Working of Hive SQL

Apache Hive is a data warehouse infrastructure built on top of Apache Hadoop® for providing data summarization, ad hoc query, and analysis of large datasets. It provides a mechanism to project structure onto the data in Hadoop and to query that data using a SQL-like language called HiveQL (HQL).

HiveQL is a superset of SQL, and it supports most of the basic SQL operations, such as SELECT, INSERT, UPDATE, and DELETE. HiveQL also supports some additional features, such as the ability to query data stored in different formats, and the ability to run complex analytical queries.

When you run a HiveQL query, Hive converts the query into a MapReduce job. MapReduce is a programming model and an associated implementation for processing and generating large data sets. MapReduce jobs are broken down into smaller tasks that are executed in parallel on a cluster of machines.

Once the MapReduce job is complete, Hive returns the results of the query to the user.

Here is an example of a HiveQL query:

Code snippet
```
SELECT * FROM table WHERE column1 = 'value1' AND column2 > 100;
```

This query will select all rows from the table where the value of column1 is equal to 'value1' and the value of column2 is greater than 100.

HiveQL is a powerful tool for querying large datasets stored in Hadoop. It provides a familiar SQL-like interface that makes it easy to use for users who are already familiar with SQL. HiveQL also supports some additional features that make it a powerful tool for data analysis.

Here are some of the benefits of using HiveQL:

- It is a SQL-like language, which makes it easy to use for users who are already familiar with SQL.
- It supports most of the basic SQL operations, such as SELECT, INSERT, UPDATE, and DELETE.
- It also supports some additional features, such as the ability to query data stored in different formats, and the ability to run complex analytical queries.
- HiveQL is a powerful tool for querying large datasets stored in Hadoop.

Here are some of the limitations of using HiveQL:

- It can be slow for interactive queries.
- It can be difficult to troubleshoot HiveQL queries.
- HiveQL is not as efficient as some other data warehouse technologies.

Overall, HiveQL is a powerful tool for querying large datasets stored in Hadoop. It is easy to use for users who are already familiar with SQL, and it supports a wide range of features. However, it can be slow for interactive queries, and it can be difficult to troubleshoot HiveQL queries.

****

Working of Hive

The working of Hive can be summarized in the following steps:

1. The user submits a HiveQL query to the Hive server.
2. The Hive server parses the query and converts it into a MapReduce job.
3. The MapReduce job is broken down into smaller tasks that are executed in parallel on a cluster of machines.
4. The results of the MapReduce job are collected by the Hive server.
5. The Hive server returns the results of the query to the user.

Here is a diagram of the Hive architecture:

The main components of the Hive architecture are:

- The Hive server: The Hive server is responsible for parsing HiveQL queries, converting them into MapReduce jobs, and collecting the results of the MapReduce jobs.
- The Metastore: The Metastore is a database that stores metadata about Hive tables, such as the table schema, the location of the data files, and the partitioning information.
- The Hadoop Distributed File System (HDFS): HDFS is a distributed file system that stores the data files for Hive tables.
- The MapReduce framework: The MapReduce framework is responsible for executing MapReduce jobs.

Hive is a powerful tool for querying large datasets stored in Hadoop. It provides a familiar SQL-like interface that makes it easy to use for users who are already familiar with SQL. Hive also supports some additional features that make it a powerful tool for data analysis.

Here are some of the benefits of using Hive:

- It is a SQL-like language, which makes it easy to use for users who are already familiar with SQL.
- It supports most of the basic SQL operations, such as SELECT, INSERT, UPDATE, and DELETE.
- It also supports some additional features, such as the ability to query data stored in different formats, and the ability to run complex analytical queries.
- Hive is a powerful tool for querying large datasets stored in Hadoop.

Here are some of the limitations of using Hive:

- It can be slow for interactive queries.
- It can be difficult to troubleshoot Hive queries.
- Hive is not as efficient as some other data warehouse technologies.

Overall, Hive is a powerful tool for querying large datasets stored in Hadoop. It is easy to use for users who are already familiar with SQL, and it supports a wide range of features. However, it can be slow for interactive queries, and it can be difficult to troubleshoot Hive queries.

****

How to create a database in Hive QL in ubuntu

To create a database in Hive QL in Ubuntu, you can use the following steps:

1. Open a terminal window.
2. Start the Hive shell by running the following command:

Code snippet

```
hive
```

3. Create the database by running the following command:

Code snippet

```
create database <database_name>;
```

For example, to create a database named `my_database`, you would run the following command:

Code snippet

```
create database my_database;
```

4. Verify that the database was created by running the following command:

Code snippet

```
show databases;
```

This will list all of the databases that have been created. The database that you created should be listed in the output.

Here is an example of the output of the `show databases` command:

Code snippet

```
Database

-------

default

my_database
```

5. Exit the Hive shell by running the following command:

Code snippet

```
exit;
```

You have now successfully created a database in Hive QL in Ubuntu.

