CHAPTER 1- ORGANIZATION OVERVIEW

Brillica Services Pvt. Ltd. is Technology Provider in Dehradun, Uttarakhand. Brillica provide specialized courses in DATA SCIENCE, INTERNET OF THINGS, PYTHON, MACHINE LEARNING with PYTHON, JAVA, CISCO, ARTIFICIAL INTELLIGENCE, ANDROID APP DEVELOPMENT, MICROSOFT and many other technologies with LIVE PROJECTS.
Here students don’t just complete the course with a single project, they complete their course with a deep practical knowledge and multiple projects.
Brillica Services has successfully executed projects in *African continent* and is still working with them towards providing emerging technology solutions for various corporates**.**Brillica services is known for its quality training because brillica believe in Quality and knowledge.

Brillica provide certified and job oriented training.Brillica has specialized in 3 important domains that is, Training, Development and Consultancy. Brillica provide online training as well as tutorial training. Brillica experts help student to prepare live project . Brillica has an excellent team of experts who understand student needs and guide them for their future and prepare them for professional world. Brillica gives training to students in accordance with today’s industry demand. Brillica provide training to students as well as professionals too.

1.2 WHY BRILLICA

Brillica Services focuses on the following key features :

- Corporate Training
- Industrial Training
- Online Training
- Language Zone
- Business and Resource Consulting
- Research and Development
- Skills for Schools and Colleges

Brillica Services is the Place where Trainer and the Students share a comfortable workplace which makes their efforts successful. They give every student and employee a fair time to share their views and thoughts so that we could make them doubt free. Brillica strive to keep our journey of innovative learning and knowledge discovery balanced for all our training courses.

1.3 REPUTED & EXPERT TRAINER

The trainers at Brillica are well versed with knowledge base and training skills which impart good quality education coupled with concept building to students. The trainers are highly supportive and help the students in clearing the doubts easily. The trainers keep students updated about the new technologies and ways to use them effectively.

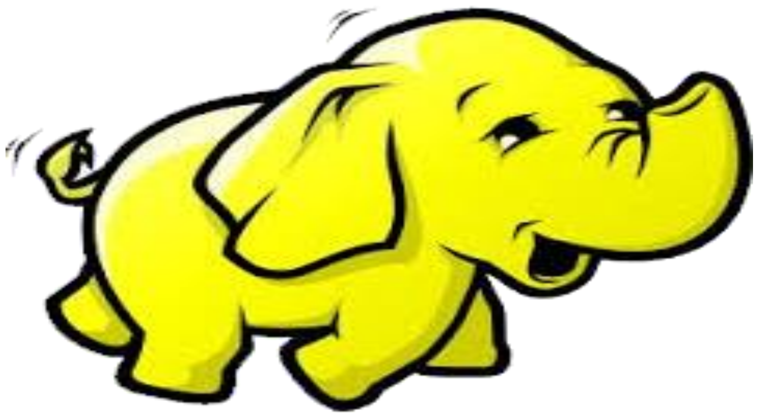
1.4 INDUSTRY ORIENTED PRACTICAL TRAINING

The students of Brillica get good exposure from the industry oriented practical training which helps them to develop excellent practical knowledge base equipped with developed thinking skills. Ms. Garima is industry level big data trainer and she has worked on many projects and she helped us on how to work on Hadoop. Students get the highest level of Industry Oriented Practical Training provided by Brillica which helps them to reach international standards. Further there are a lot of corporate interactions in which eminent technocrats' converse with the students for giving them practical exposure.

1.5 JOB ORIENTED TRAINING PROGRAM

Brillica Services Pvt. Ltd is associated with IBM as registered business partner and with Microsoft as authorized education partner to provide job oriented training programs. Brillica has exclusive partnership with MAPR and partner network with amazon web services.

CHAPTER 2- INTRODUCTION OF PROJECT/TRAINING



Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

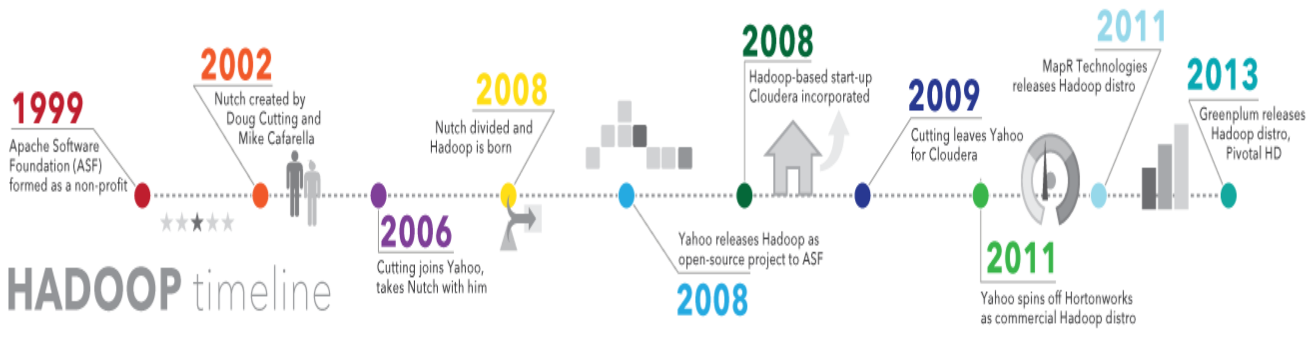
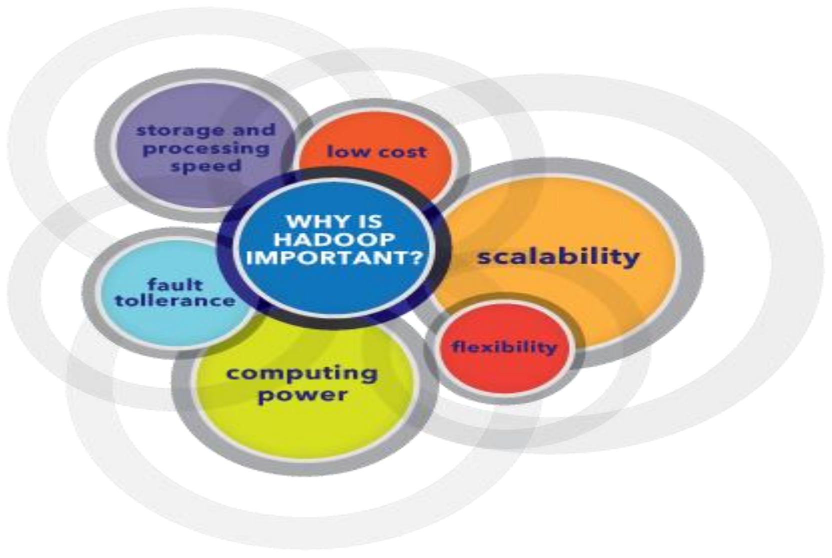

1999
Apache Software Foundation (ASF) formed as a non-profit
2002
Nutch created by Doug Cutting and Mike Cafarella
2006
Cutting joins Yahoo, takes Nutch with him
2008
Nutch divided and Hadoop is born
2008
Yahoo releases Hadoop as open-source project to ASF
2008
Hadoop-based start-up Cloudera incorporated
2009
Cutting leaves Yahoo for Cloudera
2011
Yahoo spins off Hortonworks as commercial Hadoop distro
2011
MapR Technologies releases Hadoop distro
2013
Greenplum releases Hadoop distro, Pivotal HD
HADOOP timeline

Figure 1

Hadoop was created by Doug Cutting and Mike Cafarella in 2005. It was originally developed to support distribution for the Nutch search engine project. Doug, who was working at Yahoo! at the time and is now Chief Architect of Cloudera, named the project after his son's toy elephant.

Why HADOOP is important?



What are the challenges of using Hadoop?

- MapReduce programming is not a good match for all problems**.**
- There’s a widely acknowledged talent gap
- Data security
- Full-fledged data management and governance

Here are five examples of Hadoop use cases:

1. Financial services companies use analytics to assess risk, build investment models, and create trading algorithms; Hadoop has been used to help build and run those applications.
2. Retailers use it to help analyze structured and unstructured data to better understand and serve their customers.
3. In the asset-intensive energy industry Hadoop-powered analytics are used for predictive maintenance, with input from Internet of Things (IoT) devices feeding data into big data programs.
4. Telecommunications companies can adapt all the aforementioned use cases. For example, they can use Hadoop-powered analytics to execute predictive maintenance on their

infrastructure. Big data analytics can also plan efficient network paths and recommend optimal locations for new cell towers or other network expansion. To support customer-facing operations telcos can analyze customer behavior and billing statements to inform new service offerings.

5. There are numerous public sector programs, ranging from anticipating and preventing disease outbreaks to crunching numbers to catch tax cheats.

