ASSIGNMENT #2 REPORT

Team Members and Contribution:

- **Sankalp Heranjal (A20332999) – Hadoop**
- **Akshay Ganji (A20325252) – Swift**
- **Sai Abhilash Putrevu (A20324224) – Java**

HADOOP DOCUMENTATION AND EVALUATION

Author: Sankalp Heranjal

Operating Systems Used: Windows 7 – 64Bit and Ubuntu 14.04 LTS – 64Bit (VM)

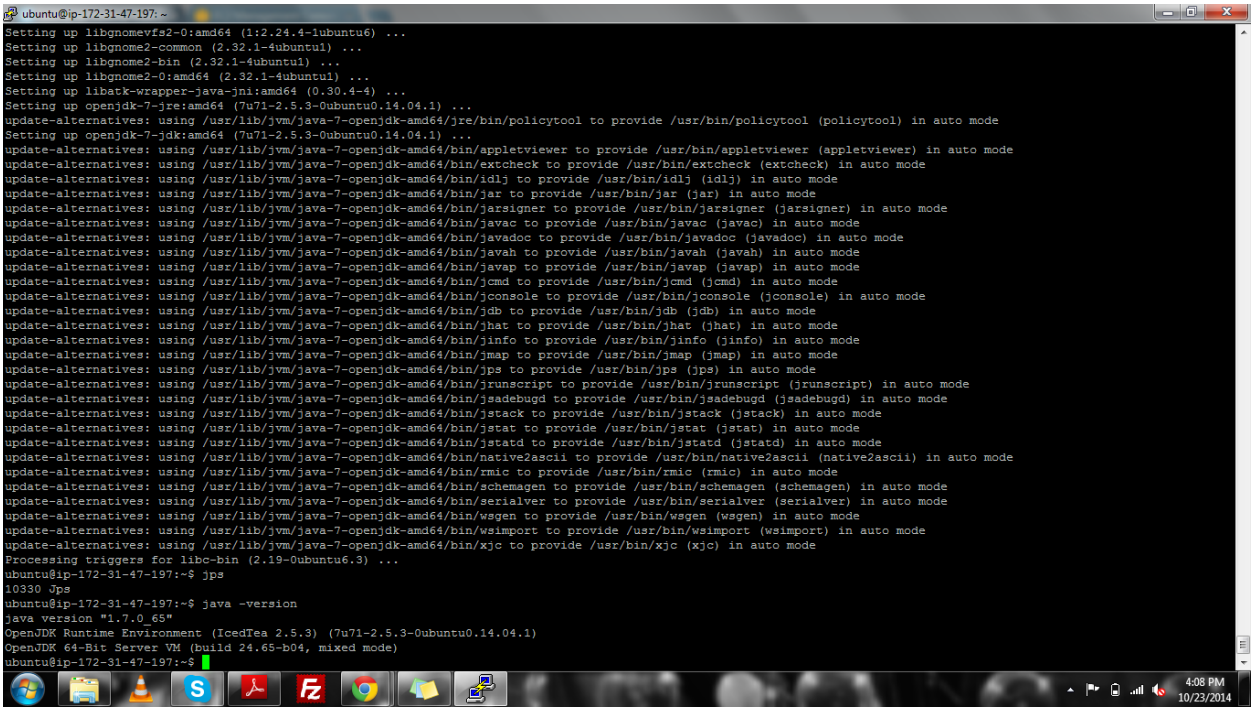
Software Used: Hadoop v-1.2.1, Java v-1.7.0_65, Eclipse 3.8, PuTTY, FileZilla

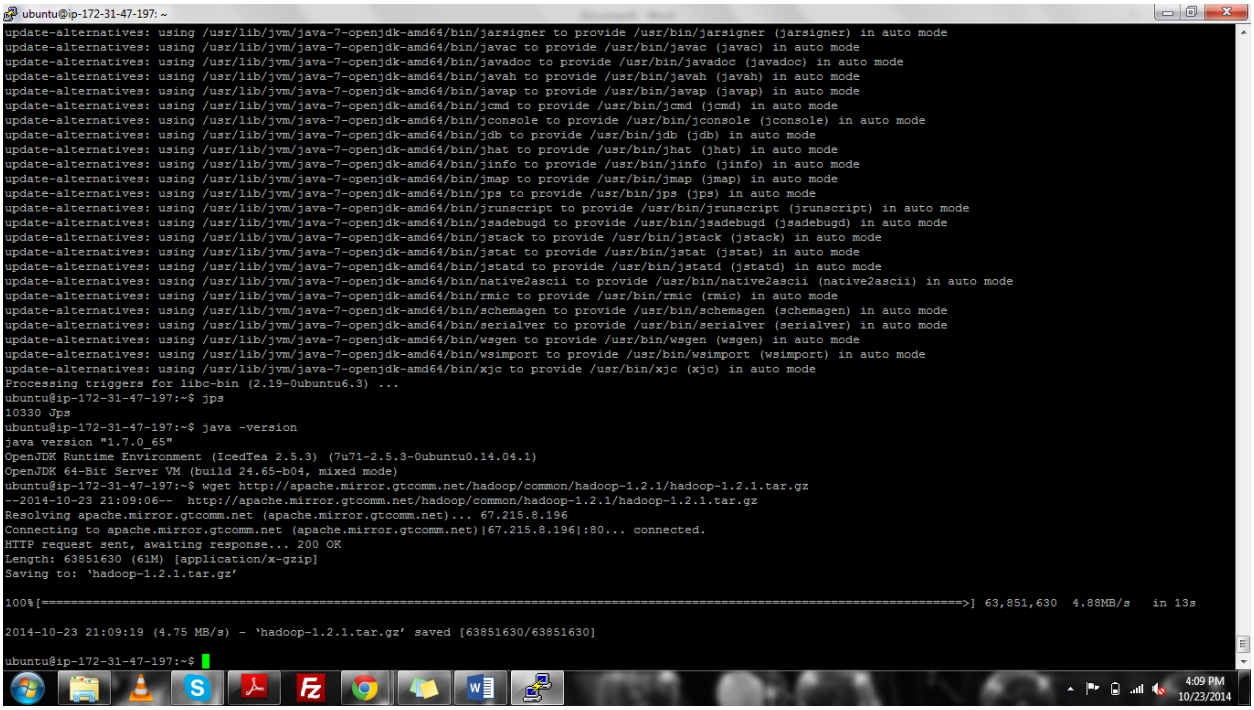
ANT Version: 1.2.1

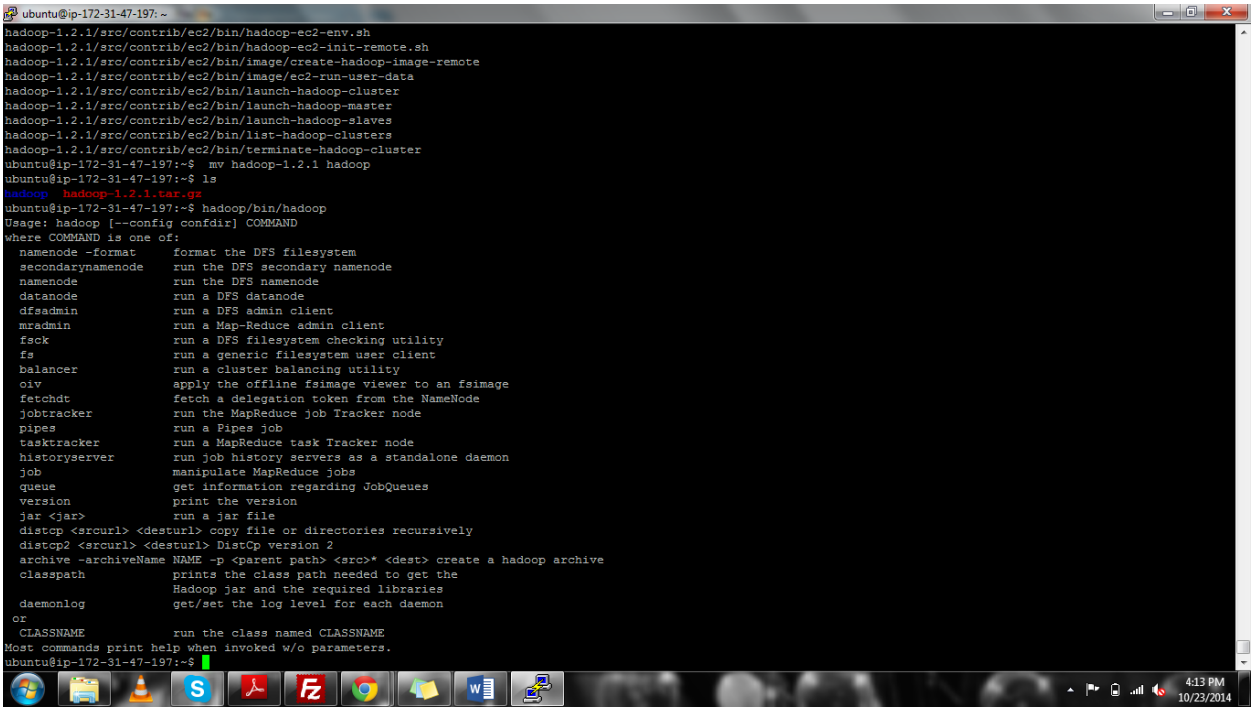
Steps Taken (With Screenshots):

Launching Amazon EC2 Instances:
We started with creating an Amazon AWS Account. After creation of the account, we started with the Cluster Setup. We started with the 16node setup first. We launched an Ubuntu 14.04 c3.large instance with 32GB EBS Storage and created a key file keypairaws.pem which we uploaded in the instance using FileZilla.
All the following execution is done in PuTTY.

Java and Hadoop Installation:
We installed Java v-1.7.0_65 and Hadoop v-1.2.1. The following screenshots show the Java and Hadoop Versions.


ubuntu@ip-172-31-47-197: ~
Setting up libgnomevfs2-0:amd64 (1:2.24.4-1ubuntu6) ...
Setting up libgnome2-common (2.32.1-4ubuntu1) ...
Setting up libgnome2-bin (2.32.1-4ubuntu1) ...
Setting up libgnome2-0:amd64 (2.32.1-4ubuntu1) ...
Setting up libatk-wrapper-java-jni:amd64 (0.30.4-4) ...
Setting up openjdk-7-jre:amd64 (7u71-2.5.3-0ubuntu0.14.04.1) ...
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/policytool to provide /usr/bin/policytool (policytool) in auto mode
Setting up openjdk-7-jdk:amd64 (7u71-2.5.3-0ubuntu0.14.04.1) ...
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/appletviewer to provide /usr/bin/appletviewer (appletviewer) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/extcheck to provide /usr/bin/extcheck (extcheck) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/idlj to provide /usr/bin/idlj (idlj) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jar to provide /usr/bin/jar (jar) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jarsigner to provide /usr/bin/jarsigner (jarsigner) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javac to provide /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javadoc to provide /usr/bin/javadoc (javadoc) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javah to provide /usr/bin/javah (javah) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javap to provide /usr/bin/javap (javap) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jcmd to provide /usr/bin/jcmd (jcmd) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jdb to provide /usr/bin/jdb (jdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jhat to provide /usr/bin/jhat (jhat) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jinfo to provide /usr/bin/jinfo (jinfo) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jmap to provide /usr/bin/jmap (jmap) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jps to provide /usr/bin/jps (jps) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jrunscript to provide /usr/bin/jrunscript (jrunscript) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jsadebugd to provide /usr/bin/jsadebugd (jsadebugd) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jstatd to provide /usr/bin/jstatd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/native2ascii to provide /usr/bin/native2ascii (native2ascii) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/rmic to provide /usr/bin/rmic (rmic) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/schemagen to provide /usr/bin/schemagen (schemagen) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/wsgen to provide /usr/bin/wsgen (wsgen) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/wsimport to provide /usr/bin/wsimport (wsimport) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/xjc to provide /usr/bin/xjc (xjc) in auto mode
Processing triggers for libc-bin (2.19-0ubuntu6.3) ...
ubuntu@ip-172-31-47-197:~$ jps
10330 Jps
ubuntu@ip-172-31-47-197:~$ java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
ubuntu@ip-172-31-47-197:~$
4:08 PM
10/23/2014