****

How to create a table in Hive QL in ubuntu

To create a table in Hive QL in Ubuntu, you can use the following steps:

1. Open a terminal window.
2. Start the Hive shell by running the following command:

Code snippet

```
hive
```

3. Create the table by running the following command:

Code snippet

```
create table <table_name> (
    <column_name> <data_type>,

    <column_name> <data_type>,

    ...

);
```

For example, to create a table named `my_table` with two columns, `name` and `age`, you would run the following command:

Code snippet

```
create table my_table (

    name string,

    age int

);
```

4. Verify that the table was created by running the following command:

Code snippet

```
show tables;
```

This will list all of the tables that have been created. The table that you created should be listed in the output.

Here is an example of the output of the `show tables` command:

Code snippet

```
Table
```

```
-------

my_table
```

5. Exit the Hive shell by running the following command:

Code snippet

```
exit;
```

You have now successfully created a table in Hive QL in Ubuntu.

Here are some additional details about the `create table` command:

- The `<table_name>` is the name of the table that you are creating.
- The `<column_name>` is the name of the column.
- The `<data_type>` is the data type of the column.
- You can specify multiple columns by separating them with commas.
- You can also specify a comment for the table or column by using the `COMMENT` keyword.

Here is an example of a `create table` command with a comment:

Code snippet

```
create table my_table (

    name string COMMENT 'The name of the person',

    age int COMMENT 'The age of the person'

);
```

****

What is metrastore in Hive QL

The Hive metastore is a central repository for storing metadata about Hive tables, including their schema, location, and partitions. It is typically implemented as a relational database, such as MySQL or PostgreSQL. The metastore is used by Hive to manage tables and their data, and it can also be queried directly using HiveQL.

The following are some of the benefits of using a metastore:

- Centralized storage of metadata: The metastore provides a central location for storing metadata about Hive tables. This makes it easy to manage and access metadata, and it also helps to ensure that metadata is consistent across all Hive clients.
- Improved performance: The metastore can improve performance by caching metadata and providing efficient access to it. This can be especially beneficial for large-scale Hive deployments.
- Increased reliability: The metastore can help to improve reliability by providing a single point of failure for metadata. This can be especially beneficial for mission-critical Hive deployments.

To use the Hive metastore, you must first configure it. This can be done by editing the Hive configuration file, hive-site.xml. Once the metastore is configured, you can start using it by creating tables and loading data into them.

To query the Hive metastore, you can use the HiveQL metastore commands. These commands allow you to retrieve metadata about Hive tables, including their schema, location, and partitions.

The following are some examples of HiveQL metastore commands:

- `show tables`: This command lists all of the tables in the Hive metastore.
- `describe table`: This command displays the schema of a Hive table.
- `show partitions`: This command lists all of the partitions of a Hive table.

The Hive metastore is a powerful tool that can be used to manage and access metadata about Hive tables. It can help to improve performance, reliability, and ease of use for Hive deployments.

Give me a brief about HDFS

Sure. HDFS stands for Hadoop Distributed File System. It is a distributed file system that is designed to run on commodity hardware. HDFS is used to store large amounts of data that is accessed by multiple users.

HDFS is divided into two main components:

- NameNode: The NameNode is responsible for managing the file system namespace. It keeps track of all the files and directories in the file system, as well as their locations.
- DataNodes: The DataNodes are responsible for storing the data. Each DataNode stores a subset of the data in the file system.

When a user wants to access a file in HDFS, they first contact the NameNode. The NameNode tells the user which DataNodes store the file. The user then contacts the DataNodes to read the file.

HDFS is designed to be fault-tolerant. If a DataNode fails, the NameNode will reassign the data that was stored on the failed DataNode to other DataNodes. This ensures that the data is always available.

HDFS is a powerful tool for storing large amounts of data. It is used by a variety of applications, including web search engines, data warehousing, and scientific computing..

Overall, HDFS is a powerful and scalable distributed file system that can be used to store large amounts of data. It is a good choice for applications that require high performance, fault tolerance, and cost-effectiveness.

Map Reduce in short

Sure. MapReduce is a programming model and an associated implementation for processing and generating large data sets. It was developed by Google in 2004. MapReduce programs are designed to run on a distributed system of computers.

The MapReduce programming model consists of two main parts:

- Mapper: The mapper takes a set of data and produces a set of intermediate key/value pairs.
- Reducer: The reducer takes a set of intermediate key/value pairs and produces a set of output values.

The MapReduce implementation provides a framework for running MapReduce programs on a distributed system. The framework handles the details of distributing the data to the mappers, running the mappers, collecting the intermediate key/value pairs from the mappers, and running the reducers.