<u>Features of Hadoop</u>

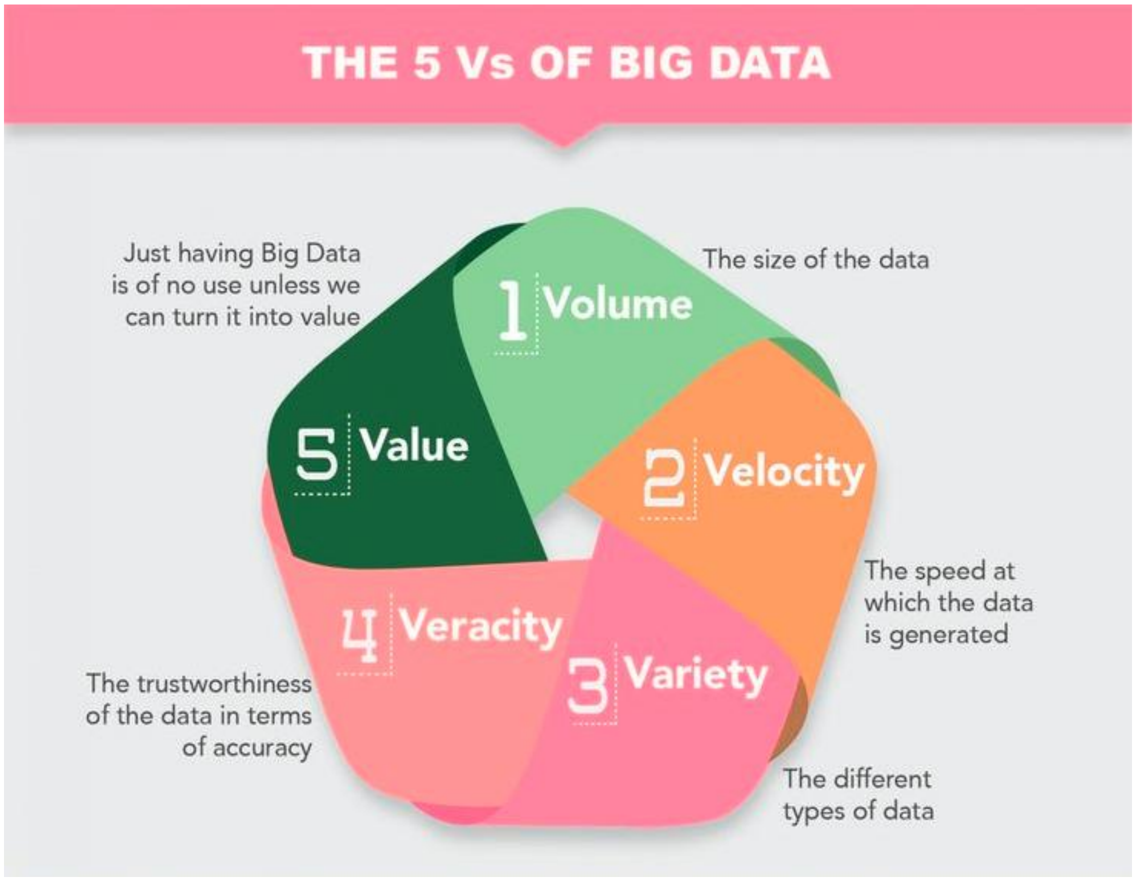

THE 5 Vs OF BIG DATA
Just having Big Data is of no use unless we can turn it into value
1 Volume
The size of the data
5 Value
2 Velocity
The speed at which the data is generated
4 Veracity
3 Variety
The trustworthiness of the data in terms of accuracy
The different types of data

Figure 2

Apache Hadoop is a Big Data framework that is part of the **Apache Software Foundation**. Hadoop is an open source software project that is extensively used by some of the biggest organizations in the world for distributed storage and processing of data on a level that is just enormous in terms of volume. That's the reason the Apache Hadoop runs its processing on large computer clusters built on commodity hardware. Some of the features of the Hadoop platform are that it can be efficiently used for data storage, processing, access, analysis, governance, security, operations and deployment.

About Project

Big data analytics in Hadoop works on a cluster of nodes and there is high availability cluster and Non – high availability cluster. I have worked on High availability cluster. A high availability cluster (HA cluster) is a set of computer hardware pieces that provide solutions for redundant operations, in the event of network component failure. Many HA cluster systems are designed to support enterprises that cannot afford to have critical interruptions to business processes if a single computer or server goes down, or if a single office is compromised in an emergency situation. So in this project , our team worked on setting up High availability cluster consisting of 4 nodes , in which my role was to set up configuration file for file block size and replication factor then at last to check if cluster is high availability or not. After this we learnt about Hadoop architecture and various tools present like apache pig , apache hive , sqoop flume , kafka etc. Among these to make my project i worked in continuation on apache pig and apache hive . In apache hive , I made minor project on time stamp setting of different vehicle and driver from tables. In major project , I Performed analytics on consumer complaint data and analysed it by suitable graphs and apache pig. Firstly, I loaded the csv file in Hadoop ecosystem and then by using apache scripts I performed analytics on it.

CHAPTER-3 MODULES OF PROJECT/TRAINING

In training, as I was working on Hadoop so first thing is thorough knowledge of Hadoop framework and How each component in its framework fulfils task allotted to it.

Hadoop Framework

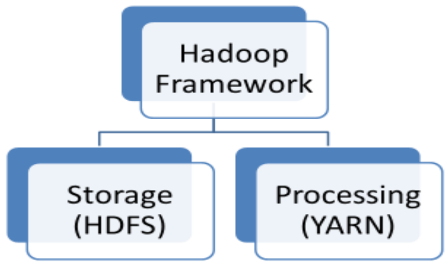


Figure 3

The **Hadoop framework** application **works** in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

MapReduce

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

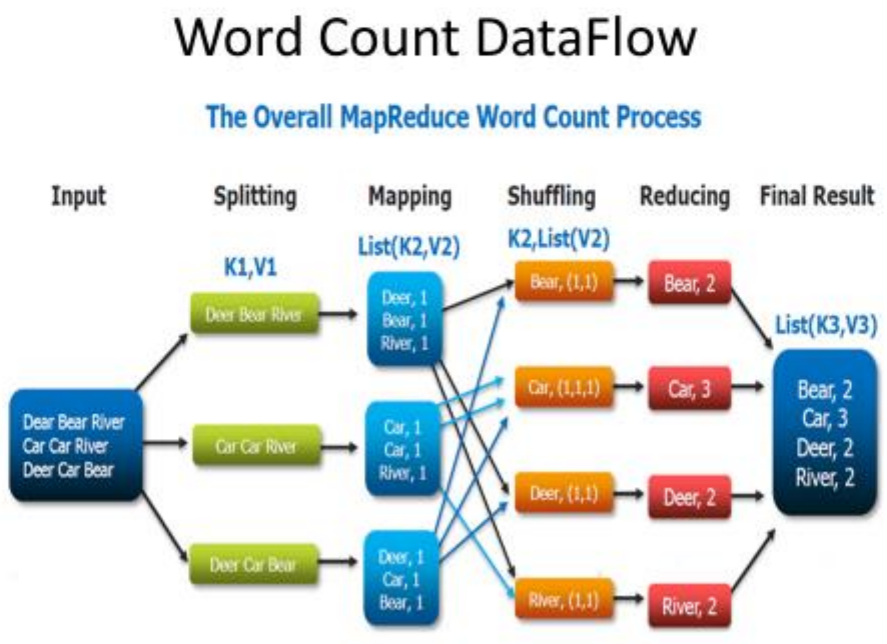


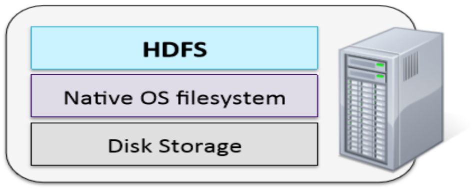
Figure 4

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules −

- **Hadoop Common** − These are Java libraries and utilities required by other Hadoop modules.
- **Hadoop YARN** − This is a framework for job scheduling and cluster resource management.



Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

HDFS Architecture

Given below is the architecture of a Hadoop File System.

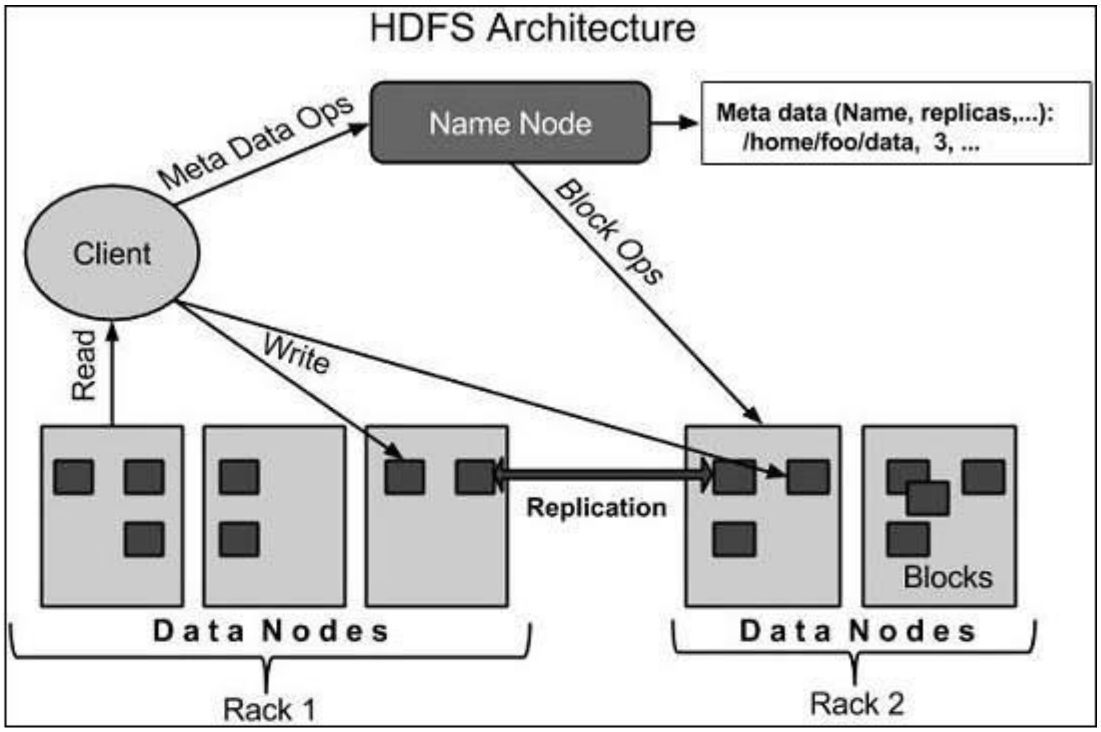


Figure 5

HDFS follows the master-slave architecture and it has the following elements.

Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks −

- Manages the file system namespace.
- Regulates client’s access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

Goals of HDFS

Fault detection and recovery − Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.

Huge datasets − HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.

Hardware at data − A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput

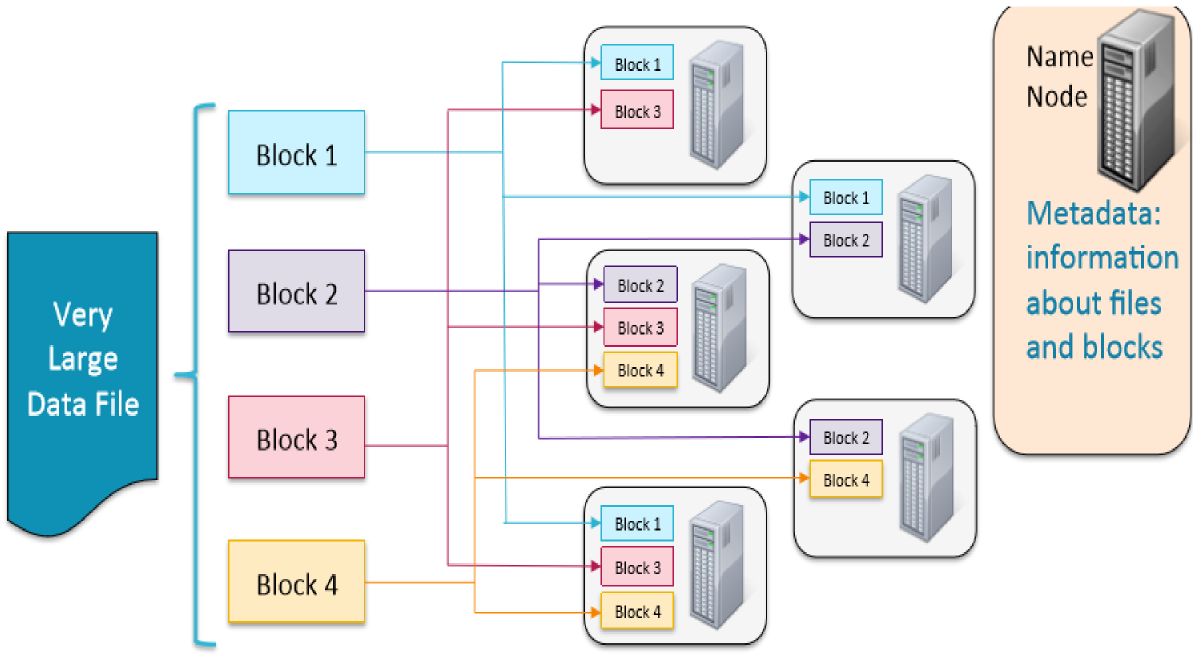


Figure 6

How Does Hadoop Work?

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs –

- Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
- These files are then distributed across various cluster nodes for further processing.
- HDFS, being on top of the local file system, supervises the processing.
- Blocks are replicated for handling hardware failure.
- Checking that the code was executed successfully.

- Performing the sort that takes place between the map and reduce stages.
- Sending the sorted data to a certain computer.
- Writing the debugging logs for each job.

WHAT IS APACHE HIVE?

Apache Hive is a data warehouse system built on top of Apache Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in various databases and file systems that integrate with Hadoop, including the MapR Data Platform with MapR XD and MapR Database. Hive offers a simple way to apply structure to large amounts of unstructured data and then perform batch SQL-like queries on that data. Hive easily integrates with traditional data center technologies using the familiar JDBC/ODBC interface.

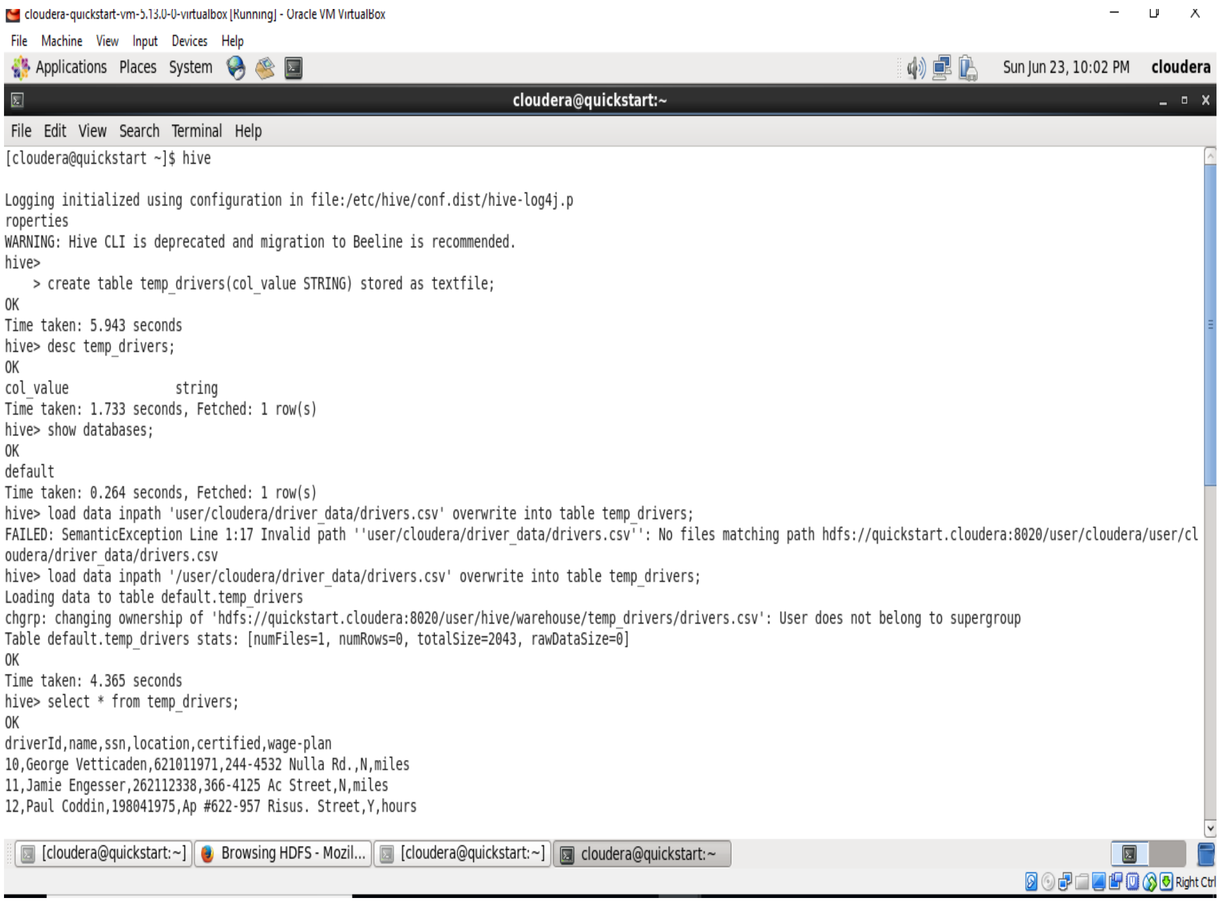
The Hive metastore provides a simple mechanism to project structure onto large amounts of unstructured data by applying a table schema on top of the data. This table abstraction of the underlying data structures and file locations presents users with a relational view of data in the file systems and NoSQL databases. Structure is applied to data at time of read, so users don’t need to worry about formatting the data when it is stored in their cluster. Data can be read from a variety of formats, from unstructured flat files with comma- or space-separated text, to semi-structured JSON files, to structured HBase tables.

Hive features an SQL-like programming interface called HiveQL to query data stored in various databases and file systems. HiveQL automatically translates SQL-like queries into batch MapReduce jobs.