ubuntu@ip-172-31-47-197: ~
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jarsigner to provide /usr/bin/jarsigner (jarsigner) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javac to provide /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javadoc to provide /usr/bin/javadoc (javadoc) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javah to provide /usr/bin/javah (javah) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/javap to provide /usr/bin/javap (javap) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jcmd to provide /usr/bin/jcmd (jcmd) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jdb to provide /usr/bin/jdb (jdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jhat to provide /usr/bin/jhat (jhat) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jinfo to provide /usr/bin/jinfo (jinfo) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jmap to provide /usr/bin/jmap (jmap) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jps to provide /usr/bin/jps (jps) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jrunscript to provide /usr/bin/jrunscript (jrunscript) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jsadebugd to provide /usr/bin/jsadebugd (jsadebugd) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/jstatd to provide /usr/bin/jstatd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/native2ascii to provide /usr/bin/native2ascii (native2ascii) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/rmic to provide /usr/bin/rmic (rmic) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/schemagen to provide /usr/bin/schemagen (schemagen) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/wsgen to provide /usr/bin/wsgen (wsgen) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/wsimport to provide /usr/bin/wsimport (wsimport) in auto mode
update-alternatives: using /usr/lib/jvm/java-7-openjdk-amd64/bin/xjc to provide /usr/bin/xjc (xjc) in auto mode
Processing triggers for libc-bin (2.19-0ubuntu6.3) ...
ubuntu@ip-172-31-47-197:~$ jps
10330 Jps
ubuntu@ip-172-31-47-197:~$ java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
ubuntu@ip-172-31-47-197:~$ wget http://apache.mirror.gtcomm.net/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
--2014-10-23 21:09:06-- http://apache.mirror.gtcomm.net/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
Resolving apache.mirror.gtcomm.net (apache.mirror.gtcomm.net)... 67.215.8.196
Connecting to apache.mirror.gtcomm.net (apache.mirror.gtcomm.net)|67.215.8.196|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 63851630 (61M) [application/x-gzip]
Saving to: 'hadoop-1.2.1.tar.gz'
100%[==========>] 63,851,630 4.88MB/s in 13s
2014-10-23 21:09:19 (4.75 MB/s) - 'hadoop-1.2.1.tar.gz' saved [63851630/63851630]
ubuntu@ip-172-31-47-197:~$
4:09 PM
10/23/2014

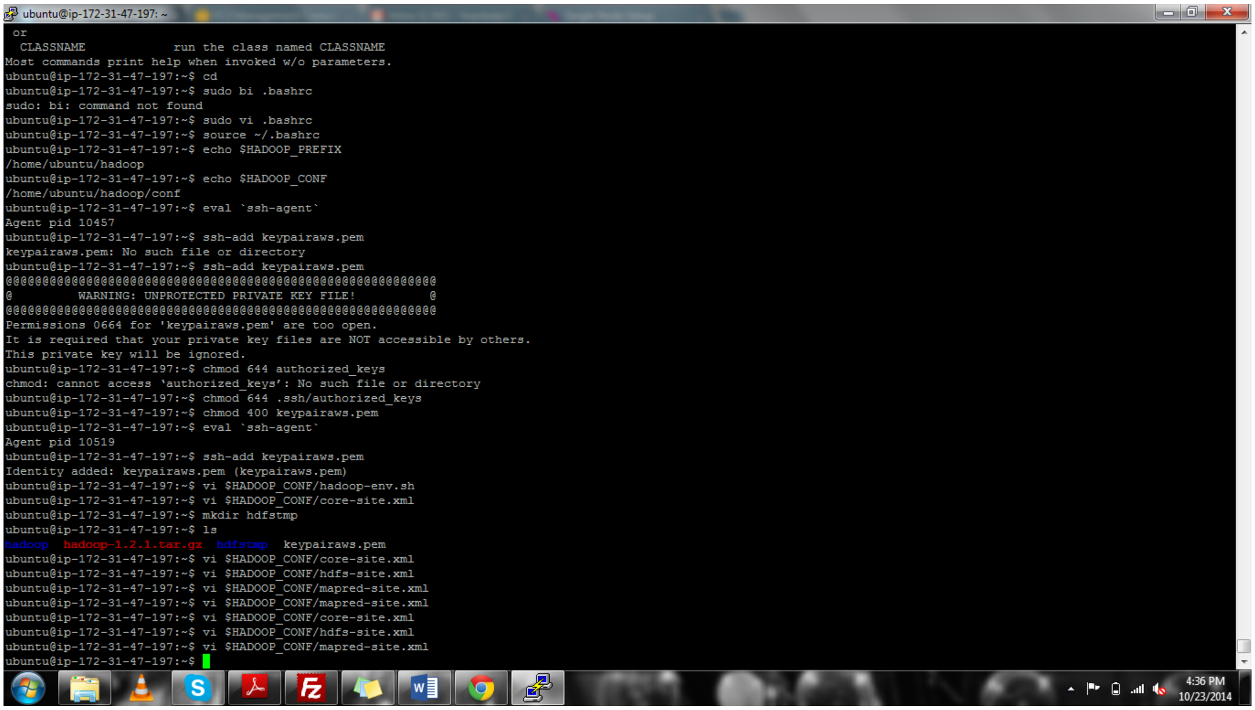


Setting up of Password-less SSH

We need to add the AWS EC2 Key File keypairaws.pem to SSH profile. In order to do that we will need to use following ssh utilities:

'ssh-agent' : Used as a background program that handles passwords for SSH private keys.
'ssh-add' : prompts the user for a private key password and adds it to the list maintained by ssh-agent.

Ssh Session is lost everytime upon shell exit and we have to repeat ssh-agent and ssh-add commands.



Setup for Running Word Count on 16 Nodes:

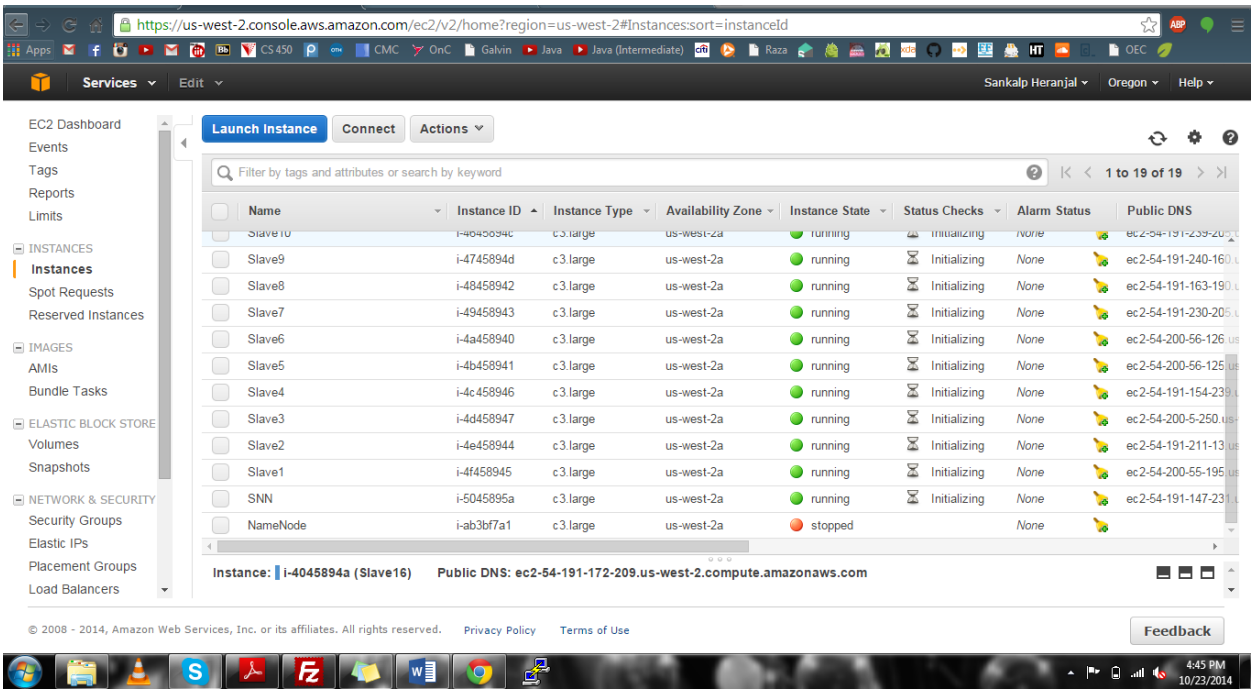
1. **Hadoop Cluster Setup**

   We went to the hadoop/conf location and edited the following Configuration Files:

   - **Hadoop-env.sh**
     In this file we gave the path of the Java Home Directory. This file contains the environment variable settings. We use this to change the aspect of Hadoop daemon behavior, such as where log files are stored, the maximum amount of heap used etc.
   - **Core-site.xml**
     This file is used for NameNode Configuration. We are providing the Private IP of the instance we are creating and the Port Number i.e. 9000.
   - **Hdfs-site.xml**
     This file is used for the configuration of HDFS daemons, the NameNode, SecondaryNameNode and Data nodes. We have provided 2 properties i.e. dfs.permissions.enabled and dfs.replication.
     The first one is set to false which means the user can do anything they want to HDFS and the latter one is set to 16 since we are using 16 slaves.
   - **Mapred-site.xml**
     This file contains the configuration settings for MapReduce daemons, the job tracker and the task-trackers. We are providing the Private IP of the instance we are creating and the Port Number i.e. 9001 on which the Job Tracker runs.

2. Image Creation and Launching the 16 Slaves

Once we were done with the configuration of the instance, we kept this as the Master or the NameNode. Next, we created an image of the instance and launched 17 more instances with that image. These 17 instances comprised of a SecondaryNameNode and 16 Slaves.

