PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

DESCRIPTION:

The purpose of this document is to describe steps to install, configure and capture the results before and after running Sorting implementation on files greater than memory size in Hadoop, Spark, Java. Also, by comparing the performance of Shared Memory Sort Java, Hadoop and Spark and determine which one is better.

Operating System: Linux Operating System.
Java version: 1.7.0
Hadoop version:2.7.2
Spark version 1.6.1
Instance type c3.large
Linux distribution kernel: 3.19.0-25-generic

HADOOP:

Hadoop single node is implemented using Java language. It consists of mapper and reducer phase. In the mapper phase the text input format is used and the key value pair for one record is extracted and push it to output collector. In the reducer phase, iterate through values with respect to the key and push to output collector in sorted order.

Configuration File Description:

1. hadoop-env.sh- This file requires environment variables settings such as JAVA_HOME which affects the Hadoop daemon behavior while storing files and increasing heap size.
2. Slaves- This file consists the public DNS of Master and Slave where the Hadoop slave daemons will run. This configuration required for multimode setup. In case of single node it consists of localhost.
3. Core-site.xml- This file consists of I/O configuration setting for Hadoop Core common t to HDFS and MapReduce.
4. Hdfs-site.xml- This file contains the configuration setting for HDFS daemon, the namenode, the secondary namenode and the datanode nodes.
5. Mapred-site.xml- This file contains the configuration setting for Hadoop daemon job tracker and task tracker
6. Yarn-site.xml- This file contains the configuration setting for YARN daemon the ResourceManager, NodeManager, and WebAppProxy.

The modifications that needed when configuring files on multi node are as follows:
For each Slave Node

1. Update the Hosts file with the private IP and public DNS of master and slave.
2. Update the Slaves file with the public DNS of master and slave.

For the Master Node

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

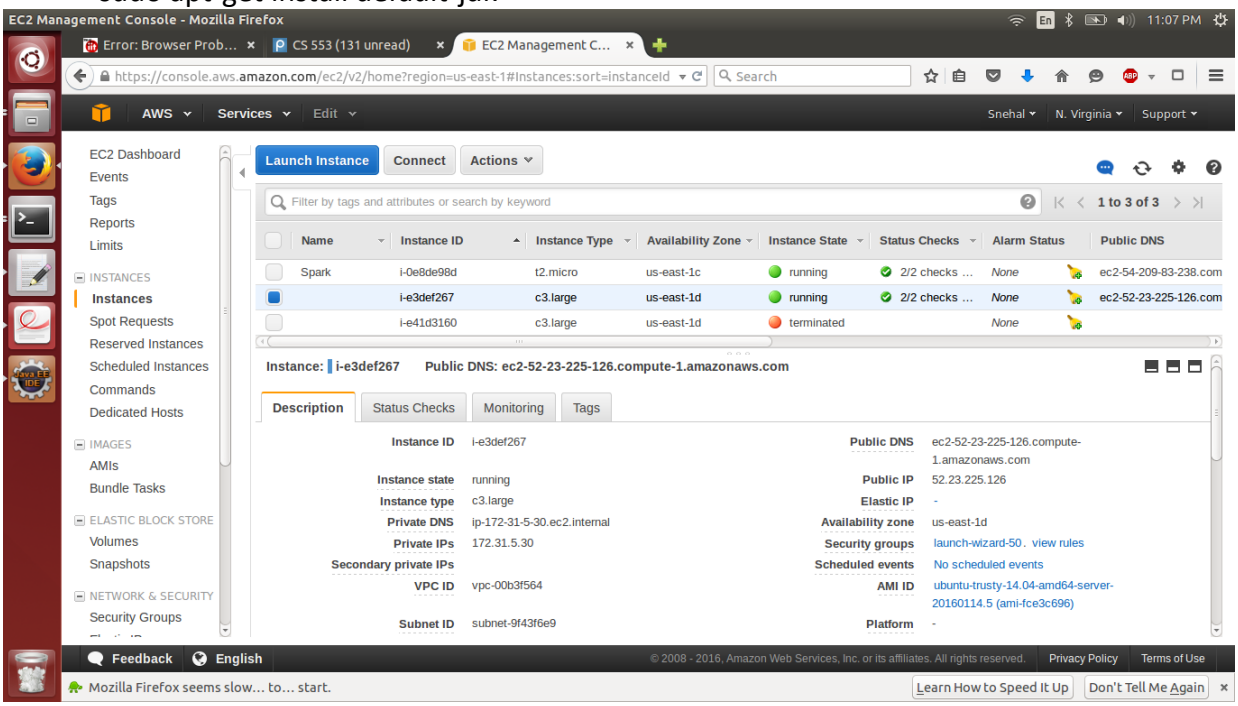
1. Update the Hosts file with the private IP and public DNS of master and all the slave nodes.
2. Update the Slaves file with the public DNS of master and all the slave nodes.

Questions:

1. What is a Master node? What is a Slaves node?
   Answer: Master node stores lots of data (HDFS) and runs parallel computation on all that data. Master node assigns work to Slaves. Slave node is the place where data is stored and where data processing takes place.
2. Why do we need to set unique available ports to those configuration files on a shared environment? What errors or side-effects will show if we use same port number for each user?
   In the configuration files, MapReduce job tracker is set to 9001, ResourceManager's resource tracker is set to 9025 and ResourceManager's scheduler is set to 9030. All are using different ports to avoid port collision. The bandwidth of the port will be shared and will raise security concerns.
3. How can we change the number of mappers and reducers from the configuration file?
   The mappers and reducers can be changed but updating the mapred.tasktracker.map.tasks.maximum and mapred.tasktracker.reduce.tasks.maximum in the mapred-site.xml.

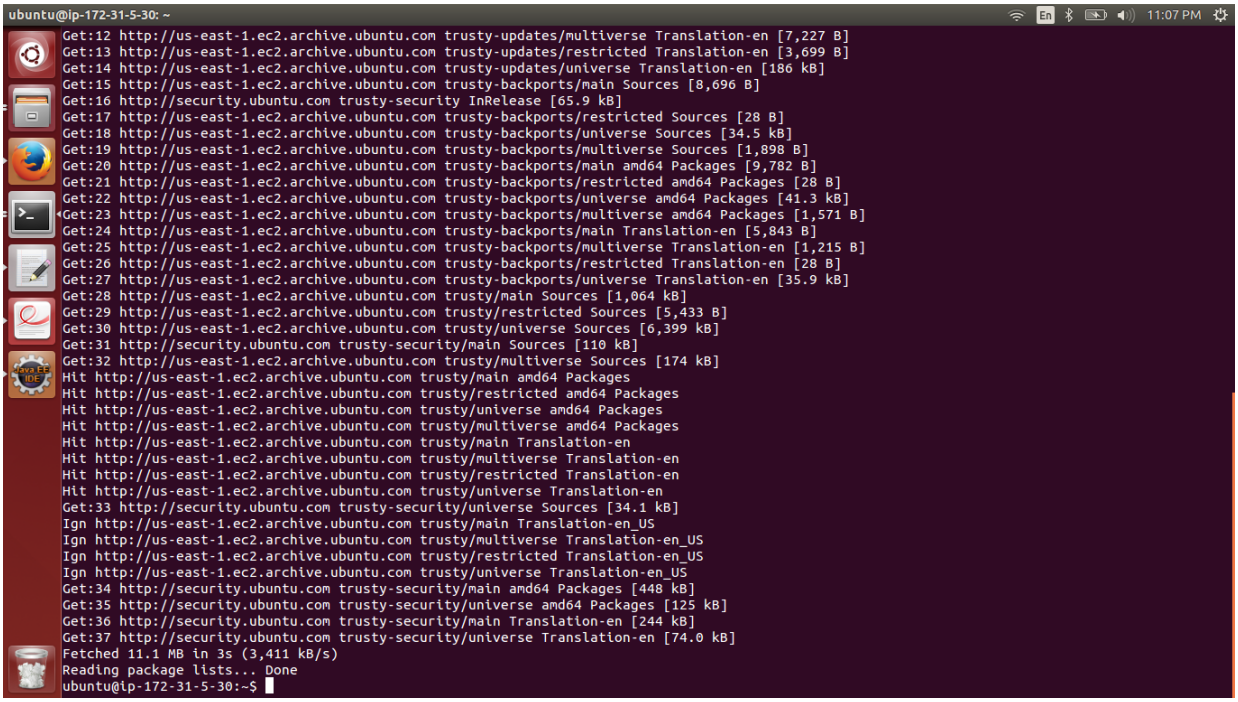
1. Installation of Hadoop 1-node:

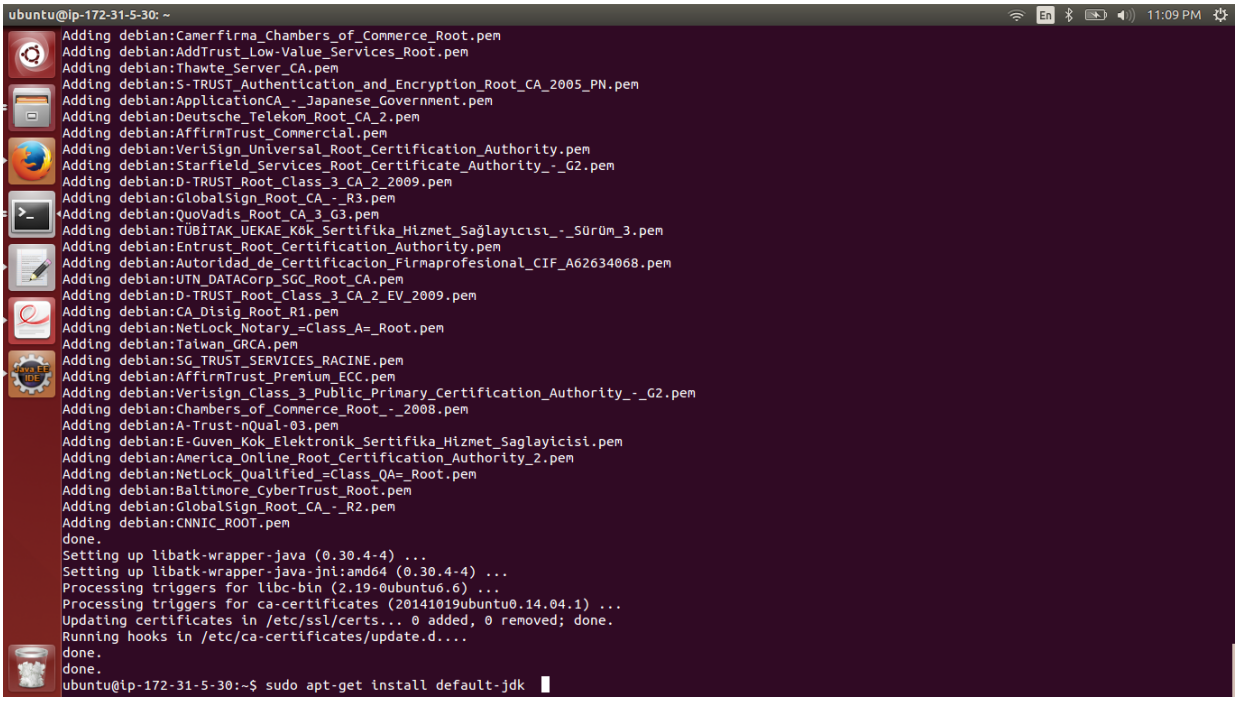
Step1: sudo apt-get update
sudo apt-get install default-jdk



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



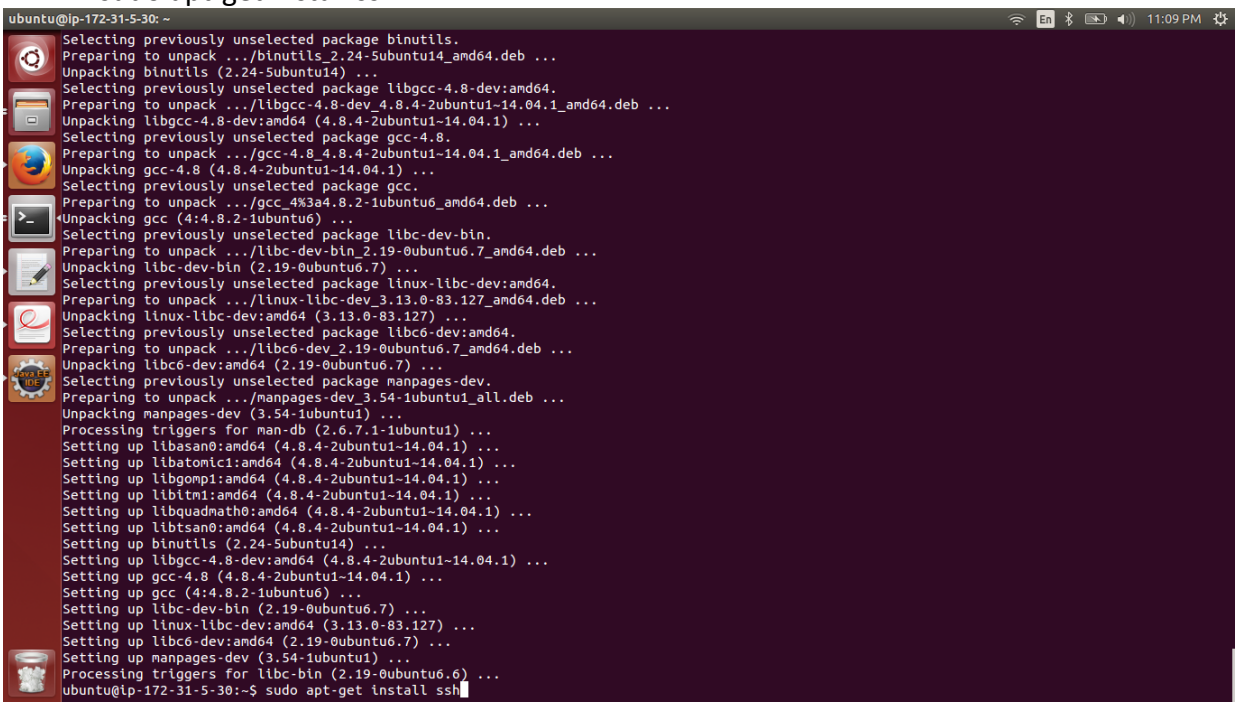


PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step2: Install ssh

sudo apt-get install ssh

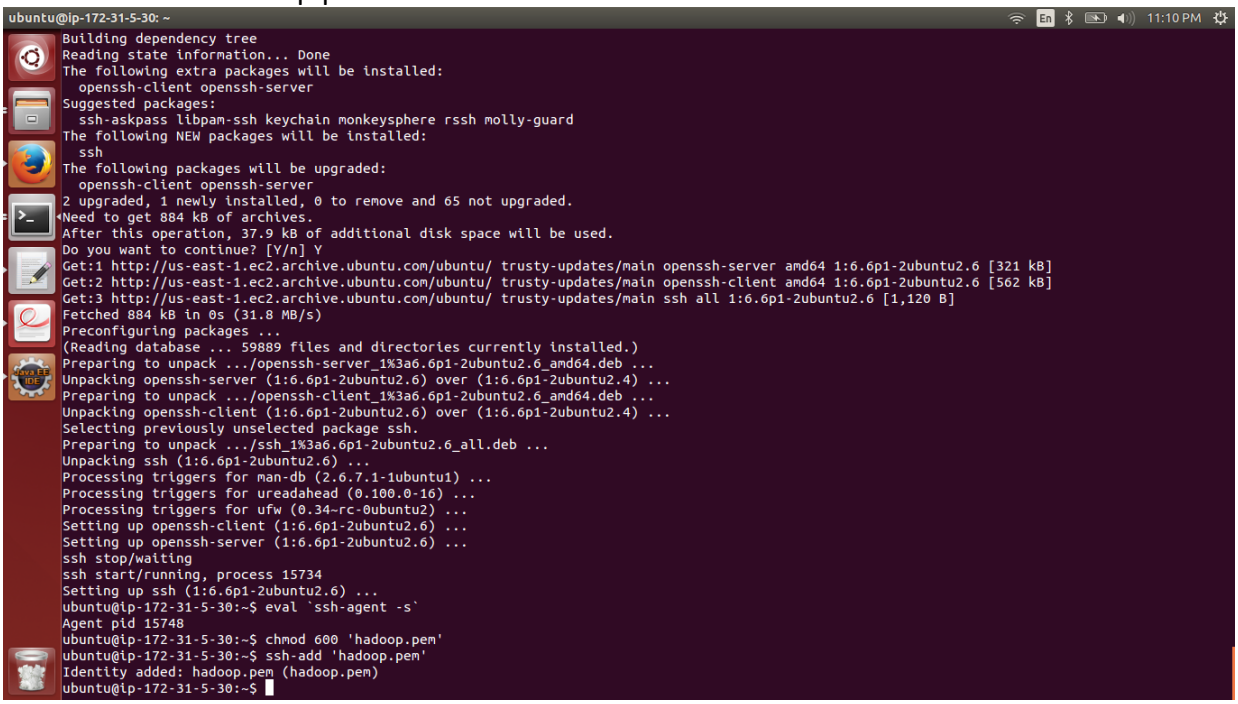


Step 3: add pem file to ssh

eval `ssh-agent -s`

chmod 600 'hadoop.pem'

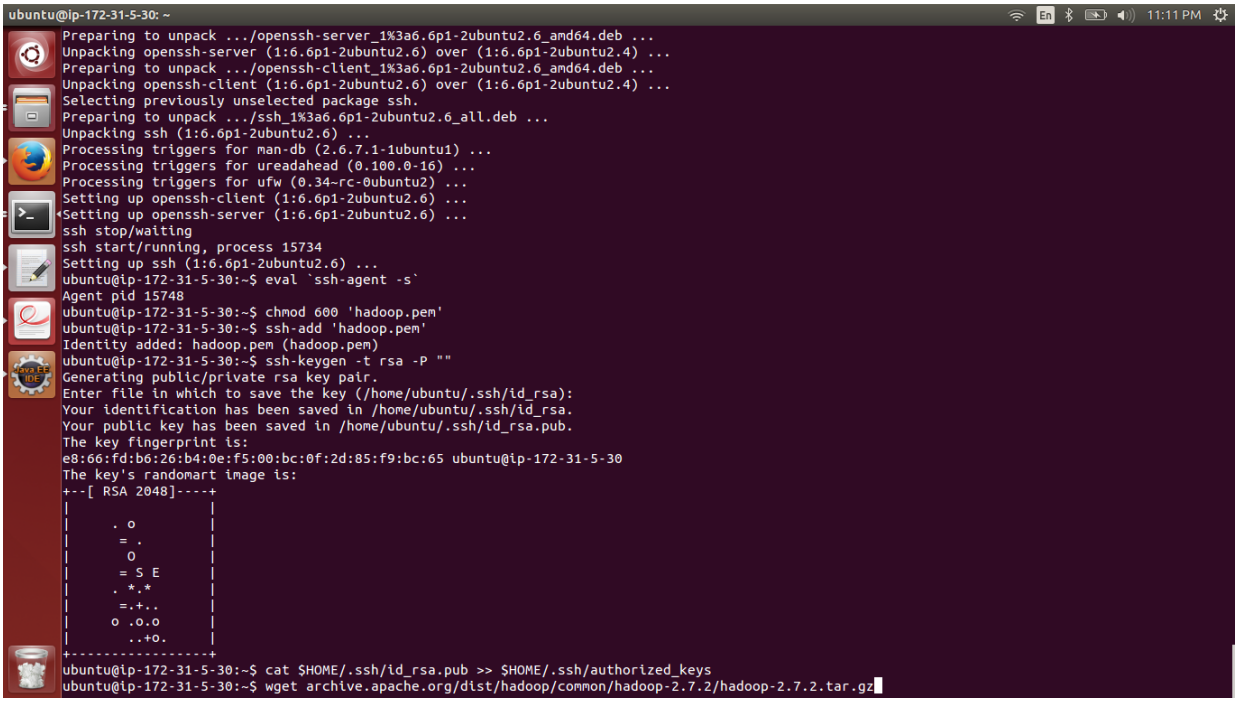
ssh-add 'hadoop.pem'



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 4: Set up SSH certificates to avoid entering password each time:

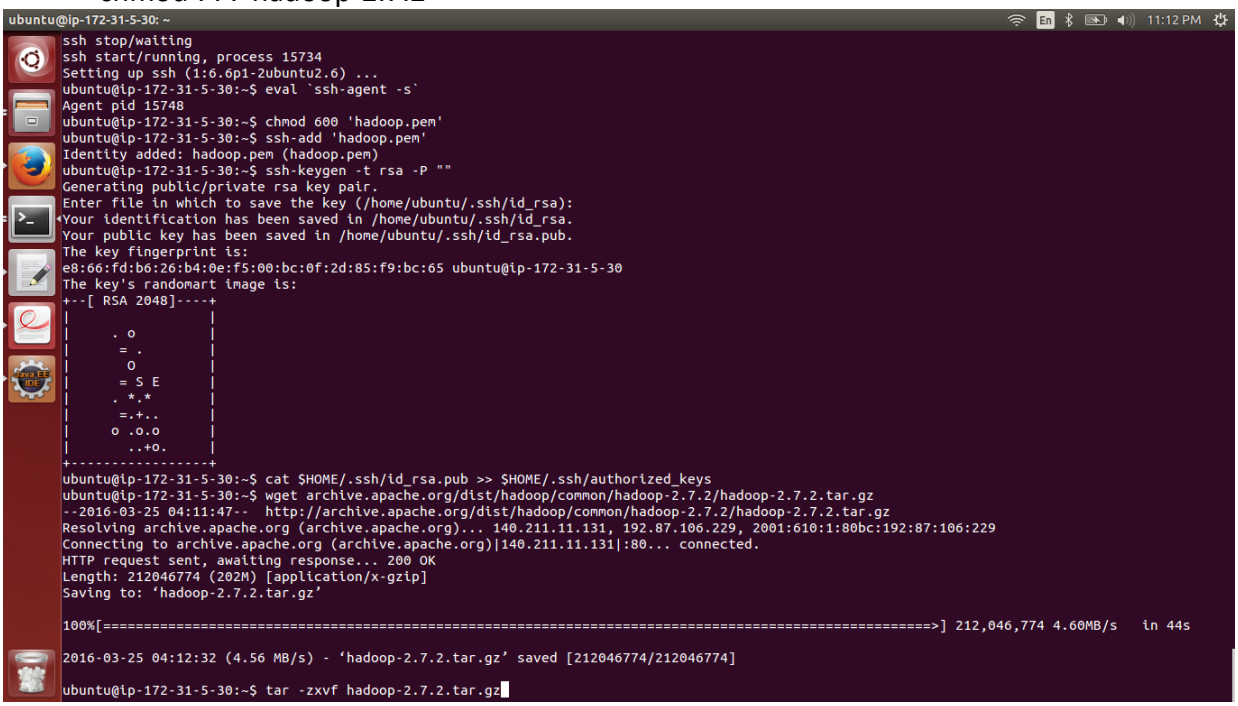


Step 5: Download and install Hadoop

wget archive.apache.org/dist/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz

tar -zxvf hadoop-2.7.2.tar.gz

chmod 777 hadoop-2.7.2



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

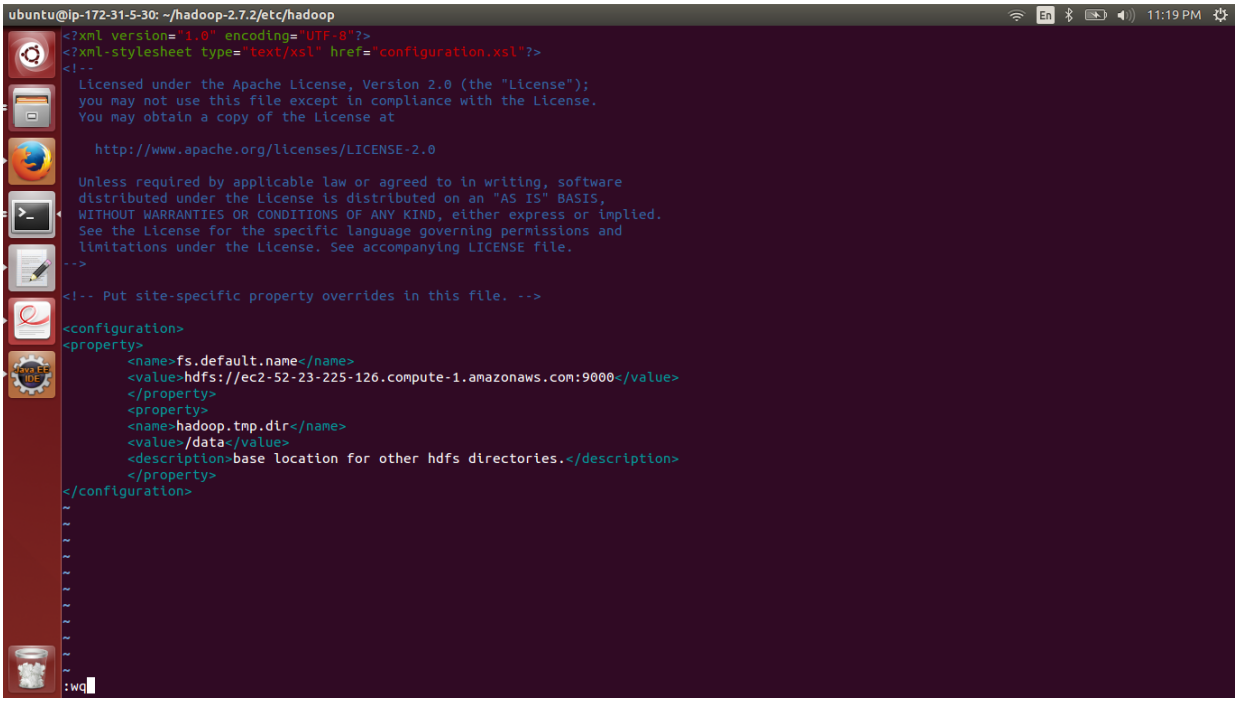
Snehal Sonawane **A20360111**

Step 6: Setup Configuration Files:

6.1. Edit environmental variables in bashrc file :

```
sudo vim .bashrc
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/home/ubuntu/hadoop-2.7.2
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HASOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS='-Djava.library.path=$HADOOP_INSTALL/lib'
```

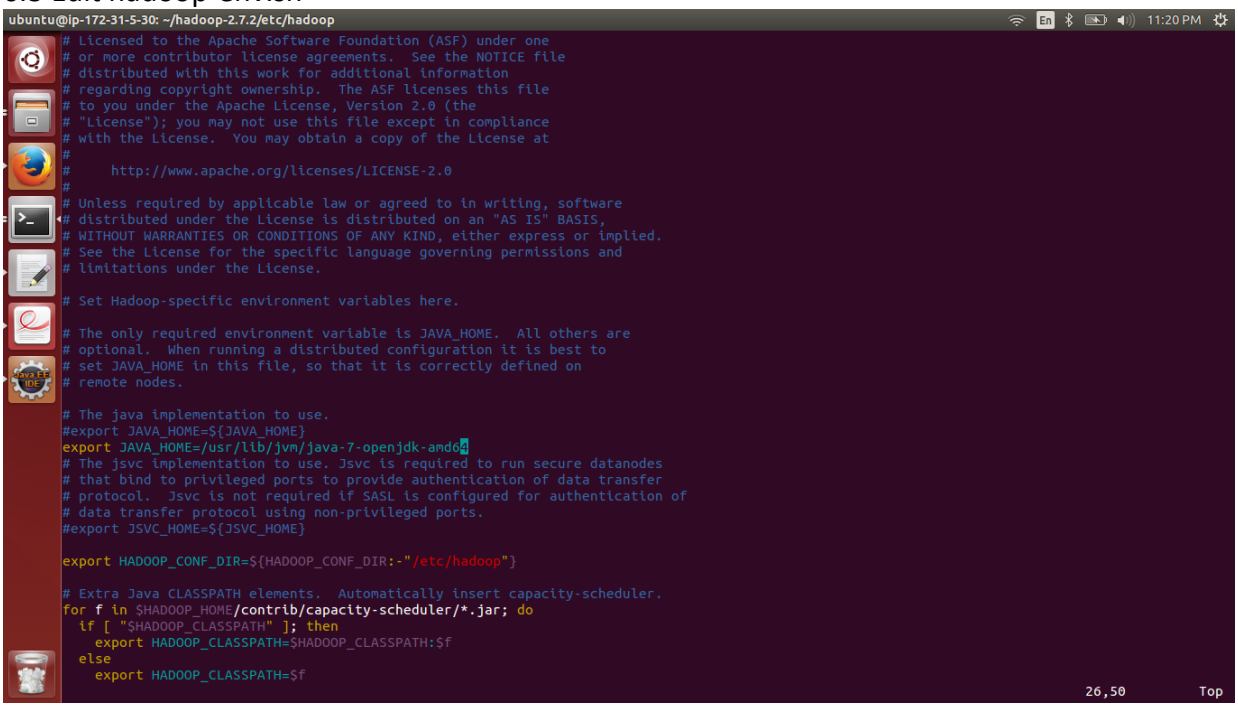
6.2. Edit core-site.xml



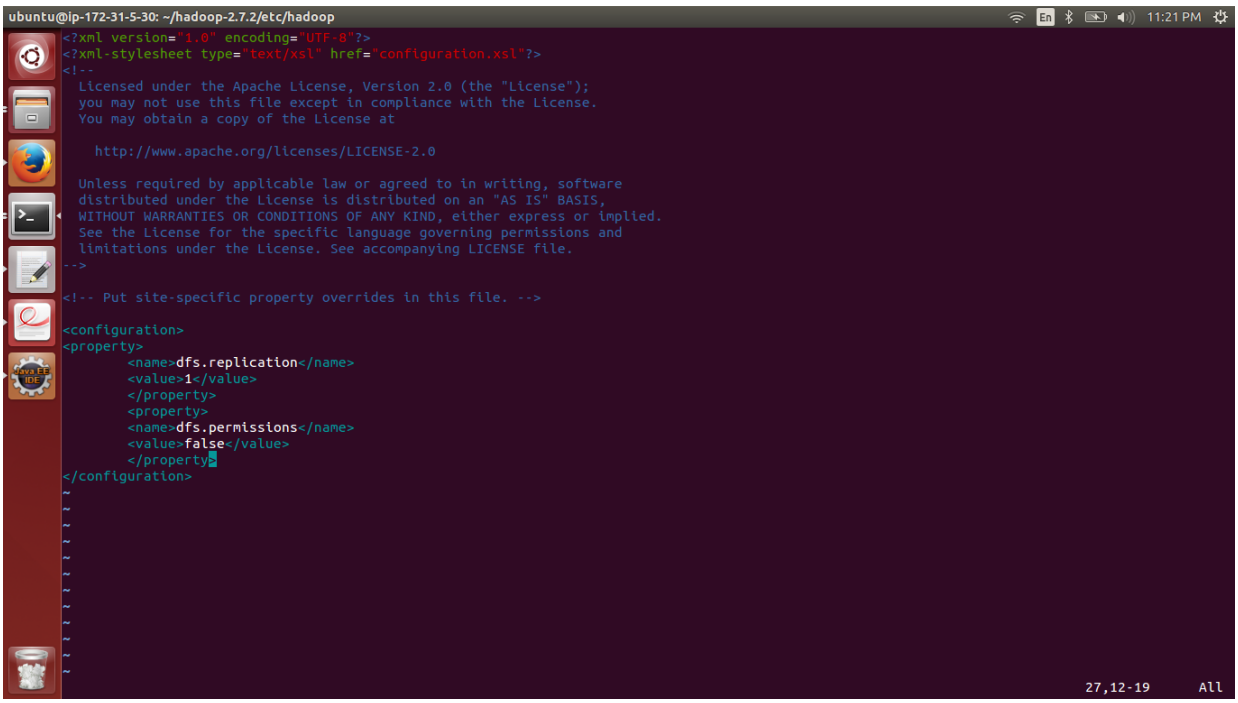
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

6.3 Edit hadoop-env.sh



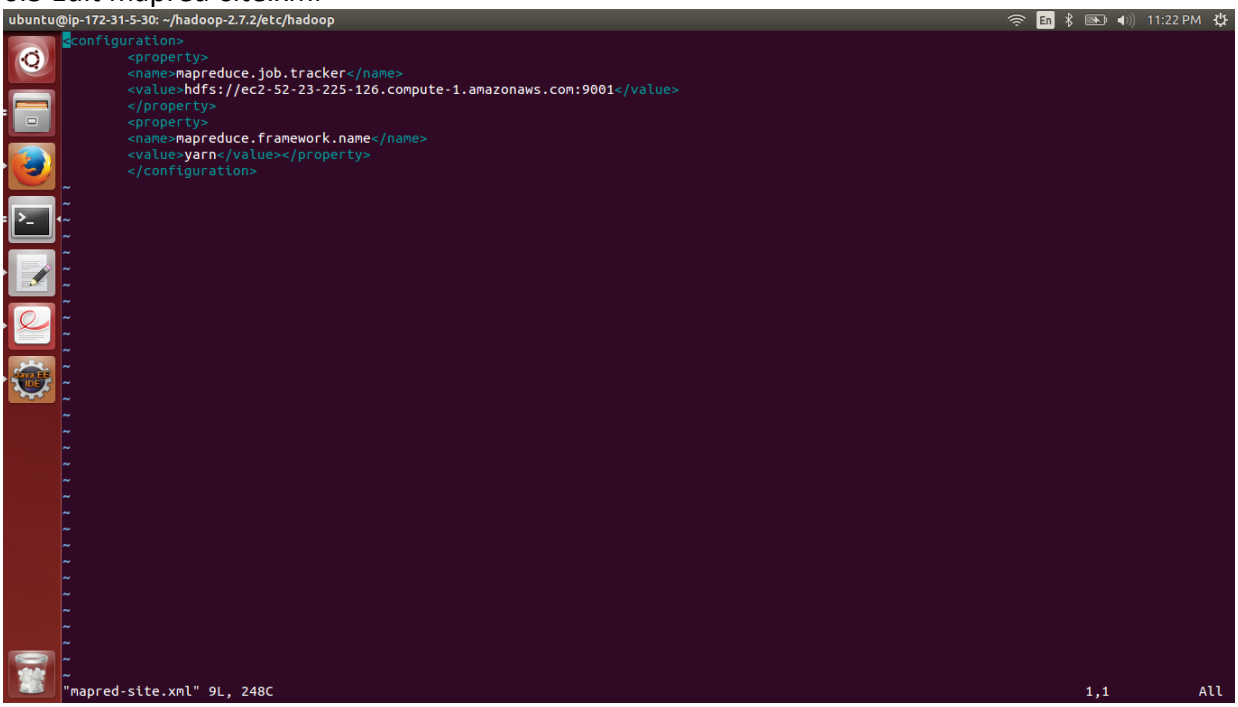
6.4 Edit hdfs-site.xml



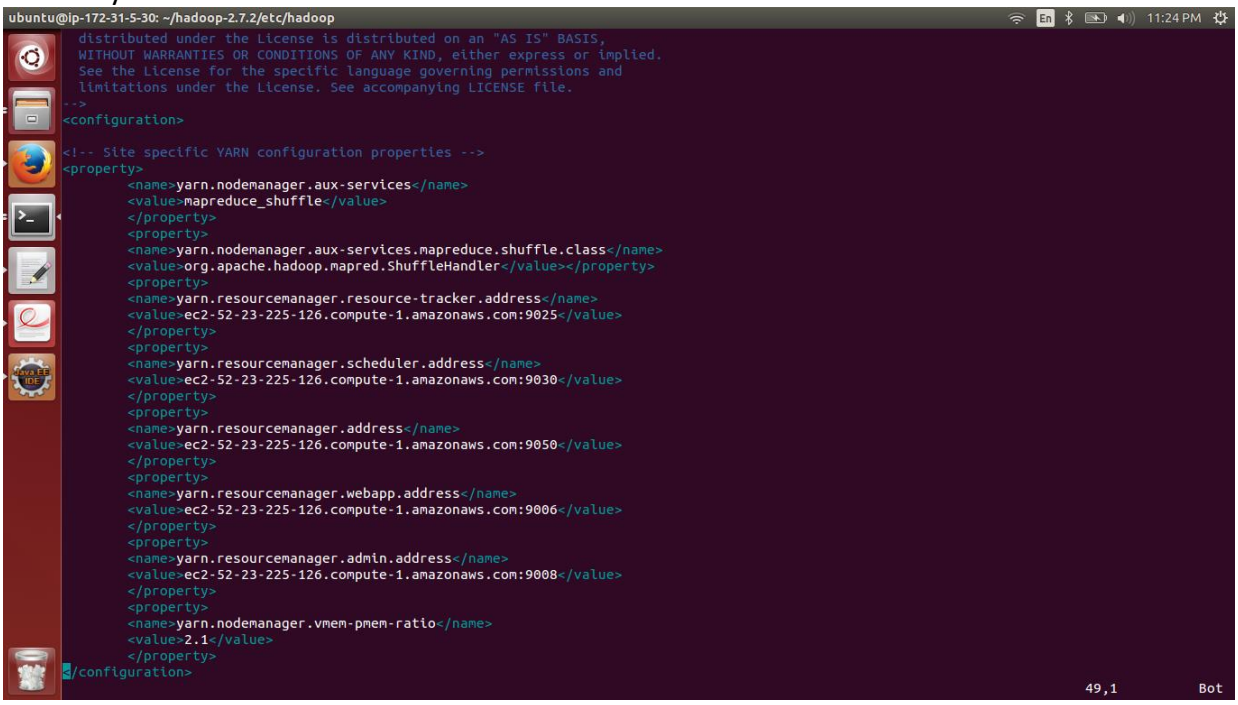
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

