

BIG DATA ANALYTICS LAB

Continuous Assessment Test- 3

TEAM MEMBERS:

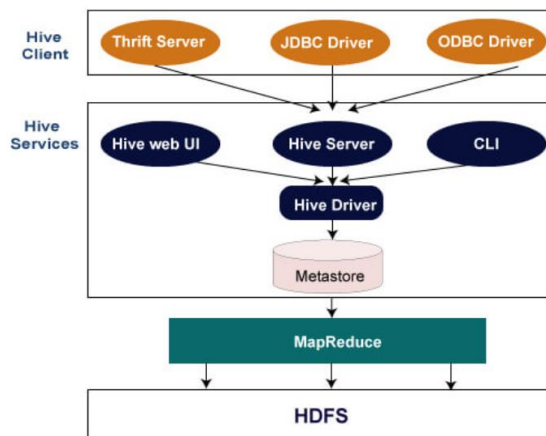
Devasena K (1934006)

Sarika M(1934040)

TOOLS: Hive and Pig

HIVE:

Architecture:



Hive installation

```
cloudera@quickstart: Microsoft Edge
Last login: Sat Apr 30 07:02:12 2022 from 192.168.56.1
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.d/hive-log4j.properties
Exception in thread "main" java.lang.RuntimeException: org.apache.hadoop.hdfs.server.namenode.SafeModeException: Cannot create directory /tmp/hive/cloudera/266a0140-ad19-44ee-a171-bf5bba3a9125.
The reported blocks 948 needs additional 2 blocks to reach the threshold 0.9990 of total blocks 950.
The number of live datanodes 1 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkNameNodeSafeMode(FSNamesystem.java:1519)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.mkDirInt(FSNamesystem.java:4461)
at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.mkDir(NameNodeRpcServer.java:676)
at org.apache.hadoop.hdfs.server.namenode.AuthorizationProviderProxyClientProtocol.mkDir(AuthorizationProviderProxyClientProtocol.java:326)
at org.apache.hadoop.hdfs.protocolPB.ClientNameNodeProtocol$ServerSideTranslatorPB.mkDir(ClientNameNodeProtocol$ServerSideTranslatorPB.java:640)
at org.apache.hadoop.hdfs.protocol.proto.ClientNameNodeProtocol$Protocol$ClientNameNodeProtocol$2.callBlockingMethod(ClientNameNodeProtocol$Protocol.java)
at org.apache.hadoop.hdfs.protocolPB.ServerSideTranslatorPB.callBlockingMethod(ClientNameNodeProtocol$Protocol.java:617)
at org.apache.hadoop.hdfs.RPCServer.call(RPC.java:1073)
at org.apache.hadoop.hdfs.ServerHandler$1.run(Server.java:2226)
at org.apache.hadoop.hdfs.ServerHandler$1.run(Server.java:2222)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.auth.Subject.doAs(Subject.java:415)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1917)
at org.apache.hadoop.hdfs.ServerHandler.run(Server.java:2220)

at org.apache.hadoop.hive.q1.session.SessionState.start(SessionState.java:571)
at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:695)
at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:134)
at sun.reflect.NativeMethodAccessorImpl.invoke(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(Native Method)
at java.lang.reflect.Method.invoke(Method.java:606)
at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
at org.apache.hadoop.util.RunJar.main(RunJar.java:110)
Caused by: org.apache.hadoop.hdfs.RemoteException(org.apache.hadoop.hdfs.server.namenode.SafeModeException): Cannot create directory /tmp/hive/cloudera/266a0140-ad19-44ee-a171-bf5bba3a9125.
Name node is in safe mode.
The reported blocks 948 needs additional 2 blocks to reach the threshold 0.9990 of total blocks 950.
The number of live datanodes 1 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkNameNodeSafeMode(FSNamesystem.java:1519)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.mkDirInt(FSNamesystem.java:4461)
at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.mkDir(FSNamesystem.java:4436)
at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.mkDir(NameNodeRpcServer.java:676)
at org.apache.hadoop.hdfs.server.namenode.AuthorizationProviderProxyClientProtocol.mkDir(AuthorizationProviderProxyClientProtocol.java:326)
at org.apache.hadoop.hdfs.protocolPB.ClientNameNodeProtocol$ServerSideTranslatorPB.mkDir(ClientNameNodeProtocol$ServerSideTranslatorPB.java:640)
at org.apache.hadoop.hdfs.protocol.proto.ClientNameNodeProtocol$Protocol$ClientNameNodeProtocol$2.callBlockingMethod(ClientNameNodeProtocol$Protocol.java)
at org.apache.hadoop.hdfs.protocolPB.ServerSideTranslatorPB.callBlockingMethod(ClientNameNodeProtocol$Protocol.java:617)
at org.apache.hadoop.hdfs.RPCServer.call(RPC.java:1073)
at org.apache.hadoop.hdfs.ServerHandler$1.run(Server.java:2226)
at org.apache.hadoop.hdfs.ServerHandler$1.run(Server.java:2222)
at java.security.AccessController.doPrivileged(Native Method)
at java.security.auth.Subject.doAs(Subject.java:415)
```