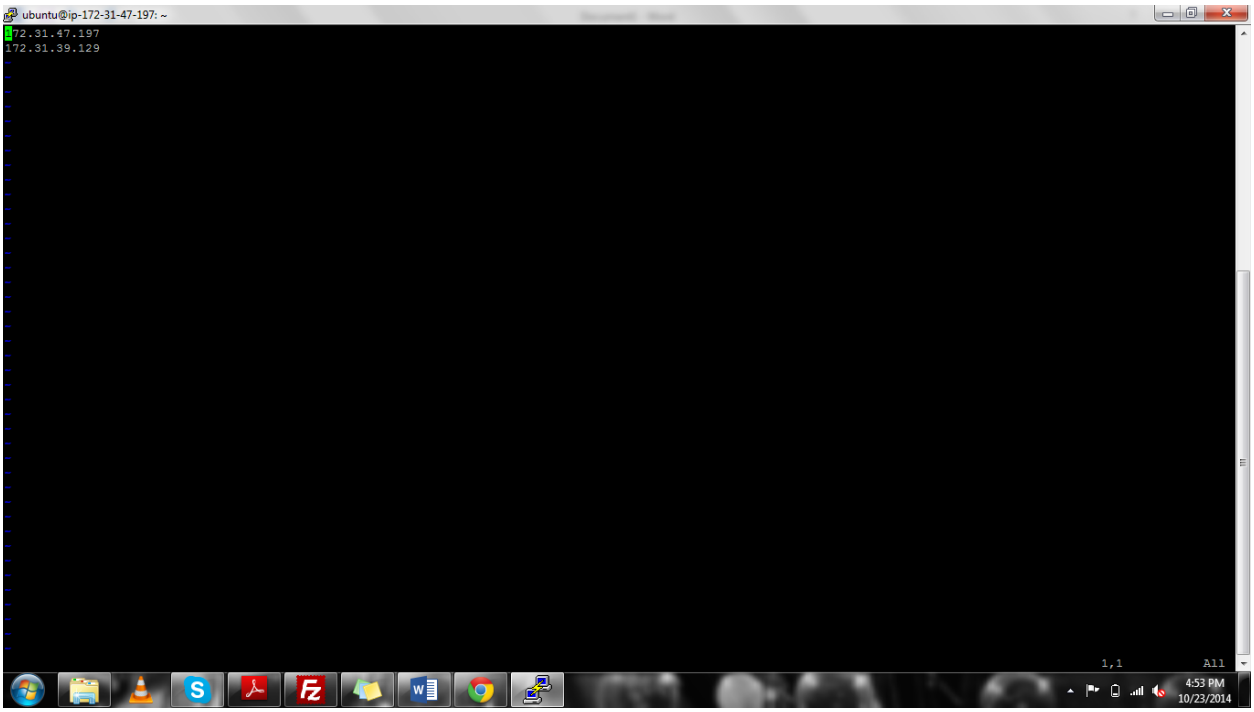


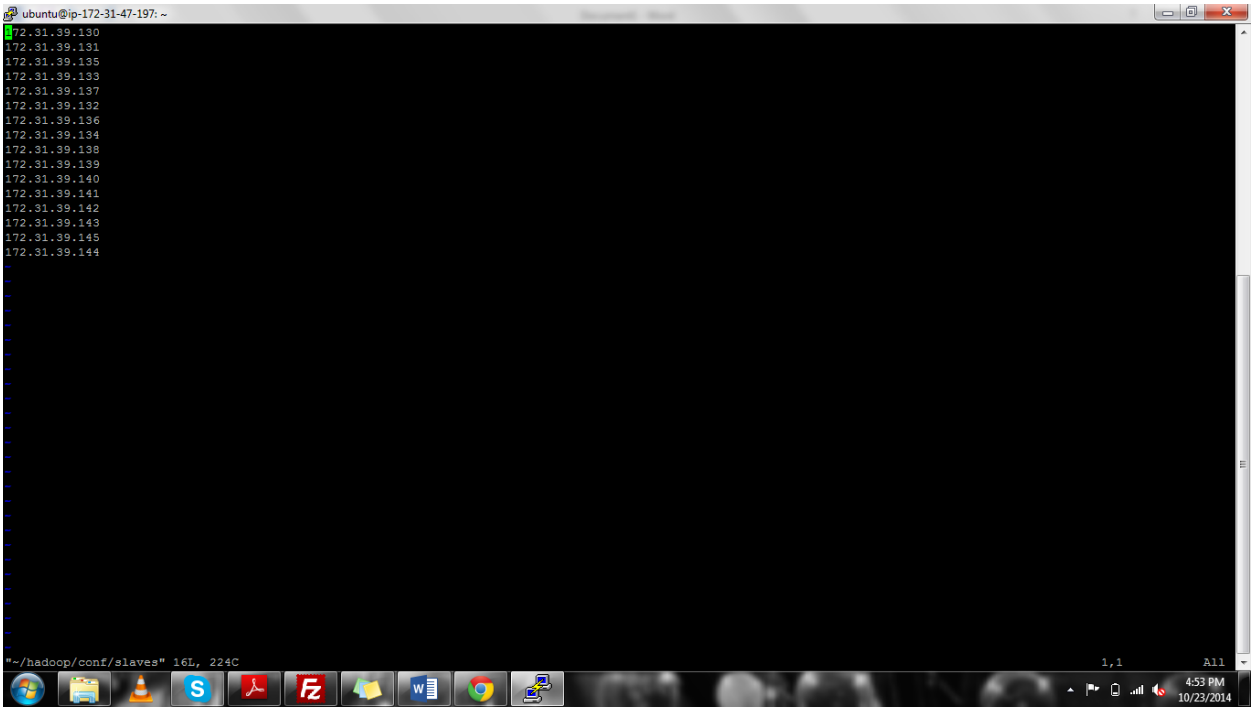
3. Configuration of Masters and Slaves Files

After launching all the instances, we configured the conf/Masters and conf/Slaves files on all the instances.

- **Master/NameNode:**
  We took the master file and gave the private IPs of both NameNode and the SecondaryNameNode one after the other. In the Slaves file, we gave the private IPs of all the 16 Slaves.
- **SecondaryNameNode:**
  In SNN, we gave the same configuration as that of the Master.
- **Slaves/DataNode:**
  We provided the private IPs of each slave in the conf/Slaves file of that slave and left the conf/Masters file blank.

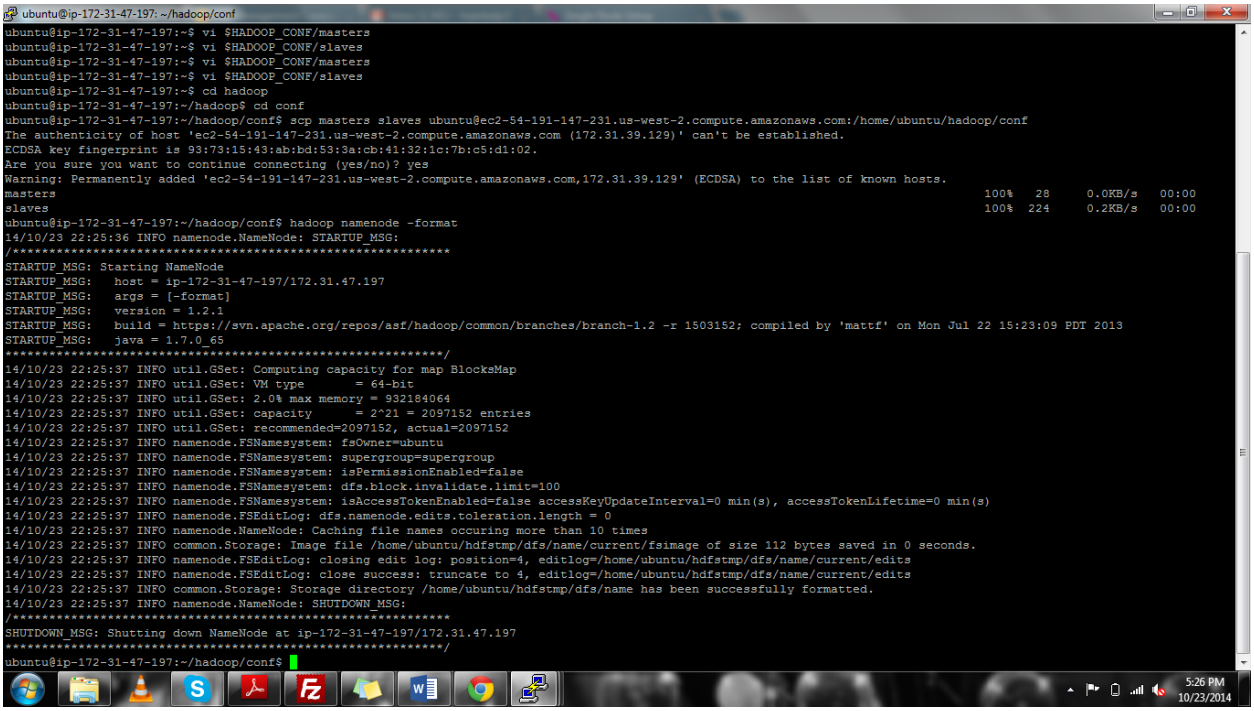
The NameNode Master and Slave File:





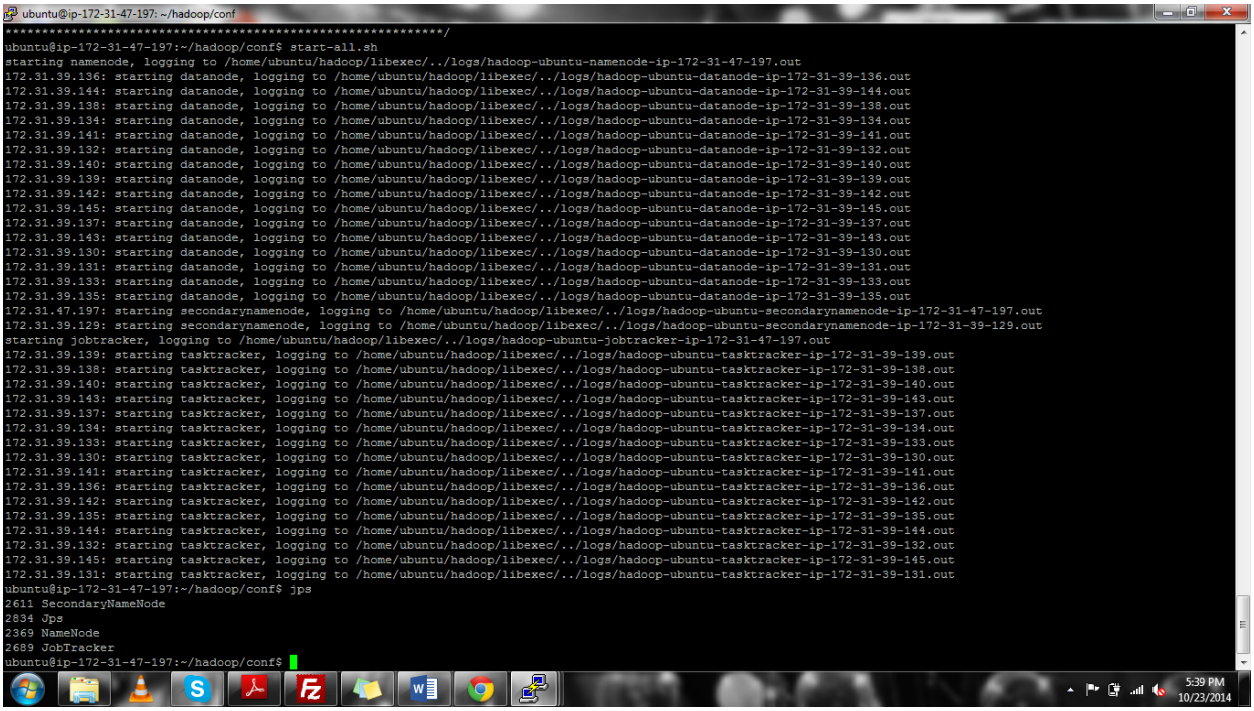
4. Hadoop Daemon Startup

After configuring all the conf files, we went to the NameNode and Started the Hadoop File System. We did this by formatting the NameNode. After Formatting the NameNode, we started the Hadoop daemons by using bin/start.sh