6.5 Edit mapred-site.xml



6.6 Edit yarn-site.xml

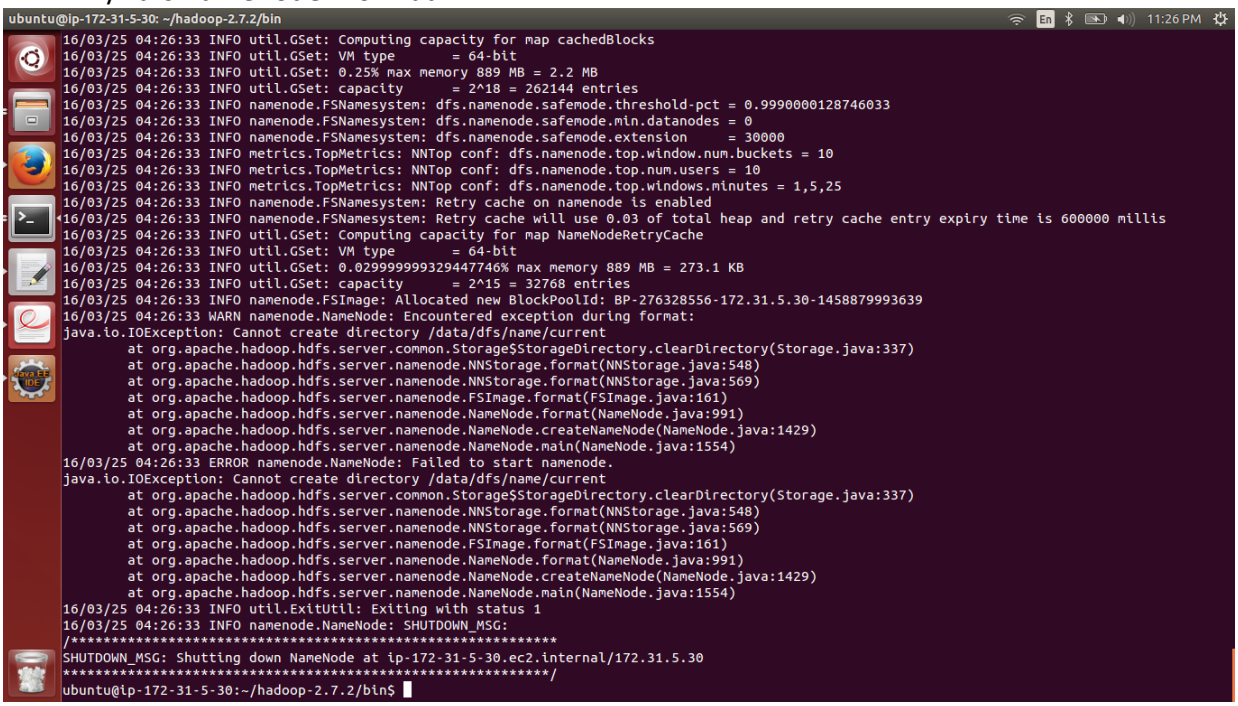


PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

6.7 Format the Namenode

./hdfs namenode –format

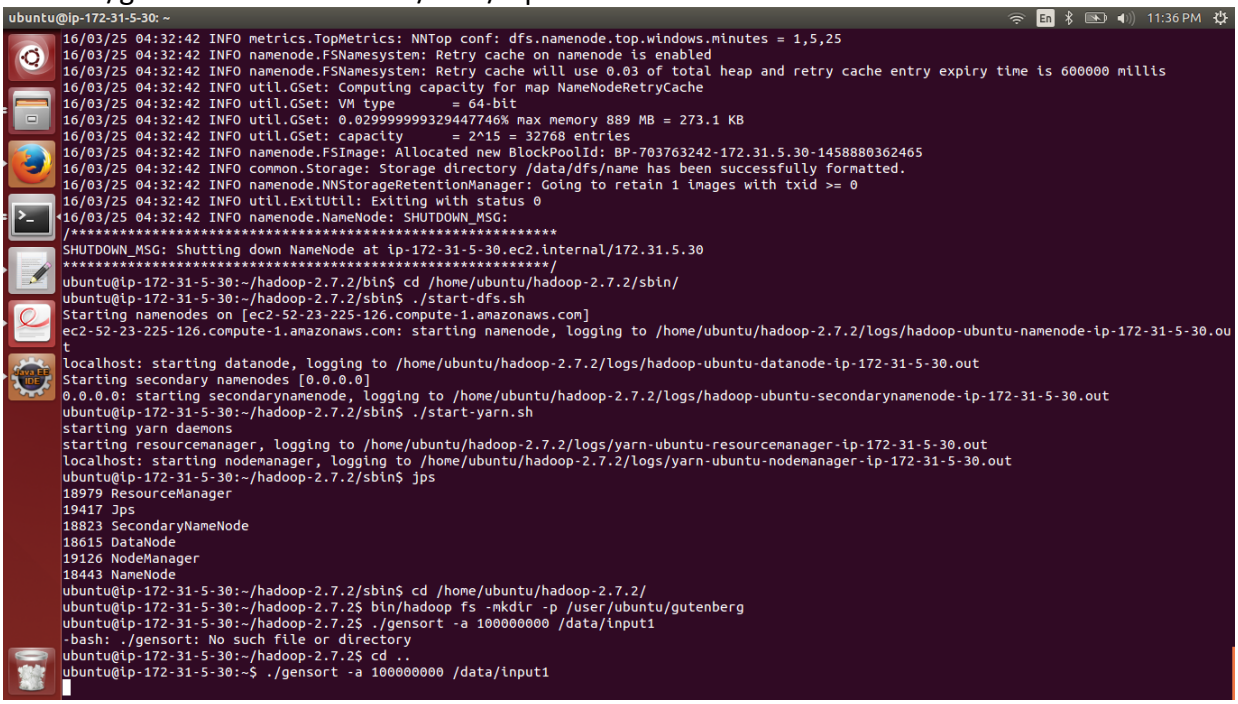


6.8 Start dfs and yarn and generate data

./start-dfs.sh

./start-yarn.sh

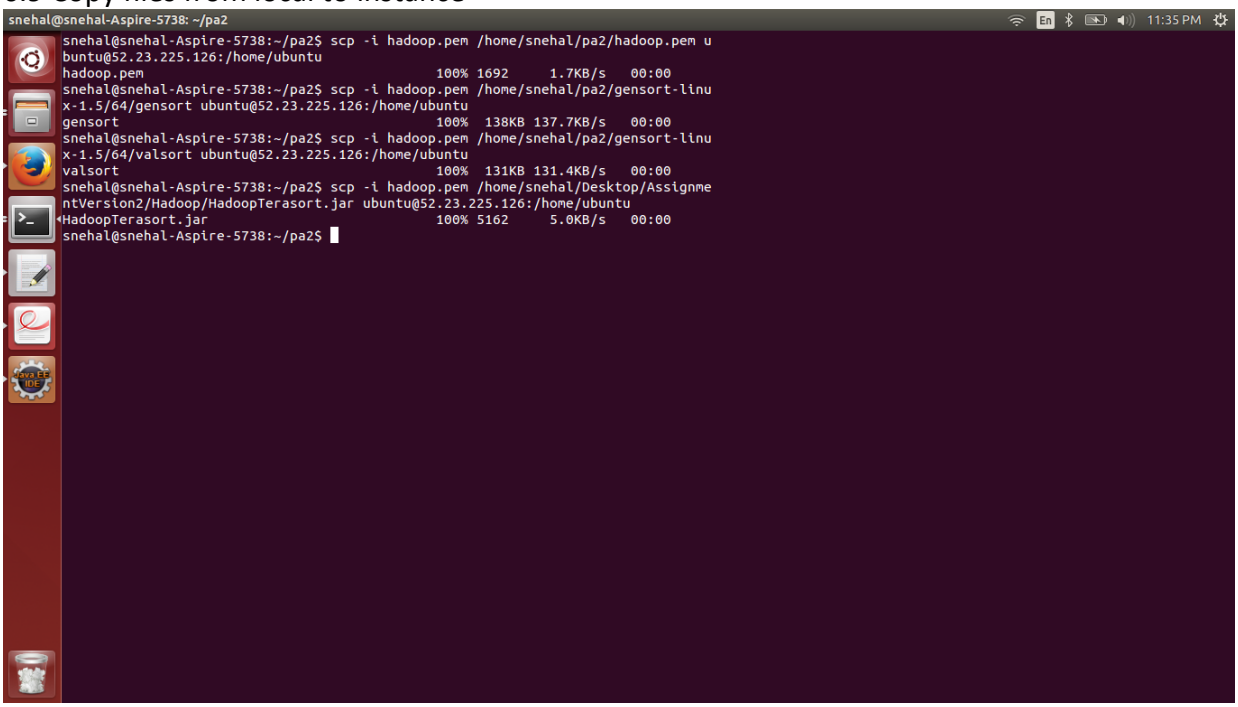
./gensort –a 100000000 /data/input1



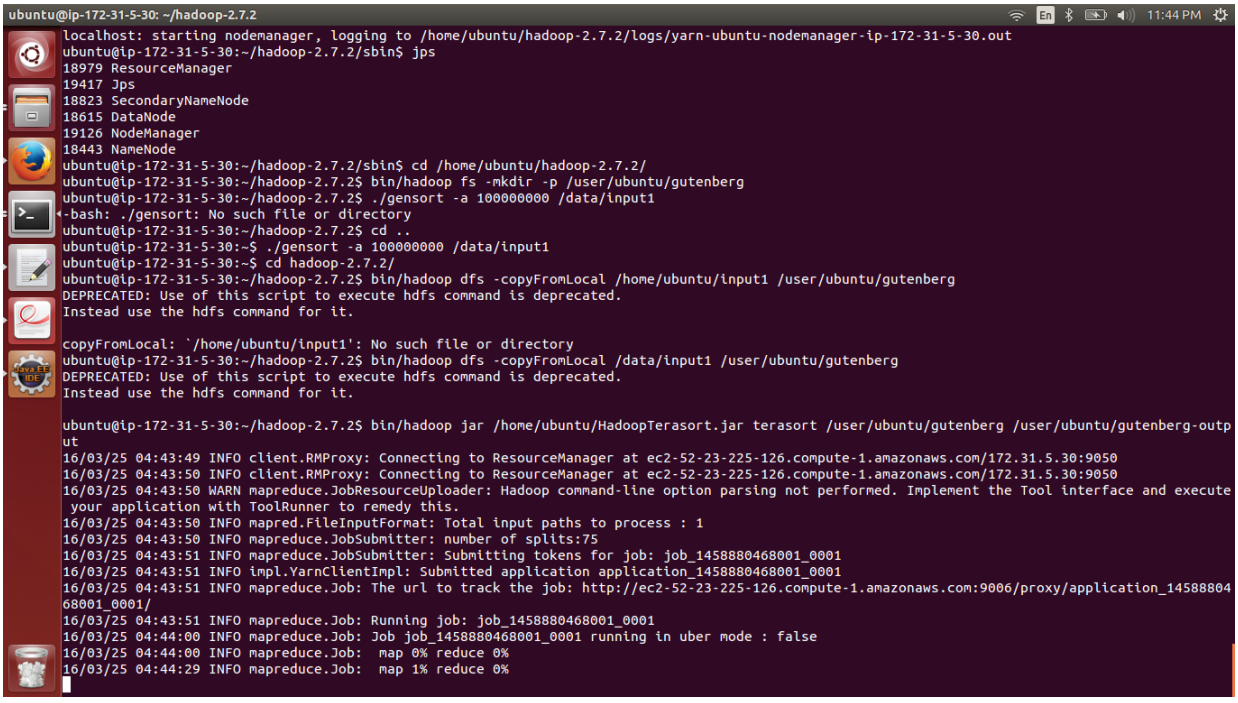
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

6.9 Copy files from local to instance

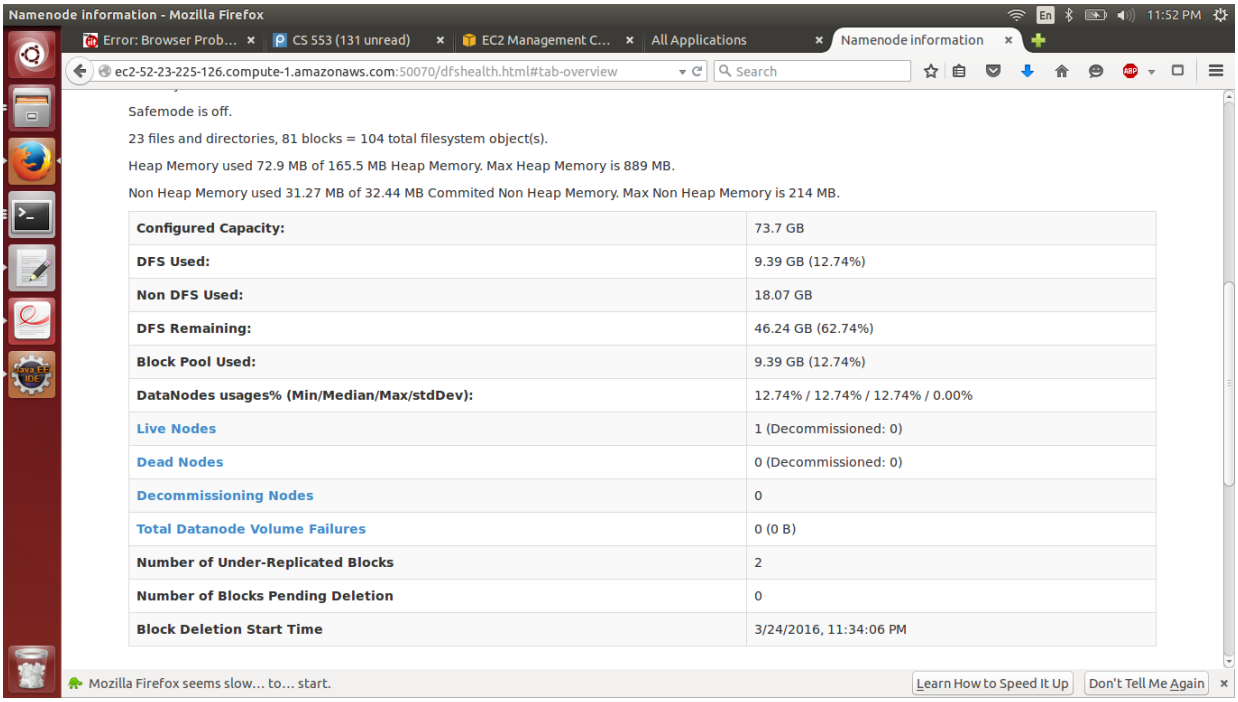


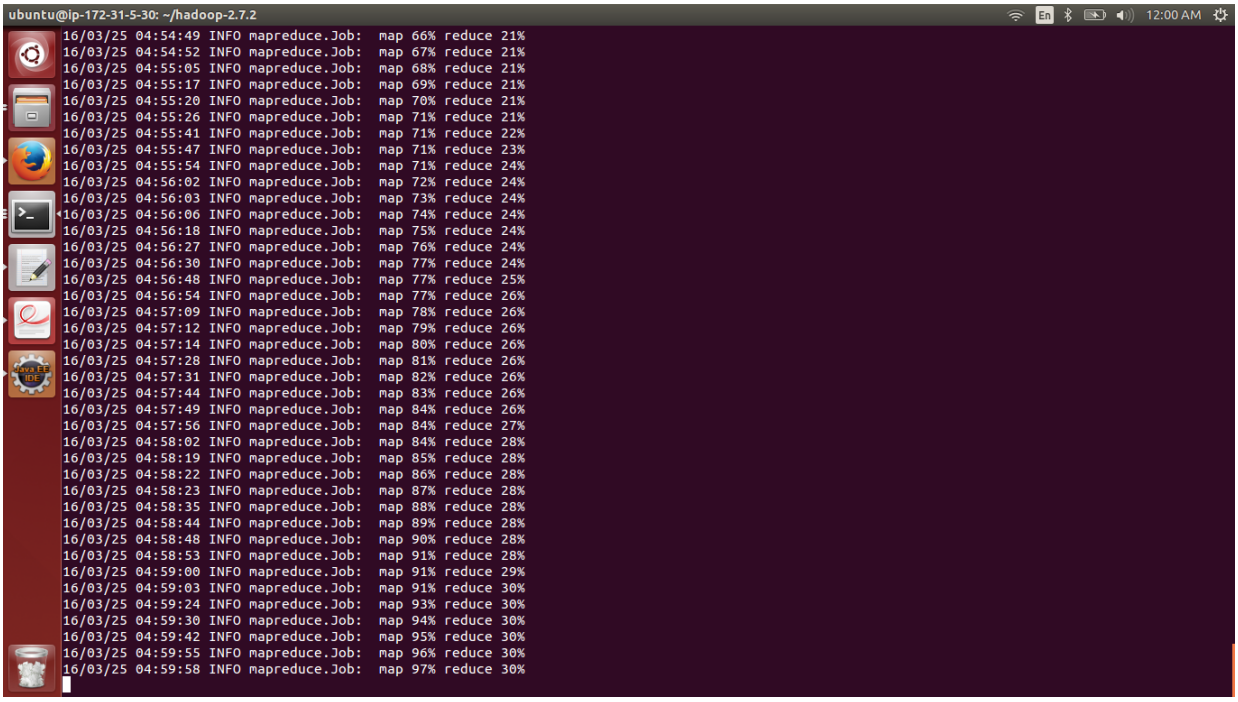
Output



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

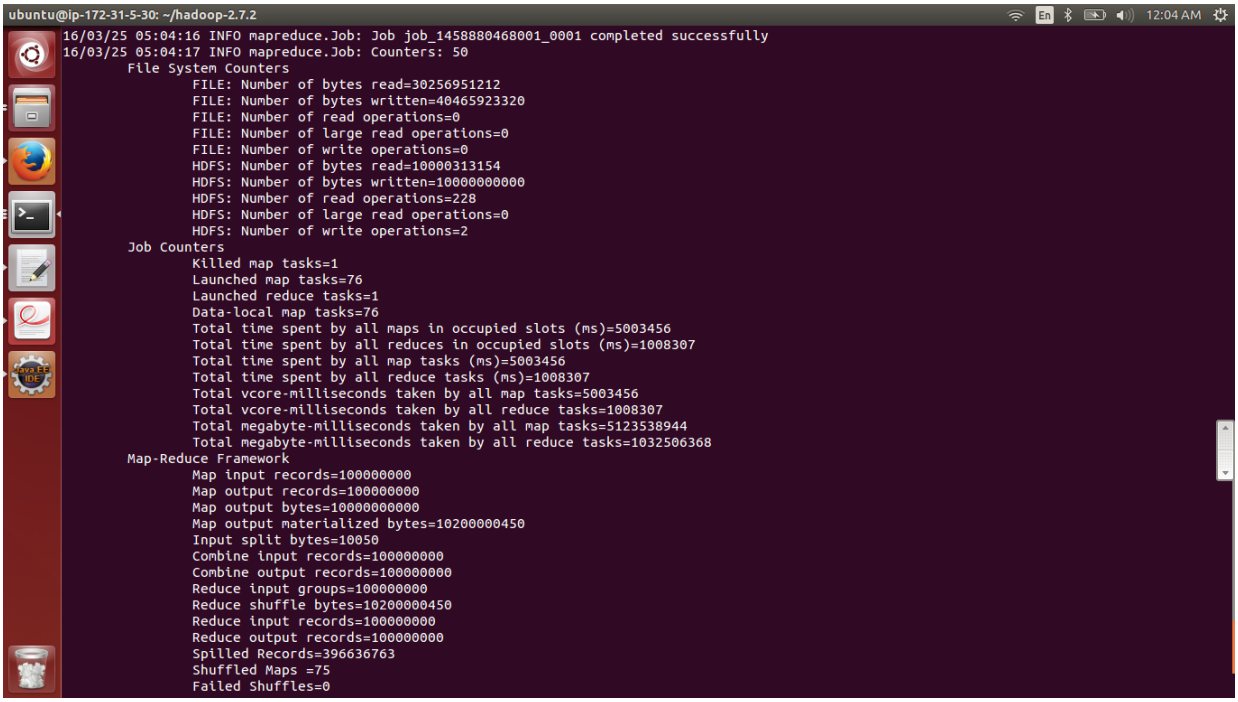
Snehal Sonawane **A20360111**

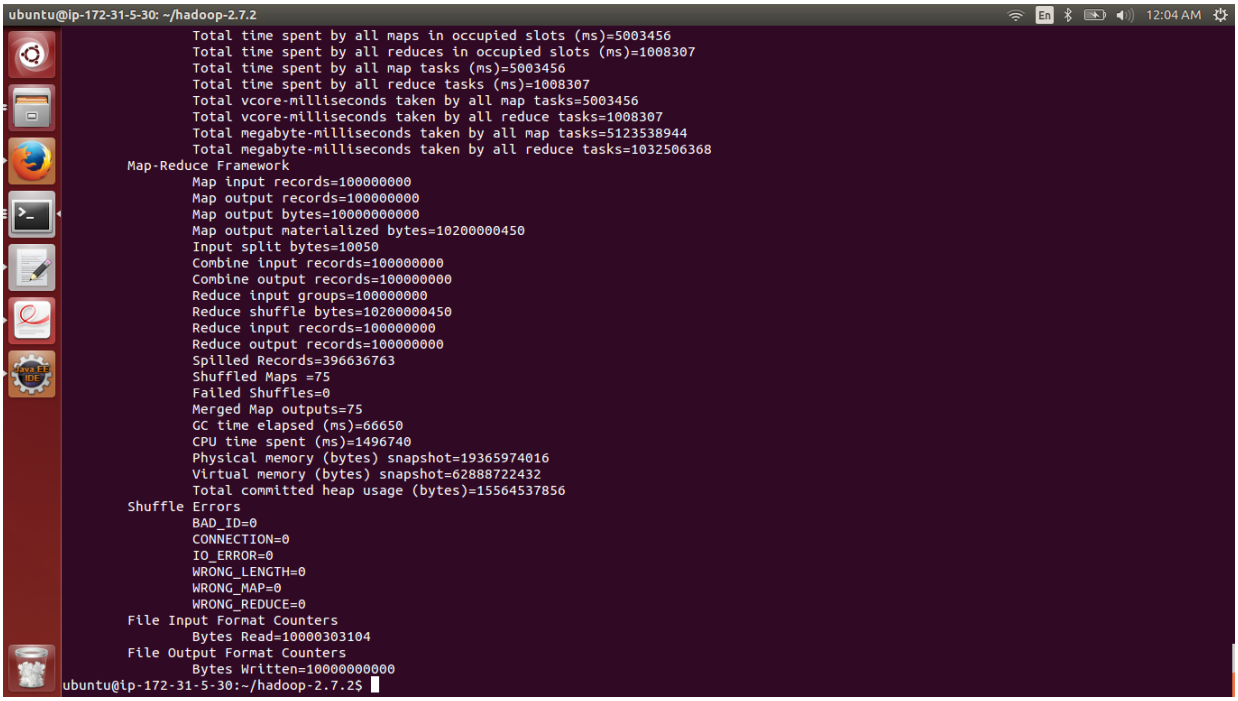




PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

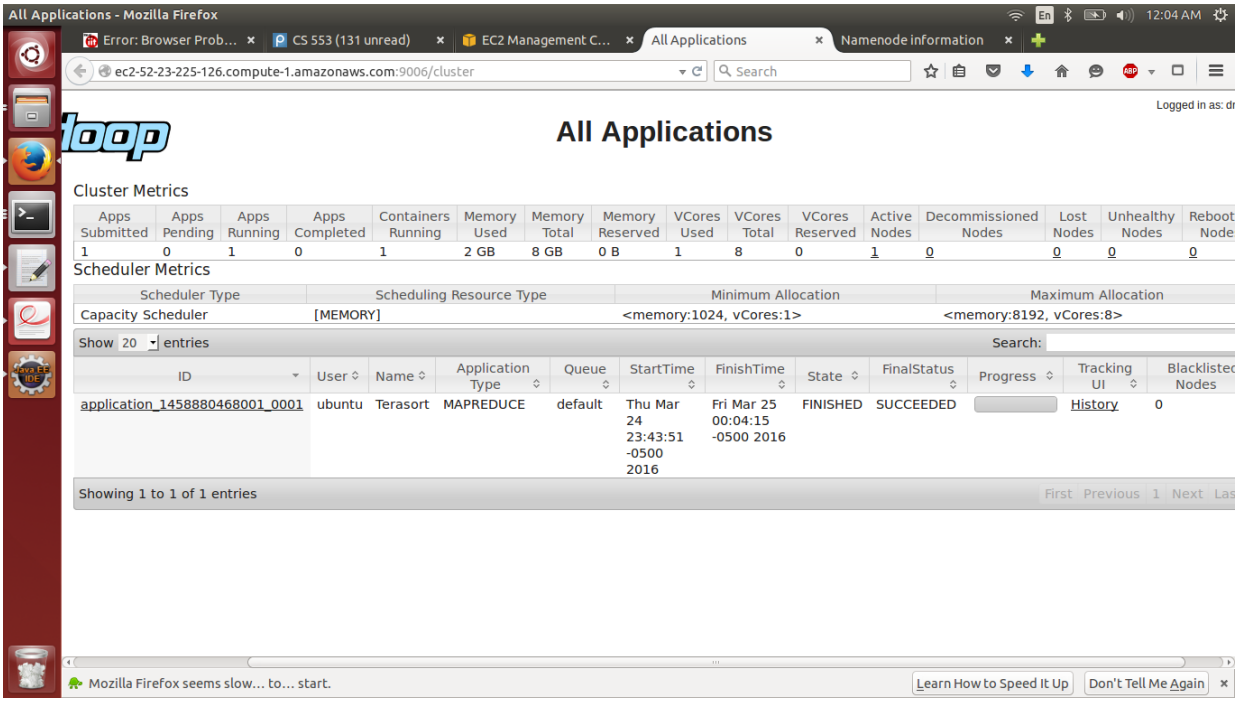
Snehal Sonawane **A20360111**



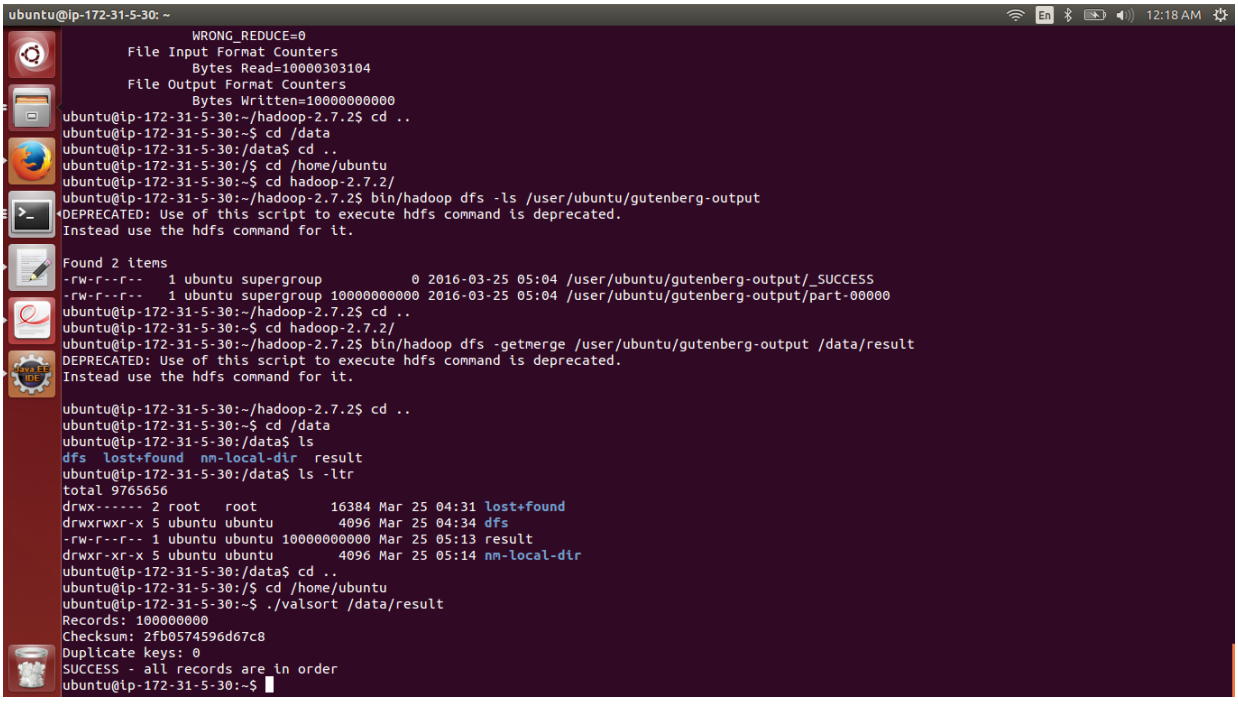


PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



Valsort



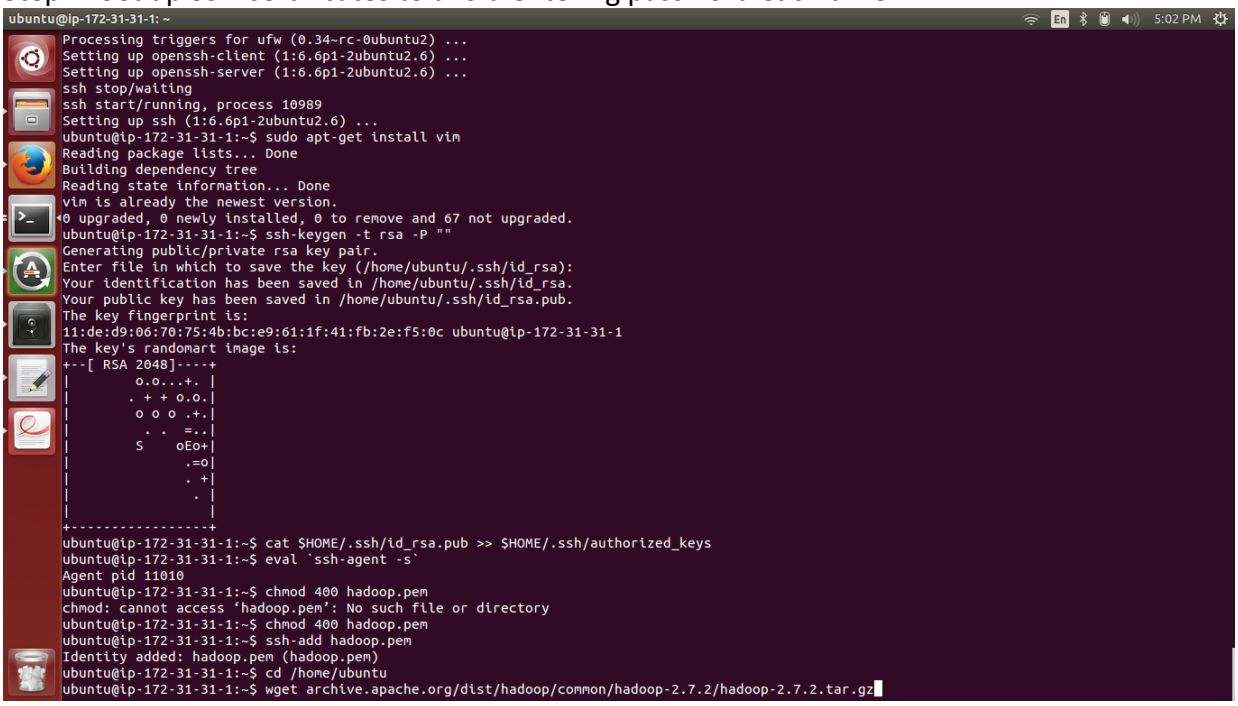
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