Got this error because name node is in safe mode , command to rectify it was

hdfs dfsadmin -safemode leave

```
cloudera@quickstart:~  
login as: cloudera  
cloudera@192.168.56.102's password:  
Last login: Wed May 25 06:44:34 2022 from 192.168.56.1  
[cloudera@quickstart ~]$ hive  
  
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.p  
roperties  
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.  
hive> show databases;  
OK  
default  
Time taken: 0.824 seconds, Fetched: 1 row(s)  
hive> █
```

Default location

```
hive> create database example;  
OK  
Time taken: 1.471 seconds  
hive> describe database example;  
OK  
example          hdfs://quickstart.cloudera:8020/user/hive/warehouse/example.db c  
loudera USER  
Time taken: 0.05 seconds, Fetched: 1 row(s)  
hive> █
```

Custom location:

```
hive> create database example1 location '/users/example1';  
OK  
Time taken: 0.098 seconds  
hive> describe database example1;  
OK  
example1          hdfs://quickstart.cloudera:8020/users/example1 cloudera  
USER  
Time taken: 0.011 seconds, Fetched: 1 row(s)  
hive> █
```

```
hive> drop database example;
OK
Time taken: 0.14 seconds
hive> show databases;
OK
default
example1
Time taken: 0.014 seconds, Fetched: 2 row(s)
hive> █
```

Create a table

```
hive> create table emp (emp_id int , emp_name string , emp_address string ) row
format delimited fields terminated by ',';
OK
Time taken: 1.951 seconds
```

Describe table

```
hive> describe emp;
OK
emp_id          int
emp_name        string
emp_address      string
Time taken: 0.236 seconds, Fetched: 3 row(s)
```

Formatted describe

```
hive> describe formatted emp;
OK
# col_name          data_type          comment

emp_id              int
emp_name             string
emp_address          string

# Detailed Table Information
Database:            default
Owner:               cloudera
CreateTime:          Tue May 31 07:27:14 PDT 2022
LastAccessTime:      UNKNOWN
Protect Mode:        None
Retention:           0
Location:             hdfs://quickstart.cloudera:8020/user/hive/warehouse/emp
Table Type:          MANAGED_TABLE
Table Parameters:
    transient_lastDdlTime 1654007234

# Storage Information
SerDe Library:        org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:          org.apache.hadoop.mapred.TextInputFormat
OutputFormat:          org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:           No
Num Buckets:          -1
Bucket Columns:       []
Sort Columns:         []
Storage Desc Params:
    field.delim        ,
    serialization.format
Time taken: 0.105 seconds, Fetched: 29 row(s)
hive>
```

Copying file from base location to db location

```
[cloudera@quickstart ~]$ hadoop fs -put hivedata.txt /user/hive/warehouse/emp
```

Verifying whether the data exists or not

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/emp
Found 1 items
-rw-r--r--  1 cloudera supergroup      49 2022-05-31 08:45 /user/hive/warehouse/emp/hivedata.txt
```

Read operation