Several efforts have emerged for faster execution of HiveQL or SQL on top of Hadoop:

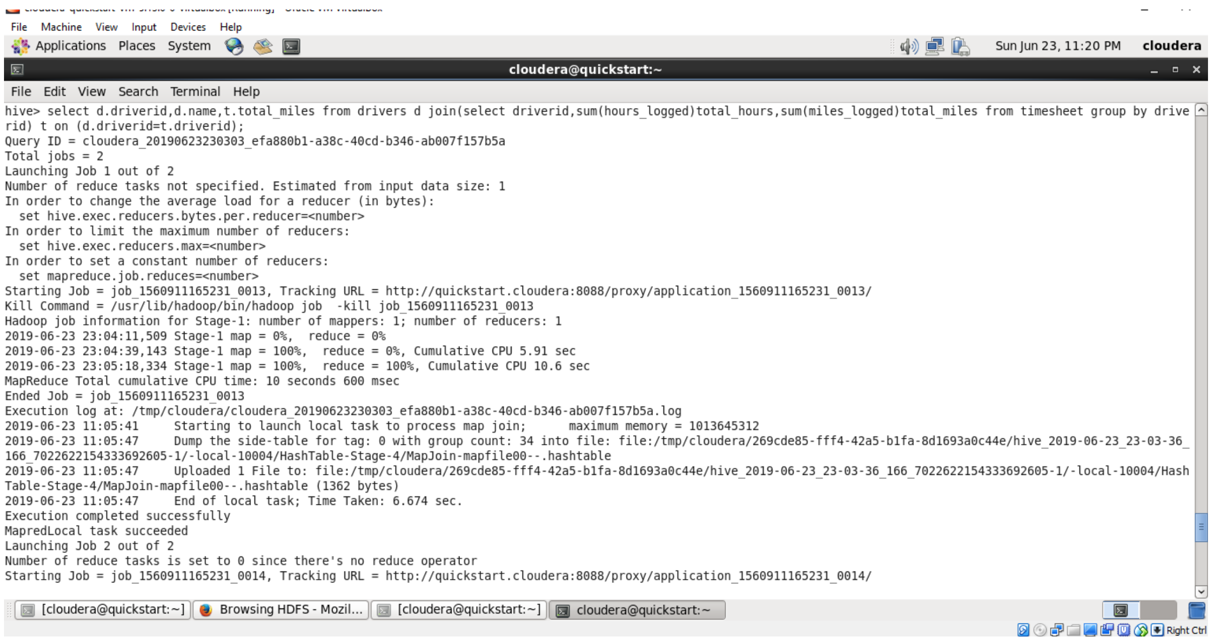
- **Apache Spark** is a powerful unified analytics engine for large-scale distributed data processing and machine learning. The Hive metastore can be used with Spark SQL and/or HiveQL can run on the Spark execution engine, optimizing workflows and offering in-memory processing to improve performance significantly.
- **Apache Drill** is an open source distributed SQL query engine offering fast in memory processing with ANSI SQL versus HiveQL. Drill provides the ability to leverage the metadata in the Hive metastore for querying. This is in addition to querying nested data with dynamic schemas.
- **Tez** has emerged as a complementary high-performance execution engine with the introduction of YARN as an independent resource manager. Hive can run on Tez, allowing queries to run significantly faster.
- **Impala** leverages Hive’s query language (HiveQL) and metastore to bring interactive SQL to Hadoop.

Create timesheet table by taking information from two table and merging them using apache hive


cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sun Jun 23, 10:02 PM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
[cloudera@quickstart:~]
Browsing HDFS - Mozil...
[cloudera@quickstart:~]
cloudera@quickstart:~
Right Ctrl

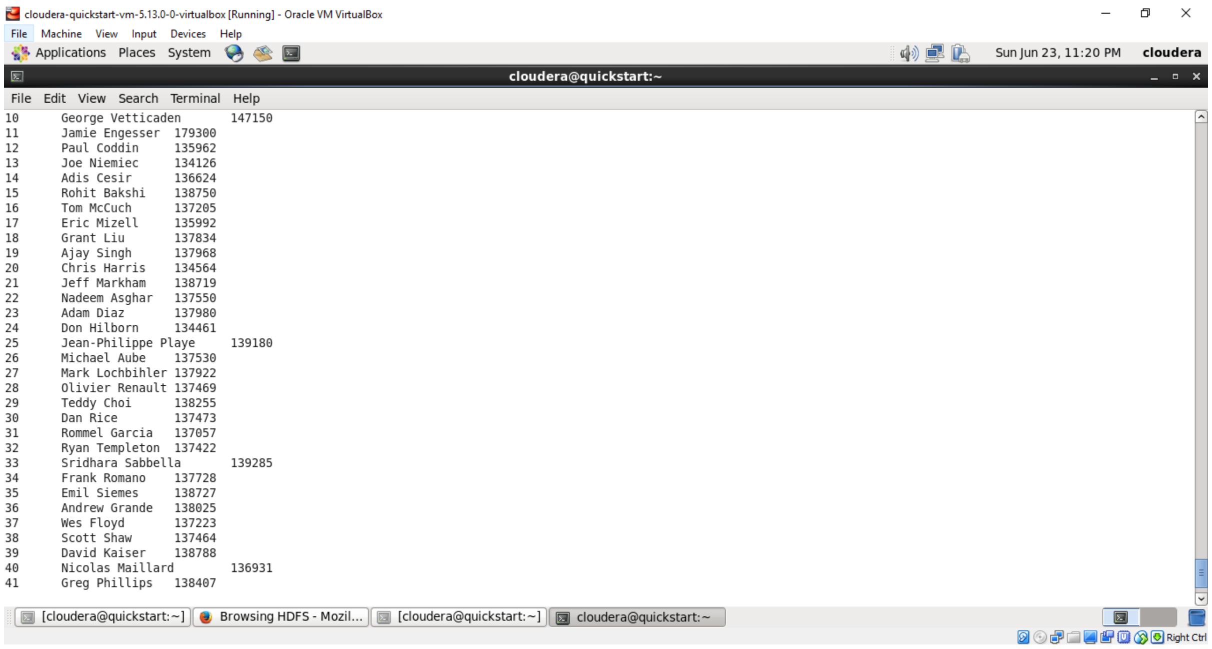
```
[cloudera@quickstart ~]$ hive

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>
    > create table temp_drivers(col_value STRING) stored as textfile;
OK
Time taken: 5.943 seconds
hive> desc temp_drivers;
OK
col_value               string
Time taken: 1.733 seconds, Fetched: 1 row(s)
hive> show databases;
OK
default
Time taken: 0.264 seconds, Fetched: 1 row(s)
hive> load data inpath 'user/cloudera/driver_data/drivers.csv' overwrite into table temp_drivers;
FAILED: SemanticException Line 1:17 Invalid path ''user/cloudera/driver_data/drivers.csv'': No files matching path hdfs://quickstart.cloudera:8020/user/cloudera/user/cl
oudera/driver_data/drivers.csv
hive> load data inpath '/user/cloudera/driver_data/drivers.csv' overwrite into table temp_drivers;
Loading data to table default.temp_drivers
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehouse/temp_drivers/drivers.csv': User does not belong to supergroup
Table default.temp_drivers stats: [numFiles=1, numRows=0, totalSize=2043, rawDataSize=0]
OK
Time taken: 4.365 seconds
hive> select * from temp_drivers;
OK
driverId,name,ssn,location,certified,wage-plan
10,George Vetticaden,621011971,244-4532 Nulla Rd.,N,miles
11,Jamie Engesser,262112338,366-4125 Ac Street,N,miles
12,Paul Coddin,198041975,Ap #622-957 Risus. Street,Y,hours
```

Create query to join the data (driverid,name,hours_logged,miles_logged)



```
hive> select d.driverid,d.name,t.total_miles from drivers d join(select driverid,sum(hours_logged)total_hours,sum(miles_logged)total_miles from timesheet group by drive
rid) t on (d.driverid=t.driverid);
Query ID = cloudera_20190623230303_efa880b1-a38c-40cd-b346-ab007f157b5a
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1560911165231_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1560911165231_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1560911165231_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2019-06-23 23:04:11,509 Stage-1 map = 0%,  reduce = 0%
2019-06-23 23:04:39,143 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.91 sec
2019-06-23 23:05:18,334 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.6 sec
MapReduce Total cumulative CPU time: 10 seconds 600 msec
Ended Job = job_1560911165231_0013
Execution log at: /tmp/cloudera/cloudera_20190623230303_efa880b1-a38c-40cd-b346-ab007f157b5a.log
2019-06-23 11:05:41     Starting to launch local task to process map join;      maximum memory = 1013645312
2019-06-23 11:05:47     Dump the side-table for tag: 0 with group count: 34 into file: file:/tmp/cloudera/269cde85-fff4-42a5-b1fa-8d1693a0c44e/hive_2019-06-23_23-03-36_
166_7022622154333692605-1/-local-10004/HashTable-Stage-4/MapJoin-mapfile00--.hashtable
2019-06-23 11:05:47     Uploaded 1 File to: file:/tmp/cloudera/269cde85-fff4-42a5-b1fa-8d1693a0c44e/hive_2019-06-23_23-03-36_166_7022622154333692605-1/-local-10004/Hash
Table-Stage-4/MapJoin-mapfile00--.hashtable (1362 bytes)
2019-06-23 11:05:47     End of local task; Time Taken: 6.674 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 2 out of 2
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1560911165231_0014, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1560911165231_0014/
```

This is how the data looks:



```
10      George Vetticaden       147150
11      Jamie Engesser  179300
12      Paul Coddin     135962
13      Joe Niemiec     134126
14      Adis Cesir      136624
15      Rohit Bakshi    138750
16      Tom McCuch      137205
17      Eric Mizell     135992
18      Grant Liu       137834
19      Ajay Singh      137968
20      Chris Harris    134564
21      Jeff Markham    138719
22      Nadeem Asghar   137550
23      Adam Diaz       137980
24      Don Hilborn     134461
25      Jean-Philippe Playe     139180
26      Michael Aube    137530
27      Mark Lochbihler 137922
28      Olivier Renault 137469
29      Teddy Choi      138255
30      Dan Rice        137473
31      Rommel Garcia   137057
32      Ryan Templeton  137422
33      Sridhara Sabbella       139285
34      Frank Romano    137728
35      Emil Siemes     138727
36      Andrew Grande   138025
37      Wes Floyd       137223
38      Scott Shaw      137464
39      David Kaiser    138788
40      Nicolas Maillard        136931
41      Greg Phillips   138407
```

What is Apache Pig?

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with **Hadoop**; we can perform all the data manipulation operations in Hadoop using Apache Pig.

To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using **Apache Pig**, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as **Pig Engine** that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

Why Do We Need Apache Pig?

Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers.

