MILESTONE-3

Overall good job

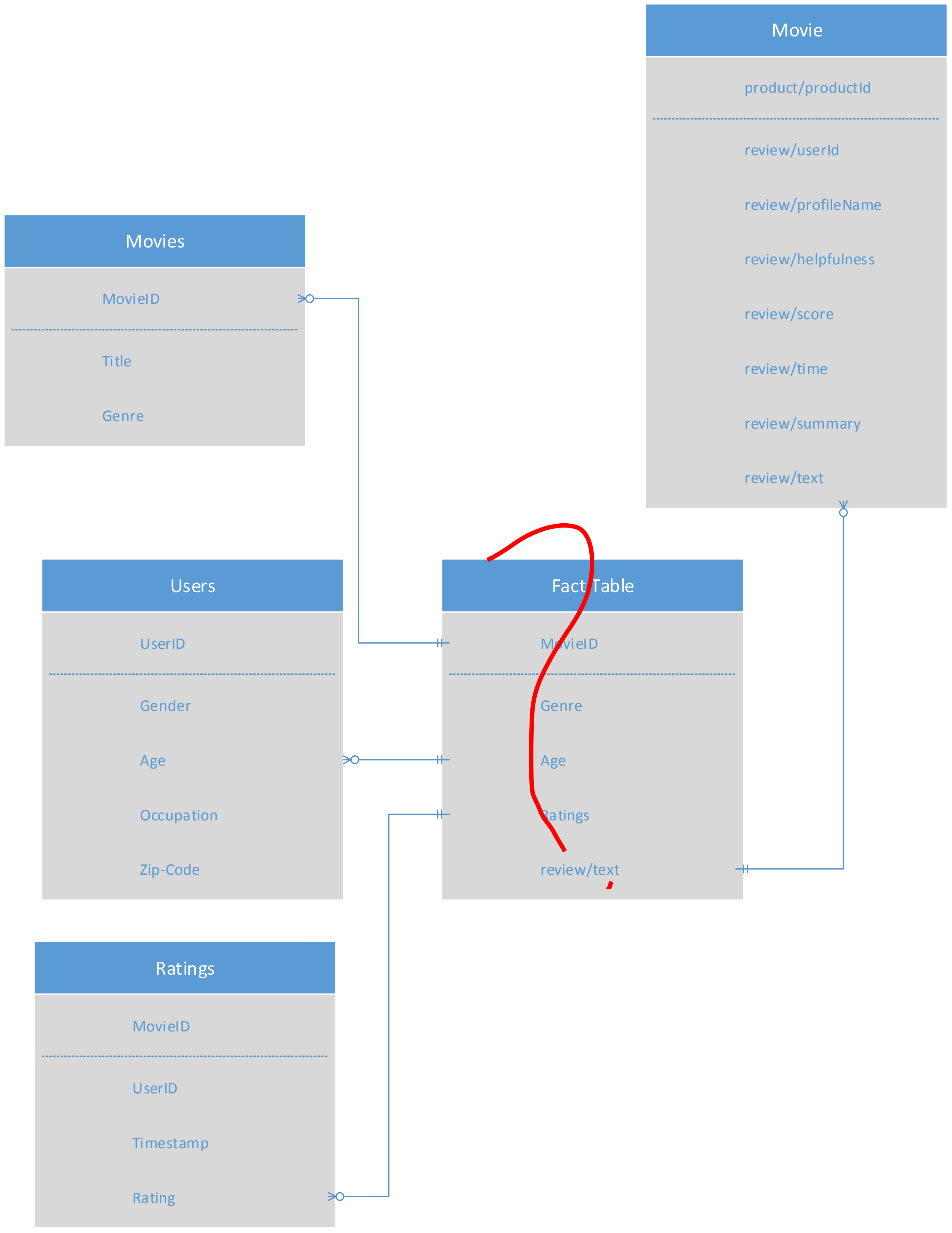
95

IT - 478

MAYANK SHRIVANSH
MADHURI CHOUDHARY
MAYANK AGRAWAL
YESH POLISHETTY

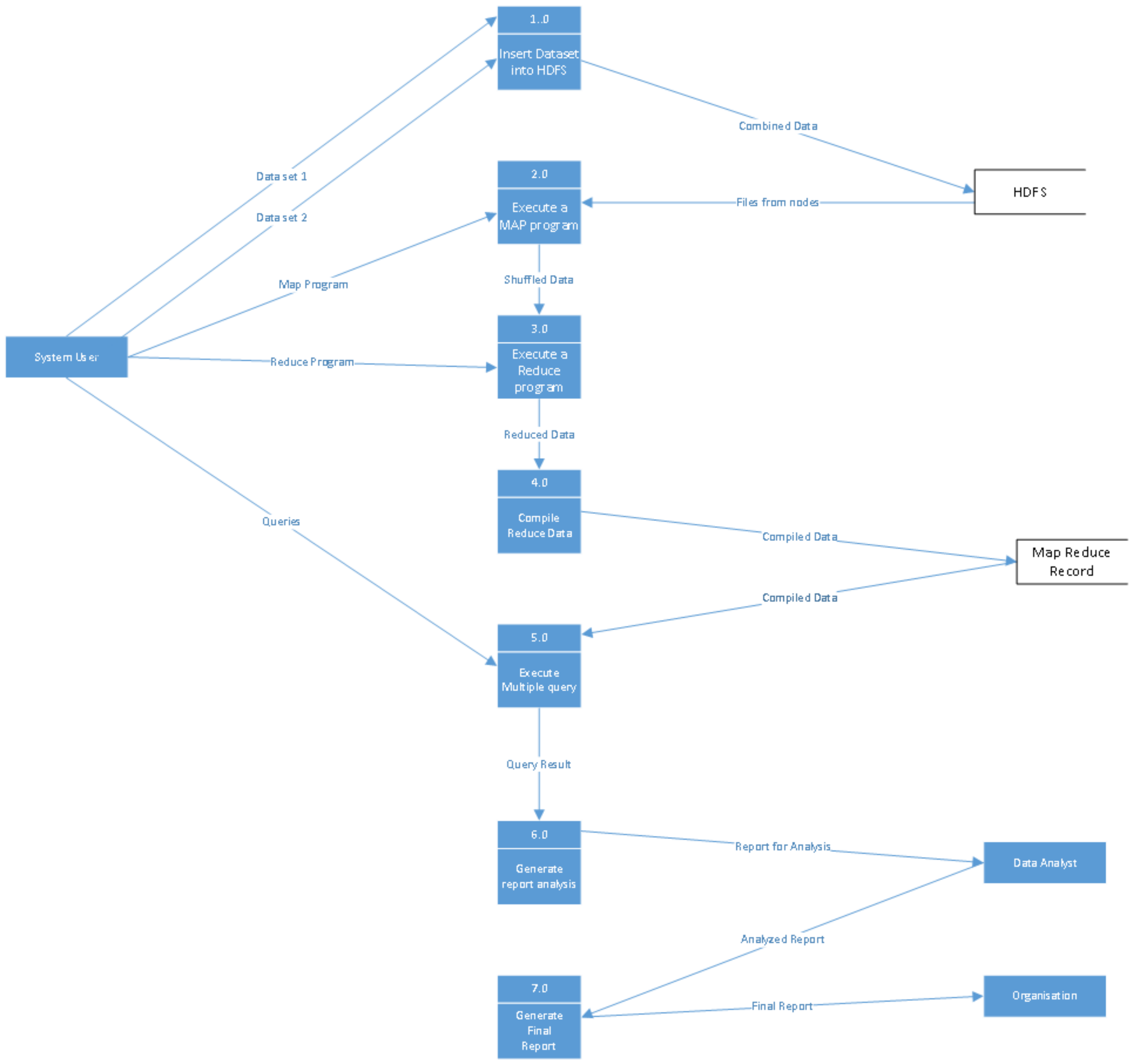
Table of Content

Data Model


Movie
product/productId
review/userId
review/profileName
review/helpfulness
review/score
review/time
review/summary
review/text
Movies
MovieID
Title
Genre
Users
UserID
Gender
Age
Occupation
Zip-Code
Fact Table
MovieID
Genre
Age
Ratings
review/text
?
Ratings
MovieID
UserID
Timestamp
Rating

In our Project we have two different type of Dataset. First Dataset consist of 3 files {USERS, MOVIES and RATINGS} and another dataset consist of a movie file. IN the first dataset user file consist of 5 attributes (User ID, Gender, Age, Occupation, and Zip code) from this we are using Age attribute which help us to select movie based on the particular age group. IN movie file there are three attribute Movie ID, title and genre from this we are using Movie ID and genre which help us to select movie based on genre. IN Rating file we have user ID, Movie ID and Ratings from which it will help us to find the interest of particular genre movie based on the age group. The second Dataset consist of a movie file which has product/product ID, review/user ID, review/profile Name, review/helpfulness, review/score, review/time, review/summary, review/text from which we are using review/text unstructured data.

Data flow Diagram



DFD Description

At first, system user will insert the datasets in to the HDFS which consists of a commodity hardware over which master node and slave nodes runs. In the HDFS the combined data which is coming from the process is stored in the form of files on the slave nodes and the metadata of these files stored in the master node.

