs in a separate JVM or a different machine than that of the Hive service and metastore service which are running in the same JVM.

**3. Remote Metastore:**

In the remote metastore configuration, the metastore service runs on its own separate JVM and not in the Hive service JVM. Other processes communicate with the metastore server using Thrift Network APIs. You can have one or more metastore servers in this case to provide more availability. The main advantage of using remote metastore is you do not need to share JDBC login credential with each Hive user to access the metastore database.

**Hive Data model:**

Data in Hive can be categorized into three types on the granular level:

* Table
* Partition
* Bucket

**Tables:**

Tables in Hive are the same as the tables present in a Relational Database. You can perform filter, project, join and union operations on them. There are two types of tables in Hive:

***1. Managed Table:***

As the name suggests (managed table), Hive is responsible for managing the data of a managed table.

If you load the data from a file present in HDFS into a Hive Managed Table and issue a DROP command on it, the table along with its metadata will be deleted.

***2. External Table:***

For external table, Hive is not responsible for managing the data.

In this case, when you issue the LOAD command, Hive moves the data into its warehouse directory. Then, Hive creates the metadata information for the external table. If you issue a DROP command on the external table, only metadata information regarding the external table will be deleted.

**Partitions:**

Hive organizes tables into partitions for grouping similar type of data together based on a column or partition key. Each Table can have one or more partition keys to identify a particular partition. This allows us to have a faster query on slices of the data.

**Buckets:**

To divide each partition or the unpartitioned table into Buckets based on the hash function of a column in the table. Actually, each bucket is just a file in the partition directory or the table directory (unpartitioned table).

Bucketing makes the sampling process more efficient and therefore, allows us to decrease the query time.