- Using **Pig Latin**, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses **multi-query approach**, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.
- Pig Latin is **SQL-like language** and it is easy to learn Apache Pig when you are familiar with SQL.
- Apache Pig provides many built-in operators to support data operations like joins, filters, ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

Features of Pig

Apache Pig comes with the following features −

- **Rich set of operators** − It provides many operators to perform operations like join, sort, filer, etc.
- **Ease of programming** − Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.
- **Optimization opportunities** − The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.
- **Extensibility** − Using the existing operators, users can develop their own functions to read, process, and write data.
- **UDF's** − Pig provides the facility to create **User-defined Functions** in other programming languages such as Java and invoke or embed them in Pig Scripts.
- **Handles all kinds of data** − Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

Hardware requirements

Device	: laptop, desktop computer
Processor	: Intel i3/AMD
RAM	: 8GB
Hard Disk	: 128GB

Software Requirements

Operating system : centos and cloudera

1. Oracle virtual Box/VMware workstation

2. Virtual image of Centos operating system

CHAPTER – 4 IMPLEMENTATION OF PROJECT

Setting Up High availability cluster

Following are the steps :

1.Stop the secondary namenode on tiger machine

```
$ sudo service hadoop-hdfs-secondarynamenode stop
```

2. Stop the namenode on Elephant machine

```
$ sudo service hadoop-hdfs-namenode stop
```

3. Stop datanodes on all the 4 machines seperately

```
$ sudo service hadoop-hdfs-datanode stop
```

****

4.On Elephant machine:

```
$sudo vim /etc/hadoop/conf/core-site.xml
```

Press "ESC" key and press "INSERT" key and change the value of the property fs.defaultFS to :

hdfs://mycluster

add new property in same file core-site.xml before </configuration>

```
<property>

    <name>ha.zookeeper.quorum</name>

    <value>elephant:2181,tiger:2181,horse:2181</value>

  </property>
```

Press "ESC" key and press colon (:) and type 'wq' to save the file and close.

****

5. On Elephant machine:

```
sudo vim /etc/hadoop/conf/hdfs-site.xml
```

Press "ESC" key and press "INSERT" key add the below properties without deleting existing properties before </configuraion>

```
<property>

   <name>dfs.nameservices</name>

   <value>mycluster</value>

 </property>

 <property>

   <name>dfs.ha.namenodes.mycluster</name>

   <value>nn1,nn2</value>

 </property>

 <property>

   <name>dfs.namenode.rpc-address.mycluster.nn1</name>

   <value>elephant:8020</value>

 </property>

 <property>

   <name>dfs.namenode.rpc-address.mycluster.nn2</name>

   <value>tiger:8020</value>

 </property>

 <property>

   <name>dfs.namenode.http-address.mycluster.nn1</name>
```

```
    <value>elephant:50070</value>
  </property>
  <property>
    <name>dfs.namenode.http-address.mycluster.nn2</name>
    <value>tiger:50070</value>
  </property>
  <property>
    <name>dfs.namenode.shared.edits.dir</name>
    <value>qjournal://elephant:8485;tiger:8485;horse:8485/mycluster</value>
  </property>
  <property>
    <name>dfs.journalnode.edits.dir</name>
    <value>/disk1/dfs/jn</value>
  </property>
  <property>
    <name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</valu
e>
  </property>
  <property>
    <name>dfs.ha.fencing.methods</name>
    <value>shell(/bin/true)</value>
  </property>
  <property>
    <name>dfs.ha.automatic-failover.enabled</name>
    <value>true</value>
  </property>
```

Press "ESC" key and press colon (:) and type 'wq' to save the file and close.

****

6. On elephant machine

```
$copy_configuration.sh
```

****

7. On elephant, tiger, and horse machines seperately

```
$ sudo yum install --assumeyes hadoop-hdfs-journalnode
```

****

8. On elephant, tiger, and horse machines seperately

```
$ sudo mkdir -p /disk1/dfs/jn
```

****

9. On elephant, tiger, and horse:

```
$ sudo chown -R hdfs:hadoop /disk1/dfs/jn
```

****

10. Reduce the heap size of the JournalNodes

on elephant machine

```
$ sudo vi /etc/hadoop/conf/hadoop-env.sh (enter)
```

Press "ESC" key and press "INSERT" key and change the value

```
export HADOOP_JOURNALNODE_OPTS="-Xmx64m"
```

Press "ESC" key and press colon (:) and type 'wq' to save the file and close.

****

10. On Elephant machine:

```
$copy_configuration.sh
```

****

11.On elephant, tiger, and horse machines seperately

```
$ sudo service hadoop-hdfs-journalnode start

$ sudo jps
```

****

12. On elephant machine

```
$ sudo -u hdfs hdfs namenode -initializeSharedEdits
```

****

13. On Elephant machine

```
$ sudo service hadoop-hdfs-namenode start

$ sudo jps
```

Verify that the NameNode started correctly

****

14. Verify that the service state of the nn1 NameNode running on elephant is Standby.

On elephant machine

```
$ sudo -u hdfs hdfs haadmin -getServiceState nn1
```

****

STEP 3:

ENABLING HA AND STANDBY NAMENODE

15. On tiger machine

```
$ sudo yum install --assumeyes hadoop-hdfs-namenode
```

```
$ sudo -u hdfs hdfs namenode -bootstrapStandby
```

```
$ sudo service hadoop-hdfs-namenode start
```

```
$ sudo -u hdfs hdfs haadmin -getServiceState nn2 (must show as standby)
```

****

16. Install the ZooKeeper Failover Controller software.

On elephant and tiger machines seperately

```
$ sudo yum install --assumeyes hadoop-hdfs-zkfc
```

****

17. Reduce the heap size of the ZooKeeper Failover Controllers by adding the following line to /etc/hadoop/conf/hadoop-env.sh on elephant:

```
$ sudo vi /etc/hadoop/conf/hadoop-env.sh (enter)
```

Press "ESC" key and press "INSERT" key and change the value

```
export HADOOP_ZKFC_OPTS="-Xmx64m"
```

Press "ESC" key and press colon (:) and type 'wq' to save the file and close.

```
:wq
```

****
****

18. On Elephant machine:

```
$copy_configuration.sh
```

```
$ sudo -u hdfs hdfs zkfc -formatZK
```

****

19. On Elephant and Tiger machines seperately

```
$ sudo service hadoop-hdfs-zkfc start
```

****

20. Run the sudo jps command and to verify that the ZooKeeper Failover Controllers on elephant and tiger started correctly.

On horse in your cluster:

```
$ sudo -u hdfs haadmin -getServiceState nn1
```

```
$ sudo -u hdfs haadmin -getServiceState nn2
```

****

21. On all four hosts (machines seperately) in your cluster:

```
$ sudo service hadoop-hdfs-datanode start
```

****

* cross check with sudo jps command whether datanode daemons started successfully on all 4 machines.

22. On horse machine:

```
$ sudo service hadoop-yarn-resourcemanager restart
```

****

23. Restart nodemanagers on all the 4 machines seperately

```
$ sudo service hadoop-yarn-nodemanager restart
```

****

24. Restart job history server on Monkey machine

```
$ sudo service hadoop-mapreduce-historyserver restart
```

****
25. On all four hosts in your cluster:

$ sudo jps

On elephant

NameNode,

JournalNode,

DFSZKFailoverController,

QuorumPeerMain,

DataNode

NodeManager

On tiger

NameNode,

JournalNode,

DFSZKFailoverController,

QuorumPeerMain,

DataNode,

NodeManager

On horse

JournalNode,

QuorumPeerMain,

DataNode,

ResourceManager,

NodeManager

On monkey

DataNode,

NodeManager,

JobHistoryServer

Note: The QuorumPeerMain process runs the ZooKeeper server

****

STEP 4:

Testing

Stop namenode on elephant machine

```
$ sudo service hadoop-hdfs-namenode stop
```

Using the hdfs haadmin -getServiceState command, verify that the service state of the nn2 NameNode is

now Active, and that you can no longer query the service state of the nn1 NameNode.

```
$ sudo -u hdfs hdfs haadmin -getServiceState nn1
```

```
$ sudo -u hdfs hdfs haadmin -getServiceState nn2
```