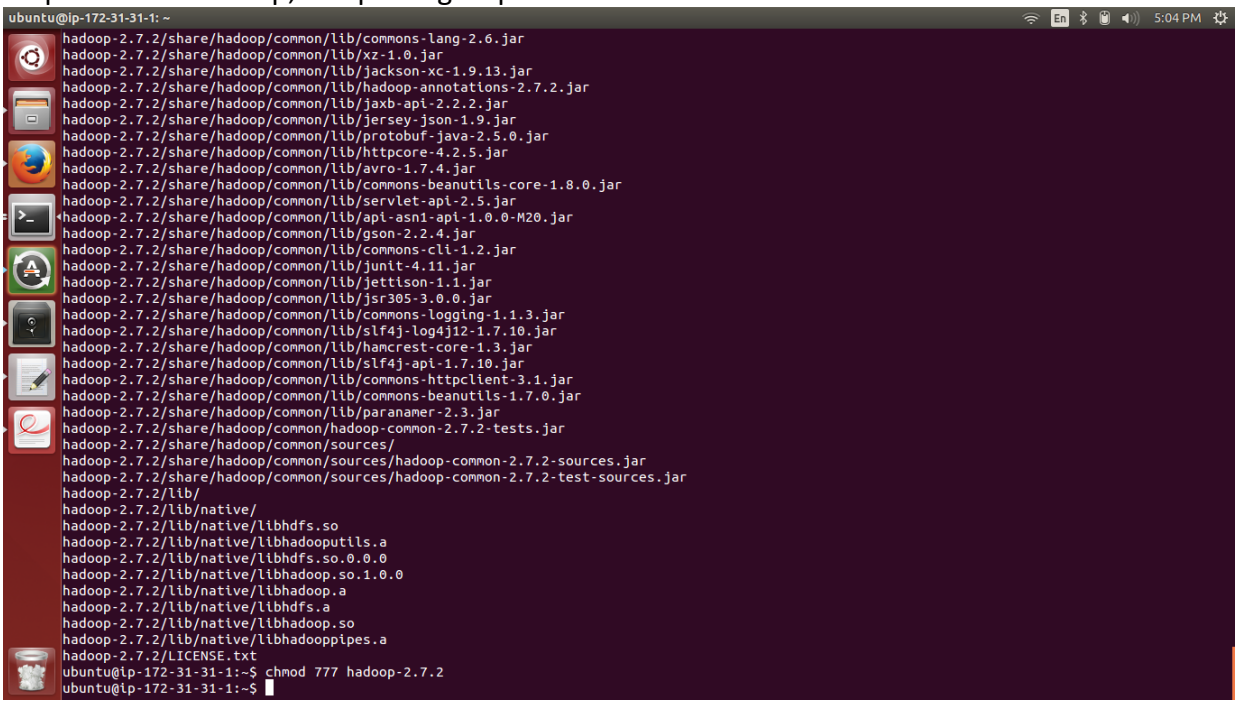
2.Installation of Hadoop 16- nodes

Step 1: sudo apt-get update
sudo apt-get install default-jdk
sudo apt-get install ssh

Step 2: Set up SSH certificates to avoid entering password each time.



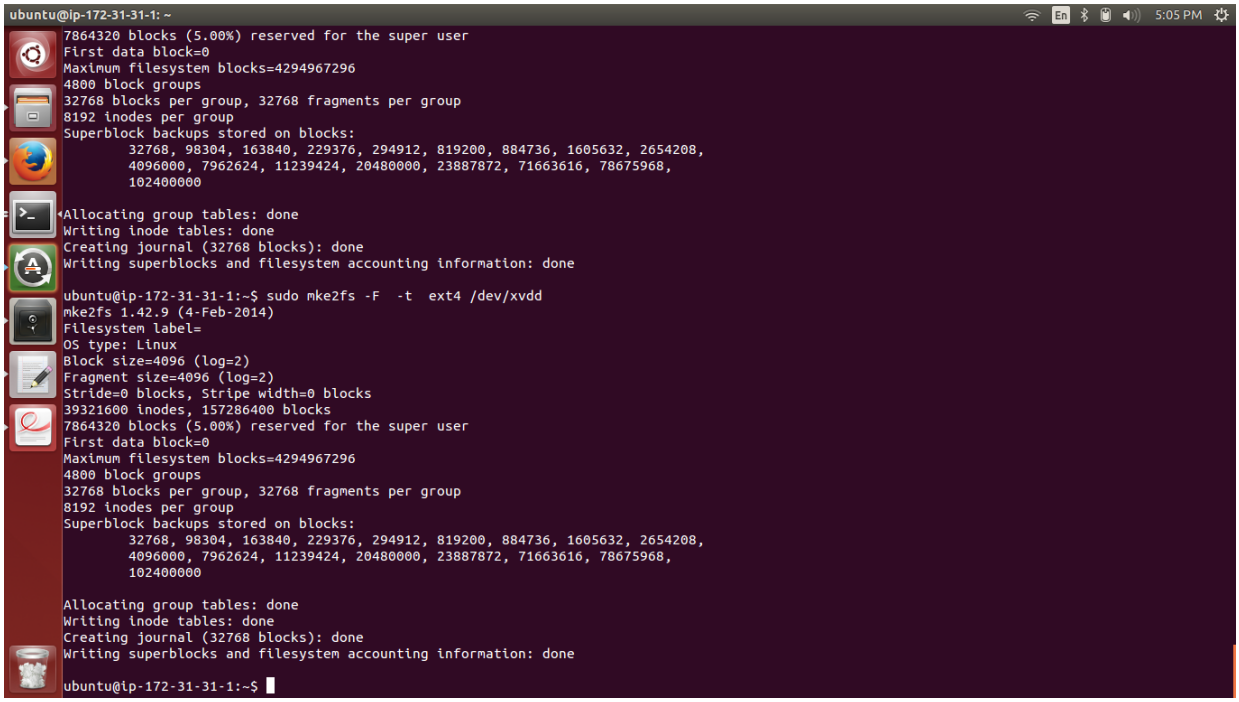
Step 3: Install Hadoop, unzip and give permissions



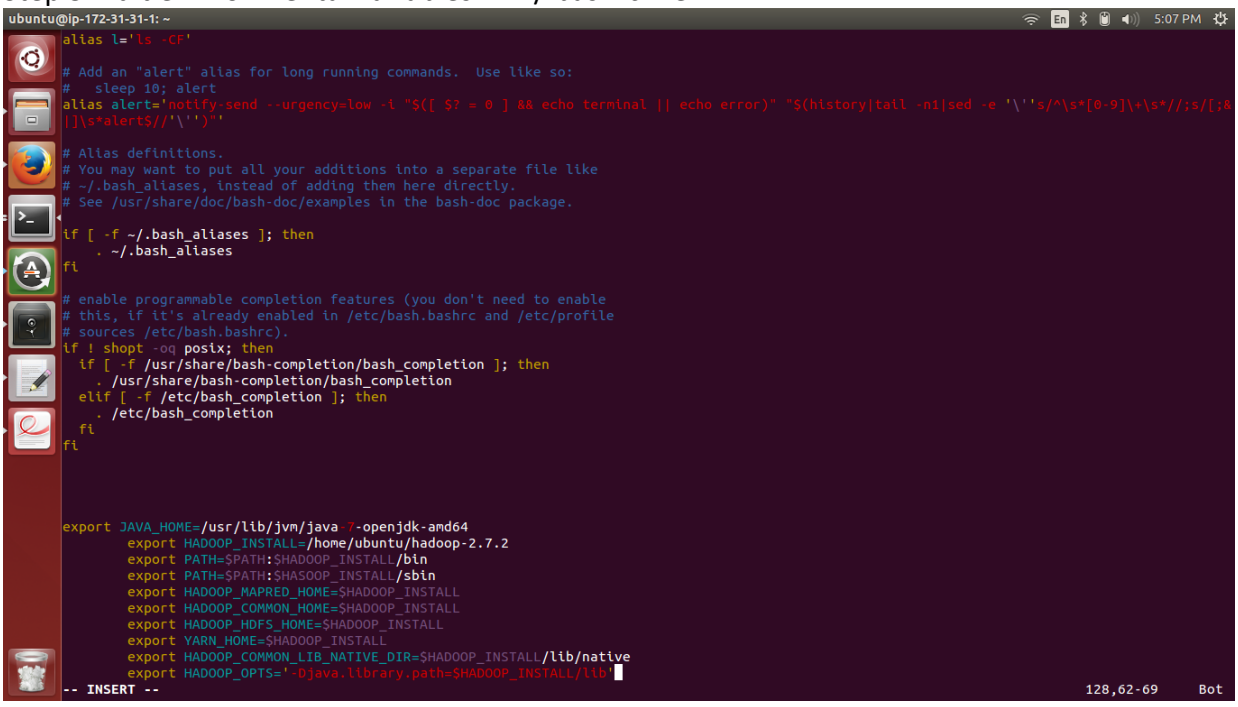
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 4: Mount EBS volume



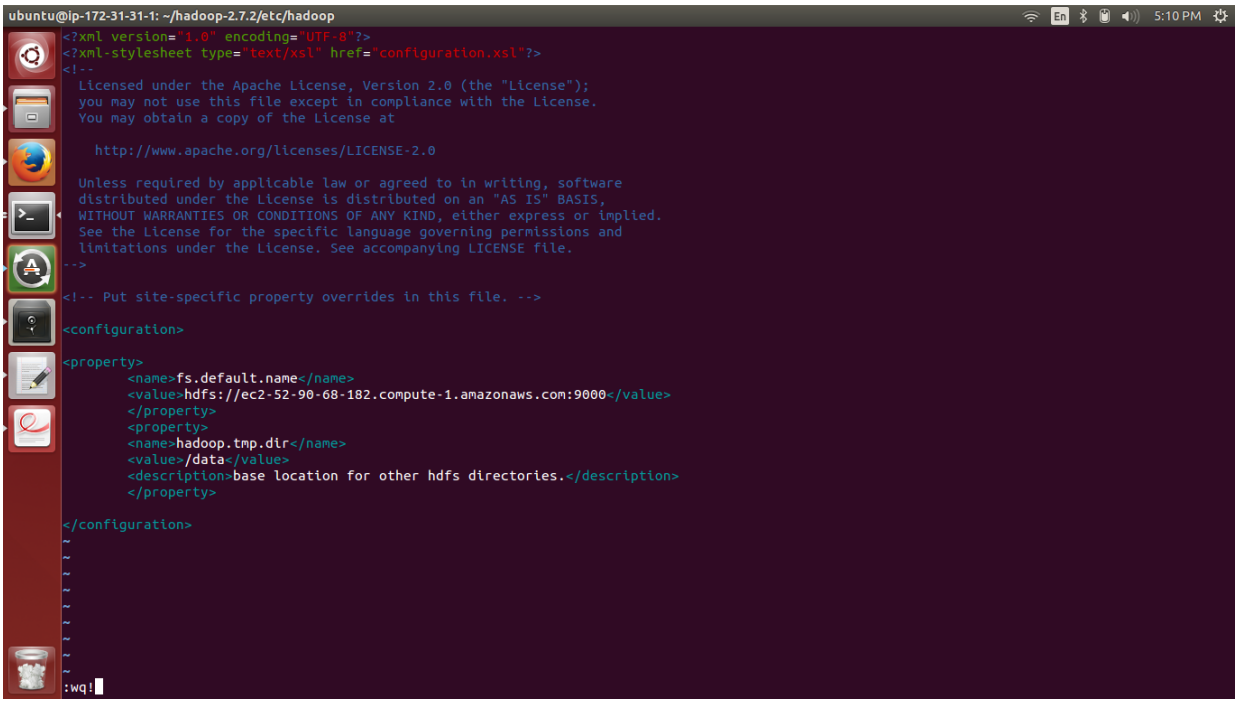
Step 5: Edit environmental variables in ~/.bashrc file :



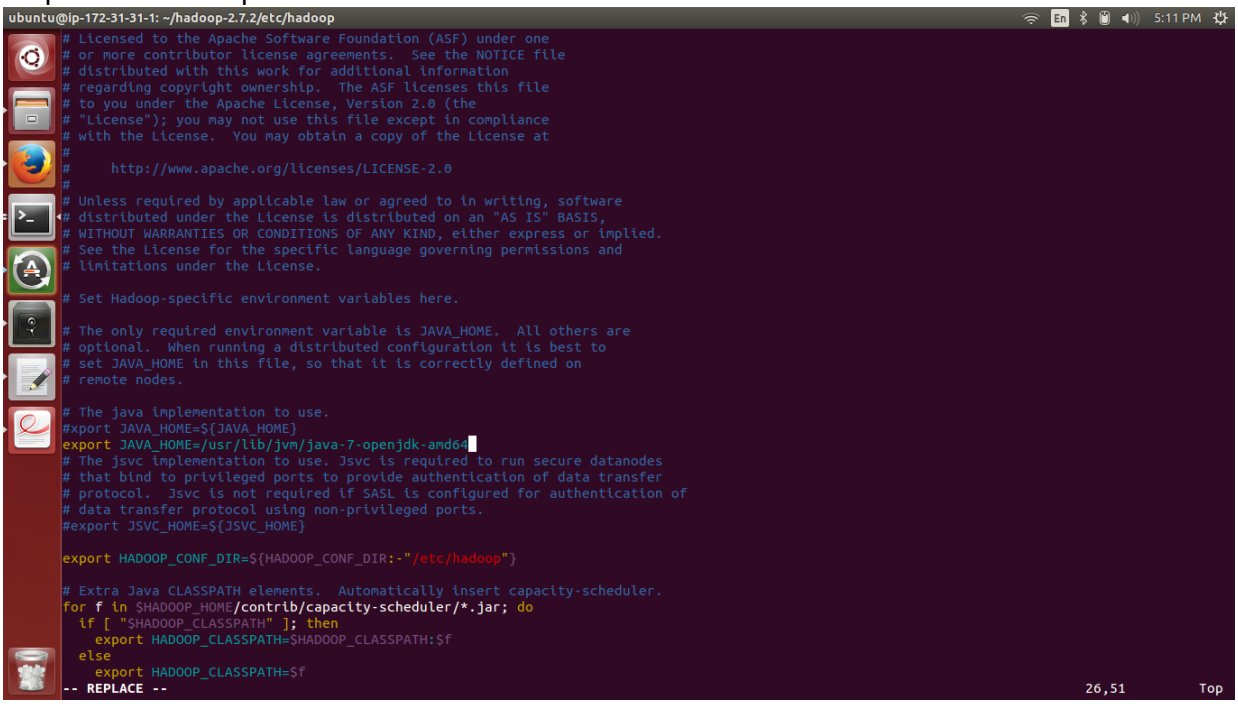
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 6: Edit core-site.xml