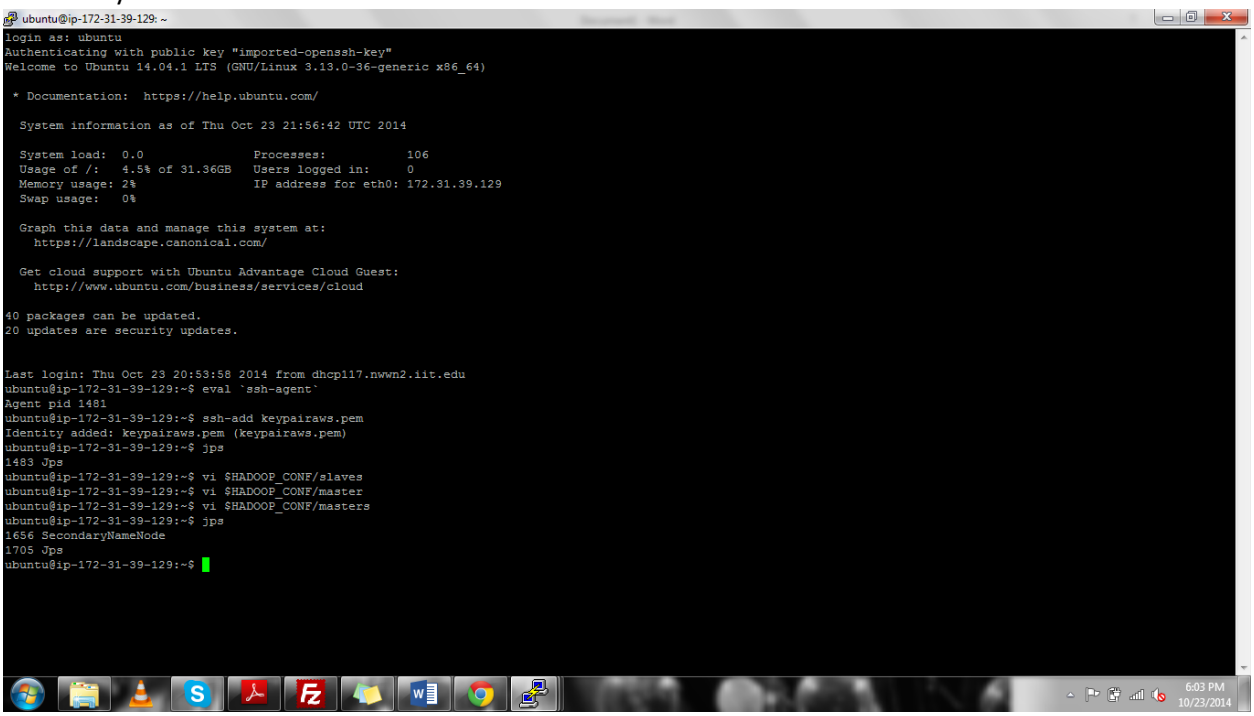


We then ran jps on every instance to check whether the connection is established by the master with the slaves.

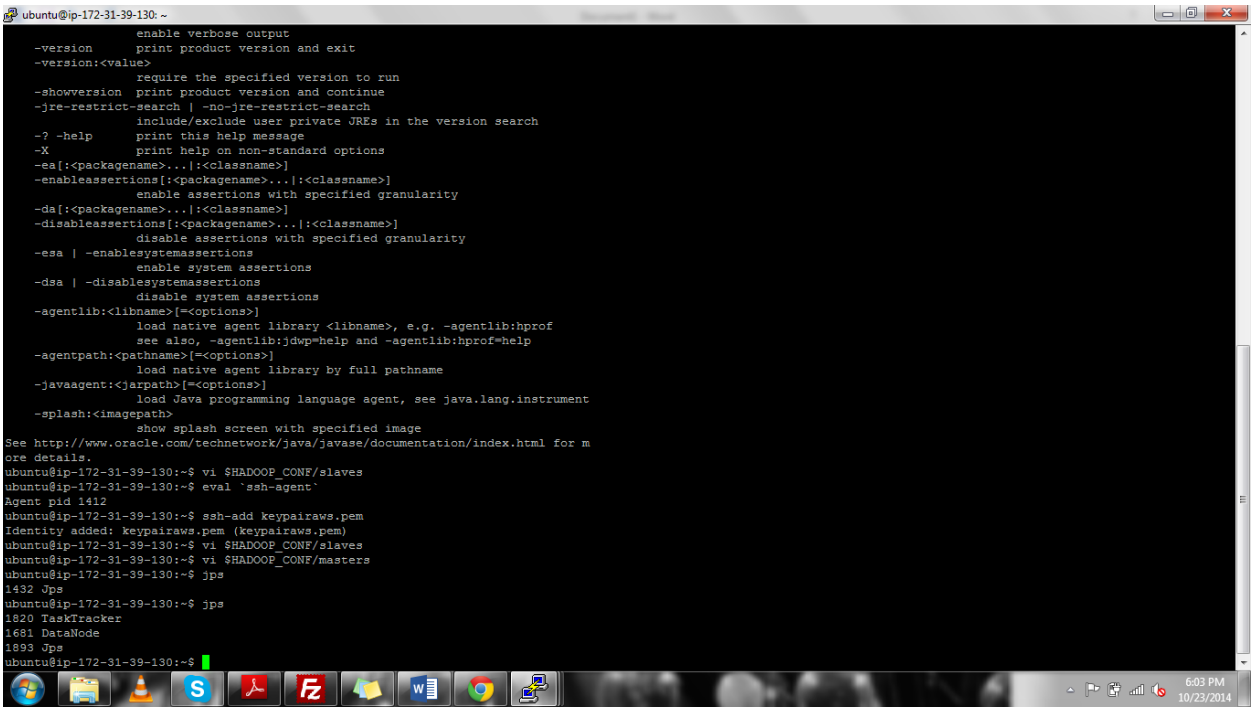
Master:



SecondaryNameNode:



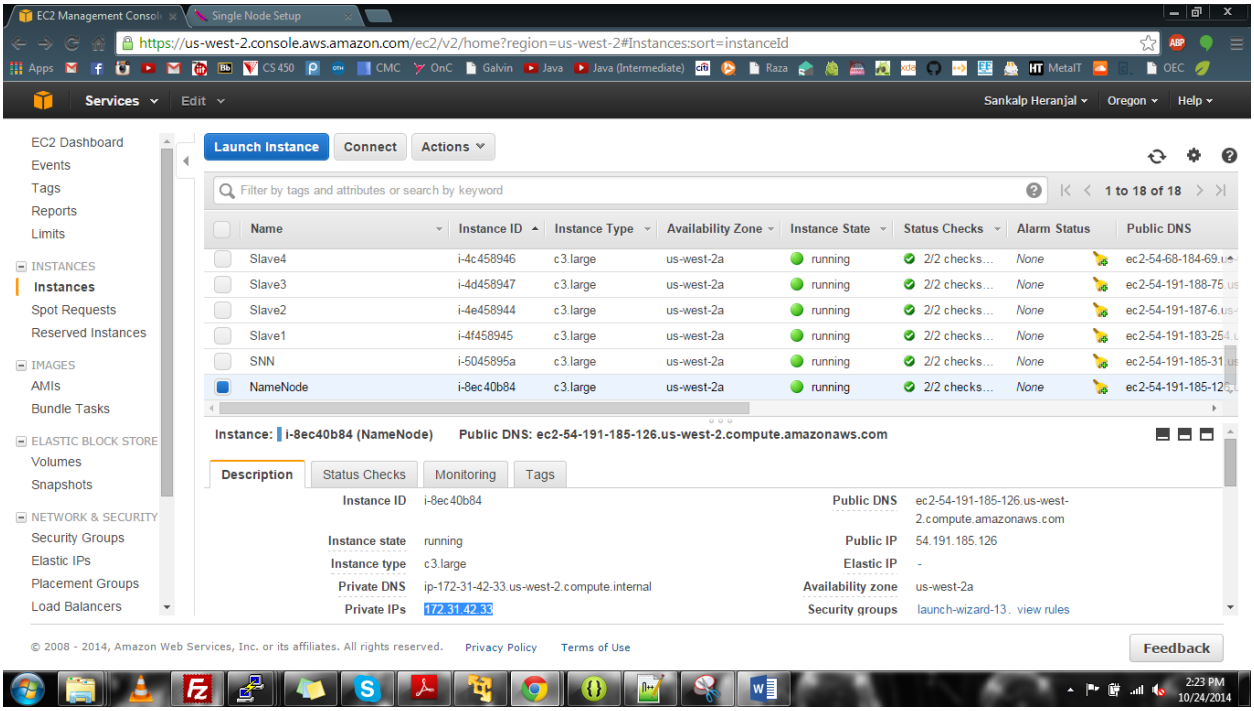
16 Slaves:



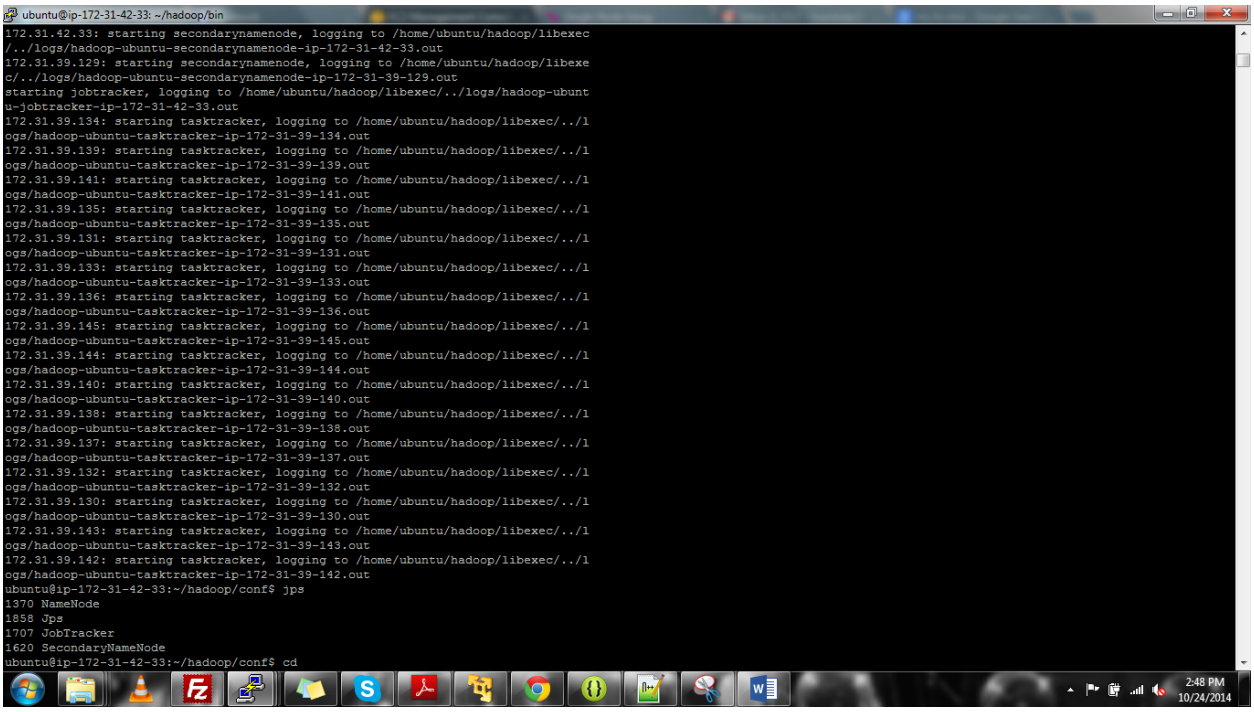
Problem and Solution: At this point of time, we had an issue with the program to run. But we thought it was a problem in the Master Node Configuration. So we had to terminate the old NameNode (172.31.39.130) because of some configuration problems and created a new one (172.31.42.33) and

changed the configuration and masters/slaves files for the NameNode, SecondaryNameNode and the DataNodes according to that. It worked fine after that.