You can also check using WEBUI of namenode of elephant and tiger

http://elephant:50070

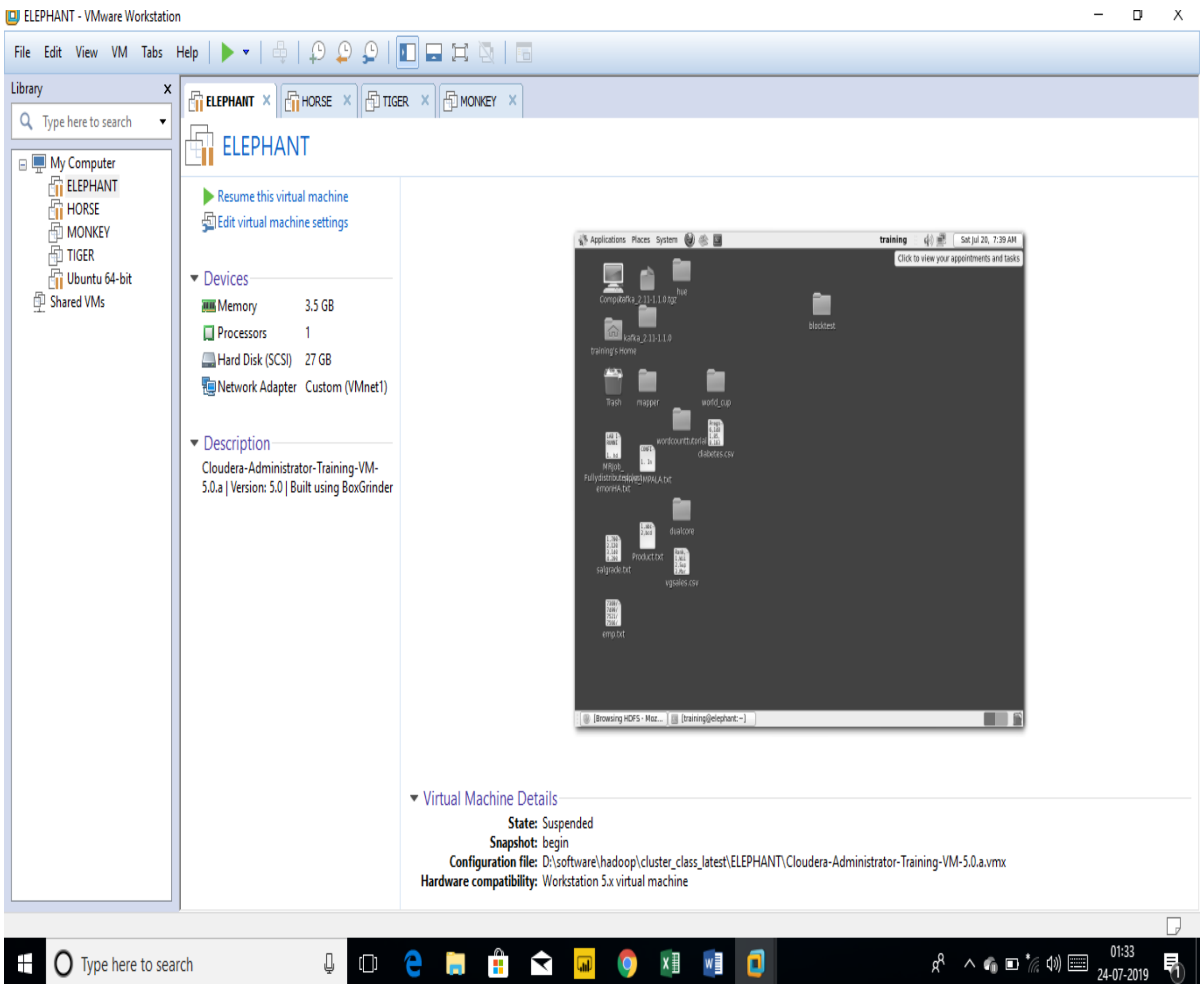
http://tiger:50070

Bring the nn1 NameNode back up.

On Elephant

```
$ sudo service hadoop-hdfs-namenode start
```

Working Area



ELEPHANT

Devices

Memory	3.5 GB
Processors	1
Hard Disk (SCSI)	27 GB
Network Adapter	Custom (VMnet1)

Description

Cloudera-Administrator-Training-VM-5.0.a | Version: 5.0 | Built using BoxGrinder

Virtual Machine Details

State:	Suspended
Snapshot:	begin
Configuration file:	D:\software\hadoop\cluster_class_latest\ELEPHANT\Cloudera-Administrator-Training-VM-5.0.a.vmx
Hardware compatibility:	Workstation 5.x virtual machine

01:33
24-07-2019

Customer Complaints Analysis

The POC is based on Consumer Complains recorded by US government.

These are complaints received from US citizens about financial products and services.

Public DATASET available at below website
https://catalog.data.gov/dataset/consumer-complaint-database

Industry: Finance

Data: Publicly available dataset with attributes like:

Complaint ID, Product, Sub-product, Issue,Sub-issue, State, ZIP code, Submitted via, Date received, Date sent to company, Company, Company response, Timely response, Consumer disputed.

Problem Statement: Analyze the data in Hadoop Eco-system to:

1. Get the number of complaints filed for each company.

2. Get the number of complaints filed under each product.

3. Get the total number of complaints filed from a particular location

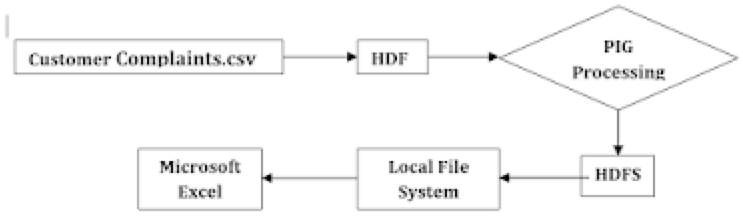
4. Get the list of company grouped by location which has no timely response

Input File Format - Comma Separated Values File (csv file)

Attributes of Input file.

1) Complaint ID, 2) Product, 3) Sub-product, 4) Issue, 5) Sub-issue, 6) State, 7) ZIP code, 8) Submitted via, 9) Date received, 10) Date sent to company, 11) Company, 12) Company response, 13) Timely response 14) Consumer disputed

Processing Logic – Customer Complaints data in Hadoop Eco-System



Pig Script

1. Get the number of complaints filed for each company.

2. Get the number of complaints filed under each product.

3. Get the total number of complaints filed from a particular location

4. Get the list of company grouped by location which has no timely response

Four Output files will be created.

Pig Code

Customer_Complain_Analysis.pig

```
CUSTOMER_COMPLAIN = LOAD '/hdfs/amit/POC/Consumer_Complaints1.csv' using
PigStorage(',') as (Complain_id:int, Product:chararray, Sub_Product:chararray,
Issue:chararray, Sub_Issue:chararray, State:chararray, Zip:int, Submitted_via:chararray,
Date_Received:chararray, Date_Sent_To_Company:chararray, Company:chararray,
Company_Response:chararray, Timely_Responsesponse:chararray,
Customer_Dispute:chararray);
GRP_COMPANY = GROUP CUSTOMER_COMPLAIN by Company;
CNT_COMPANY = FOREACH GRP_COMPANY GENERATE
group,COUNT(CUSTOMER_COMPLAIN);
STORE CNT_COMPANY INTO '/hdfs/amit/POC/Complains_by_Company/' using
PigStorage(',');
GRP_PRODUCT = GROUP CUSTOMER_COMPLAIN by Product;
CNT_PRODUCT = FOREACH GRP_PRODUCT GENERATE
group,COUNT(CUSTOMER_COMPLAIN);
STORE CNT_PRODUCT INTO '/hdfs/amit/POC/Complains_by_Product/' using
PigStorage(',');
GRP_LOCATION = GROUP CUSTOMER_COMPLAIN by State;
CNT_LOCATION = FOREACH GRP_LOCATION GENERATE
group,COUNT(CUSTOMER_COMPLAIN);
STORE CNT_LOCATION INTO '/hdfs/amit/POC/Complains_by_Location/' using
PigStorage(',');
FLTR_TIME_RESPONSE = FILTER CUSTOMER_COMPLAIN by
Timely_Responsesponse=='No';
GRP_COMPANY_LOCATION = GROUP FLTR_TIME_RESPONSE by State;
COMPANY_AND_LOCATION = FOREACH GRP_COMPANY_LOCATION GENERATE
group,FLATTEN(FLTR_TIME_RESPONSE.Company);
```

```
STORE COMPANY_AND_LOCATION INTO
'/hdfs/amit/POC/Complains_by_Response_No/' using PigStorage(',');
```

Shell Script

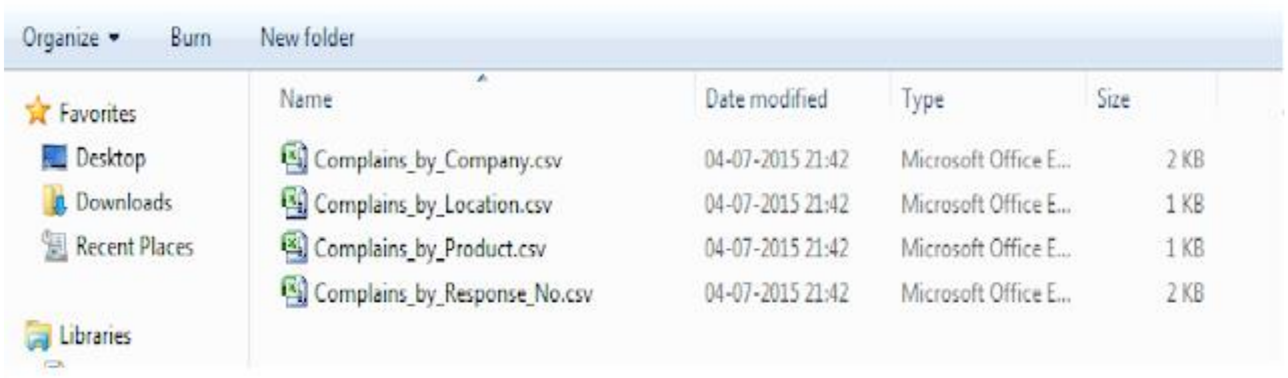
Purpose of this shell script is to perform cleanup (delete existing output files) and execute the Pig Script to get Customer Complaints Analysis and store the resultant file in CSV format.