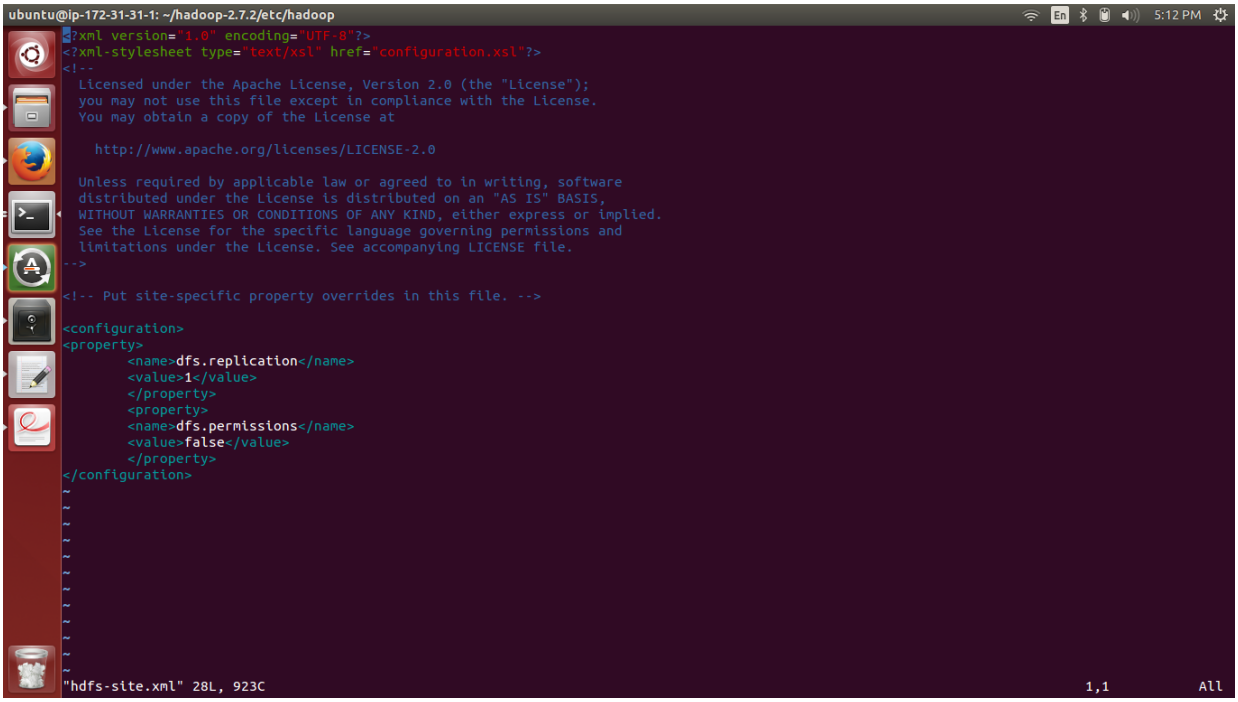
Step 7: Edit hadoop-env.sh



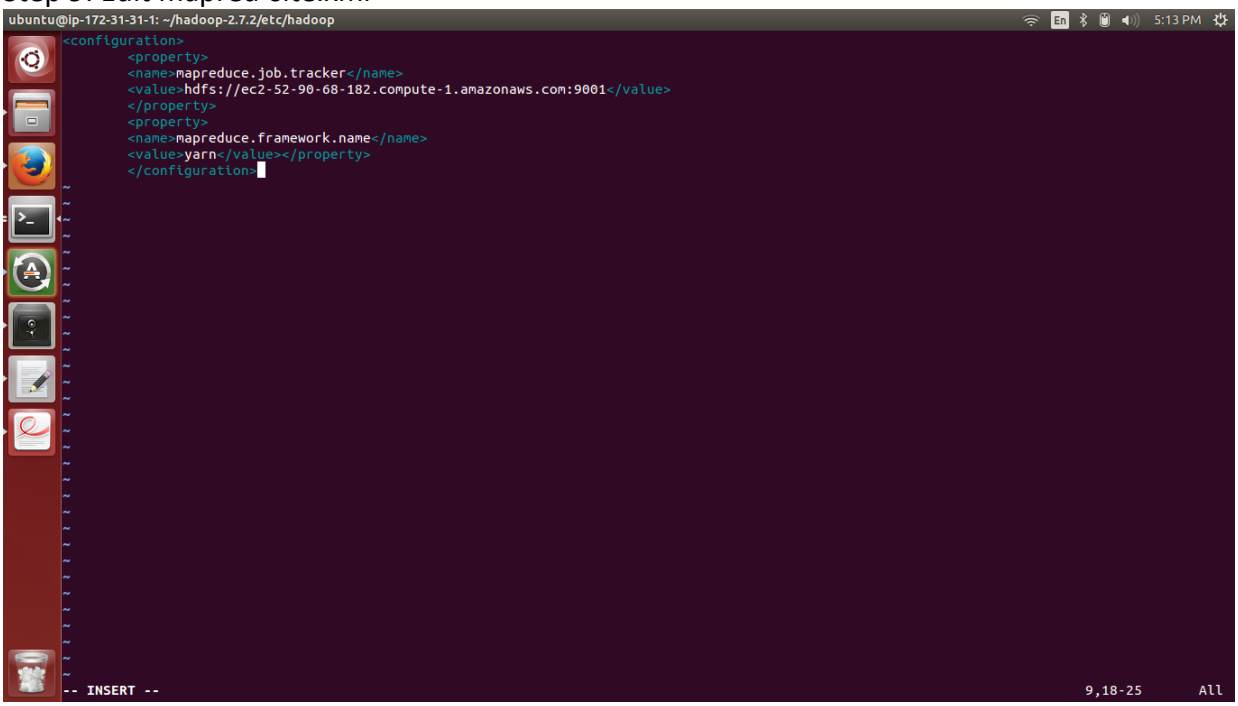
Step 8: Edit hdfs-site.xml

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



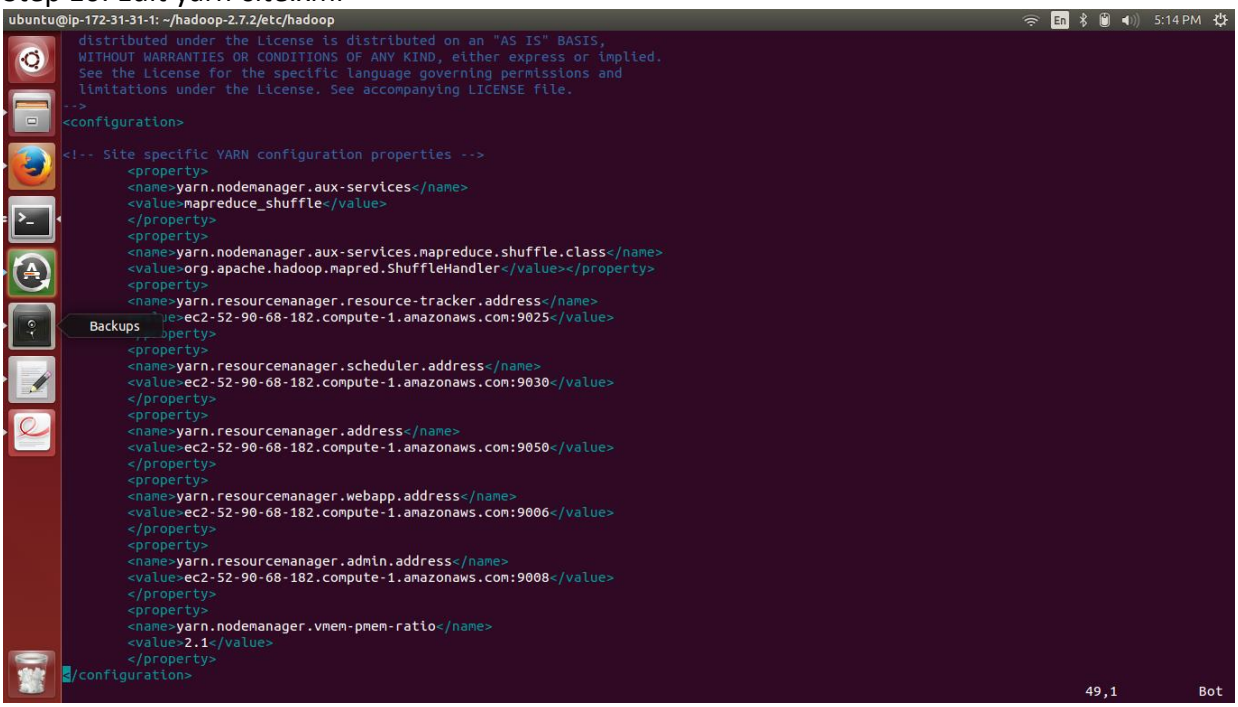
Step 9: Edit mapred-site.xml



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

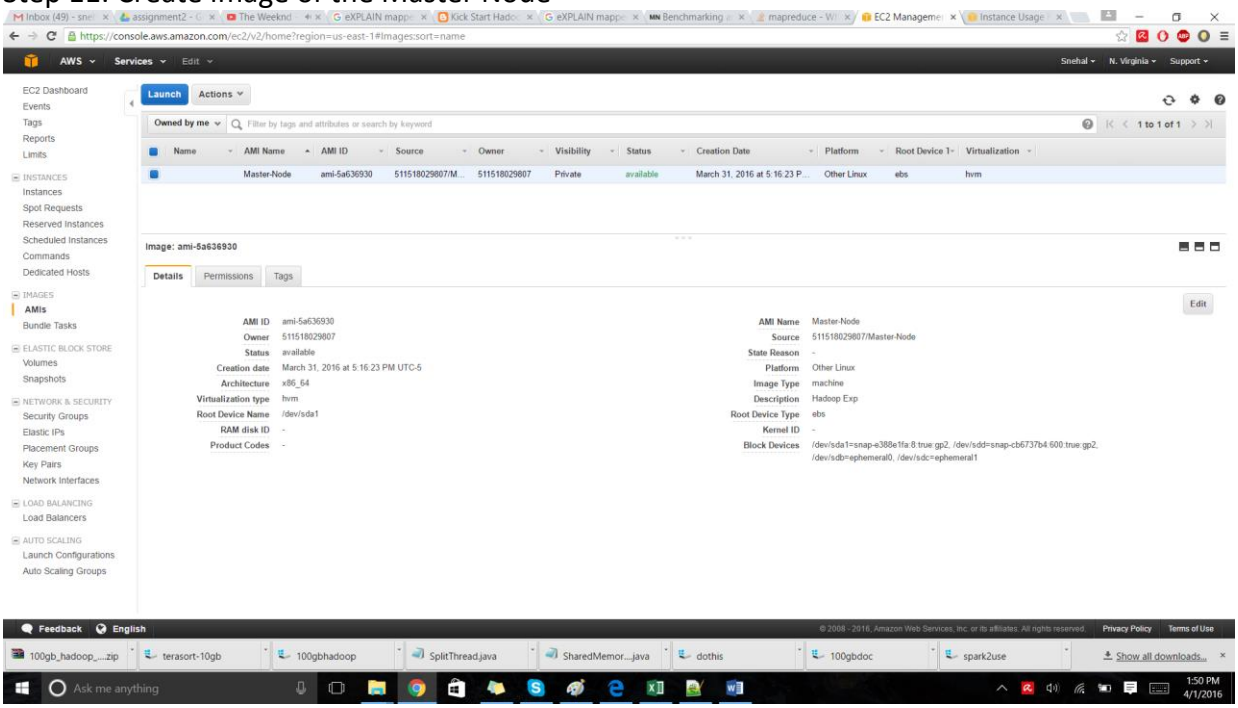
Snehal Sonawane **A20360111**

Step 10: Edit yarn-site.xml




```
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

<!-- Site specific YARN configuration properties -->
        <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
        </property>
        <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value></property>
        <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>ec2-52-90-68-182.compute-1.amazonaws.com:9025</value>
        </property>
        <property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>ec2-52-90-68-182.compute-1.amazonaws.com:9030</value>
        </property>
        <property>
        <name>yarn.resourcemanager.address</name>
        <value>ec2-52-90-68-182.compute-1.amazonaws.com:9050</value>
        </property>
        <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>ec2-52-90-68-182.compute-1.amazonaws.com:9006</value>
        </property>
        <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>ec2-52-90-68-182.compute-1.amazonaws.com:9008</value>
        </property>
        <property>
        <name>yarn.nodemanager.vmem-pmem-ratio</name>
        <value>2.1</value>
        </property>
</configuration>
```

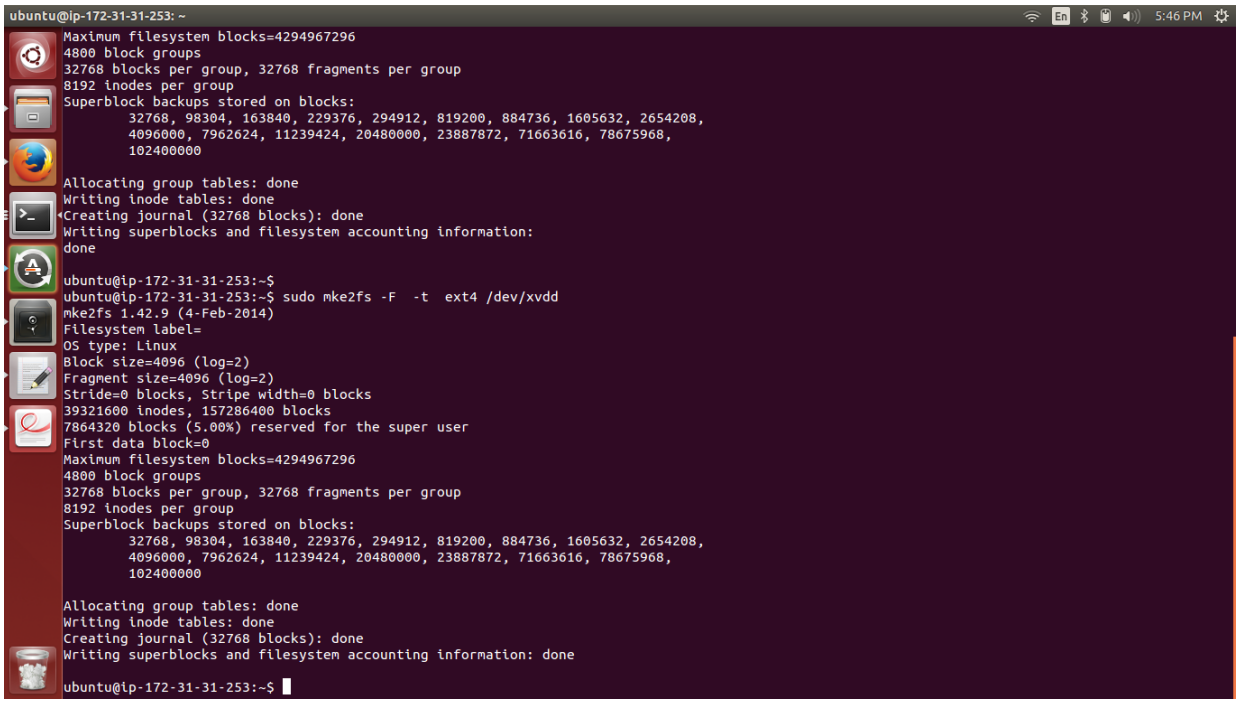
Step 11: Create image of the Master Node



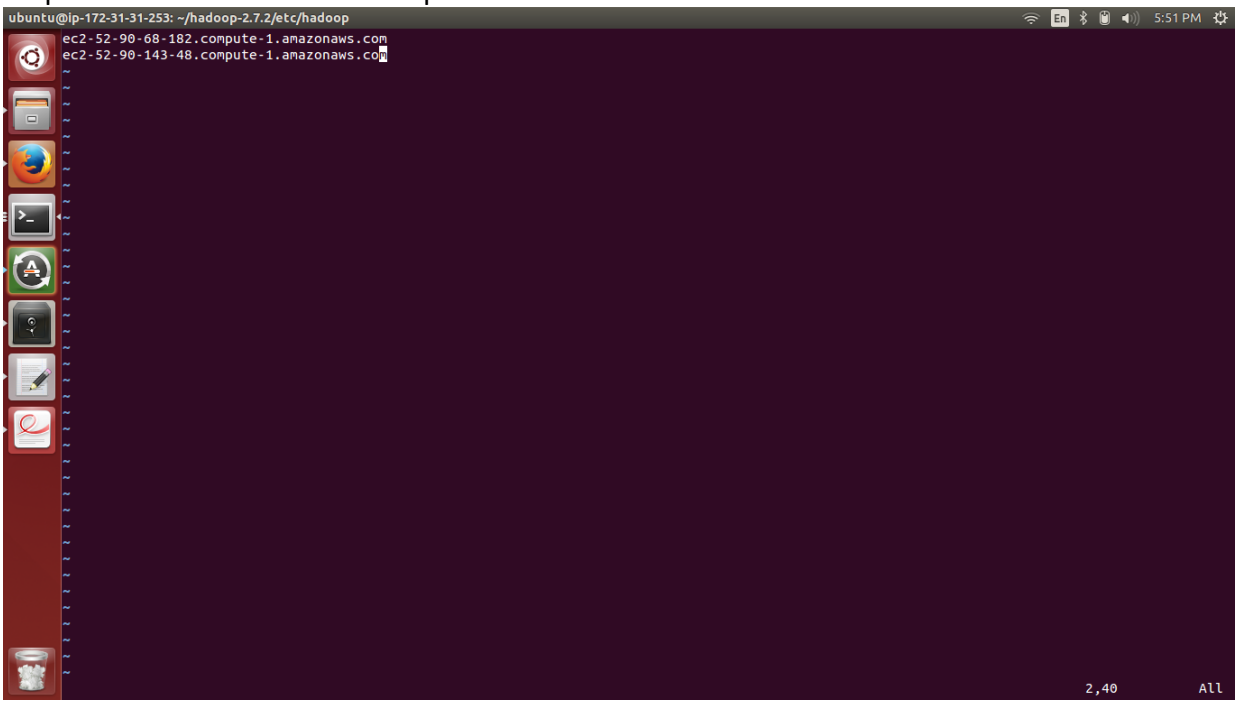
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 12: Connect to each slave and mount the EBS volume.



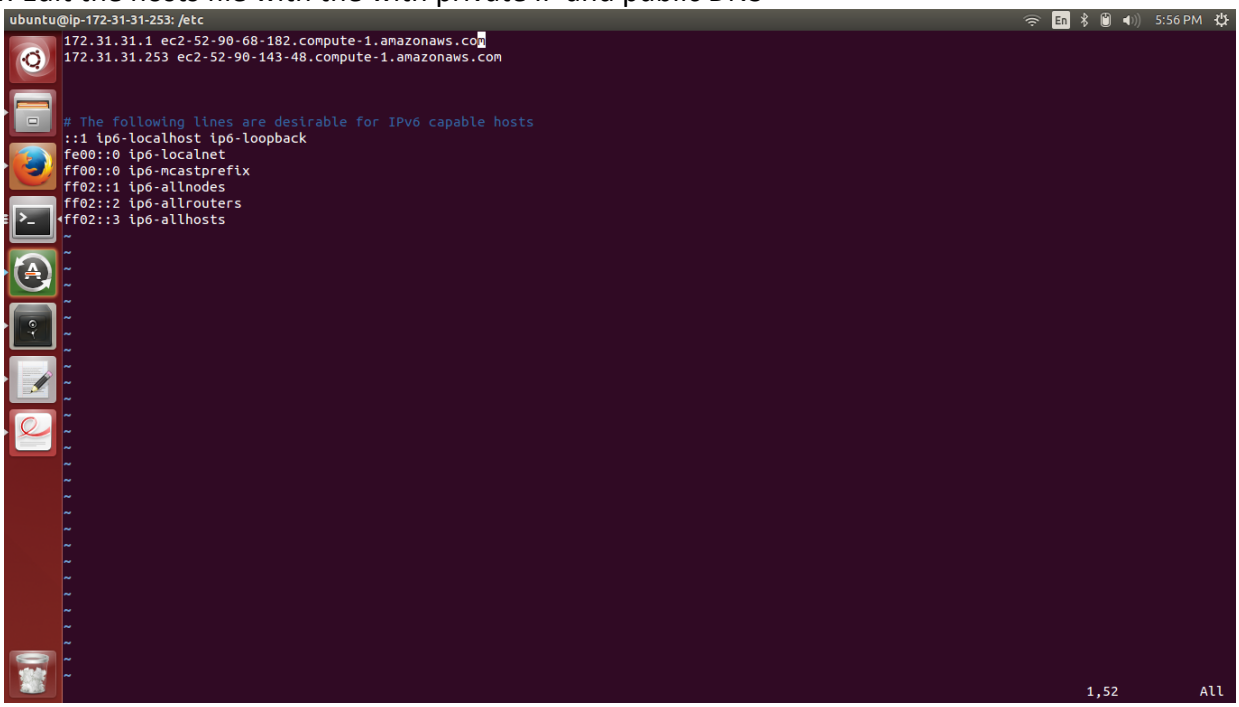
Step 13: Edit the slaves file with public DNS of master and slave



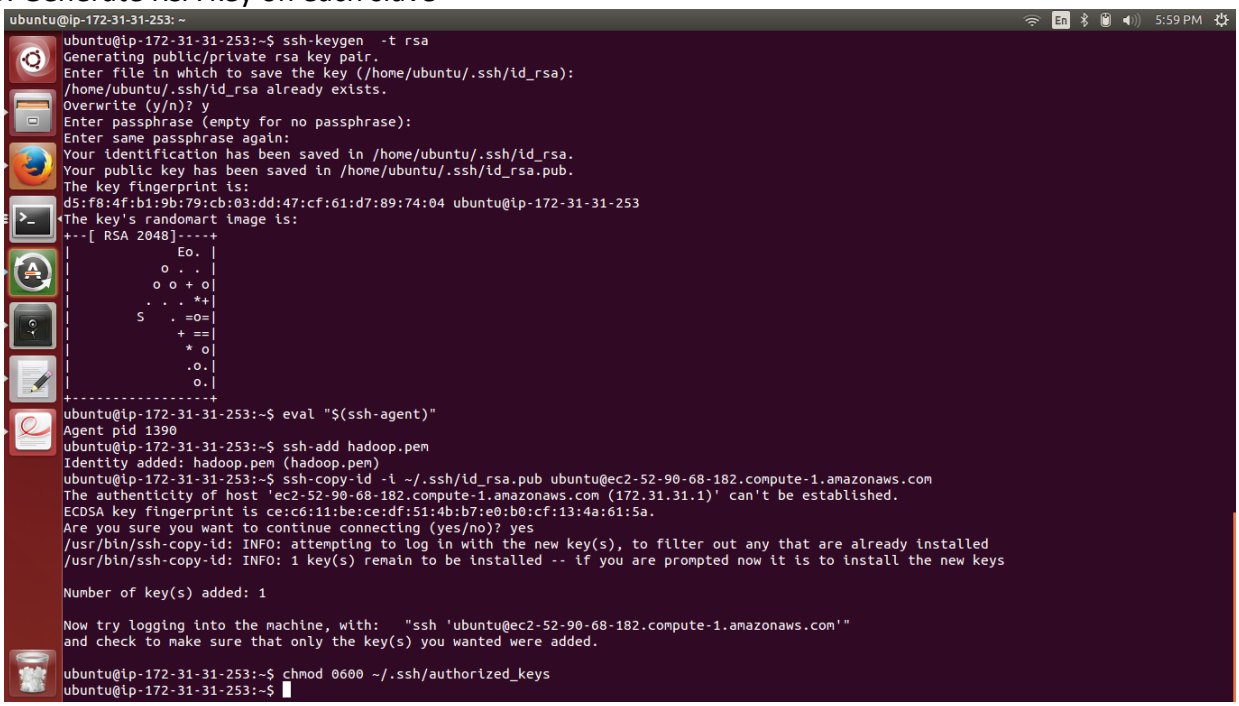
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 14: Edit the hosts file with the with private IP and public DNS



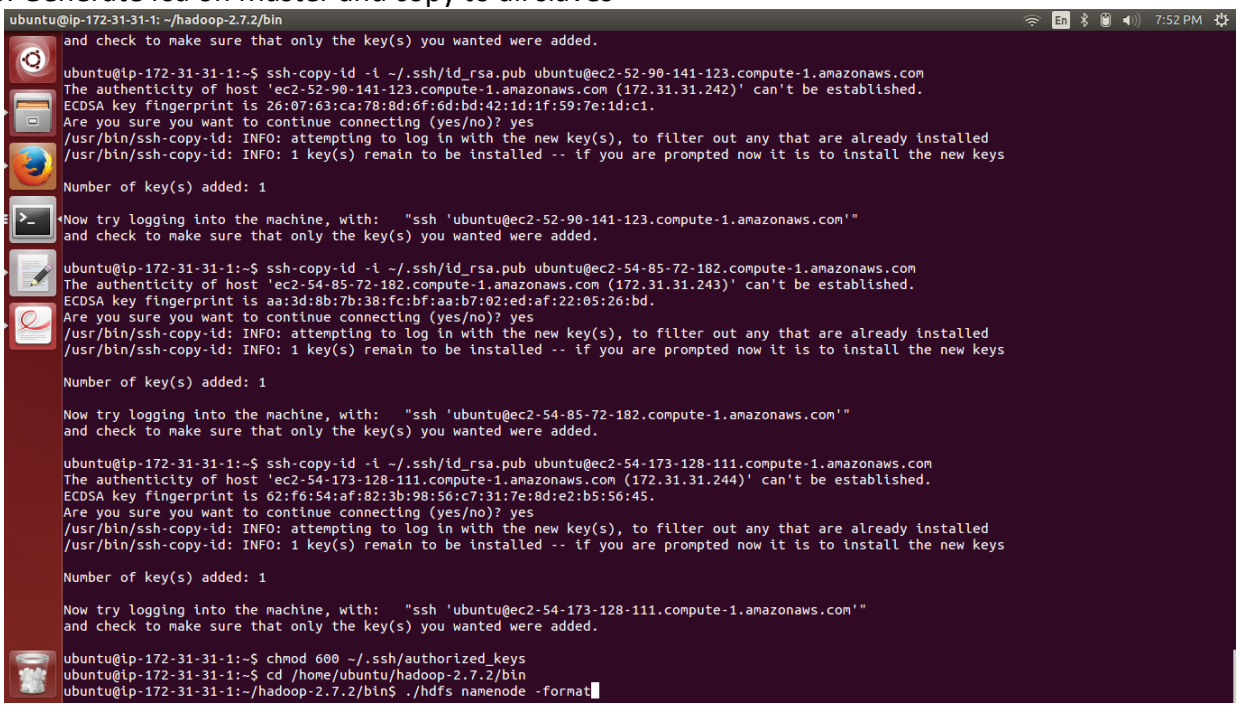
Step 15: Generate RSA key on each slave



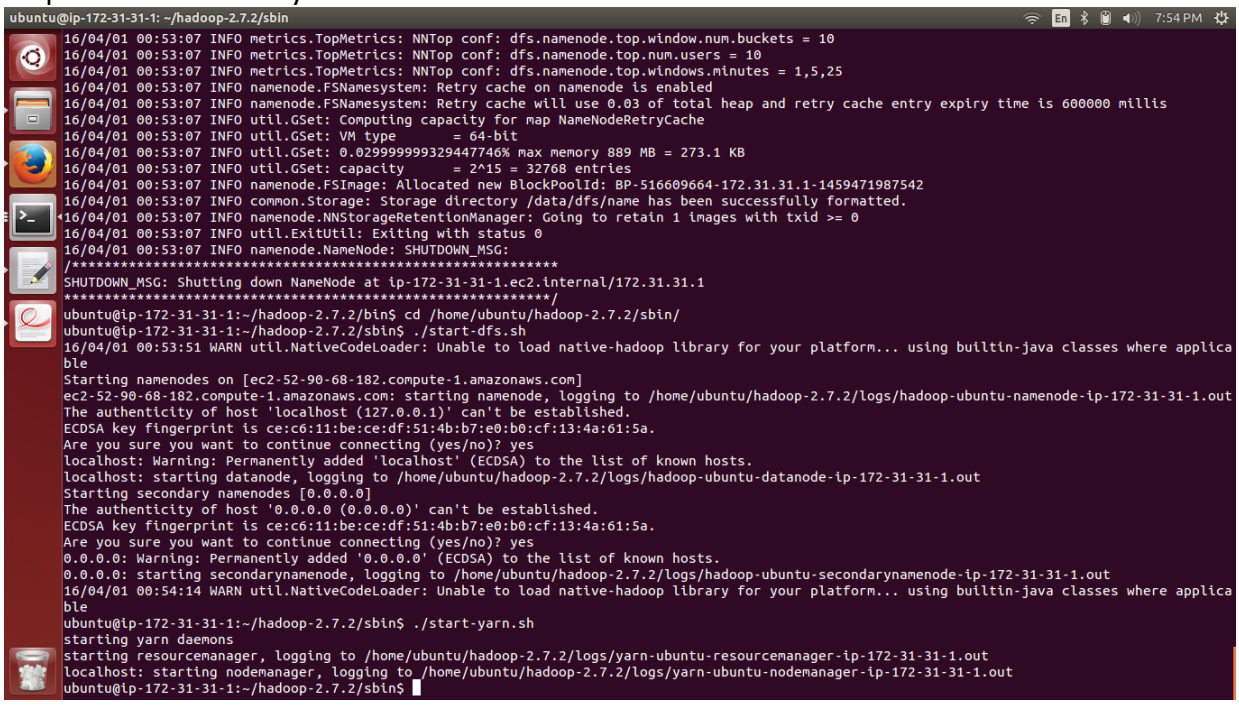
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 16: Generate rsa on Master and copy to all slaves