Now, the system user will write a map program to acquire desire result from the data which is stored in the HDFS in the form of files, and after executing a map program in the process, the shuffled data files (maps) will be generated which serve as an input to the next process.

In the next process, system user will write a reduce program which is dependent on the maps that is generated in the previous process. Here in this process we are executing the reduced program to retrieve reduced data corresponding to the maps that we have generated in our previous process.

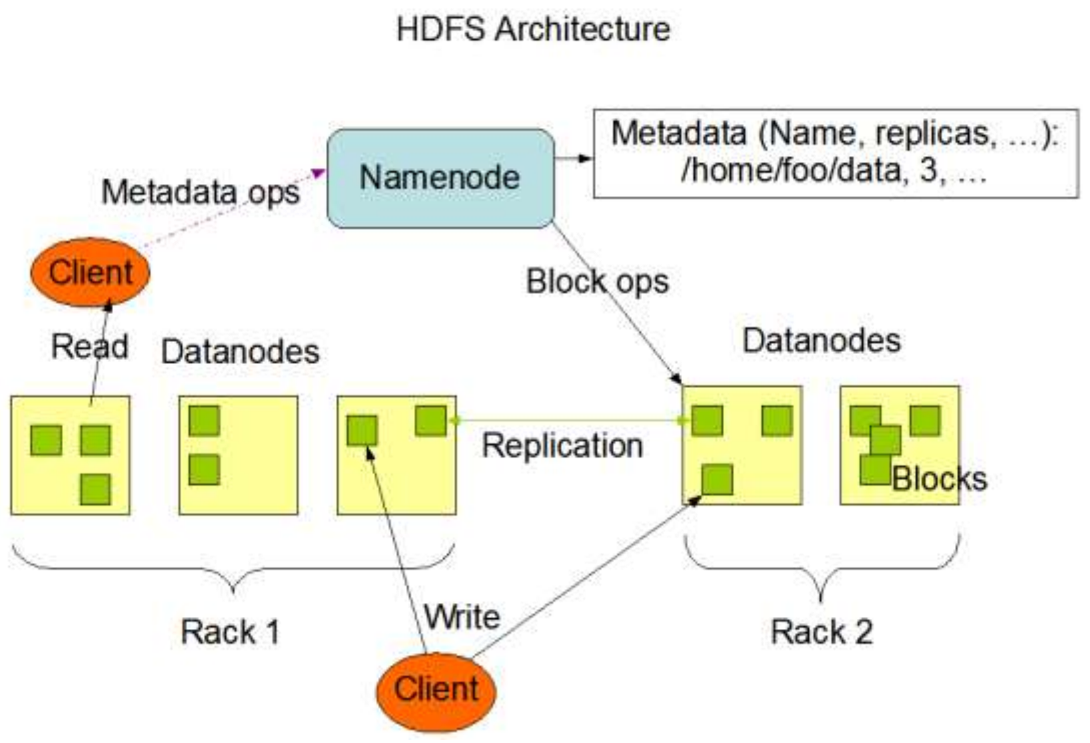
After that, the reduced data that came from the previous process will serve as the input for our next process of compiling reduced data. In this process the data will be compiled, filtered and arranged in a systematic way so that it can be stored in the map-reduce record database.

Now, the compiled result from the map-reduce database will serve as a input in our next process of executing multiple queries and this process will also need queries from the system user to generate query result in the form of sorted and meaningful items which we can use to generate analysis report.

In the next process we are generating the analysis report which is based on the query result of the previous process. This report will be analyzed by data analyst to find the solution of the client's problem and we will generate the final report which will be presented to the client so the client will be able to take better decision.