Shell Code

Customer_Complain_Analysis.sh

```
rm /home/training/Downloads/POC/Complains_by_Company.csv
rm /home/training/Downloads/POC/Complains_by_Product.csv
rm /home/training/Downloads/POC/Complains_by_Location.csv
rm /home/training/Downloads/POC/Complains_by_Response_No.csv
hadoop fs -rmr /hdfs/amit/POC/Complains_by_Company
hadoop fs -rmr /hdfs/amit/POC/Complains_by_Product
hadoop fs -rmr /hdfs/amit/POC/Complains_by_Location
hadoop fs -rmr /hdfs/amit/POC/Complains_by_Response_No
pig /home/amit/POC/Customer_Complain_Analysis.pig
hadoop fs -get /hdfs/amit/POC/Complains_by_Company/part-r-00000
/home/training/Downloads/POC/Complains_by_Company.csv
hadoop fs -get /hdfs/amit/POC/Complains_by_Product/part-r-00000
/home/training/Downloads/POC/Complains_by_Product.csv
hadoop fs -get /hdfs/amit/POC/Complains_by_Location/part-r-00000
/home/training/Downloads/POC/Complains_by_Location.csv
hadoop fs -get /hdfs/amit/POC/Complains_by_Response_No/part-r-00000
/home/training/Downloads/POC/Complains_by_Response_No.csv
```

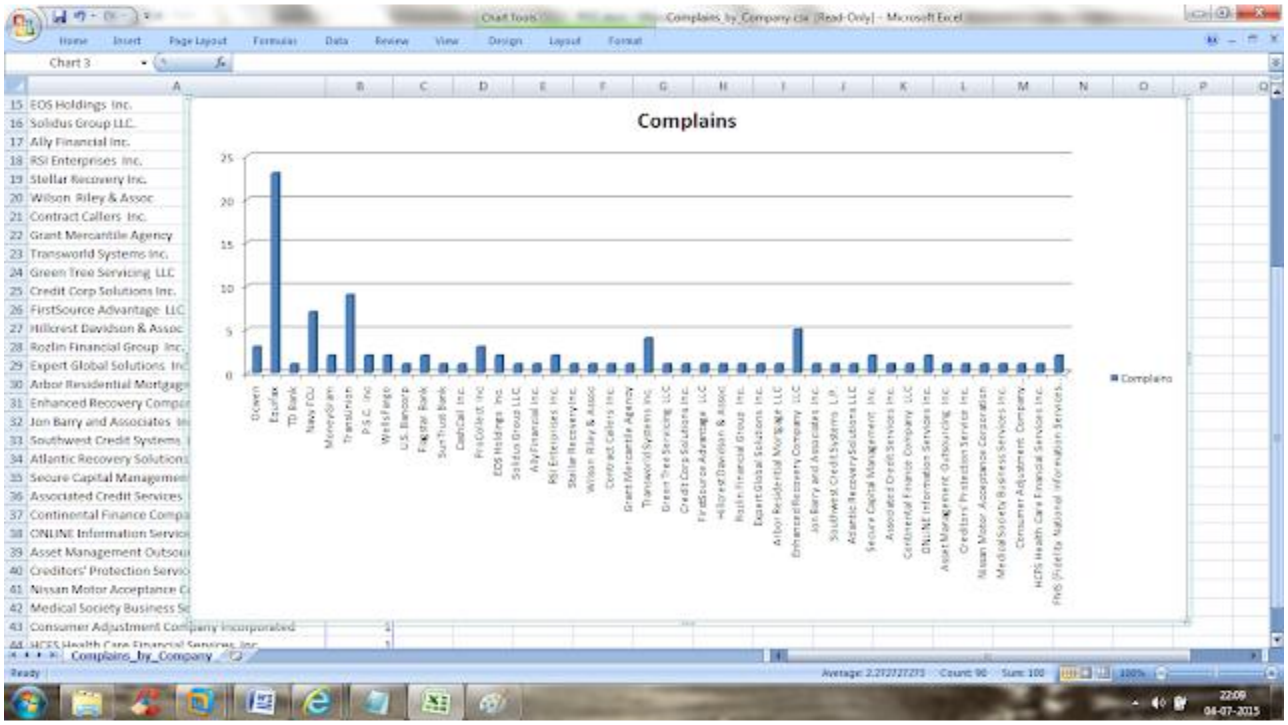
Output



Name	Date modified	Type	Size
Complains_by_Company.csv	04-07-2015 21:42	Microsoft Office E...	2 KB
Complains_by_Location.csv	04-07-2015 21:42	Microsoft Office E...	1 KB
Complains_by_Product.csv	04-07-2015 21:42	Microsoft Office E...	1 KB
Complains_by_Response_No.csv	04-07-2015 21:42	Microsoft Office E...	2 KB

Chart view in Excel

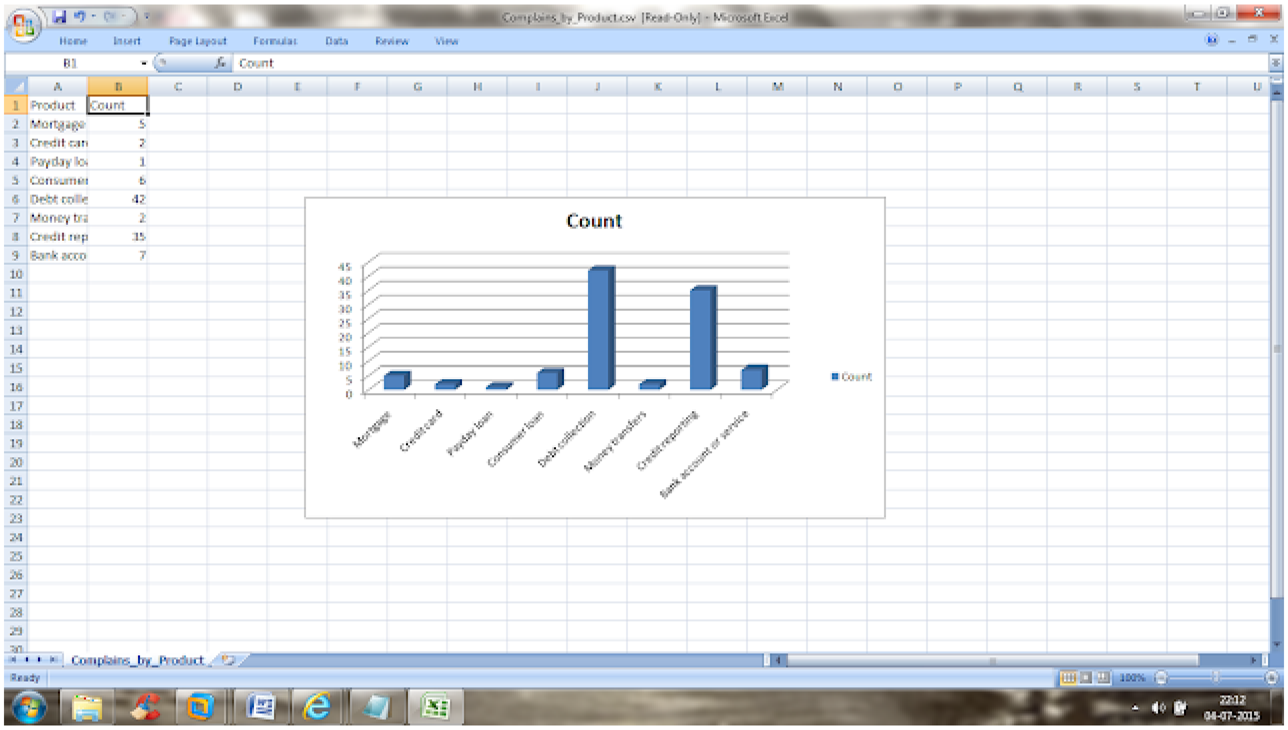
1. Get the number of complaints filed for each company.(Complains_by_Company.csv)


Complains
25
20
15
10
5
0
Ocwen
Equifax
TD Bank
Navy FCU
MoneyGram
TransUnion
P.S.C. Inc
Wells Fargo
U.S. Bancorp
Flagstar Bank
SunTrust Bank
CashCall Inc.
ProCollect Inc.
EOS Holdings Inc.
Solidus Group LLC
Ally Financial Inc.
RSI Enterprises Inc.
Stellar Recovery Inc.
Wilson Riley & Assoc
Contract Callers Inc.
Grant Mercantile Agency
Transworld Systems Inc.
Green Tree Servicing LLC
Credit Corp Solutions Inc.
FirstSource Advantage LLC
Hillcrest Davidson & Assoc
Rozlin Financial Group Inc.
Expert Global Solutions Inc.
Arbor Residential Mortgage LLC
Enhanced Recovery Company LLC
Jon Barry and Associates Inc.
Southwest Credit Systems L.P.
Atlantic Recovery Solutions LLC
Secure Capital Management Inc.
Associated Credit Services Inc.
Continental Finance Company LLC
ONLINE Information Services Inc.
Asset Management Outsourcing Inc.
Creditors' Protection Service Inc.
Nissan Motor Acceptance Corporation
Medical Society Business Services Inc.
Consumer Adjustment Company
HCFS Health Care Financial Services Inc.
FNIS (Fidelity National Information Services...
Complains

	A	B
15	EOS Holdings Inc.	
16	Solidus Group LLC.	
17	Ally Financial Inc.	
18	RSI Enterprises Inc.	
19	Stellar Recovery Inc.	
20	Wilson Riley & Assoc	
21	Contract Callers Inc.	
22	Grant Mercantile Agency	
23	Transworld Systems Inc.	
24	Green Tree Servicing LLC	
25	Credit Corp Solutions Inc.	
26	FirstSource Advantage LLC	
27	Hillcrest Davidson & Assoc	
28	Rozlin Financial Group Inc.	
29	Expert Global Solutions Inc	
30	Arbor Residential Mortgag	
31	Enhanced Recovery Compa	
32	Jon Barry and Associates In	
33	Southwest Credit Systems	
34	Atlantic Recovery Solution	
35	Secure Capital Managemen	
36	Associated Credit Services	
37	Continental Finance Compa	
38	ONLINE Information Servic	
39	Asset Management Outsou	
40	Creditors' Protection Servic	
41	Nissan Motor Acceptance C	
42	Medical Society Business Se	
43	Consumer Adjustment Company Incorporated	1
44	HCFS Health Care Financial Services Inc.	1