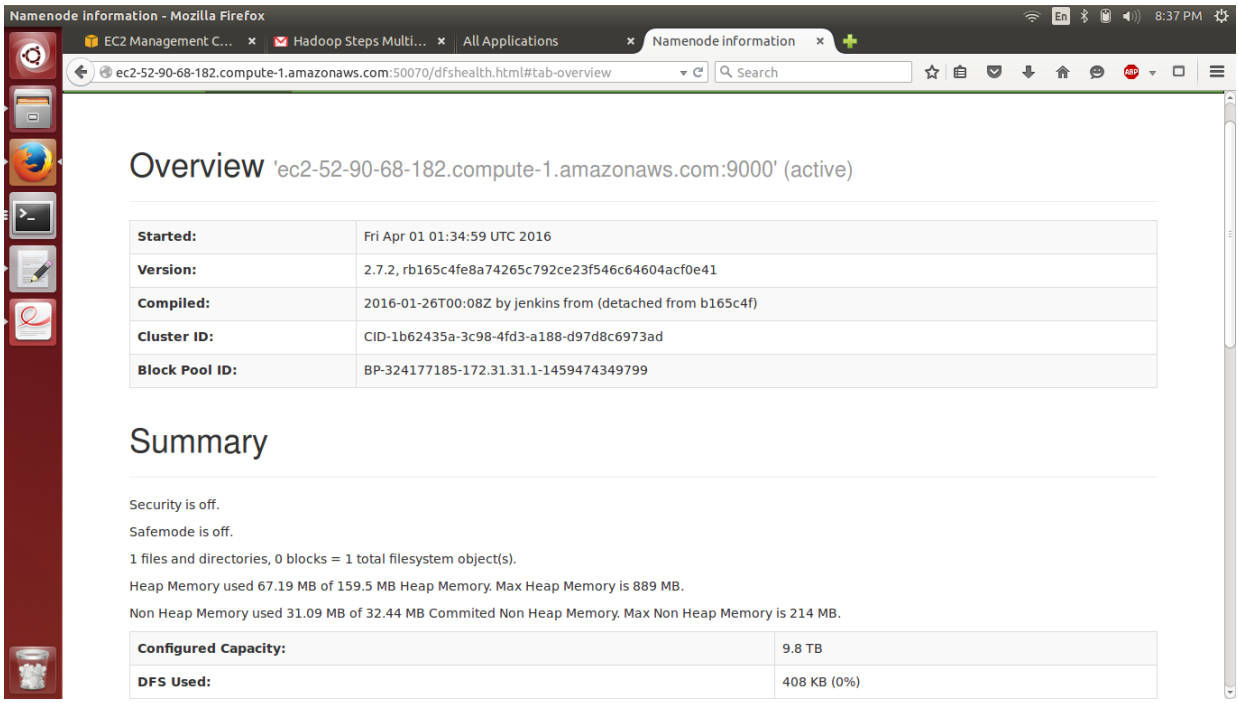
Step 17: Start dfs and yarn



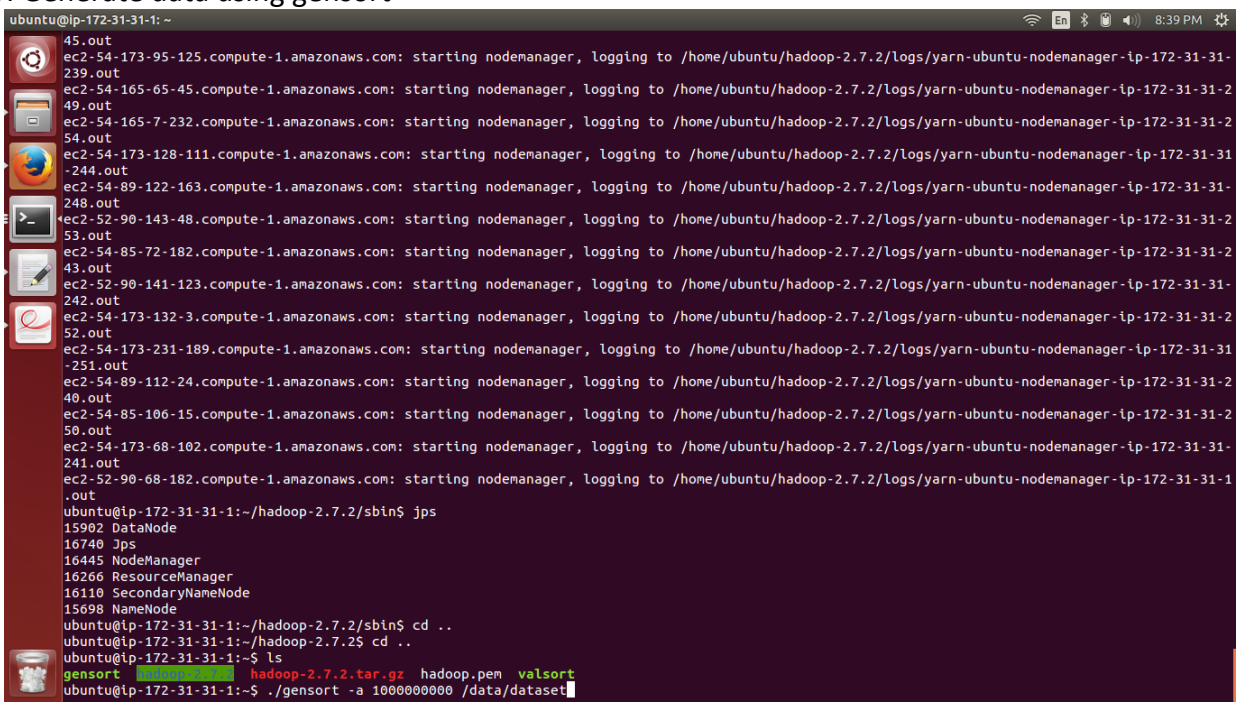
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 18: Check if the mount was successful



Step 19: Generate data using gensort



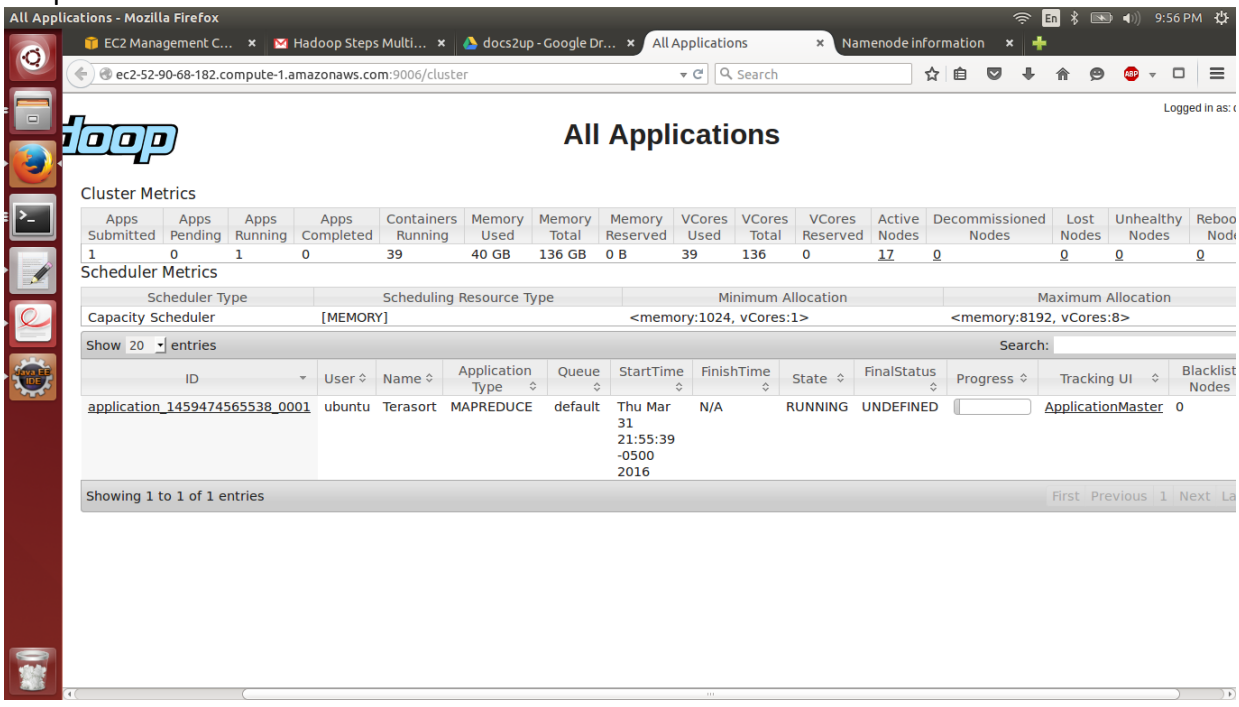
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

Step 20: Run the HadoopTerasort jar

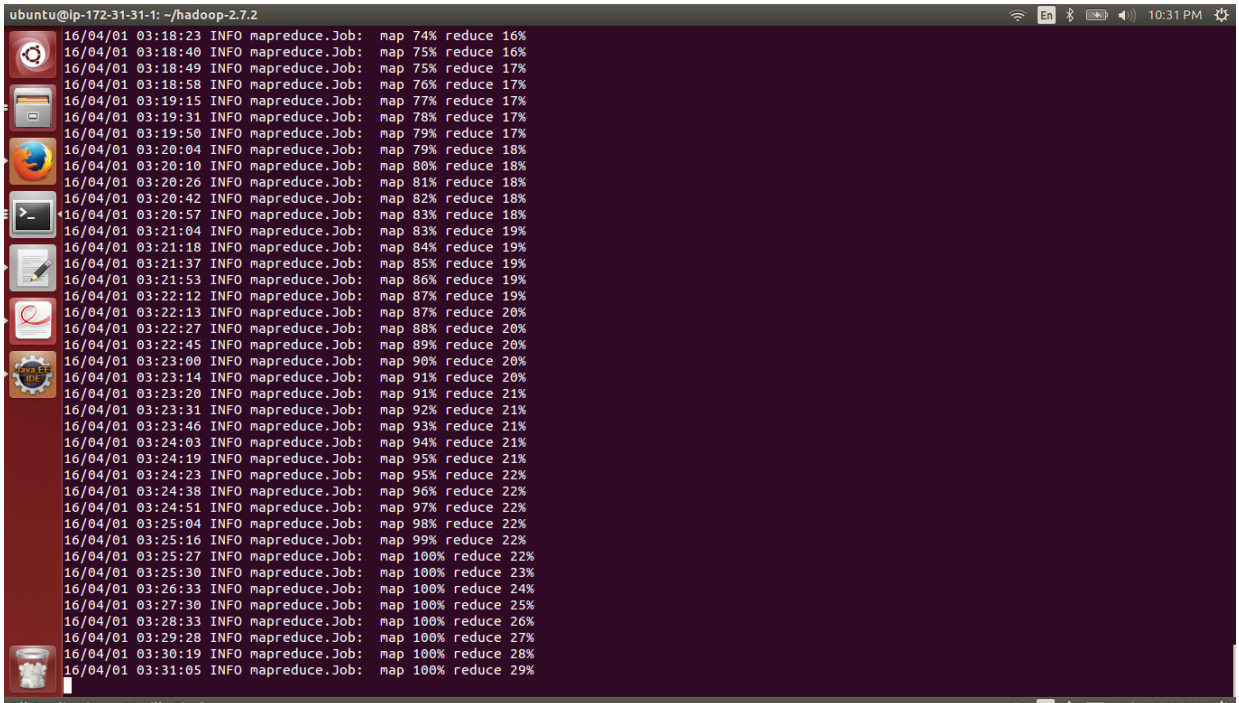
```
ubuntu@ip-172-31-31-1: ~/hadoop-2.7.2
drwxrwxr-x 5 ubuntu ubuntu         4096 Apr  1 01:35 dfs
-rwxrwxr-x 1 ubuntu ubuntu 100000000000 Apr  1 02:06 dataset
drwxr-xr-x 5 ubuntu ubuntu         4096 Apr  1 02:10 nm-local-dir
ubuntu@ip-172-31-31-1:/data$ cd /home/ubuntu/
ubuntu@ip-172-31-31-1:~$ cd hadoop-2.7.2/
ubuntu@ip-172-31-31-1:~/hadoop-2.7.2$ bin/hadoop fs -mkdir -p /user/ubuntu/gutenberg
16/04/01 02:14:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
ubuntu@ip-172-31-31-1:~/hadoop-2.7.2$ bin/hadoop dfs -copyFromLocal /data/dataset /user/ubuntu/gutenberg
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/04/01 02:15:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
ubuntu@ip-172-31-31-1:~/hadoop-2.7.2$
ubuntu@ip-172-31-31-1:~/hadoop-2.7.2$ bin/hadoop dfs -ls /user/ubuntu/gutenberg
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/04/01 02:47:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
Found 1 items
-rw-r--r--   1 ubuntu supergroup 100000000000 2016-04-01 02:42 /user/ubuntu/gutenberg/dataset
ubuntu@ip-172-31-31-1:~/hadoop-2.7.2$ bin/hadoop jar /home/ubuntu/HadoopTerasort.jar
16/04/01 02:55:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
16/04/01 02:55:36 INFO client.RMProxy: Connecting to ResourceManager at ec2-52-90-68-182.compute-1.amazonaws.com/172.31.31.1:9050
16/04/01 02:55:36 INFO client.RMProxy: Connecting to ResourceManager at ec2-52-90-68-182.compute-1.amazonaws.com/172.31.31.1:9050
16/04/01 02:55:37 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute
 your application with ToolRunner to remedy this.
16/04/01 02:55:37 INFO mapred.FileInputFormat: Total input paths to process : 1
16/04/01 02:55:37 INFO net.NetworkTopology: Adding a new node: /default-rack/172.31.31.1:50010
16/04/01 02:55:38 INFO mapreduce.JobSubmitter: number of splits:745
16/04/01 02:55:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459474565538_0001
16/04/01 02:55:39 INFO impl.YarnClientImpl: Submitted application application_1459474565538_0001
16/04/01 02:55:39 INFO mapreduce.Job: The url to track the job: http://ec2-52-90-68-182.compute-1.amazonaws.com:9006/proxy/application_145947456
5538_0001/
16/04/01 02:55:39 INFO mapreduce.Job: Running job: job_1459474565538_0001
16/04/01 02:55:47 INFO mapreduce.Job: Job job_1459474565538_0001 running in uber mode : false
16/04/01 02:55:47 INFO mapreduce.Job:  map 0% reduce 0%
```