HDFS Architecture

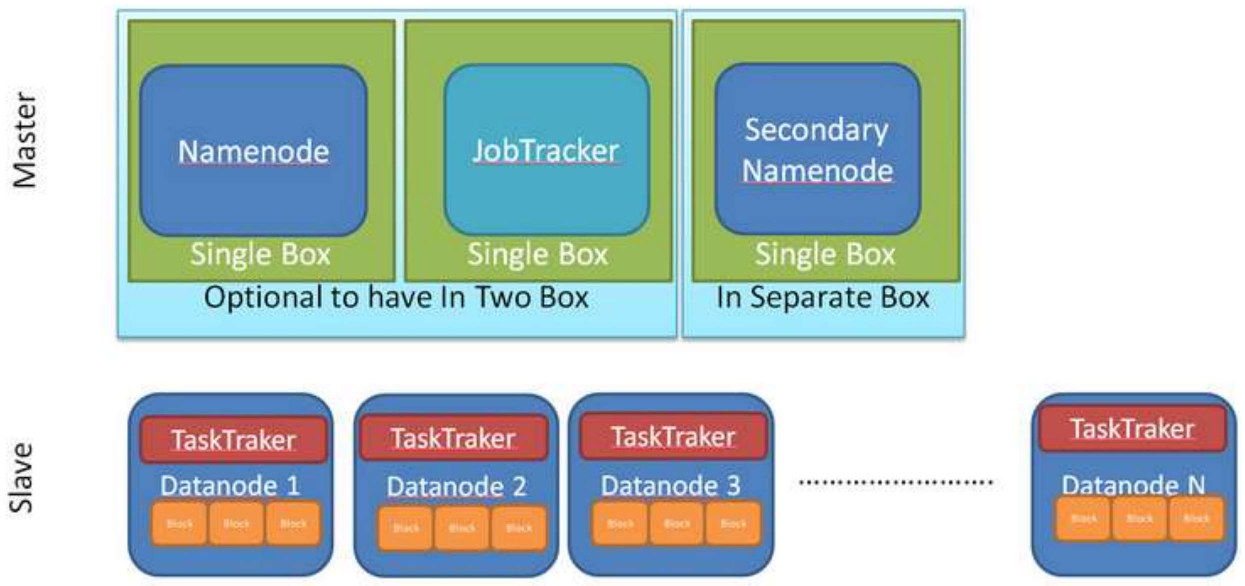
Hadoop distributed File system is a file system which is used to store data on a commodity hardware. It is highly fault tolerant and easy to install on low cost hardware. It is mainly designed for large Datasets. As the data nodes that are used in the Hadoop architecture are low cost there might be the chances of loss of data, to overcome the loss of data HDFS has provided with Failover instances **[1].** Typical architecture consists of three fail overs for one instance of the data node. The general file system in a Hadoop architecture varies from gigabytes to terabytes of data. The basic architecture of HDFS consists of master/slave nodes **[1].** Master slave is used to manage the client's and regulates access to files of the client. A file in a HDFS consists of many splits and is stored in a data node. Name node executes the file system namespace operations. Name node is responsible for determining the block size in the data node. Data node is responsible for serving read and write requests from the clients. Upon instructions from the Name node data node also performs block creation, deletion and replication of data **[2].** The base for the HDFS is Java language, so any machine that supports Java can run Name nodes and data nodes. HDFS is designed to have a data replication, each set of files that are stored in the data nodes are associated with the replication of similar type. HDFS supports write once and read many times **[3].** Data in HDFS is stored in the form of blocks. A typical block size consists of 64mb, so all the data that is coming from the client system is divided into 64mb blocks and is resided onto the different data nodes. Metadata is the data of data. It is used to store the Name node, Data node information for the client system.



Source: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Introduction

Job Tracker and Task Tracker

Job tracker runs on Name node. It reads the job files which has map tasks and reduce taks. Job files are created by the user. Using job files map and reduce files are created. Number of map tasks are equal to number of input splits. Job tracker sends Heaartbeats to the task tracker to check the availability of the data node. Typical wait time for the job tracker is 3 secs. Job tracker determines if the task tracker is alive or not, if it is not alive the job tracker assigns the data to another data node which the same replica of data. Client will split data into multiple input files **[3].**



Source: http://blogs.msdn.com/b/avkashchauhan/archive/2012/02/24/master-slave-architecture-in-hadoop.aspx

MapReduce

MapReduce is a framework that allows job based parallelized processing across data in the HDFS structure. MapReduce program consists of two procedures **Map()** and **Reduce()**. Map performs filtering and sorting such as sorting movies by customer rating into queue, one queue for each rating and Reduce performs summary operation such as counting number of movies in each queue. MapReduce manages all communications and transfer of data between the various sections of a system. It also provides fault tolerance.[4]

MapReduce can be looked as a three step computation:

1. Step: Map()

2. Step: Shuffle

3. Step: Reduce()

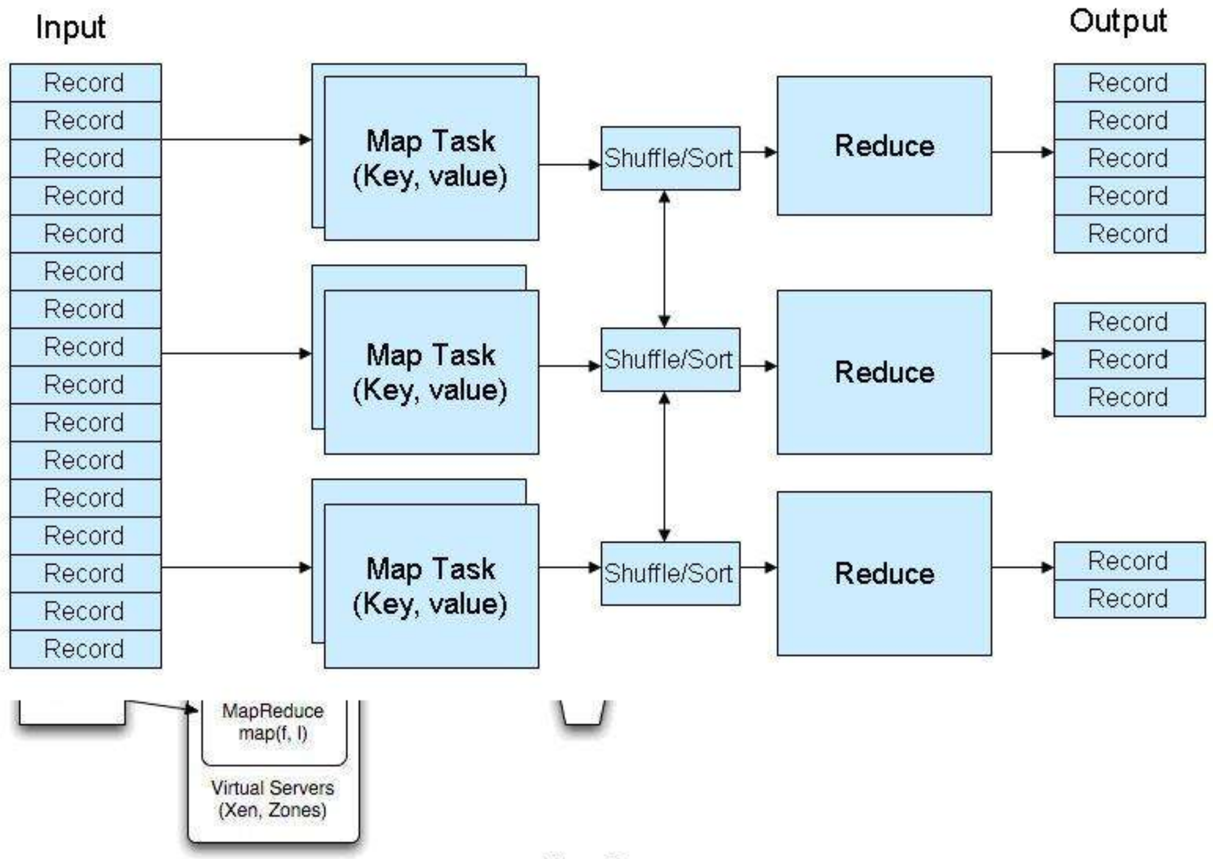


Figure 2

Source:http://mapreduce-deepak4278.blogspot.com/2010/08/hdfs-and-map-reduce-architecture.html

1. **Map():** Each slave node applies the map() function to the data and the output is stored in a temporary storage, the master node arranges that data for redundancies and only one copy is processed.

2. **Shuffle:** Slave nodes redistribute the data based on the keys produced by the map() function, such that the data that belongs to one key is to be located in same slave node.

3. **Reduce():** Now slave node will parallel process each group of output data, per key.

Map function defined with respect to data structured in pairs of key and value (key, value). The Map function applies to each pair in input, the output of this is a list of pairs, and then Reduce function arranges all pairs according to values.[4]

Use of Map Reduce Process for Movie Review (Project Description)

Of the Two data sets that's has been chosen for the Movie review HDFS Architecture, There are 4 input files. Users.dat file consists Gender, age occupation and Zip code of the system user. Second input file is Movies.dat file which has information about Movie name, movie id, and Genre. Third input file is the Rating.dat which stores information about User ID, movie ID, and ratings. The last input file is the Textual review which is unstructured and is considered as the secondary dataset and it consists of the data related to textual reviews of the movies.

Map Reduce Job tasks for the input Files

Map Tasks

- From Movie.dat file one job program is created which is map program which contains the list of movie genre's and movie id
- From rating.dat file one map program will be created which is used to find the ratings
- From the users.dat file map program is generated to find the age groups
- Textual review.dat file is used to find the Movie's textual review.

Reduce Tasks:

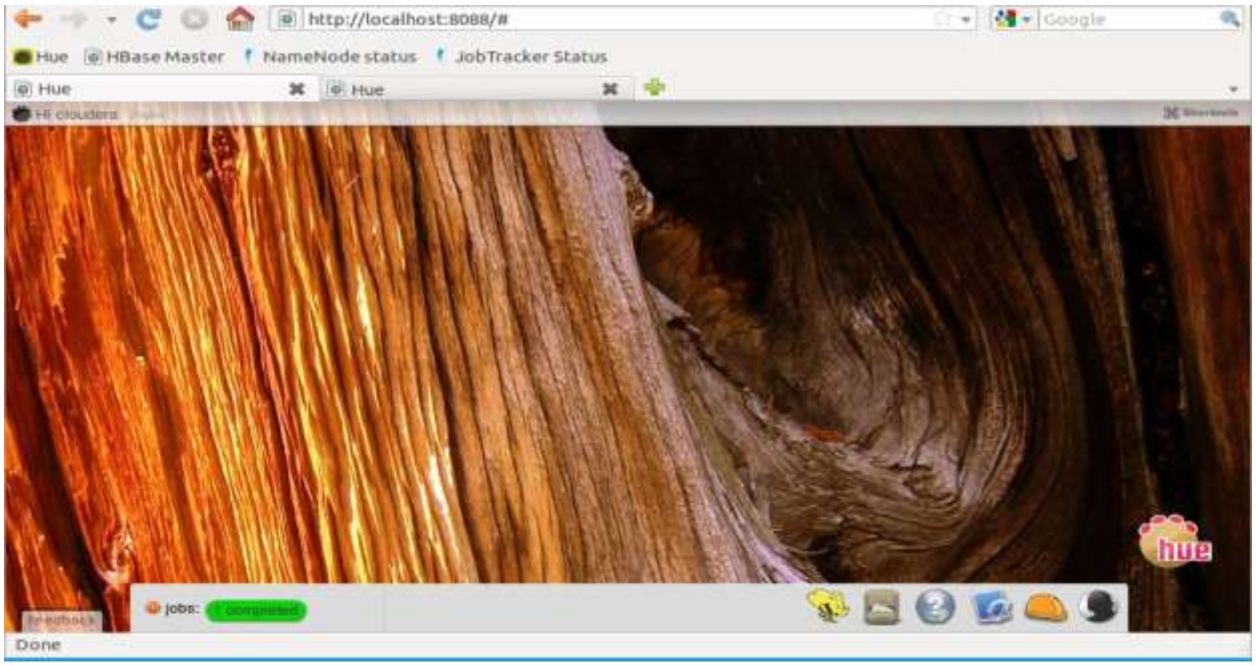
- From the above map Tasks we will create a reduce task whose output contains movie genre with high ratings corresponding to the particular age group.

Cloudera's Distributed Hadoop

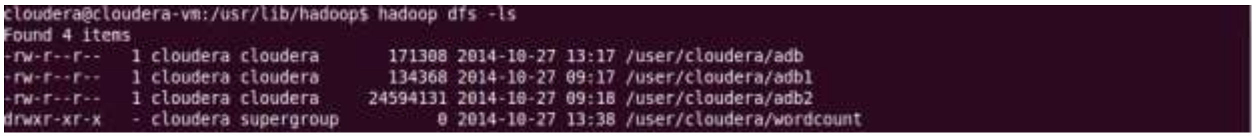
For our project we are using **Cloudera Distributions Including Apache Hadoop version 3 (CDH 3).** We chose this because it is open source and easily available for practicing Hadoop.