```
hive> select * from emp;
OK
1      sarika  cbe
2      kavya   che
3      abhi    cbe
4      jana    cbe
Time taken: 0.546 seconds, Fetched: 4 row(s)
hive> █
```

Rename table

```
hive> alter table emp rename to emp_new;
OK
Time taken: 0.229 seconds
hive> show tables;
OK
emp_new
Time taken: 0.092 seconds, Fetched: 1 row(s)
hive> █
```

```
hive> load data local inpath 'Downloads/hivedata.txt' into table emp_new;
Loading data to table default.emp_new
Table default.emp_new stats: [numFiles=2, totalSize=98]
OK
Time taken: 0.367 seconds
```

```
hive> select * from emp_new;
OK
1      sarika  cbe
2      kavya   che
3      abhi    cbe
4      jana    cbe
1      sarika  cbe
2      kavya   che
3      abhi    cbe
4      jana    cbe
Time taken: 0.065 seconds, Fetched: 8 row(s)
hive> █
```

Hive project:

Project description:

The project aims to display Beverages-to-Multiple Branches of one Coffee Shop (many-to many relation) using Hive. In other words, each Beverage might be available on many Branches, and each Branch of the Coffee shop might distribute many Beverages. Assuming each branch send their sales report as a csv file. The project aims to stage them to HDFS and further analysis to be performed using Hive for the given problem statement below.

Procedures:

Placing the given dataset in HDFS

Create directory in HDFS

Step 1: Before creating directories in HDFS, ensure all the daemons in hadoop are started. The below code is for creating directory called "projinput" as follows,

```
$ hadoop fs -mkdir /user/hive/projinput
```

Placing the input files in the HDFS directory

Step 2: Copying all the given dataset files from local to HDFS directory in a separate directory. The code as follows

```
$ hadoop fs -copyFromLocal Bev_BranchA.txt /user/hive/projinput/Bev_BranchA.txt  
$ hadoop fs -copyFromLocal Bev_BranchB.txt /user/hive/projinput/Bev_BranchB.txt  
$ hadoop fs -copyFromLocal Bev_BranchC.txt /user/hive/projinput/Bev_BranchC.txt  
$ hadoop fs -copyFromLocal Bev_ConcountA.txt /user/hive/projinput/Bev_ConcountA.txt  
$ hadoop fs -copyFromLocal Bev_ConcountB.txt /user/hive/projinput/Bev_ConcountB.txt  
$ hadoop fs -copyFromLocal Bev_ConcountC.txt /user/hive/projinput/Bev_ConcountC.txt
```

Implementation in HIVE

Step1 : created database using previous method

Database name : hadoophiveproj

Creating & Loading the HIVE tables with the given datasets

Step 2: Create separate raw tables for the Beverages-Consumercount different datasets each in "hadoophiveproj" database. The given file (Bev_Concount *.txt) consist of (A Beverage and the number of consumers).

Example Bev_Concount*.txt: 6 SAMPLE HIVE PROJECT, 21 Triple_Espresso, 38 Mild_LATTE, 73 LARGE_Coffee, 144 Cold_cappuccino, 287 SMALL_cappuccino, 57

```
Hive> use hadoophiveproj;
```

```
Hive> create table if not exists BevcountA (beverage string,count int) row format delimited  
fields terminated by ',';
```

```
Hive> create table if not exists BevcountB(beverage string,count int) row format delimited  
fields terminated by ',';
```

```
Hive> create table if not exists BevcountC (beverage string,count int) row format delimited  
fields terminated by ',';
```

Step 3: Loading the Beverage -Number of consumers' raw tables from the given text files individually.

```
Hive> load data inpath '/user/hive/projinput/Bev_ConscountA.txt' into table BevcountA;
```

```
Hive> load data inpath '/user/hive/projinput/Bev_ConscountB.txt' into table BevcountB;
```

```
Hive> load data inpath '/user/hive/projinput/Bev_ConscountC.txt' into table BevcountC;
```

Step 4: Create separate raw tables for the Beverages-Branches different datasets each in "hadoophiveproj" database.

The given file (Bev_Branch*.txt) consist of (A Beverages and the Branches it was on). Example Bev_Branch*.txt: Special_Lite, Branch6 MED_LATTE, Branch2 Triple_cappuccino, Branch9 ICY_LATTE, Branch5 SMALL_Espresso, Branch1 Double_cappuccino, Branch6 LARGE_Espresso, Branch2 Mild_

