>>**ls**

>>**ifconfig**

>>**pwd** (To check currect path)

>>**cd pop**(To go back or to select directory)

>>**hadoop fs -ls /** (to access Hadoop data files) (/ represents home directory)

>>**hdfs dfs -ls /** (to access Hadoop data files)

>>**vi text.txt** (to create a text file) to save the file Esc (esc to enter command mode) :wq

>> **hadoop fs -copyFromLocal text1.txt /temp**  == Copy = -put (To copy file from local file system and paste it to hdfs fs) (Hadoop fs = Setting Hadoop path)

>> **Hadoop fs -copyToLocal /text1.txt/** (To copy from Hadoop fs and paste it to local fs)

>>**rm** (to remove file from local system)

>> **hadoop fs -chmod 777 /text1.txt /** (Giving permission to hdfs file) (chmod= it gives permission to the file)

>>**cat text1.txt** (To check the text inside the text file)

>>**hadoop fs -cat /test0.txt /** (for Hadoop)

>> **Hadoop version** (To check Hadoop version) Hadoop 3.0.0x

>> **Clear/history/mkdir** **/get/ put/cd**

>> **Hadoop fs -mkdir /pop** = **hdfs dfs -mkdir /pop** (To create a folder in hdfs)

>> **hdfs dfs -put test2.txt /poop** (Copying file from local to hdfs folder, pop is folder)

>>**mv test0.txt/home/cloudera/poop** (To move directory from home to a folder)

>>**hdfs dfs -mv /text1.txt/poop** (To move copy from one folder to another)

>>**hdfs dfs -du /** (To check disk usage and details)

>>**hdfs dfs -df /** (To check disk space)

>>**Hadoop fs -usage mkdir** (To check command usage)

>> **hdfs fsck** /

Scaling:

**Vertical Scaling:**

In vertical scaling we increase the capacity of machine capacity in terms of storage, ram, cpu etc.

**Horizontal Scaling:**

In Horizontal scalability we and more machines in the same configuration.

**Hive:**

**Hdfs:** To store data in distributed fashion.

**MapReduce:** To do the computation part.

Hive is kind of wrapper on the top of Hadoop ecosystem.

Typical sql kind of framework.

Logic will be converted in Map-Reduce code.

Hive read the data from HDFS and convert SQL into Map-Reduce.

HDFS

Hive Reads Data from HDFS

Hive

Converts SQL into MapReduce

Convert

Result

Hive is a framework or (Processing Engine) it works like a database but it does not work like transactional database or NoSQL database. It is also used as analytical queries in big data.

**Hadoop Component:**

1. HDFS (Hadoop distributed file storage)—For File storage
2. MapReduce --- For processing data
3. YARN (Yet another resource negotiator) --- Resource Manager

MapReduce:

When we fire MapReduce job the code will go the blocks (Data Nodes).

Map –

* Map function will read the lines of data and generate key value pair. Key will be the data and value will number of occurrences. [name: No of occurrence] [John:01]
* Each node will have different values.

Reduce –

* Reduce function will read intermediate input generated by the MAP and will try to do aggregation.
* Reduce will hold all the keys and values which is generated by MAP and combine all the data from different nodes.

Hive:

Hive components:

JDBC/ODBC

CLI Thrift Web Interface

Driver

Parcer Planner

Meta Store

Execution Optimizer

MapReduce RDBMS

HDFS

JDBC/ODBC – Are connectors which connects spark with Hive.

CLI – Command line interface to run the command.

Thrift – It will work along with JDBC. It works in backend.

Web interface – Web based Command line. Like HUE

Driver –

* It acts like a controller for HQL (Hive query language) which receives hive query
* Creates sessions for query.
* Maintains lifecycle of HQL
* It stores necessary Meta data whatever is needed for execution.
* Main Meta Data for execution (Information about space, file location…).
* Collects results generated by the reducer and display it.

Parser –

* Parser/compiler it checks syntax, libraries, everything is correct or not.
* Syntax check
* Execution Plan
* Steps to get the output
* Raise compile time error.
* Parser/ Compiler and planner works at the same time.

Executor –

* Once compilation part is done the executor executes the task.
* Task is nothing but every execution which is performed. Ex allocating 1000 blocks which 1000 Task.

Optimization –