We are using CDH3 via VMware Player which startup like below:

At first we insert our datasets in to the HDFS via Hue which is the Cloudera's GUI for the Hadoop environment which looks like below:



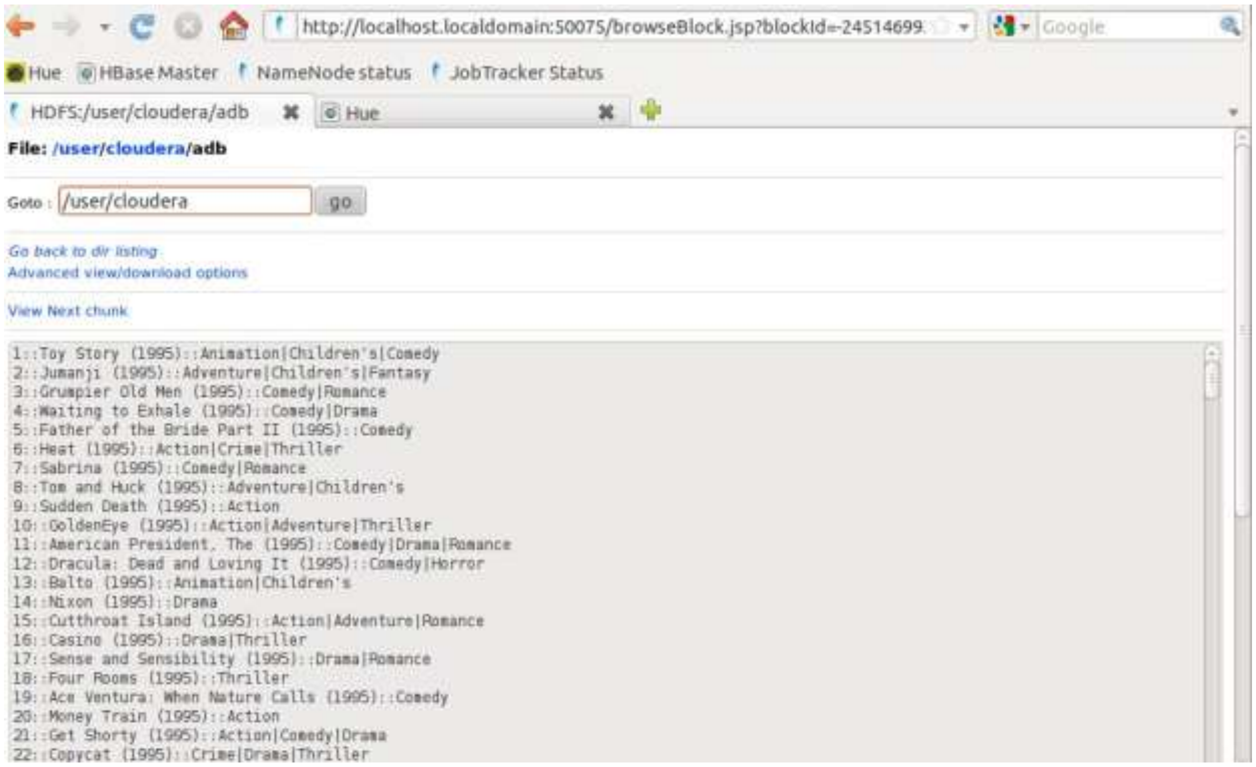
Here is the files that we inserted in to the HDFS:



```
cloudera@cloudera-vm:/usr/lib/hadoop$ hadoop dfs -ls
Found 4 items
-rw-r--r--   1 cloudera cloudera      171308 2014-10-27 13:17 /user/cloudera/adb
-rw-r--r--   1 cloudera cloudera      134368 2014-10-27 09:17 /user/cloudera/adb1
-rw-r--r--   1 cloudera cloudera    24594131 2014-10-27 09:18 /user/cloudera/adb2
drwxr-xr-x   - cloudera supergroup         0 2014-10-27 13:38 /user/cloudera/wordcount
```

The files name are adb, adb1, adb2 as shown above, we need to insert one more file to the HDFS, that contains the textual review of the movies but the size of this file is 2.96 GB and we come up with some problem loading this in to the HDFS, so we decided to split this file and load this to the system latter.

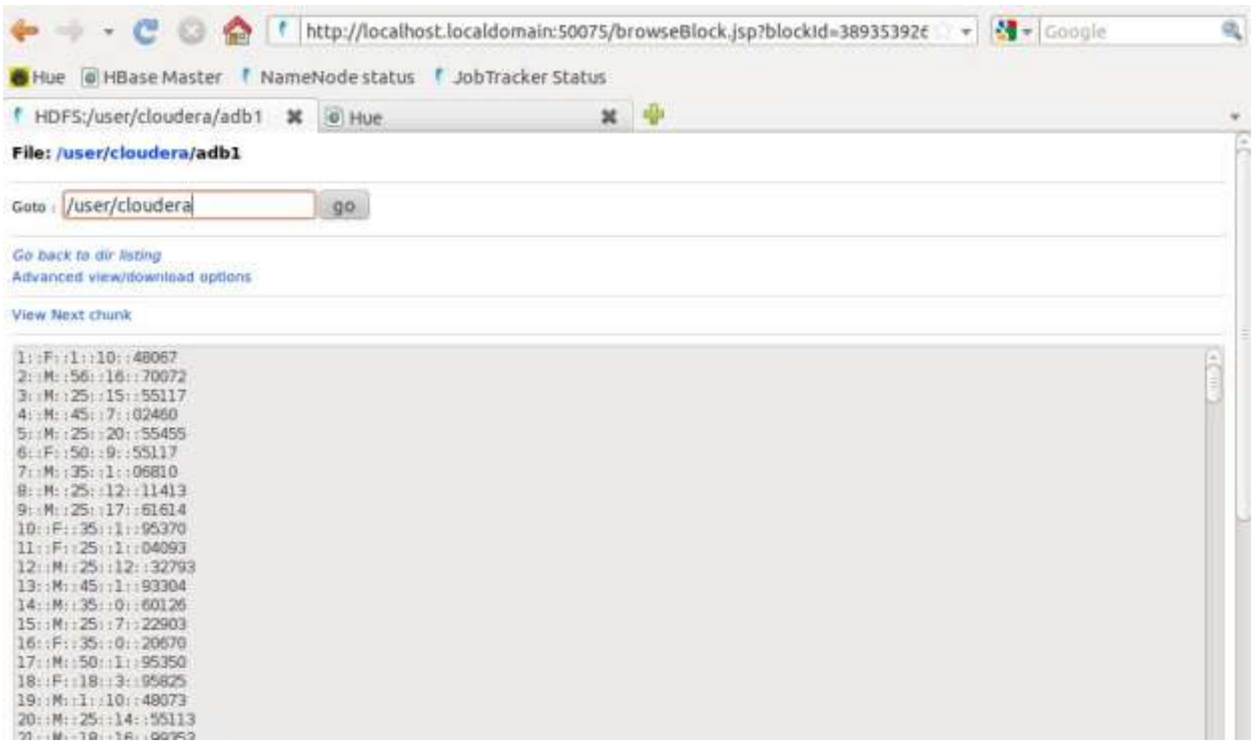
The data file adb contains the information about the movies as we can see below:



File: /user/cloudera/adb

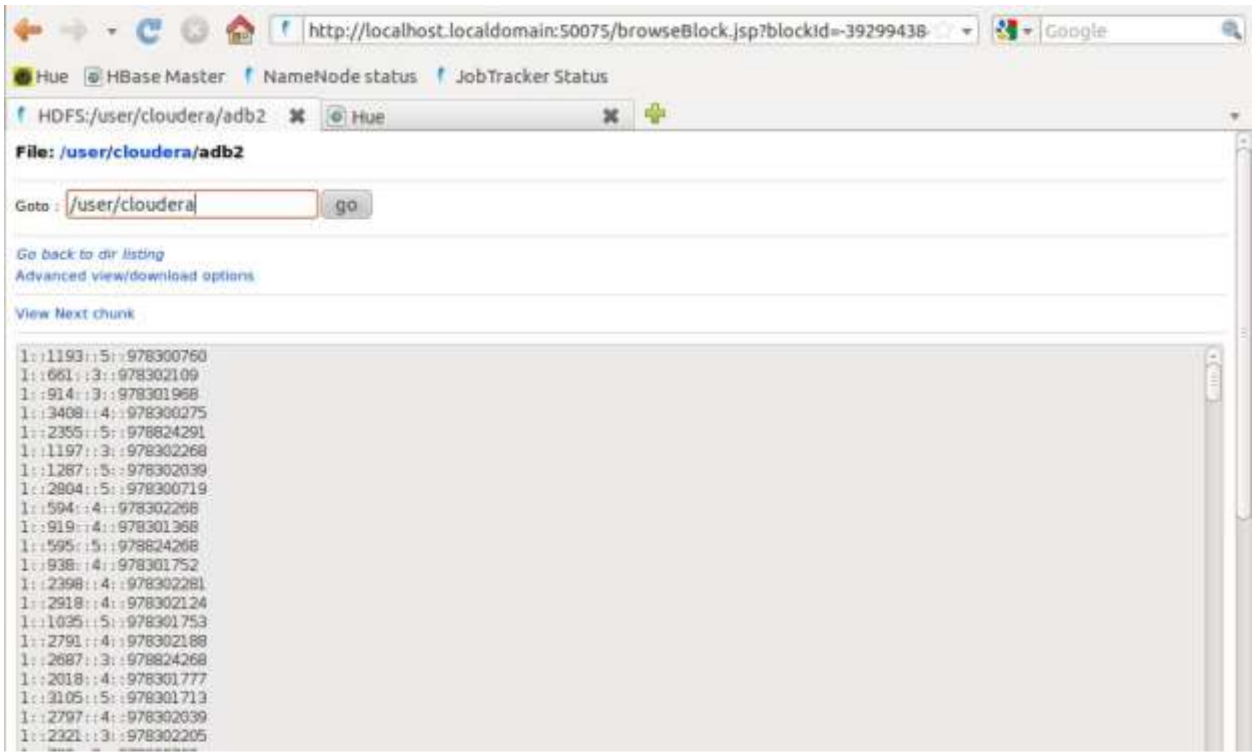
```
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thriller
11::American President, The (1995)::Comedy|Drama|Romance
12::Dracula: Dead and Loving It (1995)::Comedy|Horror
13::Balto (1995)::Animation|Children's
14::Nixon (1995)::Drama
15::Cutthroat Island (1995)::Action|Adventure|Romance
16::Casino (1995)::Drama|Thriller
17::Sense and Sensibility (1995)::Drama|Romance
18::Four Rooms (1995)::Thriller
19::Ace Ventura: When Nature Calls (1995)::Comedy
20::Money Train (1995)::Action
21::Get Shorty (1995)::Action|Comedy|Drama
22::Copycat (1995)::Crime|Drama|Thriller
```

The data file adb1 contains the information about the users (like gender, age, zip code, etc.) as we can see below:



```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
9::M::25::17::61614
10::F::35::1::95370
11::F::25::1::04093
12::M::25::12::32793
13::M::45::1::93304
14::M::35::0::60126
15::M::25::7::22903
16::F::35::0::20670
17::M::50::1::95350
18::F::18::3::95825
19::M::1::10::48073
20::M::25::14::55113
```

The data file adb2 contains the information about the movie ratings (out of 5) as we can see below:

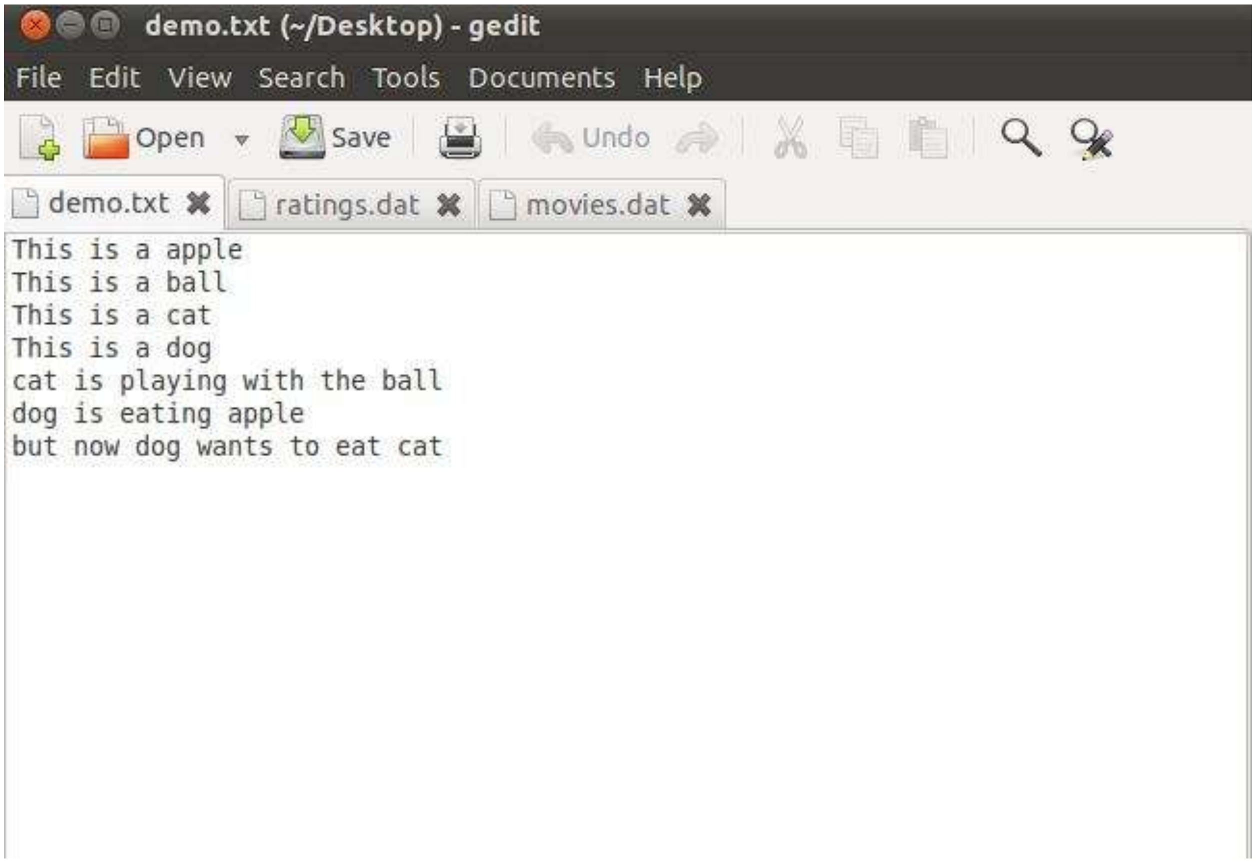


```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
1::595::5::978824268
1::938::4::978301752
1::2398::4::978302281
1::2918::4::978302124
1::1035::5::978301753
1::2791::4::978302188
1::2687::3::978824268
1::2018::4::978301777
1::3105::5::978301713
1::2797::4::978302039
1::2321::3::978302205
```