```
Hive> create table if not exists BevbranchA(beverage string,branch string)  
row format delimited fields terminated by ',';
```

```
Hive> create table if not exists BevbranchB(beverage string, branch string)  
row format delimited fields terminated by ',';
```

```
Hive> create table if not exists BevbranchC(beverage string, branch string)  
row format delimited fields terminated by ',';
```

Step 5: Loading the Beverage type-Branch raw tables from the given text files individually.

```
hive> load data inpath '/user/hive/projinput/Bev_BranchA.txt' into table BevbranchA;
```

```
hive> load data inpath '/user/hive/projinput/Bev_BranchB.txt' into table BevbranchB;
```

```
hive> load data inpath '/user/hive/projinput/Bev_BranchC.txt' into table BevbranchC;
```

Problem Scenario 1:

What is the most consumed beverage on Branch1?

Explanation Step 1: Creating a single table “Branch1Constotcount” with sub queries in which the Branch1 consumers alone selected and counted.

Explanation Step 2: Selecting the aggregate count from the previously created table Branch1Constotcount and ordering the data in descending.

Query:

```
Hive> select beverage,sum(totalcount) totcount from Branch1Constotcount group by beverage order by totcount desc limit 1;
```

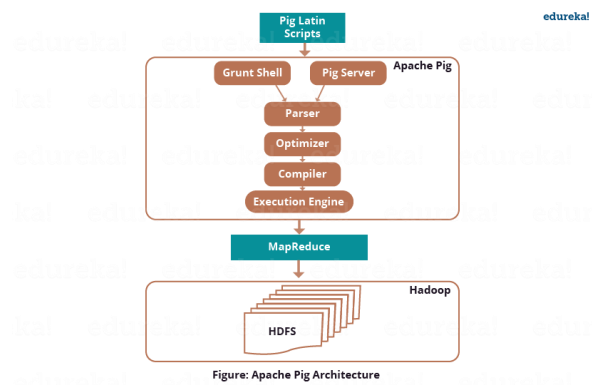
Output:

Special_cappuccino 108163

PIG:

Pig Represents Big Data as data flows. Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce. It provides a high-level scripting language, known as Pig Latin which is used to develop the data analysis codes. First, to process the data which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language. Internally Pig Engine(a component of Apache Pig) converted all these scripts into a specific map and reduced tasks. But these are not visible to the programmers in order to provide a high-level of abstraction. Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of always stored in the HDFS

Architecture of Pig :



Features of Apache Pig:

- For performing several operations Apache Pig provides rich sets of operators like the filters, join, sort, etc.
- Easy to learn, read and write. Especially for SQL-programmers, Apache Pig is a boon.
- Apache Pig is extensible so that you can make your own user-defined functions and processes.
- Join operation is easy in Apache Pig. Fewer lines of code. Apache Pig allows splits in the pipeline.
- The data structure is multivalued, nested, and richer. Pig can handle the analysis of both structured and unstructured data



Applications of Apache Pig:

- For exploring large datasets Pig Scripting is used.
- Provides the supports across large data-sets for Ad-hoc queries. In the prototyping of large data-sets processing algorithms. Required to process the time sensitive data loads.
- For collecting large amounts of datasets in form of search logs and web crawls. Used where the analytical insights are needed using the sampling.

Types of Data Models in Apache Pig