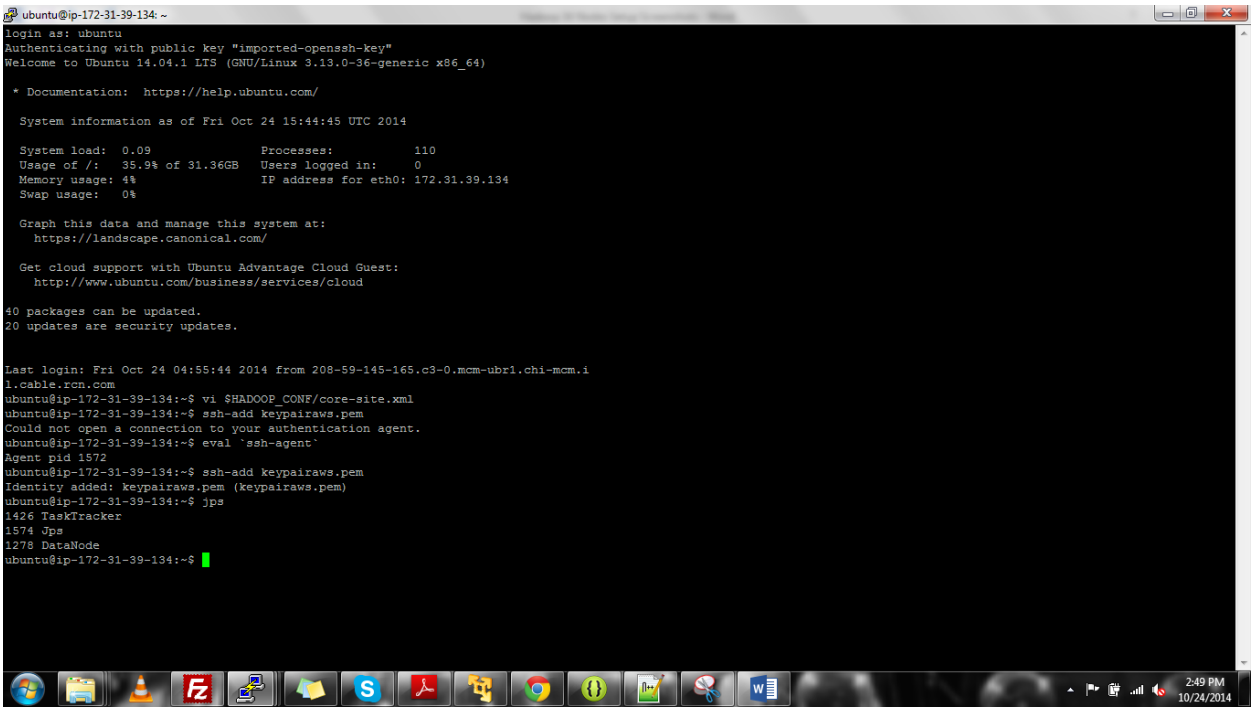
The following are the screenshots of the instances running after we created another master:



Master: NameNode, JobTracker and SecondaryNameNode are running

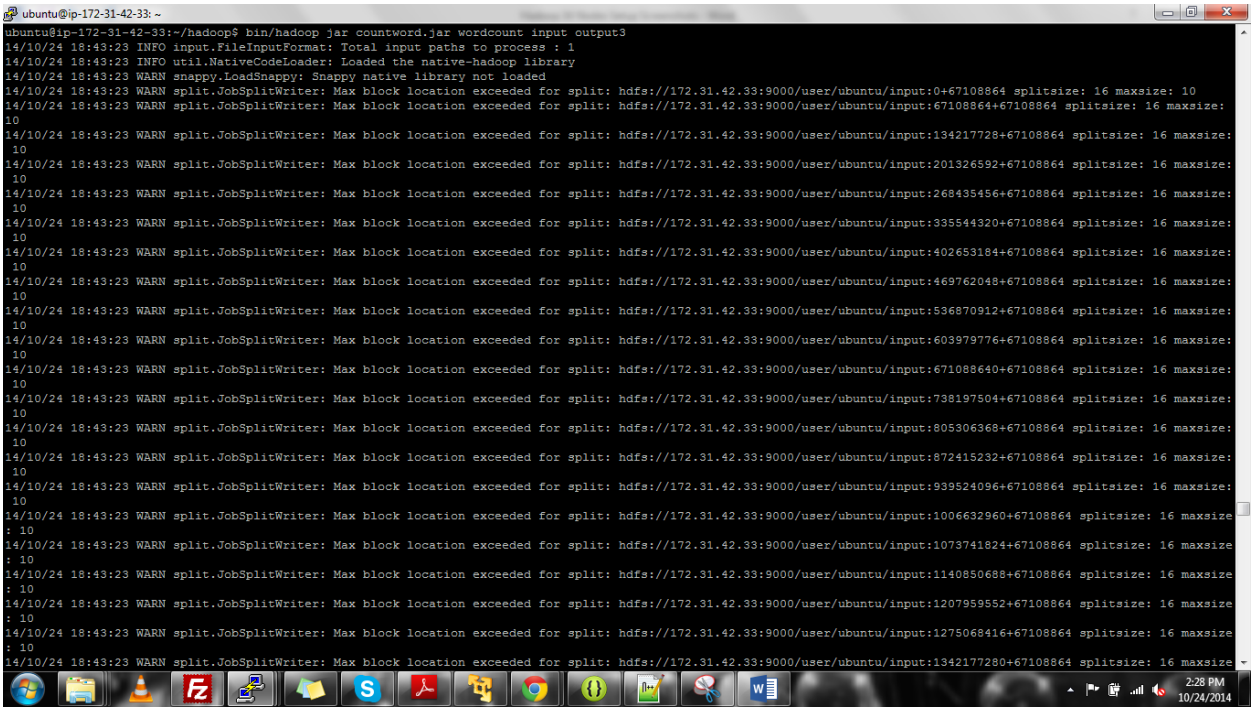


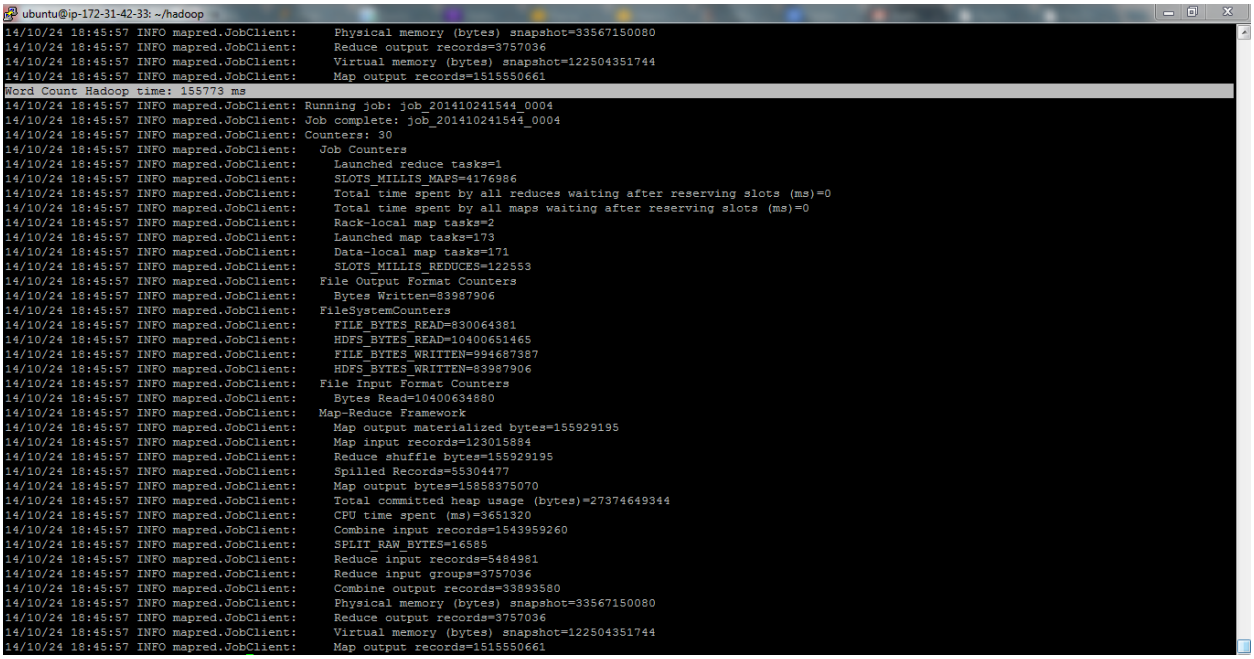
16 Slaves: DataNode and TaskTracker are running



5. Running the Word Count Program:

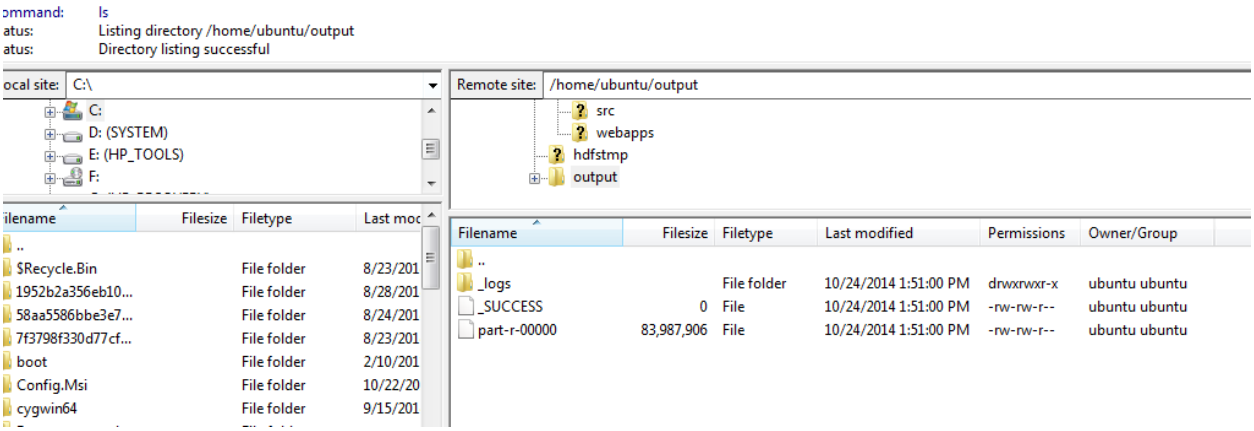
We created a Hadoop MapReduce word count program on eclipse and created a JAR file of the program > countword.jar
We copied the 10GB i.e. wiki10gb file into the Hadoop Filesystem in a folder Input.
Then we transferred the JAR file countword.jar in the home/ubuntu on our master instance.
After copying the required files, we ran the Jar file:

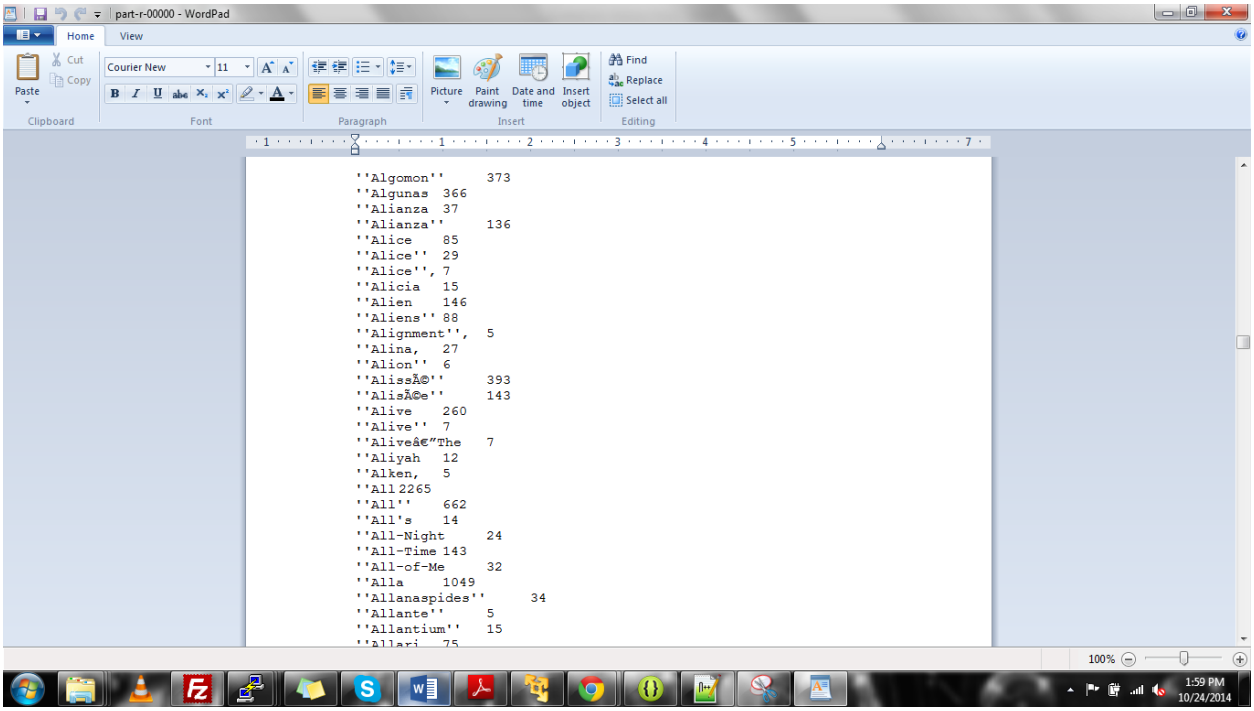




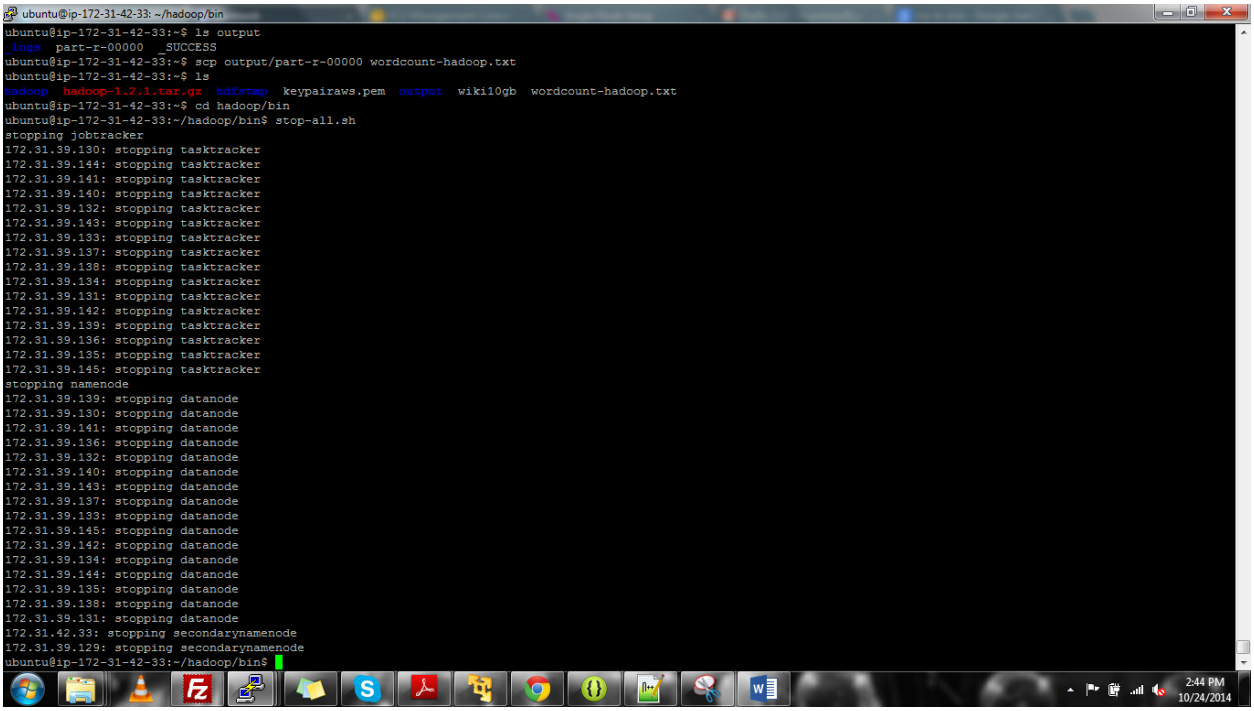
The Program executed successfully on the 16 Nodes and we got an execution Time of **155773 milliseconds** which you can see in the screenshot above.

The Output Folder and the Output are shown below:





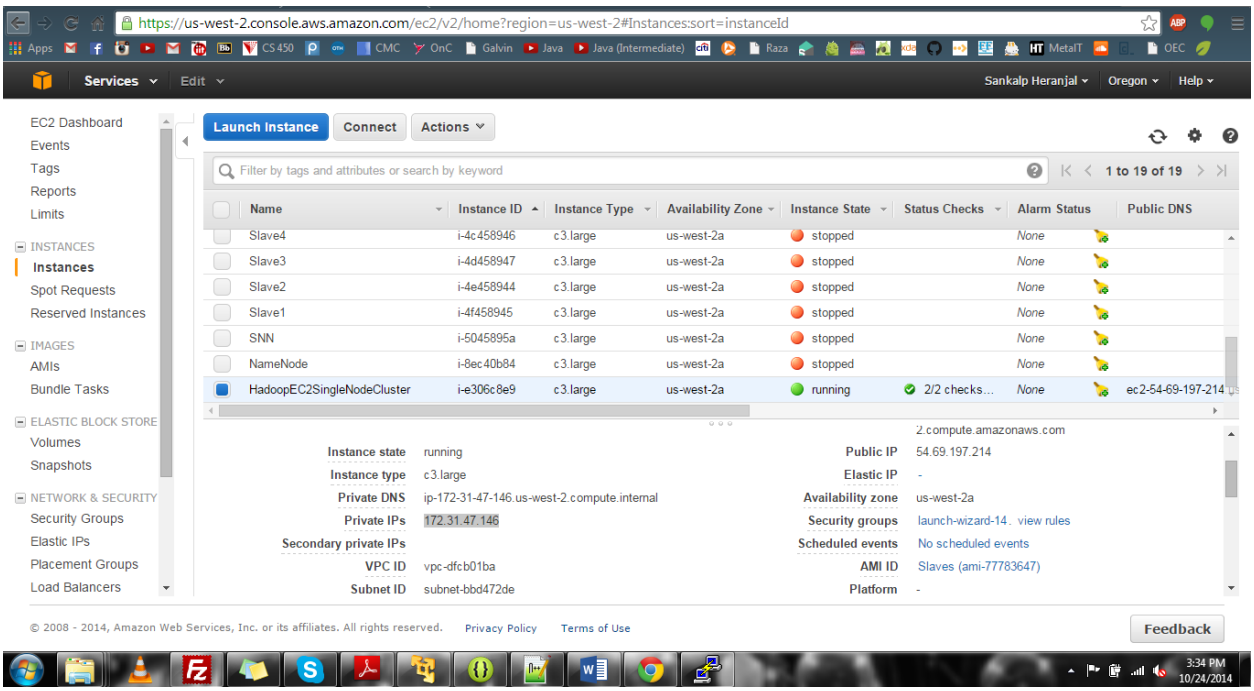
6. **Stopping all the Daemons:**
   After we got the Output, we stopped all the daemons using bin/stop.sh



Setup for Running Word Count on a Single Node:

1. **Launch an Instance**

   We launched another Ubuntu 14.04 c3.large instance using the image with 32GB EBS Storage and used the same key file keypairaws.pem and uploaded it in the instance using FileZilla.



Then we started puTTY to start with our Hadoop word count. We setup a passphraseless ssh to the localhost using:

ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa

cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys

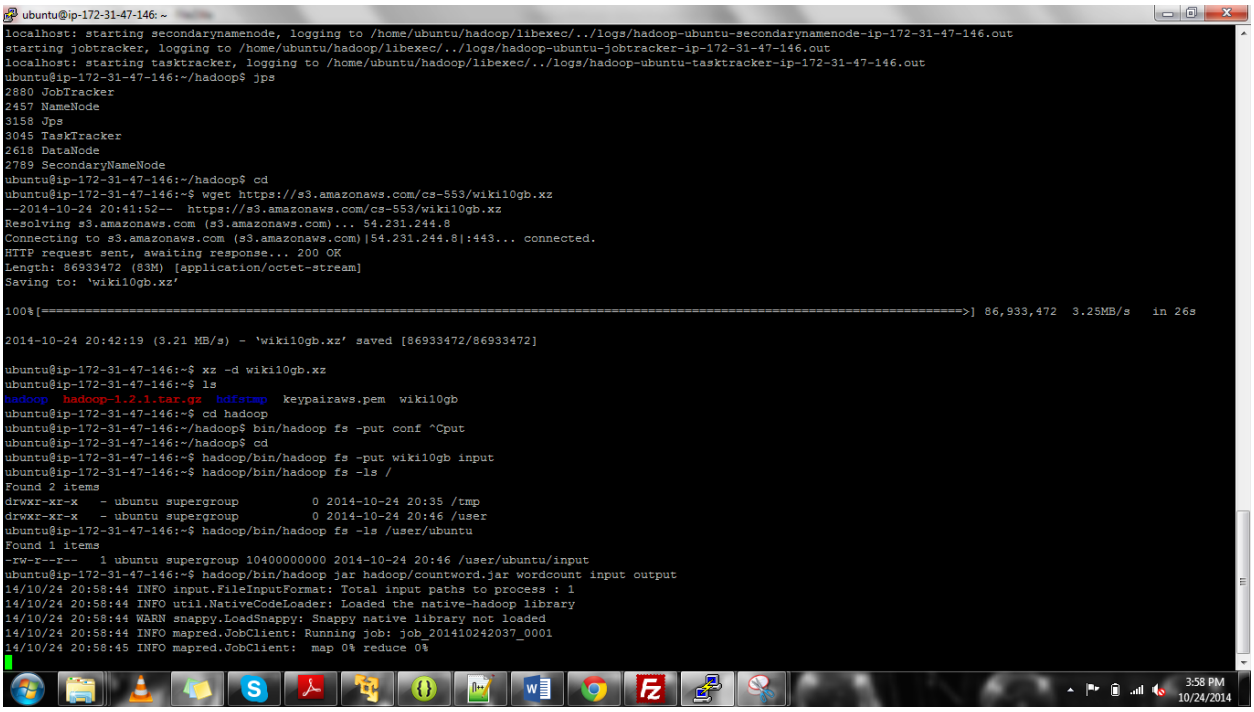
2. **Hadoop Configuration Files**
   We went to the hadoop/conf location and edited the following Configuration Files again for single node (since we created an instance of the image we took last time):
   - **Core-site.xml**
     We have kept it same as that in 16Nodes
   - **Hdfs-site.xml**
     We have provided the 2 properties i.e. dfs.permissions.enabled and dfs.replication.
     The first one is set to false and the latter one is set to 1 since we are using 1 Node.
   - **Mapred-site.xml**
     Even this file is kept the same as 16 Nodes.
   - **Masters**
     We removed IPs given earlier and gave the localhost
   - **Slaves**
     Even in Slaves we removed the previous slave IPs and gave the localhost.