We are currently working on a MapReduce tasks for our datasets but we are not able to find the desired result that we need for our project. So for this milestone and just to show how MapReduce work in the

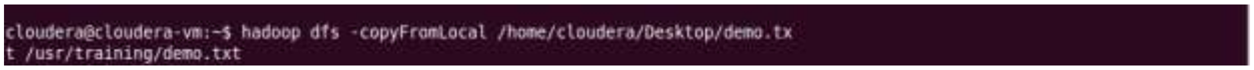
Hadoop environment, we demonstrate MapReduce tasks with the help of simple text document and the simple Map program for word count.

Sample Text File demo.txt contains the set of statements as below:


demo.txt (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
demo.txt
ratings.dat
movies.dat

```
This is a apple
This is a ball
This is a cat
This is a dog
cat is playing with the ball
dog is eating apple
but now dog wants to eat cat
```

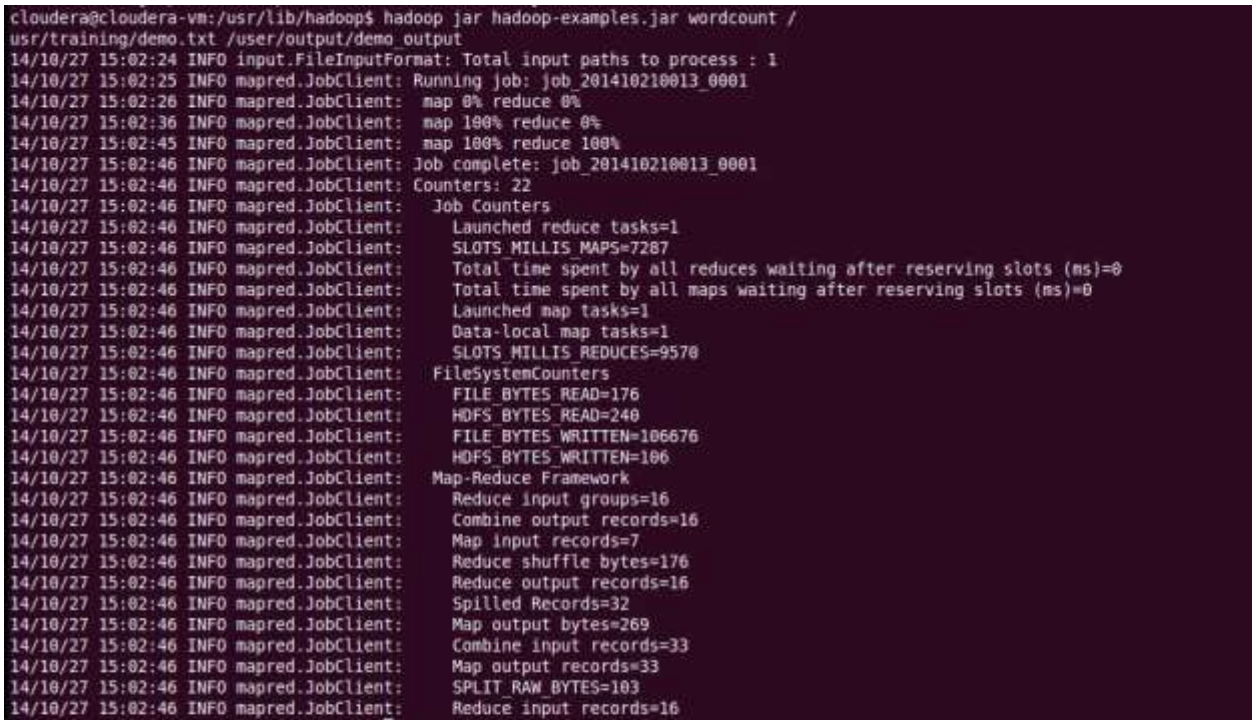
Then we load this file into HDFS:



```
cloudera@cloudera-vm:~$ hadoop dfs -copyFromLocal /home/cloudera/Desktop/demo.tx
t /usr/training/demo.txt
```