- It consist of the 4 types of data models as follows:
 - ◆ Atom: It is an atomic data value which is used to store as a string. The main use of this model is that it can be used as a number and as well as a string.
 - ◆ Tuple: It is an ordered set of the fields.
 - ◆ Bag: It is a collection of the tuples.
 - ◆ Map: It is a set of key/value pairs.

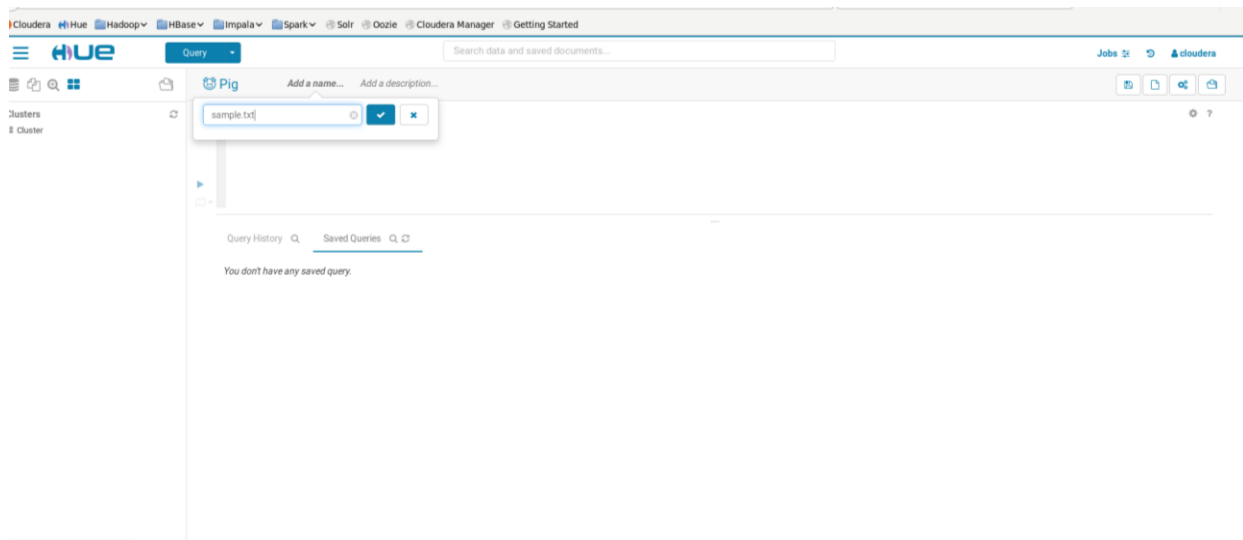
Hive Vs Pig

Features		
1. Language	Hive uses a declarative language called HiveQL	With Pig Latin, a procedural data flow language is used
2. Schema	Hive supports schema	Creating schema is not required to store data in Pig
3. Data Processing	Hive is used for batch processing	Pig is a high-level data-flow language

4. Partitions	Yes	No. Pig does not support partitions although there is an option for filtering
5. Web interface	Hive has a web interface	Pig does not support web interface
6. User Specification	Data analysts are the primary users	Programmers and researchers use Pig
7. Used for	Reporting	Programming
8. Type of data	Hive works on structured data. Does not work on other types of data	Pig works on structured, semi-structured and unstructured data

9. Operates on	Works on the server-side of the cluster	Works on the client-side of the cluster
10. Avro File Format	Hive does not support Avro	Pig supports Avro
11. Loading Speed	Hive takes time to load but executes quickly	Pig loads data quickly
12. JDBC/ ODBC	Supported, but limited	Unsupported

PIG in Cloudera:



Terminal:

```
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ pig
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell)
.
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2022-06-03 06:42:23,114 [main] INFO org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.13.0 (reexported) compiled Oct 04 2017, 11:09:03
2022-06-03 06:42:23,117 [main] INFO org.apache.pig.Main - Logging error message
to: /home/cloudera/pig_1654263743092.log
2022-06-03 06:42:23,140 [main] INFO org.apache.pig.impl.util.Utils - Default bootstrap file /home/cloudera/.pigbootstrap not found
2022-06-03 06:42:23,991 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2022-06-03 06:42:23,991 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-06-03 06:42:23,991 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://quickstart.cloudera:8020
2022-06-03 06:42:25,949 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2022-06-03 06:42:25,949 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker at: localhost:8021
2022-06-03 06:42:25,949 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-06-03 06:42:26,004 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-06-03 06:42:26,006 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2022-06-03 06:42:26,068 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-06-03 06:42:26,068 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
```

Loading the data :

```
grunt> input1= LOAD '/user/cloudera/Pig/sample.txt' AS (f1:chararray);
grunt> █
```