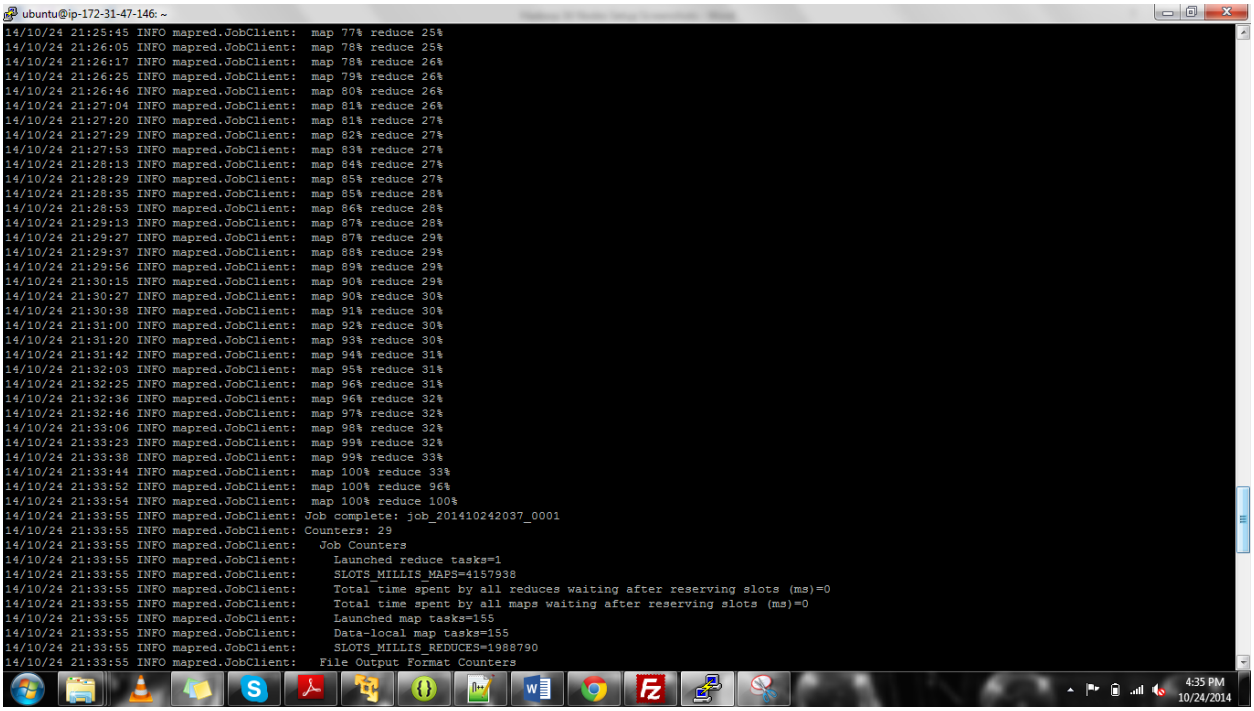
3. **Hadoop Daemon Startup**
   After configuring all the conf files, we started the Hadoop File System. We did this by formatting the NameNode. After Formatting the NameNode, we started the Hadoop daemons by using bin/start.sh

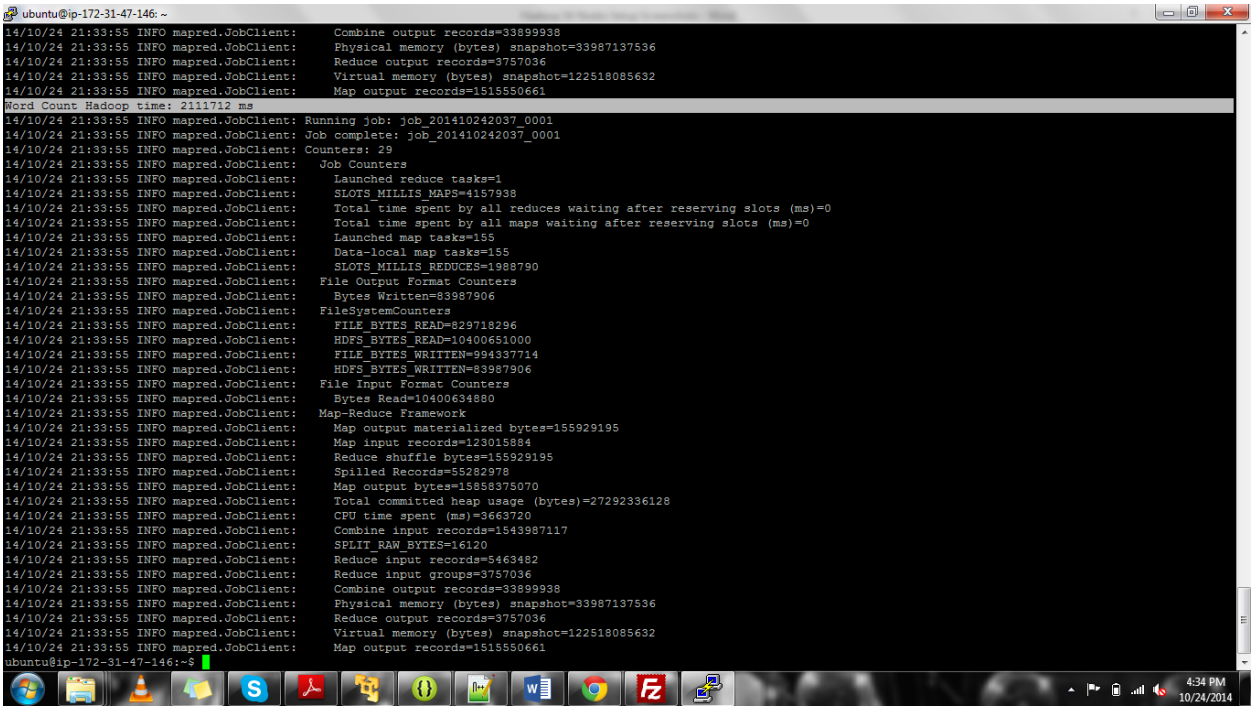
We then ran jps and we can see that the NameNode, SecondaryNameNode, JobTracker, TaskTracker and the DataNode is working in the above screen.

4. Running the Word Count Program:

We took the previously used Hadoop MapReduce word count program > countword.jar for the single node.
We again copied the 10GB i.e. wiki10gb file into the Hadoop Filesystem in a folder Input.
Then we transferred the JAR file countword.jar in the home/ubuntu on our instance.
After copying the required files, we ran the Jar file:



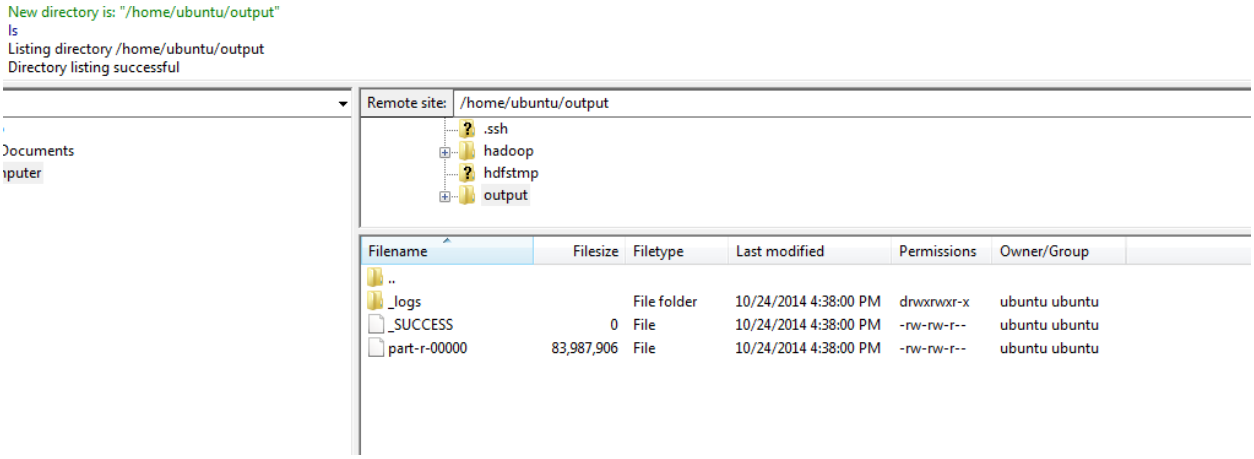


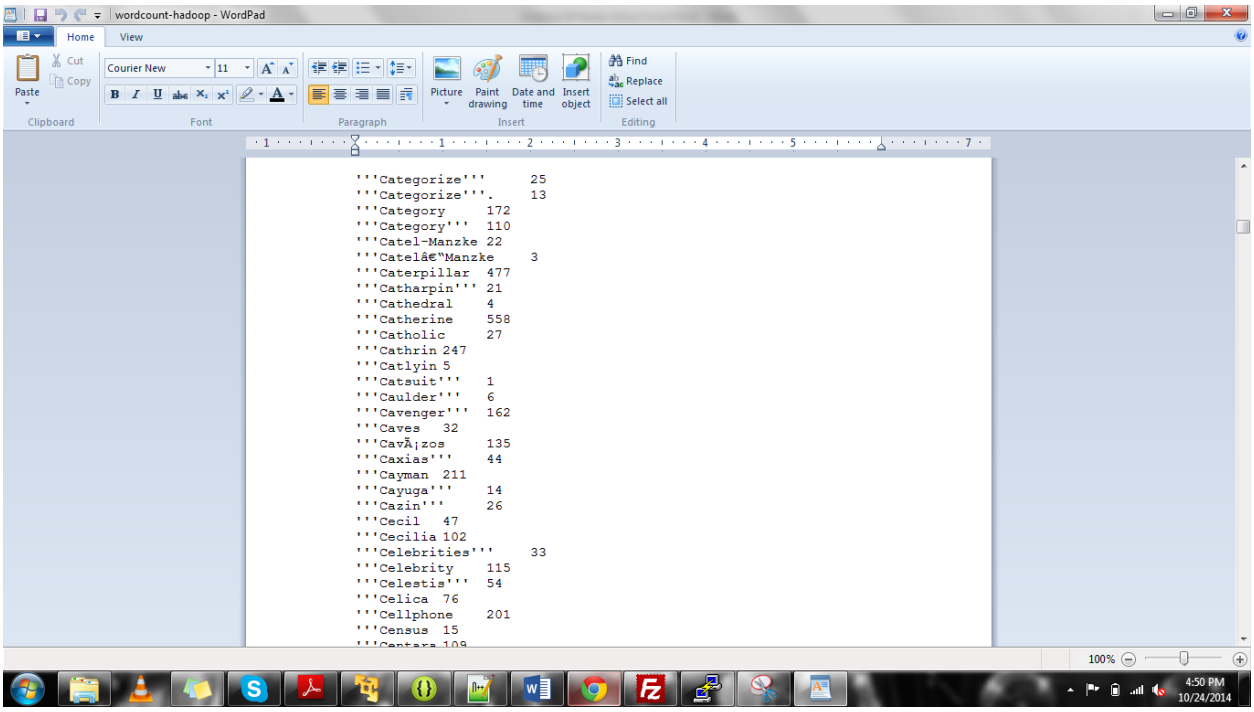


The Program executed successfully on the Single Node and we got an execution Time of **2111712 milliseconds** which you can see in the screenshot above.