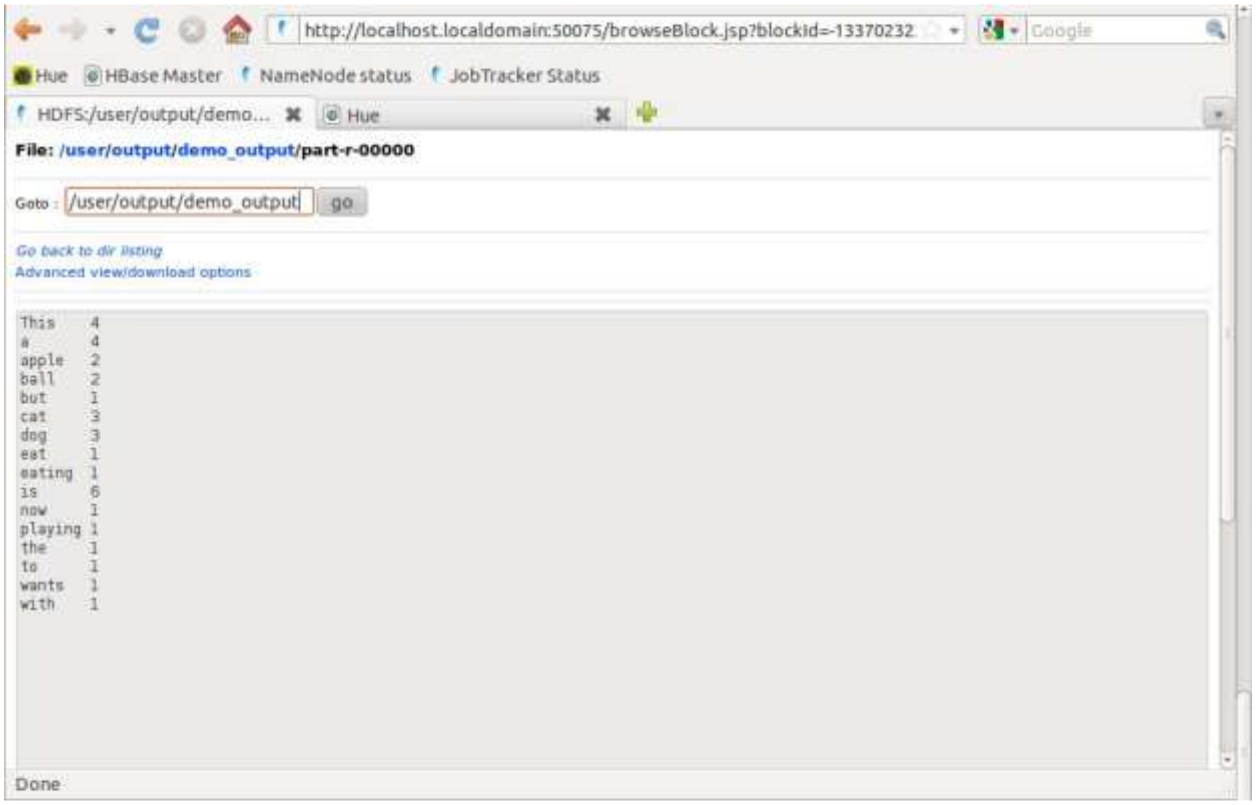
After that we apply Map task on the file to get the word count of a particular word. The source of Map program is http://www.cloudera.com/content/cloudera/en/documentation/HadoopTutorial/CDH4/Hadoop-Tutorial/ht_wordcount1_source.html

The detailed execution of Map task is shown below:



```
cloudera@cloudera-vm:/usr/lib/hadoop$ hadoop jar hadoop-examples.jar wordcount /
usr/training/demo.txt /user/output/demo_output
14/10/27 15:02:24 INFO input.FileInputFormat: Total input paths to process : 1
14/10/27 15:02:25 INFO mapred.JobClient: Running job: job_201410210013_0001
14/10/27 15:02:26 INFO mapred.JobClient:  map 0% reduce 0%
14/10/27 15:02:36 INFO mapred.JobClient:  map 100% reduce 0%
14/10/27 15:02:45 INFO mapred.JobClient:  map 100% reduce 100%
14/10/27 15:02:46 INFO mapred.JobClient: Job complete: job_201410210013_0001
14/10/27 15:02:46 INFO mapred.JobClient: Counters: 22
14/10/27 15:02:46 INFO mapred.JobClient:   Job Counters
14/10/27 15:02:46 INFO mapred.JobClient:     Launched reduce tasks=1
14/10/27 15:02:46 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=7287
14/10/27 15:02:46 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
14/10/27 15:02:46 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
14/10/27 15:02:46 INFO mapred.JobClient:     Launched map tasks=1
14/10/27 15:02:46 INFO mapred.JobClient:     Data-local map tasks=1
14/10/27 15:02:46 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=9570
14/10/27 15:02:46 INFO mapred.JobClient:   FileSystemCounters
14/10/27 15:02:46 INFO mapred.JobClient:     FILE_BYTES_READ=176
14/10/27 15:02:46 INFO mapred.JobClient:     HDFS_BYTES_READ=240
14/10/27 15:02:46 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=106676
14/10/27 15:02:46 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=106
14/10/27 15:02:46 INFO mapred.JobClient:   Map-Reduce Framework
14/10/27 15:02:46 INFO mapred.JobClient:     Reduce input groups=16
14/10/27 15:02:46 INFO mapred.JobClient:     Combine output records=16
14/10/27 15:02:46 INFO mapred.JobClient:     Map input records=7
14/10/27 15:02:46 INFO mapred.JobClient:     Reduce shuffle bytes=176
14/10/27 15:02:46 INFO mapred.JobClient:     Reduce output records=16
14/10/27 15:02:46 INFO mapred.JobClient:     Spilled Records=32
14/10/27 15:02:46 INFO mapred.JobClient:     Map output bytes=269
14/10/27 15:02:46 INFO mapred.JobClient:     Combine input records=33
14/10/27 15:02:46 INFO mapred.JobClient:     Map output records=33
14/10/27 15:02:46 INFO mapred.JobClient:     SPLIT_RAW_BYTES=103
14/10/27 15:02:46 INFO mapred.JobClient:     Reduce input records=16
```

The result of the Map task is as follows:


http://localhost.localdomain:50075/browseBlock.jsp?blockId=-13370232
Google
Hue
HBase Master
NameNode status
JobTracker Status
HDFS:/user/output/demo...
Hue
Done

File: /user/output/demo_output/part-r-00000

Goto : /user/output/demo_output go

Go back to dir listing
Advanced view/download options

```
This    4
a       4
apple   2
ball    2
but     1
cat     3
dog     3
eat     1
eating  1
is      6
now     1
playing 1
the     1
to      1
wants   1
with    1
```

Conclusion

We concluded that we successfully able to deploy Hadoop environment for our project and successfully configure the Hadoop cluster according to the requirements. We already inserted the files in to the Hadoop environment (HDFS) and we also started designing our Map Reduce tasks for the project and we will complete them in the next milestone.

References

[1]. Borthakur, D. (2008). HDFS architecture guide. HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs

[2]. Dai Yuefa, W. B., Yaqiang, G., Quan, Z., & Chaojing, T. (2009, November). Data security model for cloud computing. In Proceedings of the 2009 International Workshop on Information Security and Application (IWISA 2009) Qingdao, China.

[3]. Yang, G. (2011, October). The application of MapReduce in the cloud computing. In Intelligence Information Processing and Trusted Computing (IPTC), 2011 2nd International Symposium on (pp. 154-156). IEEE.

[4]. http://en.wikipedia.org/wiki/MapReduce

[5]. YouTube Tutorial 'Hadoop MapReduce Fundamentals' by Lynn Langit