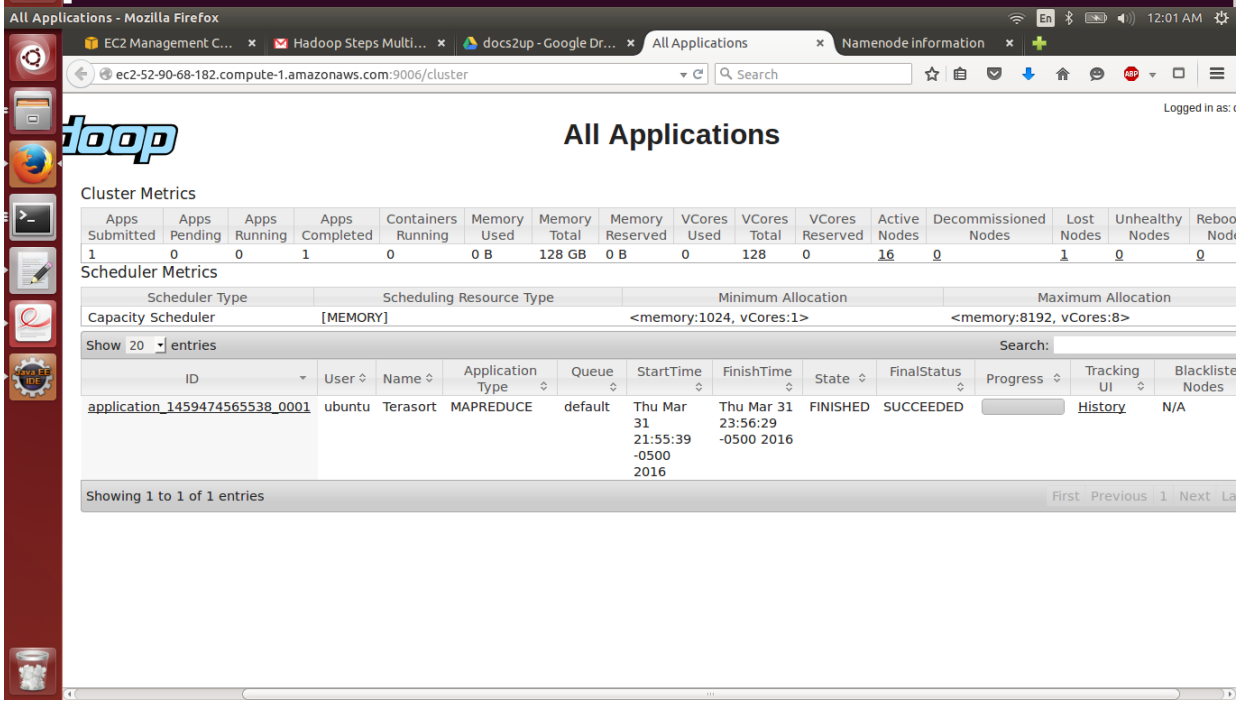
Output



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

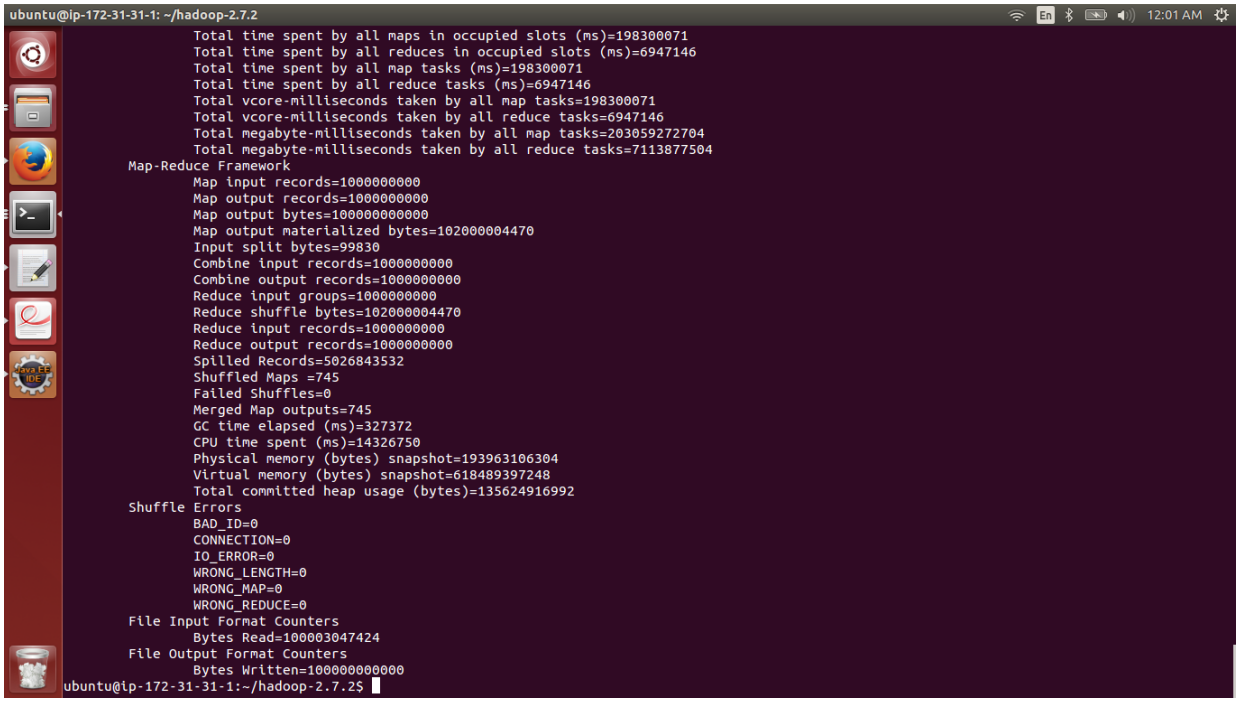
Snehal Sonawane **A20360111**



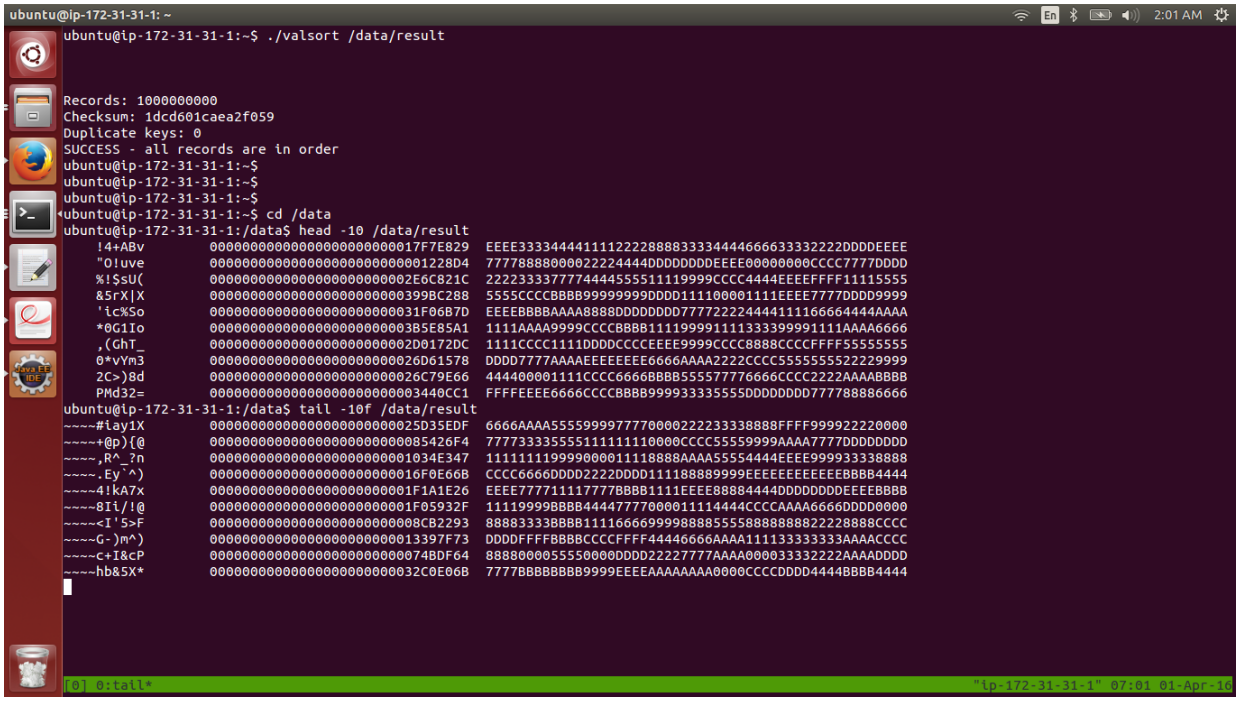


PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



Valsort

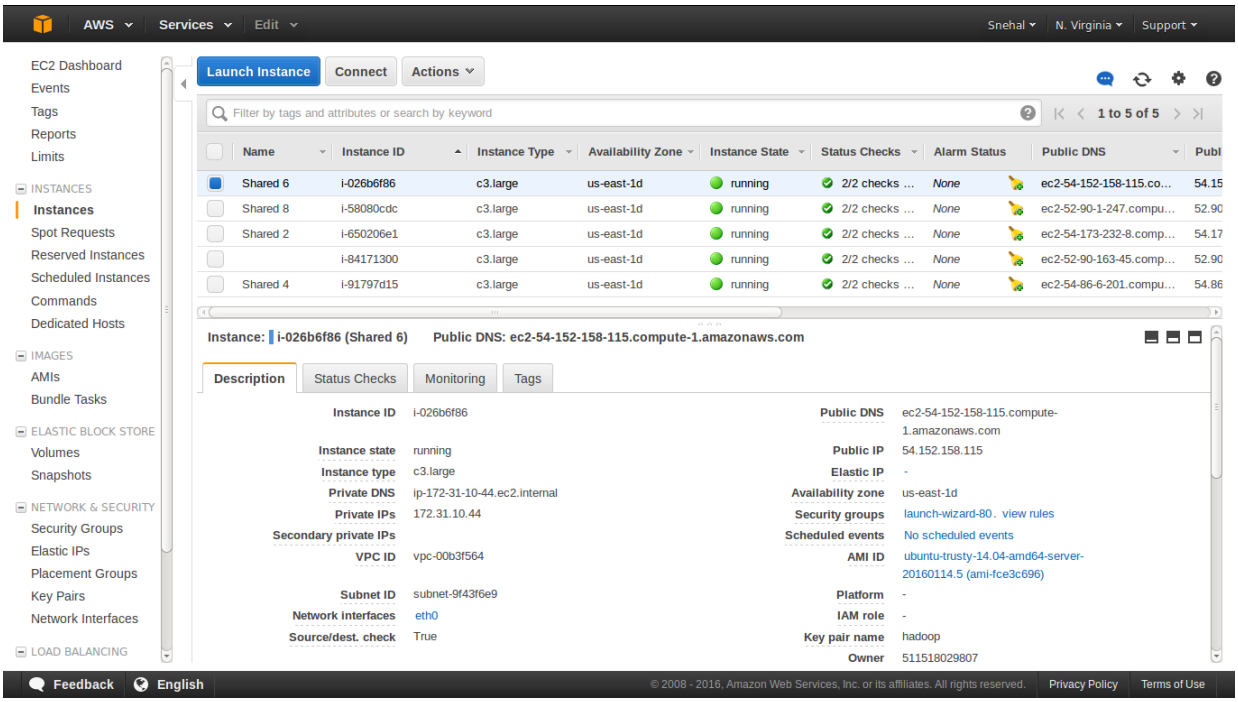


PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

SHARED MEMORY SORT

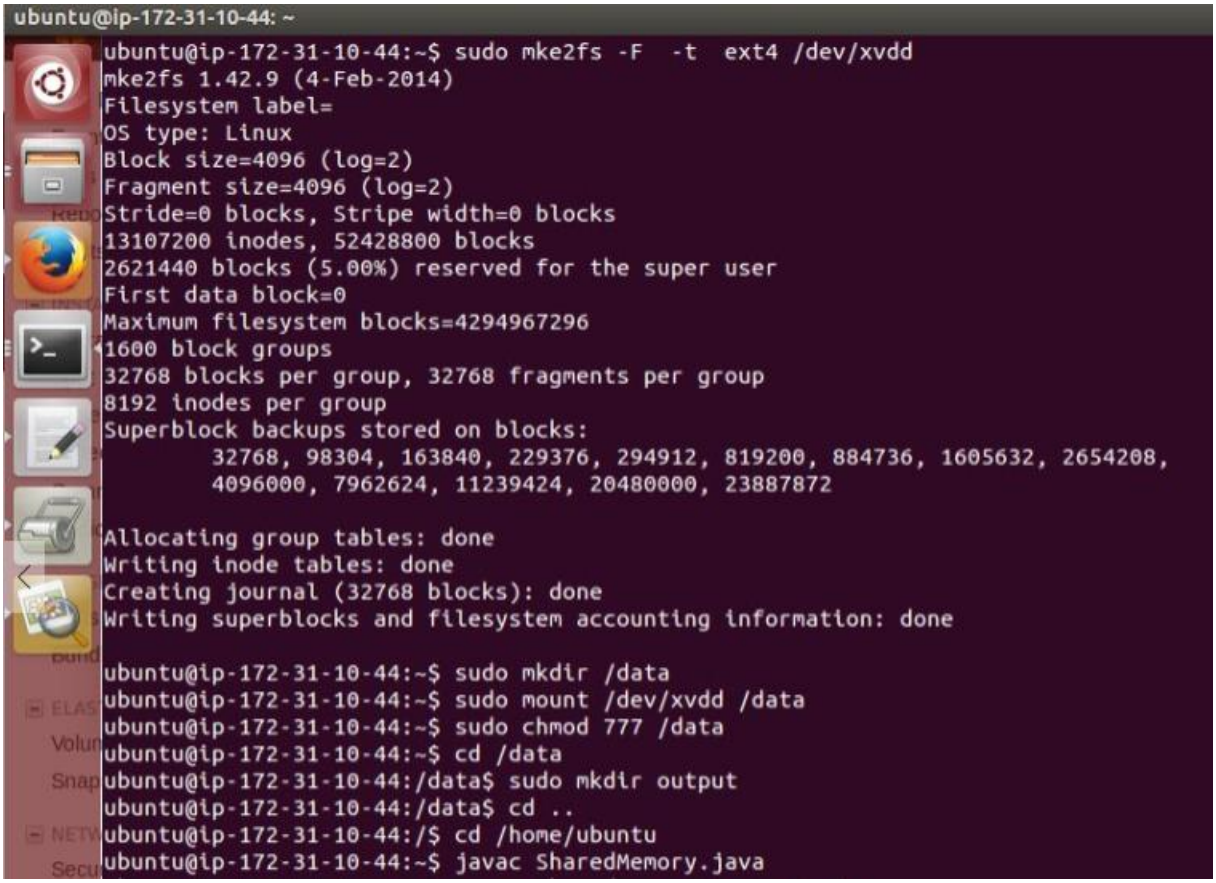
This Shared Memory Sort is implemented in java. Multithreading is used for its implementation. ExecutorsService class for is used for multithreading.
ExecutorService exec = Executors.*newFixedThreadPool*(noOfThreads);
The Shared Memory Sort is tested for 1, 2, 4, and 8 threads. The implementation is designed for single node with 10GB dataset. I ran the code simultaneously on 5 instance .



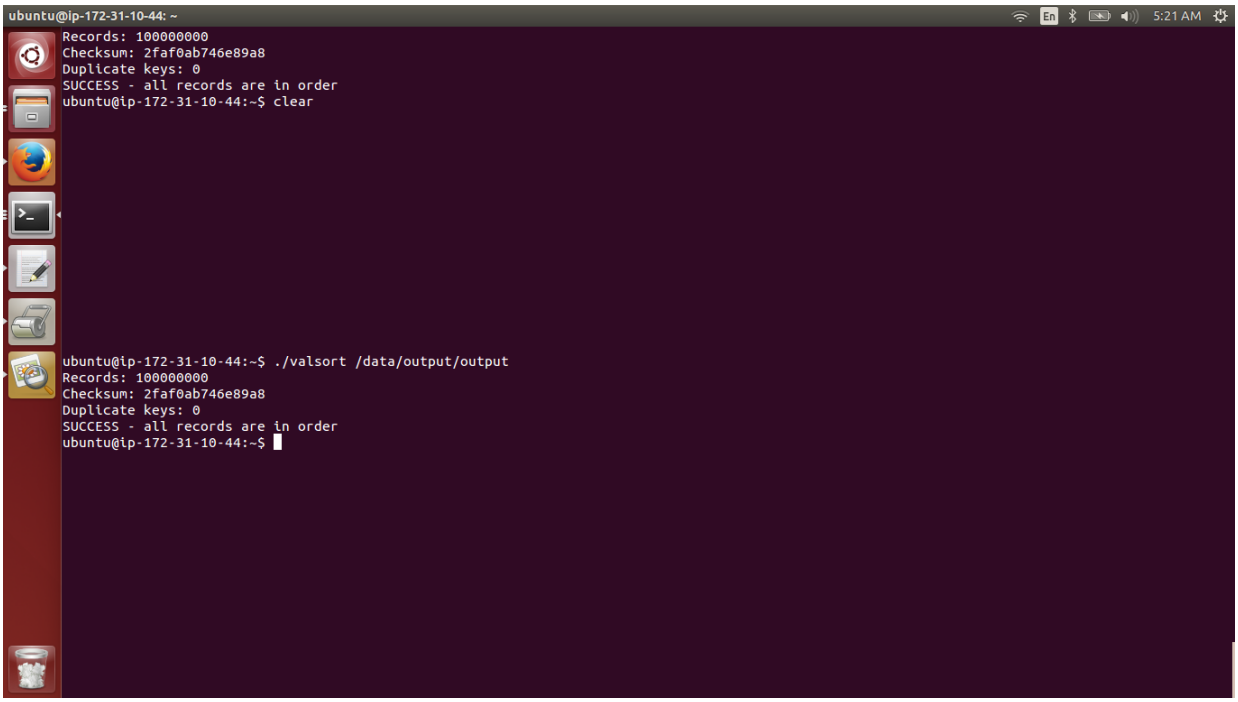
Compiling the code and running the code with 6 threads.

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

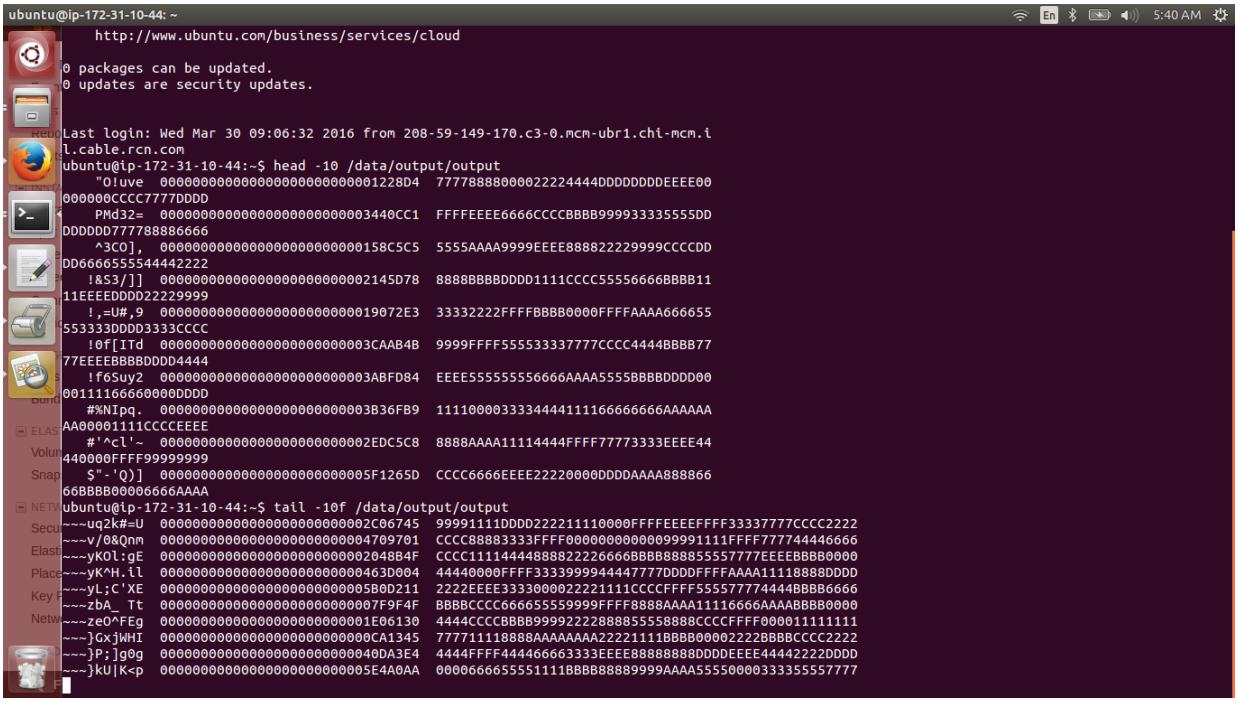


Valsort

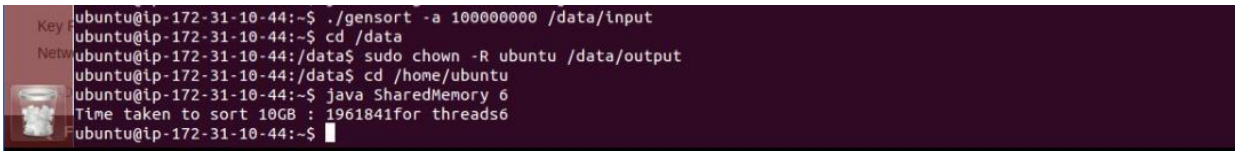


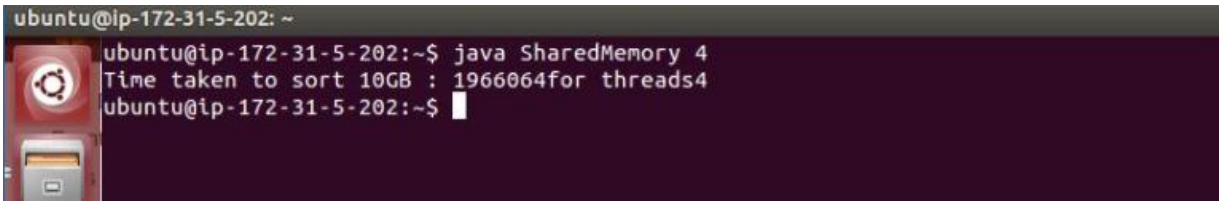
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

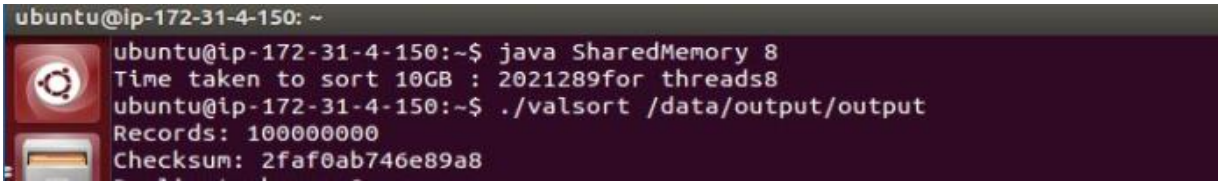
Snehal Sonawane **A20360111**

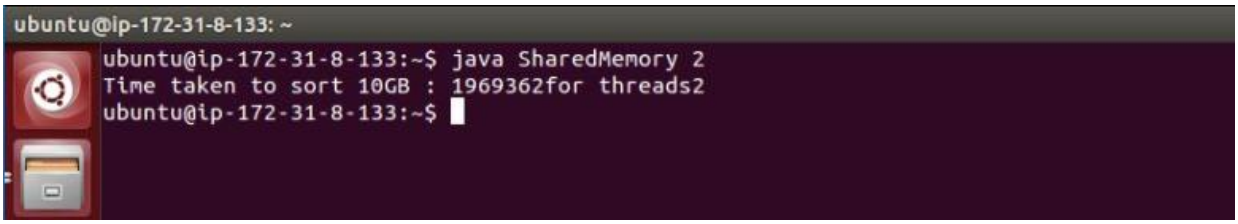


Result



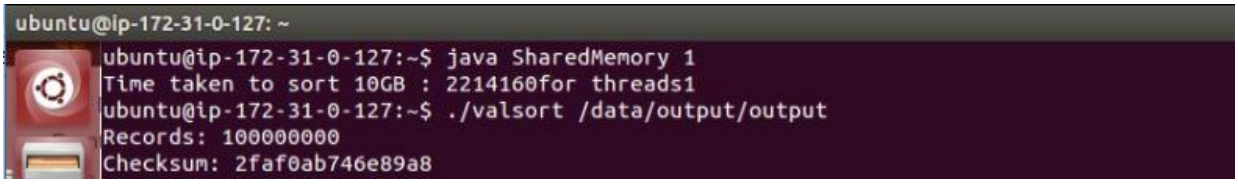






PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

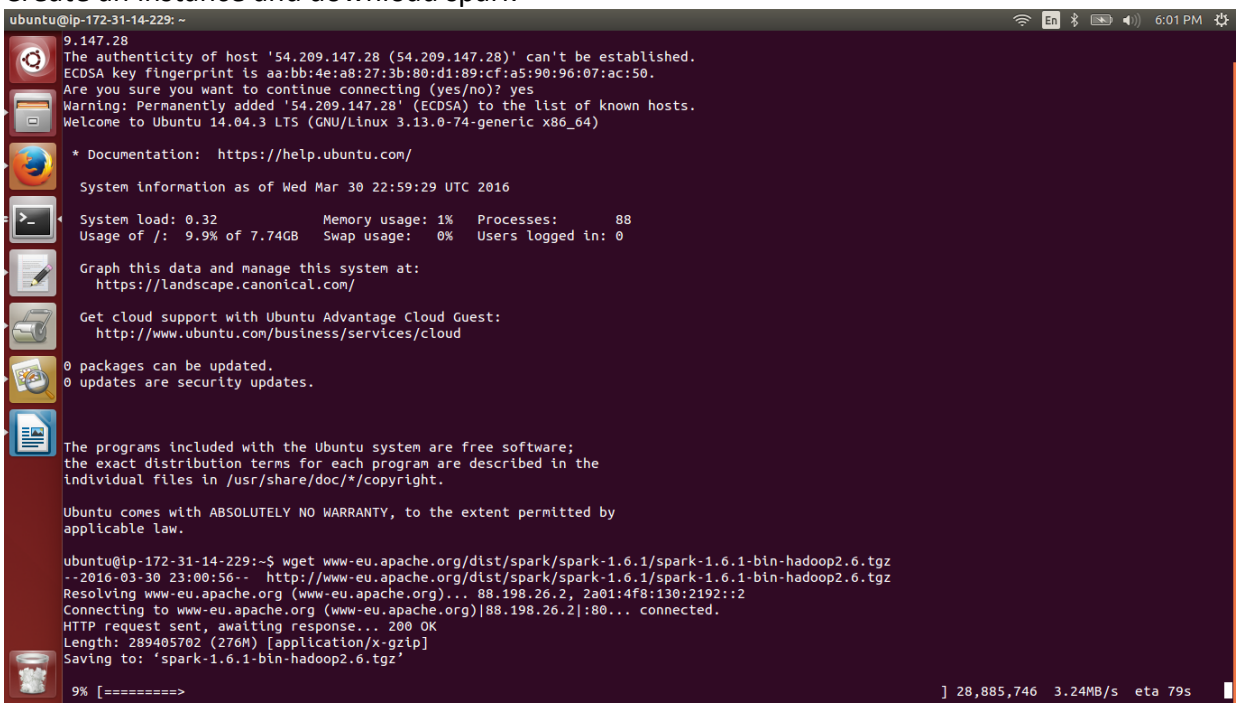


SPARK

This Spark code is written in Scala. The inbuilt function sortByKey is used to sort the input file generated using gensort.