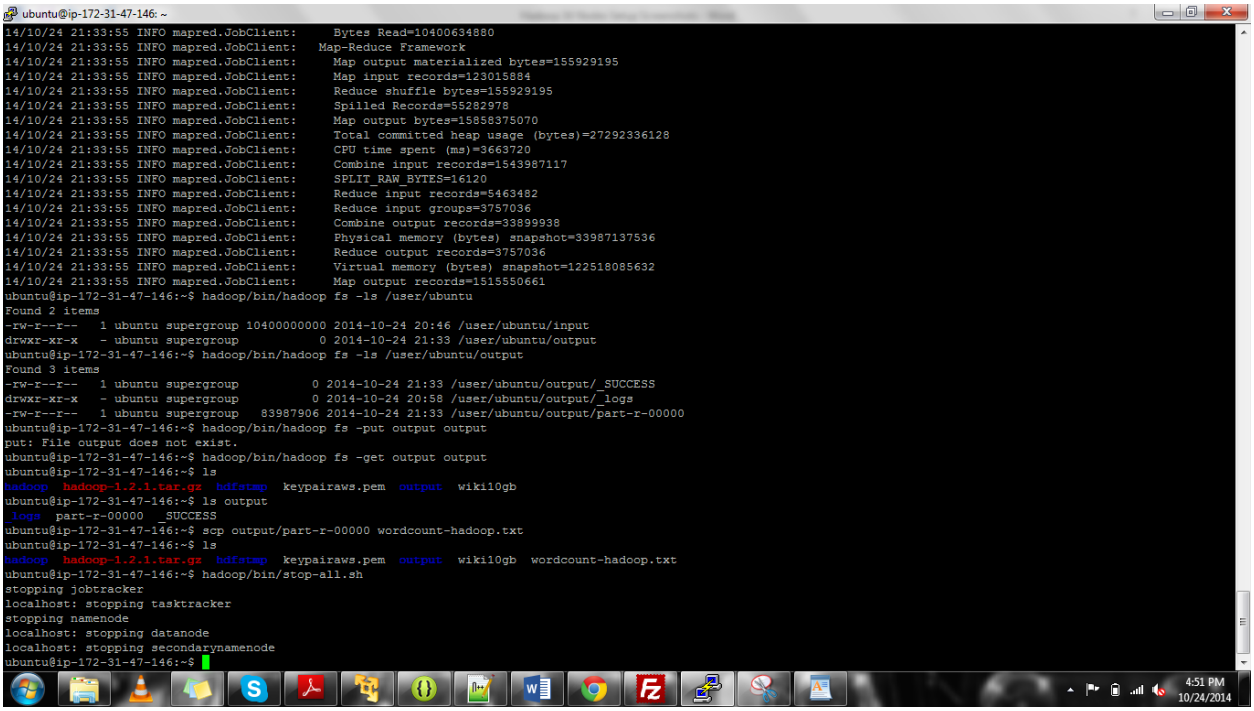
The Output Folder and the Output are shown below:

```
ubuntu@ip-172-31-47-146:~$ hadoop/bin/hadoop fs -ls /user/ubuntu
Found 2 items
-rw-r--r--   1 ubuntu supergroup 10400000000 2014-10-24 20:46 /user/ubuntu/input
drwxr-xr-x   - ubuntu supergroup          0 2014-10-24 21:33 /user/ubuntu/output
ubuntu@ip-172-31-47-146:~$ hadoop/bin/hadoop fs -ls /user/ubuntu/output
Found 3 items
-rw-r--r--   1 ubuntu supergroup          0 2014-10-24 21:33 /user/ubuntu/output/_SUCCESS
drwxr-xr-x   - ubuntu supergroup          0 2014-10-24 20:58 /user/ubuntu/output/_logs
-rw-r--r--   1 ubuntu supergroup   83987906 2014-10-24 21:33 /user/ubuntu/output/part-r-00000
ubuntu@ip-172-31-47-146:~$ hadoop/bin/hadoop fs -put output output
put: File output does not exist.
ubuntu@ip-172-31-47-146:~$ hadoop/bin/hadoop fs -get output output
ubuntu@ip-172-31-47-146:~$ ls
hadoop  hadoop-1.2.1.tar.gz  hdfstmp  keypairaws.pem  output  wiki10gb
ubuntu@ip-172-31-47-146:~$ ls output
_logs  part-r-00000  _SUCCESS
ubuntu@ip-172-31-47-146:~$
```

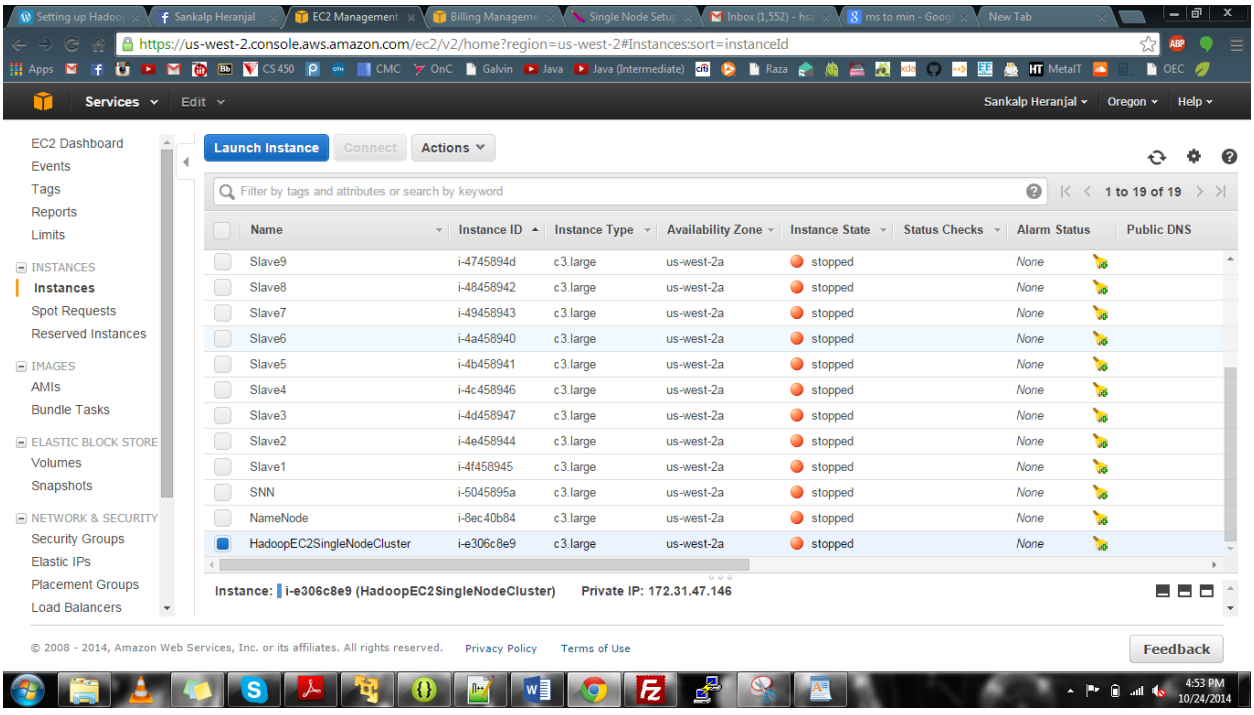




5. **Stopping all the Daemons:**
   After we got the Output, we stopped all the daemons using bin/stop.sh



After the Execution of the Program, we stopped all the running instances:



Performance Evaluation:

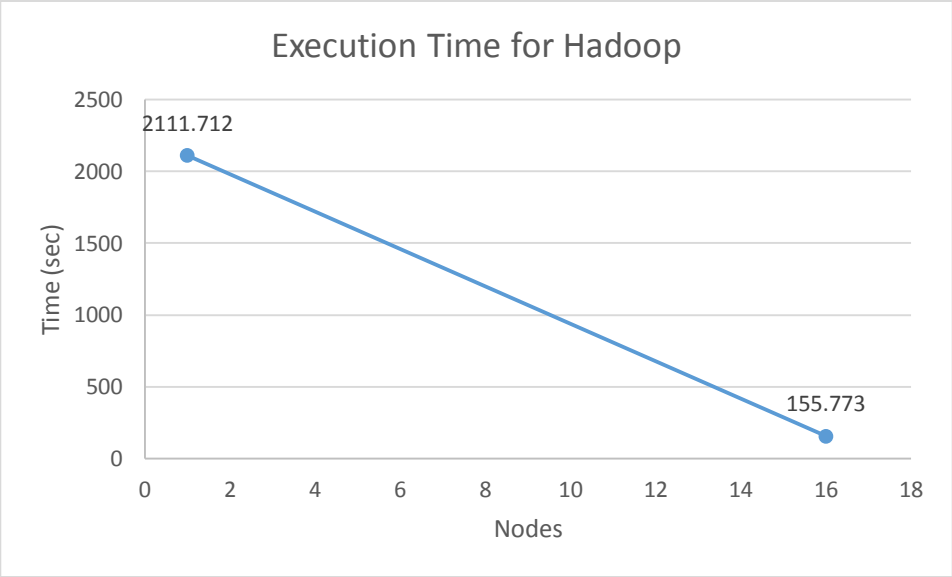
The Execution Time for running the Word Count Program on a Single node = **2111712 milliseconds**

= 2111.712 seconds

The Execution Time for running the Word Count Program 16 nodes = **155773 milliseconds**

= 155.773 seconds

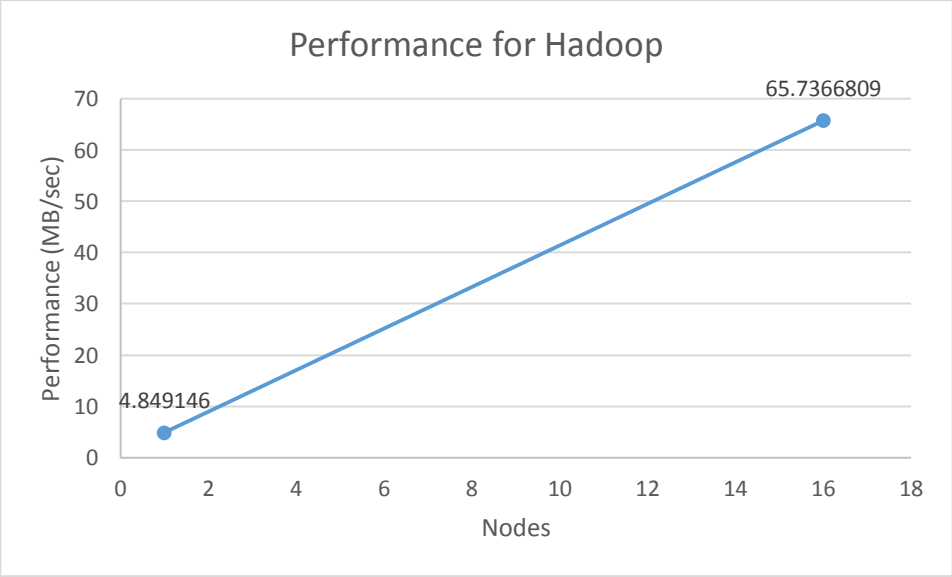
Graph of Execution Time for Hadoop vs Number of Nodes:



Performance in a Single Node = Total Size of File / Execution Time = **4.849146MB/sec**

Performance in 16 Nodes = Total Size of File / Execution Time = **65.7366809MB/sec**

Graph of Performance for Hadoop vs Number of Nodes:



Output Table:

Number of Nodes	File Size	Execution Time	Performance
1	**10240MB**	**2111.712sec**	**4.849146MB/sec**
16	**10240MB**	**155.773sec**	**65.7366809MB/sec**

Observation:

We observe that we get a better performance on a 16 node system than a single node system. Since there are 16 slaves working on the same problem, the work is equally divided between all the slaves and at a time each slave is doing the computation. This reduces the overall execution time when 16 Nodes are used. In a single node, that node is doing the whole computation by itself, so it takes a lot of time to execute.

What is a Master node? What is a Slaves node?

Master Node: The Master Node is the one which stores all the data in the HDFS (Hadoop Distributed File System) and runs Map Reduce on all the data. Here the NameNode is used to store the data and the JobTracker does the parallel computation on that data i.e. Map Reduce.

Slave Node: The Slave Nodes are the ones which do all the computation work. The Slaves consists of both Data Node and Task Tracker. They receive the instructions from the master and do the required computations.

How can we change the number of mappers and reducers from the configuration file?
The number of reducers is controlled by mapred.reduce.tasks and the number of maps is controlled by mapred.map.tasks in the mapred conf file in the mapred-site.xml file.

Why do we need to set unique available ports to those configuration files on a shared environment?
The port number is used by Hadoop to find the path on the HDFS whose NameNode is running at <HDFSip>:<port>

Conclusion: We haven't provided the output file for the word count in Hadoop as it was 80MB each and according to the TA's mail, we have to keep the assignment folder small. So we have kept a Screenshot of the output in the Documentation. We have the copy of the output files in our system if you need it in the future.