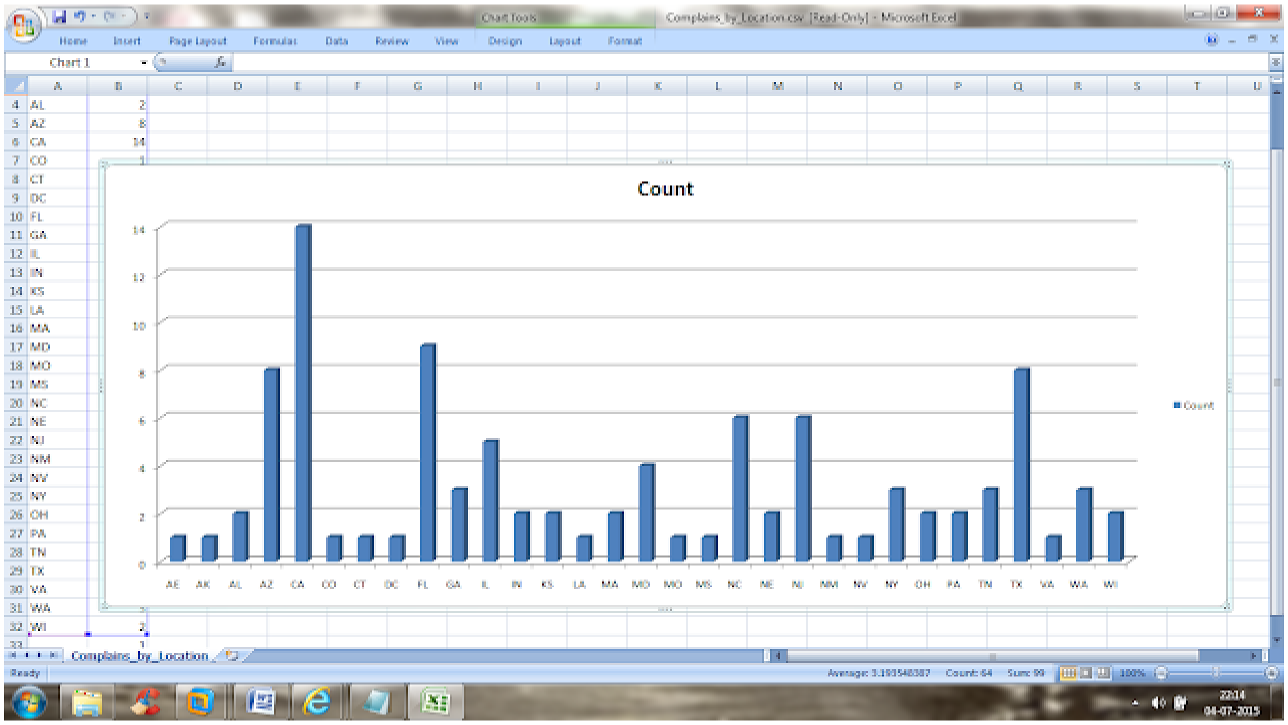
Average: 2.272727273 Count: 90 Sum: 100

2. Get the number of complaints filed under each product. (Complains_by_Product.csv)



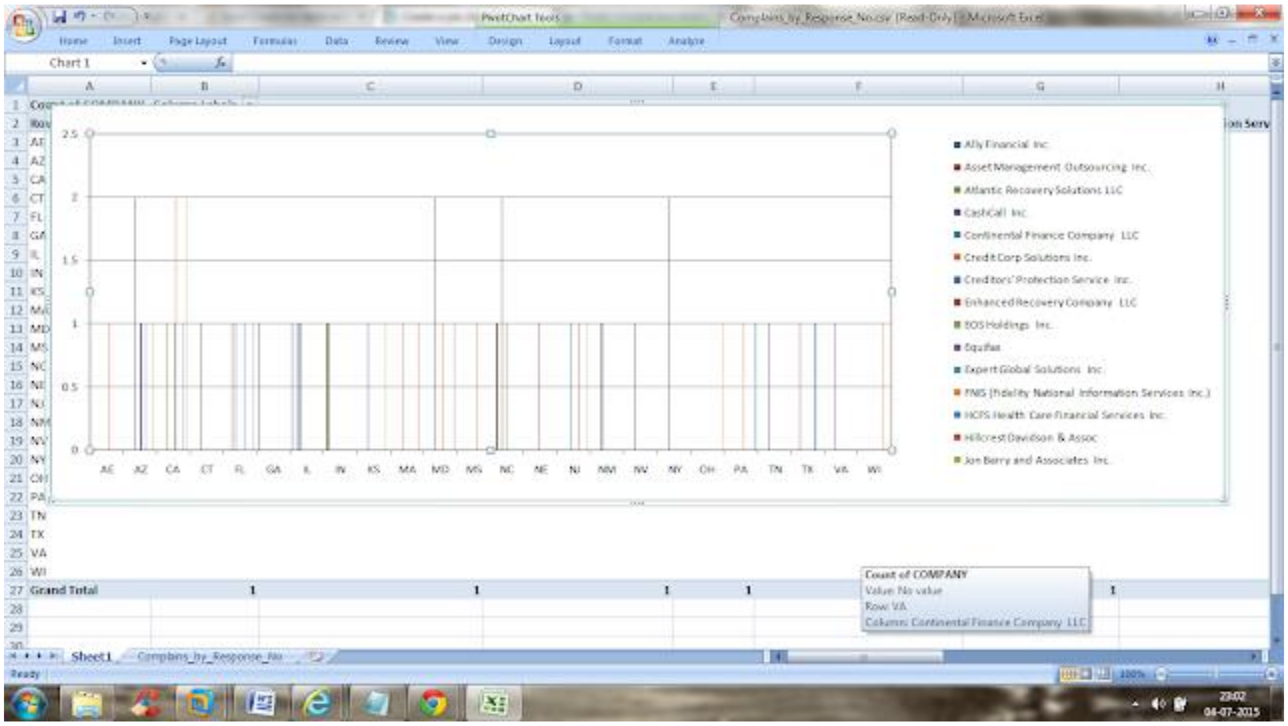
	A	B
1	Product	Count
2	Mortgage	5
3	Credit car	2
4	Payday lo	1
5	Consumer	6
6	Debt colle	42
7	Money tra	2
8	Credit rep	35
9	Bank acco	7

3. Get the total number of complaints filed from a particular location (Complains_by_Location.csv)



	A	B
4	AL	2
5	AZ	8
6	CA	14
7	CO	1
8	CT	
9	DC	
10	FL	
11	GA	
12	IL	
13	IN	
14	KS	
15	LA	
16	MA	
17	MD	
18	MO	
19	MS	
20	NC	
21	NE	
22	NJ	
23	NM	
24	NV	
25	NY	
26	OH	
27	PA	
28	TN	
29	TX	
30	VA	
31	WA	
32	WI	2
33		1

4. Get the list of company grouped by location which has no timely response (Complains_by_Response_No.csv)


Complains_by_Response_No.csv [Read-Only] - Microsoft Excel
PivotChart Tools
Home Insert Page Layout Formulas Data Review View Design Layout Format Analyze
Chart 1
0 0.5 1 1.5 2 2.5
AE AZ CA CT FL GA IL IN KS MA MD MS NC NE NJ NM NV NY OH PA TN TX VA WI
Ally Financial Inc.
Asset Management Outsourcing Inc.
Atlantic Recovery Solutions LLC
CashCall Inc.
Continental Finance Company LLC
Credit Corp Solutions Inc.
Creditors' Protection Service Inc.
Enhanced Recovery Company LLC
EOS Holdings Inc.
Equifax
Expert Global Solutions Inc.
FNIS (Fidelity National Information Services Inc.)
HCFS Health Care Financial Services Inc.
Hillcrest Davidson & Assoc
Jon Barry and Associates Inc.
Grand Total
Sheet1
Complains_by_Response_No
Ready

Count of COMPANY
Value: No value
Row: VA
Column: Continental Finance Company LLC

CHAPTER-6 CONCLUSIONS

Practical knowledge means the visualization of the knowledge, which we read in our books. For this, we perform experiments and get observations. Practical knowledge is very important in every field. One must be familiar with the problems related to that field so that he may solve them and become a successful person.

After achieving the proper goal in life, an engineer has to enter in professional life. According to this life, he has to serve an industry, may be public or private sector or self-own. For the efficient work in the field, he must be well aware of the practical knowledge as well as theoretical knowledge.

Due to all above reasons and to bridge the gap between theory and practical, our Engineering curriculum provides a practical training of 45 days. During this period a student work in the industry and get well all type of experience and knowledge about the working of companies and hardware and software tools.

I have undergone my 45 days summer training in 7th sem at **Brillica Services Pvt Ltd**. This report is based on the knowledge, which I acquired during my 45 days of summer training.

CHAPTER 7: FUTURE SCOPE

With the knowledge I have gained by working on this project, I am confident that in

the future I can work on Big data more effectively by adding some more features .

- Setting up high availability cluster which consists of more nodes so that huge amount of data   could be stored on them.
- Increasing the effectiveness of analytics by using advance tools like apache spark.
- To make it make it more interpretable we can analytics tools like tableau powerbi etc.
- This project could be used analysis of product behaviour and demands of user on the basis of complaints found using the project.

BIBLIOGRAPHY

- Brillica services Student Reference Guide
- **https://www.oreilly.com/data/free/files/cloudera-impala.pdf**
- Cloudera
- Hortonworks