MapReduce is a powerful tool for processing large data sets. It is used by a variety of applications, including web search, data mining, and machine learning.

Overall, MapReduce is a powerful and scalable programming model for processing large data sets. It is a good choice for applications that require high performance, fault tolerance, and ease of use.

Hadoop architecture

Hadoop architecture is a distributed computing model that is used to process large data sets. Hadoop is a framework that consists of two main components:

- Hadoop Distributed File System (HDFS): HDFS is a distributed file system that is used to store large data sets.
- MapReduce: MapReduce is a programming model and an associated implementation for processing and generating large data sets.

Hadoop architecture is a scalable and fault-tolerant architecture that can be used to process large data sets.

The Hadoop architecture consists of the following components:

- NameNode: The NameNode is the master server in HDFS. It is responsible for managing the file system namespace and storing the location of all the blocks in the file system.
- DataNodes: The DataNodes are the slave servers in HDFS. They are responsible for storing the data blocks in the file system.
- JobTracker: The JobTracker is the master server in MapReduce. It is responsible for scheduling jobs and managing the resources in the cluster.
- TaskTracker: The TaskTrackers are the slave servers in MapReduce. They are responsible for executing the tasks that are assigned to them by the JobTracker.

The Hadoop architecture is a scalable and fault-tolerant architecture that can be used to process large data sets.

Inserting Data into HDFS

There are a few ways to insert data into HDFS.

- Using the hadoop fs command: The hadoop fs command can be used to copy files from the local file system to HDFS. For example, to copy the file `/home/user/data.txt` to HDFS, you would use the following command:

Code snippet
```
hadoop fs -put /home/user/data.txt /user/hdfs/data.txt
```

- Using the Hadoop API: The Hadoop API can be used to insert data into HDFS programmatically. For example, to insert the data in the file `/home/user/data.txt` into HDFS, you would use the following code:

Code snippet
```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class InsertDataIntoHDFS {

  public static void main(String[] args) throws Exception {

    // Get the filesystem object for HDFS.
    FileSystem fs = FileSystem.get(new Configuration());

    // Create a new path object for the file to be inserted.
    Path path = new Path("/user/hdfs/data.txt");
    // Copy the file from the local file system to HDFS.
    fs.copyFromLocalFile(new Path("/home/user/data.txt"), path);

  }

}
```

- Using a Hadoop client library: There are a number of Hadoop client libraries available for different programming languages. These libraries can be used to insert data into HDFS programmatically.

Once the data has been inserted into HDFS, it can be accessed by Hadoop applications.

Commands to start hadoop

There are a few different ways to start Hadoop.

Using the start-all.sh script

The `start-all.sh` script starts all of the Hadoop daemons, including the NameNode, DataNode, JobTracker, and TaskTracker. To start Hadoop using the `start-all.sh` script, you would use the following command:

Code snippet
```
bin/start-all.sh
```

Using the start-dfs.sh and start-yarn.sh scripts

The `start-dfs.sh` script starts the NameNode and DataNode daemons, while the `start-yarn.sh` script starts the JobTracker and TaskTracker daemons. To start Hadoop using the `start-dfs.sh` and `start-yarn.sh` scripts, you would use the following commands:

Code snippet
```
bin/start-dfs.sh
bin/start-yarn.sh
```

Using the hadoop-daemon.sh script

The `hadoop-daemon.sh` script can be used to start a specific Hadoop daemon. To start a Hadoop daemon using the `hadoop-daemon.sh` script, you would use the following command:

Code snippet
```
bin/hadoop-daemon.sh start <daemon-name>
```

For example, to start the NameNode daemon, you would use the following command:

Code snippet
```
bin/hadoop-daemon.sh start namenode
```

Using the jps command

The `jps` command can be used to list the running Java processes. To list the running Hadoop daemons, you would use the following command:

Code snippet
```
jps
```

The output of the `jps` command will show the name of each running Hadoop daemon. For example, if the NameNode and DataNode daemons are running, the output of the `jps` command would look like this:

Code snippet
```
NameNode
DataNode
```

Once you have started Hadoop, you can access it using the `hadoop fs` command. The `hadoop fs` command has a number of options for accessing data, including the following:

- `-ls`: This option lists the contents of a directory.

- `-cat`: This option prints the contents of a file.
- `-cp`: This option copies a file from one location to another.
- `-mv`: This option moves a file from one location to another.
- `-rm`: This option deletes a file.

For example, to list the contents of the HDFS directory `/user/data`, you would use the following command:

Code snippet
```
hadoop fs -ls /user/data
```

To print the contents of the HDFS file `/user/data/data.txt`, you would use the following command:

Code snippet
```
hadoop fs -cat /user/data/data.txt
```

To copy the HDFS file `/user/data/data.txt` to the local file `/home/user/data.txt`, you would use the following command:

Code snippet
```
hadoop fs -cp /user/data/data.txt /home/user/data.txt
```

To move the HDFS file `/user/data/data.txt` to the new path `/user/new_data`, you would use the following command:

Code snippet
```
hadoop fs -mv /user/data/data.txt /user/new_data
```

To delete the HDFS file `/user/data/data.txt`, you would use the following command:

Code snippet
```
hadoop fs -rm /user/data/data.txt
```

Jps java process status in brief

Sure. jps is a command-line tool that displays information about Java processes running on the local system. It is a part of the Java Development Kit (JDK).

The jps command can be used to list all Java processes running on the local system, or to list only Java processes that match a specific criteria. The criteria can be a process name, a process ID, or a combination of both.

The output of the jps command includes the following information for each Java process:

- Process ID
- Process name
- State of the process
- Virtual machine name
- Arguments passed to the process

For example, the following output shows the information for three Java processes:

Code snippet
```
12345 NameNode
67890 DataNode
98765 JobTracker
```

The jps command can be used to troubleshoot Java applications and to monitor the performance of Java applications. It can also be used to kill Java processes that are no longer needed.

Here are some of the options of jps command:

- -a: This option lists all Java processes running on the local system.
- -l: This option lists only Java processes that are in the foreground.
- -m: This option lists only Java processes that are in the background.
- -v: This option lists the virtual machine name for each Java process.
- -s: This option lists the arguments passed to each Java process.

For example, to list all Java processes running on the local system, you would use the following command:

Code snippet
```
jps -a
```

To list only Java processes that are in the foreground, you would use the following command:

Code snippet
```
jps -l
```

To list only Java processes that are in the background, you would use the following command:

Code snippet
```
jps -m
```

To list the virtual machine name for each Java process, you would use the following command:

Code snippet
```
jps -v
```

To list the arguments passed to each Java process, you would use the following command:

Code snippet
```
jps -s
```

NameNode,SecondaryNameNode, DataNode

Sure. Here are the details of NameNode, SecondaryNameNode, and DataNode:

- NameNode: The NameNode is the master server in HDFS. It is responsible for managing the file system namespace and storing the location of all the blocks in the file system.
- SecondaryNameNode: The SecondaryNameNode is a helper server to the NameNode. It periodically merges the edits log with the fsimage file.
- DataNode: The DataNodes are the slave servers in HDFS. They are responsible for storing the data blocks in the file system.

The NameNode and SecondaryNameNode are usually run on the same machine, while the DataNodes can be run on any number of machines.

The NameNode is the most critical component of HDFS. If the NameNode fails, the entire HDFS file system will be unavailable. The SecondaryNameNode helps to mitigate the risk of a NameNode failure by periodically merging the edits log with the fsimage file. This way, if the NameNode fails, the SecondaryNameNode can be used to restore the file system.

The DataNodes are responsible for storing the data blocks in the file system. If a DataNode fails, the data blocks that it is storing will be unavailable. However, the other DataNodes will still be able to access the data blocks that they are storing.

HDFS is a distributed file system that is designed to be fault-tolerant. The NameNode and SecondaryNameNode help to mitigate the risk of a NameNode failure, while the DataNodes help to mitigate the risk of a DataNode failure.

Resources manager, Nodemanager

Sure. Here are the details of Resources Manager and NodeManager:

- Resource Manager: The Resource Manager is the master server in YARN. It is responsible for managing the cluster resources and scheduling jobs.
- NodeManager: The NodeManager is a slave server in YARN. It is responsible for managing the resources on a single node and running containers.

The Resource Manager and NodeManager are the two main components of YARN. They work together to manage the cluster resources and schedule jobs.

The Resource Manager maintains a pool of resources that can be used by jobs. When a job is submitted to the Resource Manager, the Resource Manager will allocate resources from the pool to the job. The Resource Manager will also track the progress of the job and ensure that the job is completed.

The NodeManager is responsible for managing the resources on a single node. This includes managing the CPU, memory, and disk resources on the node. The NodeManager will also track the health of the node and report any problems to the Resource Manager.

The NodeManager will also run containers on the node. A container is a unit of execution in YARN. A container is allocated a set of resources from the NodeManager and will run a single application.

The Resource Manager and NodeManager work together to provide a scalable and fault-tolerant platform for running applications.

Here are some of the key differences between Resource Manager and NodeManager:

- Resource Manager:
  - The Resource Manager is a master server that manages the cluster resources and schedules jobs.
  - It is responsible for allocating resources to jobs and tracking the progress of jobs.

    - It is also responsible for restarting jobs that fail.
- NodeManager:
    - The NodeManager is a slave server that manages the resources on a single node.
    - It is responsible for managing the CPU, memory, and disk resources on the node.
    - It is also responsible for running containers on the node.

The Resource Manager and NodeManager work together to provide a scalable and fault-tolerant platform for running applications.