Installation Single Node

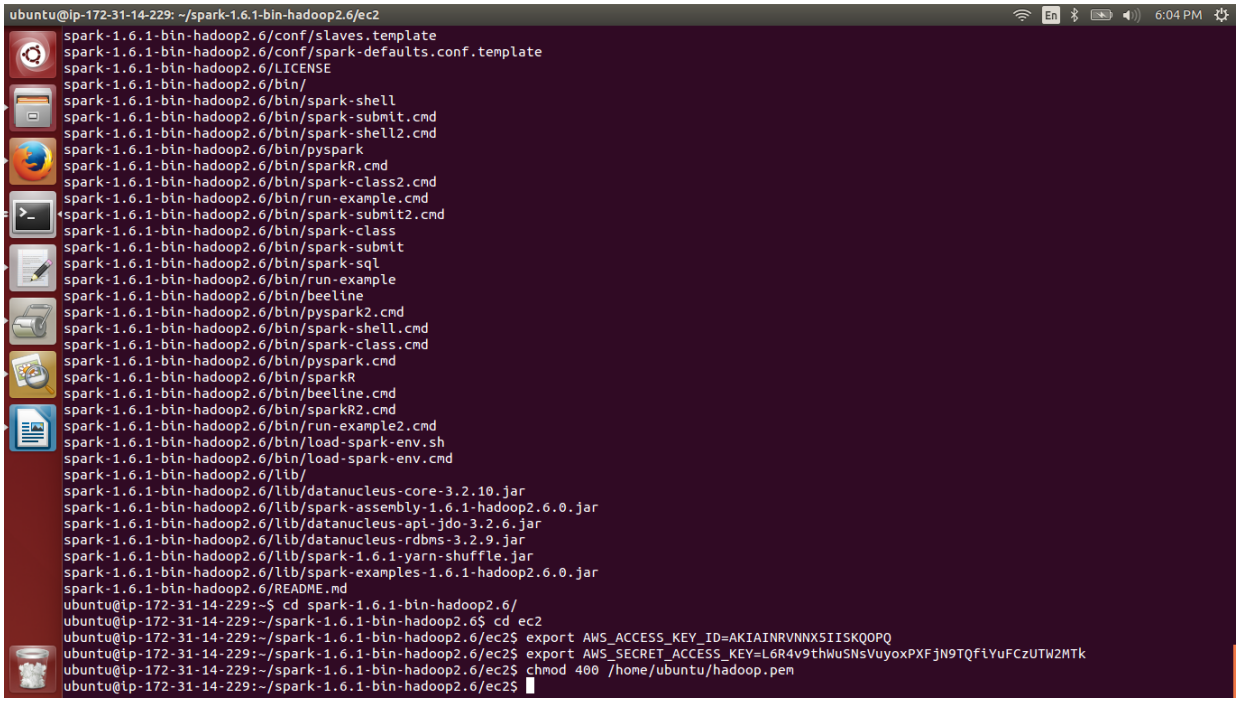
1. Create an instance and download spark



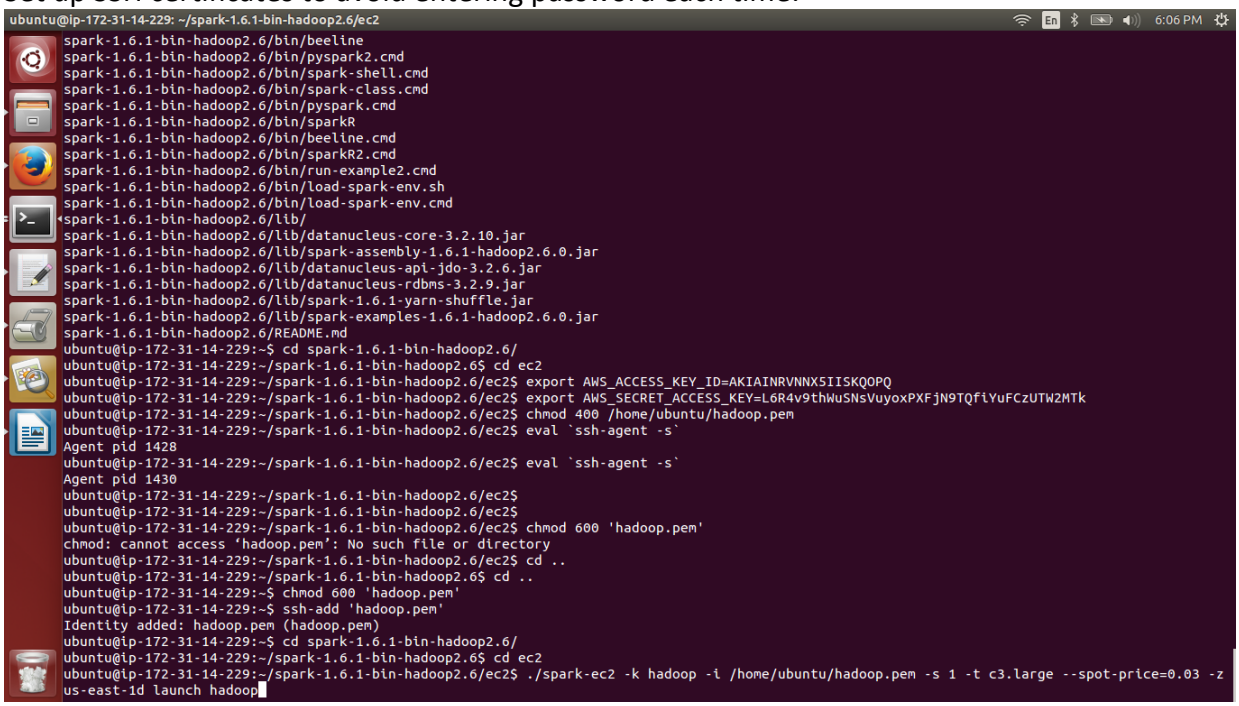
2. Unzip the downloaded file and export ACCESS KEY and SECRET ACCESS KEY

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



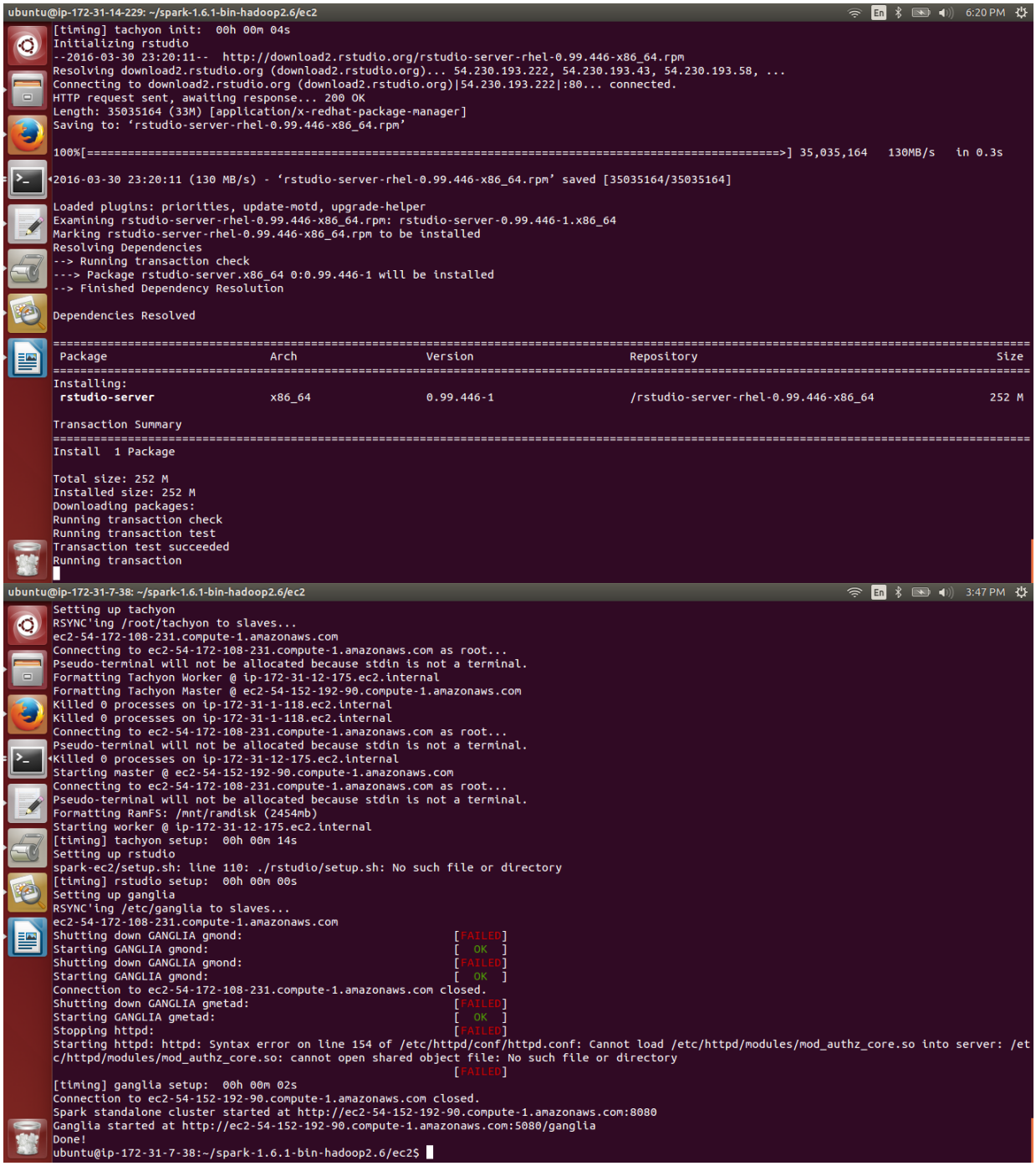
3. Set up SSH certificates to avoid entering password each time:



4. Launch Spark

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

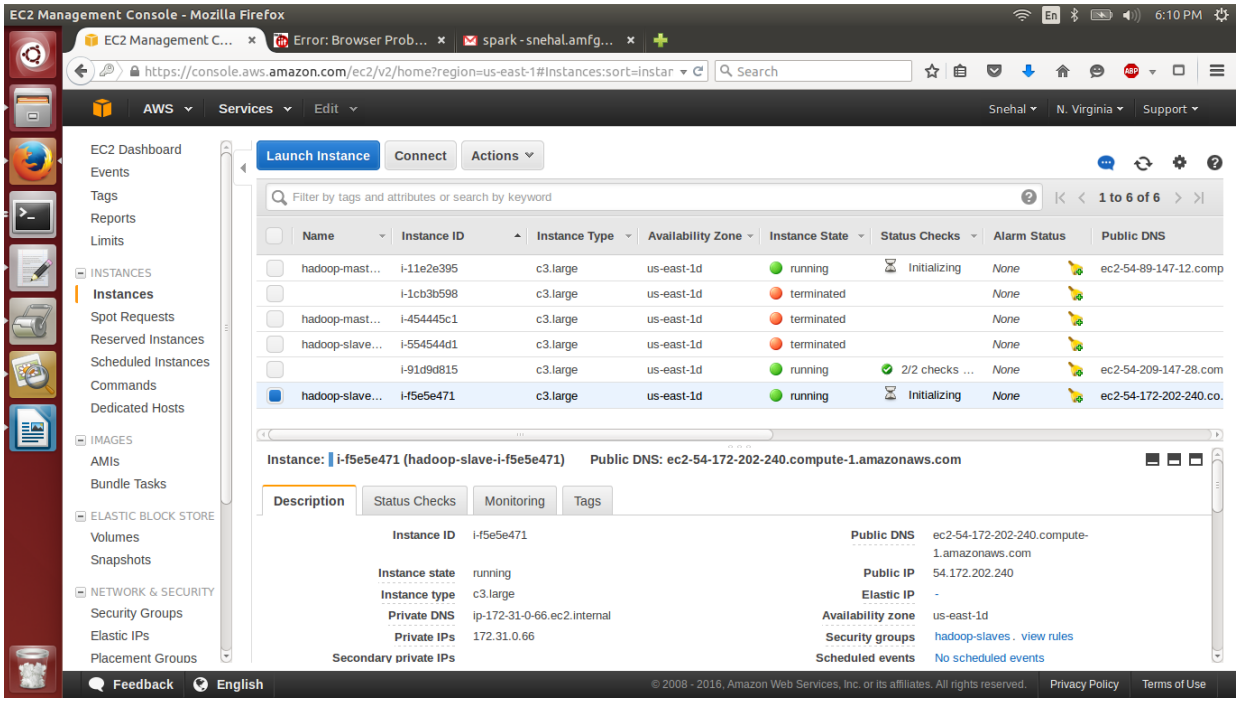
Snehal Sonawane **A20360111**



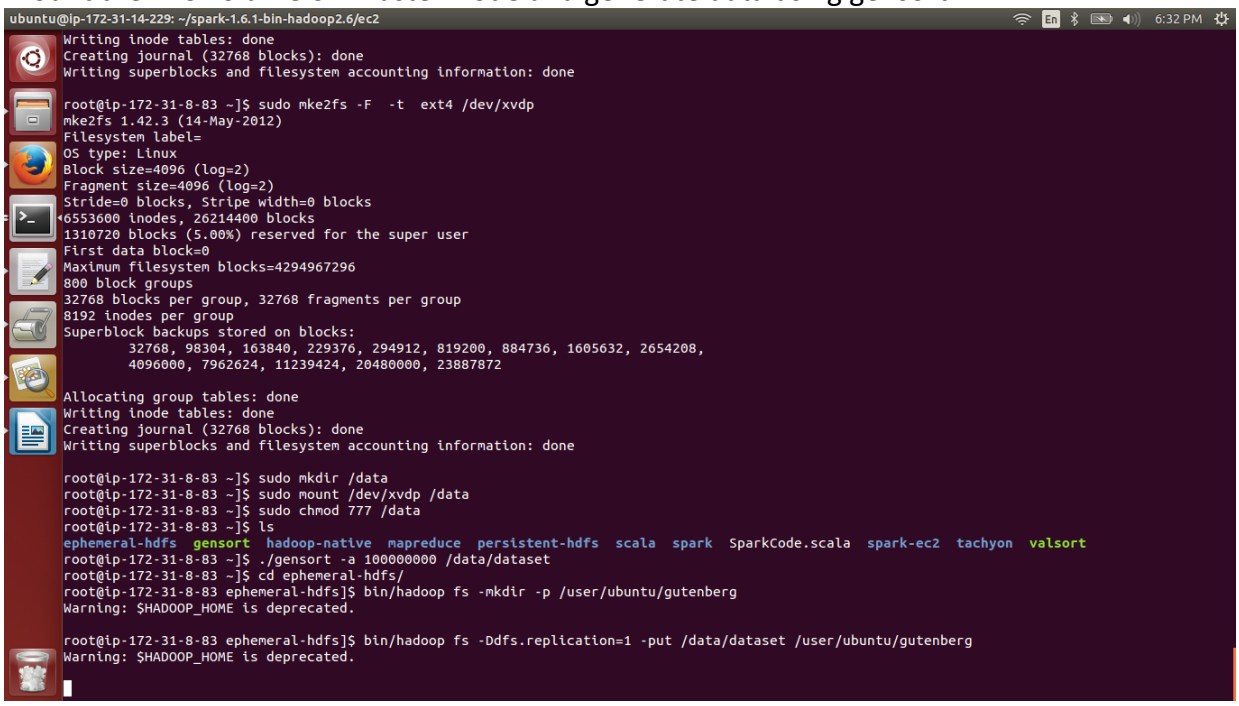
5. Spark has created a master and slave

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



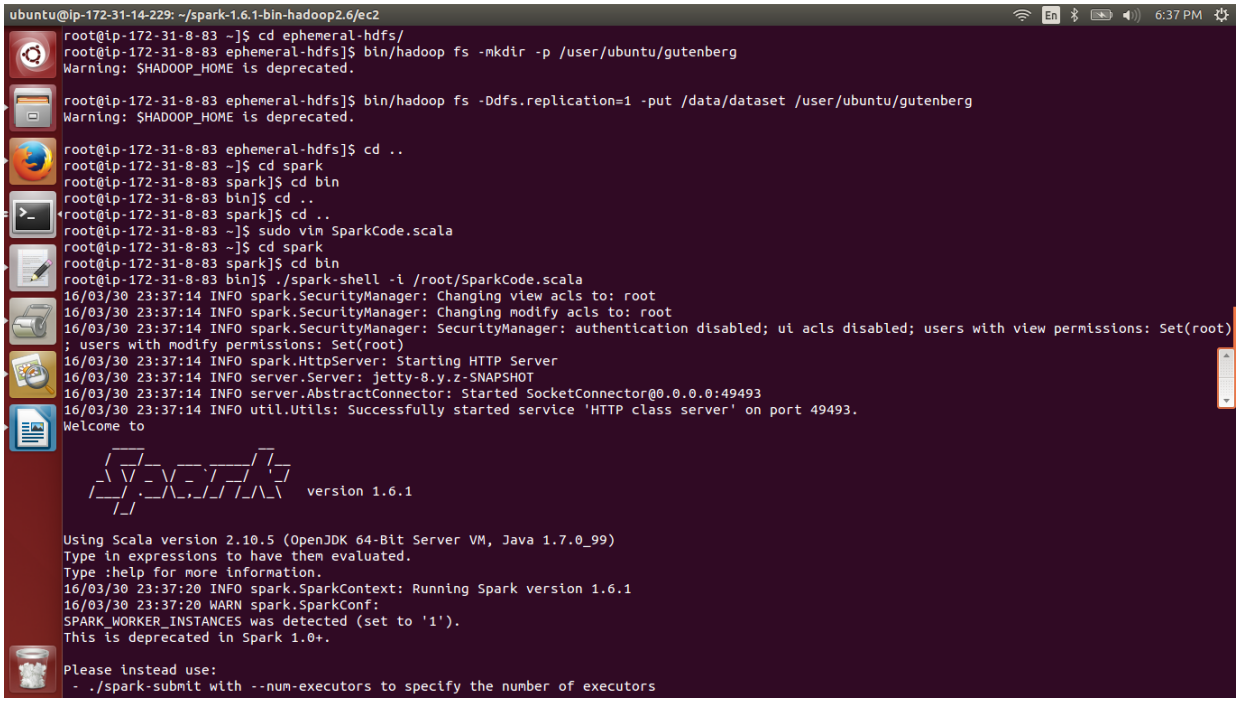
6. Mount the EBS volume on Master Node and generate data using gensort



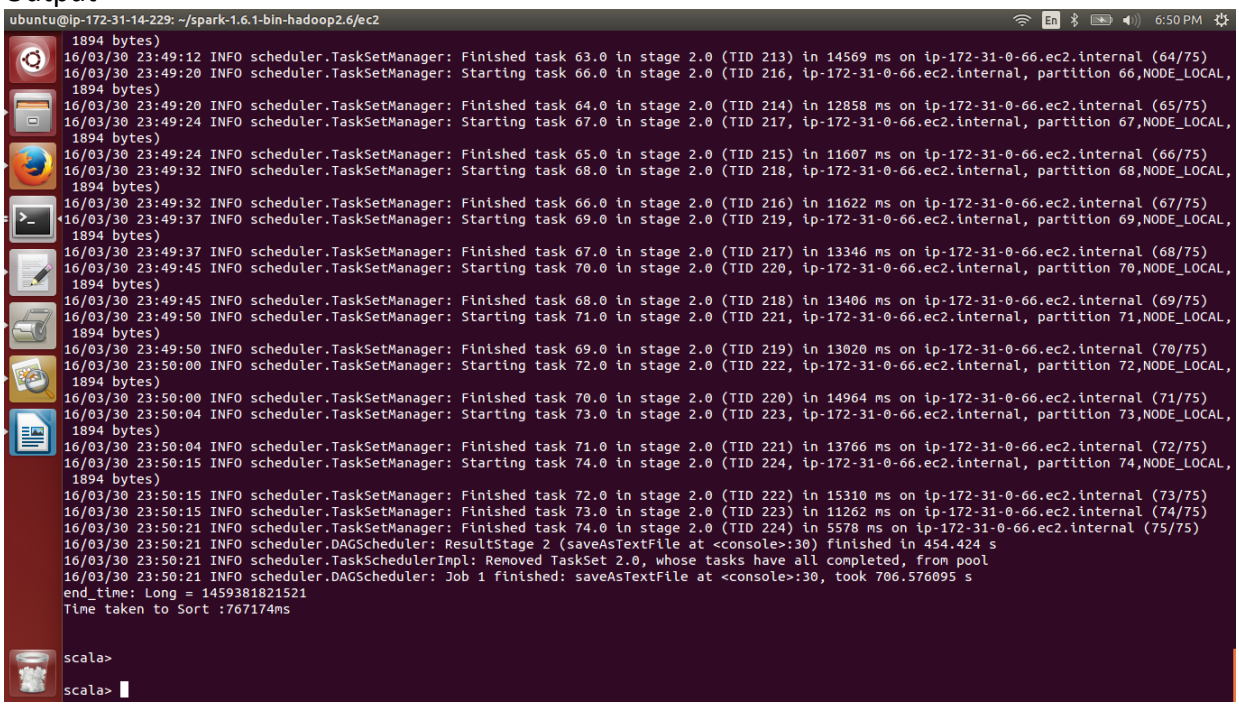
7. Run the Scala code

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



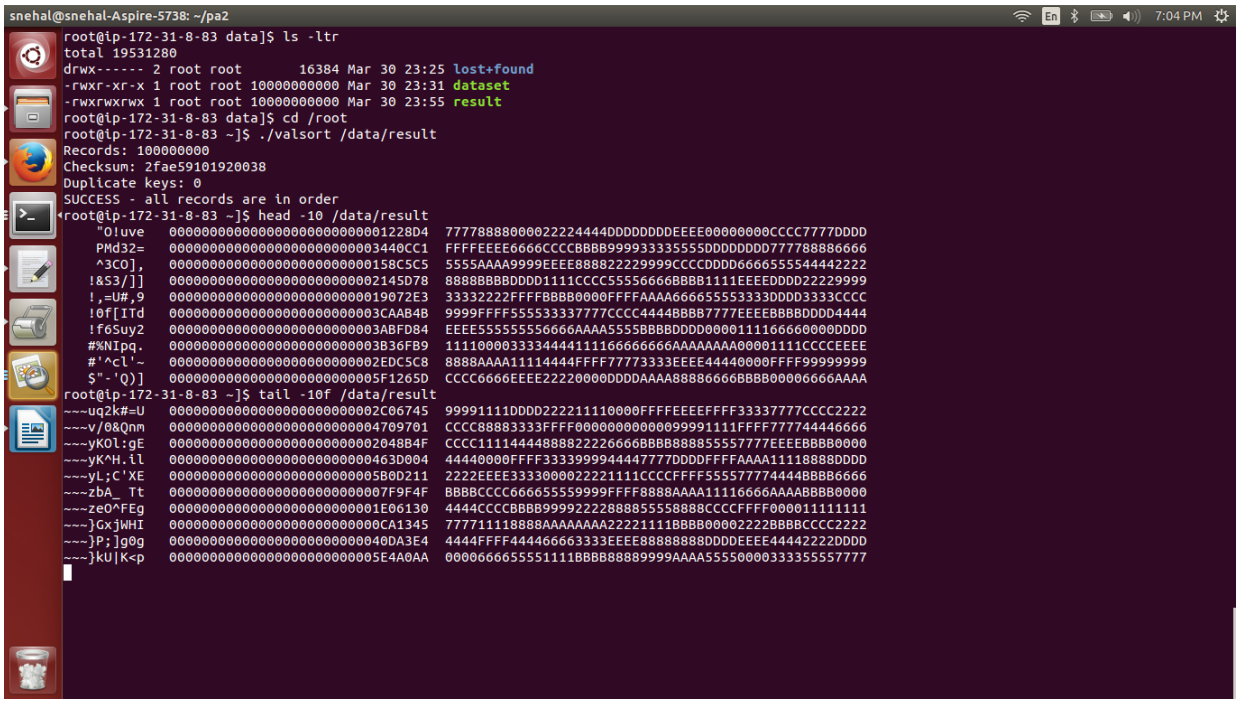
Output



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

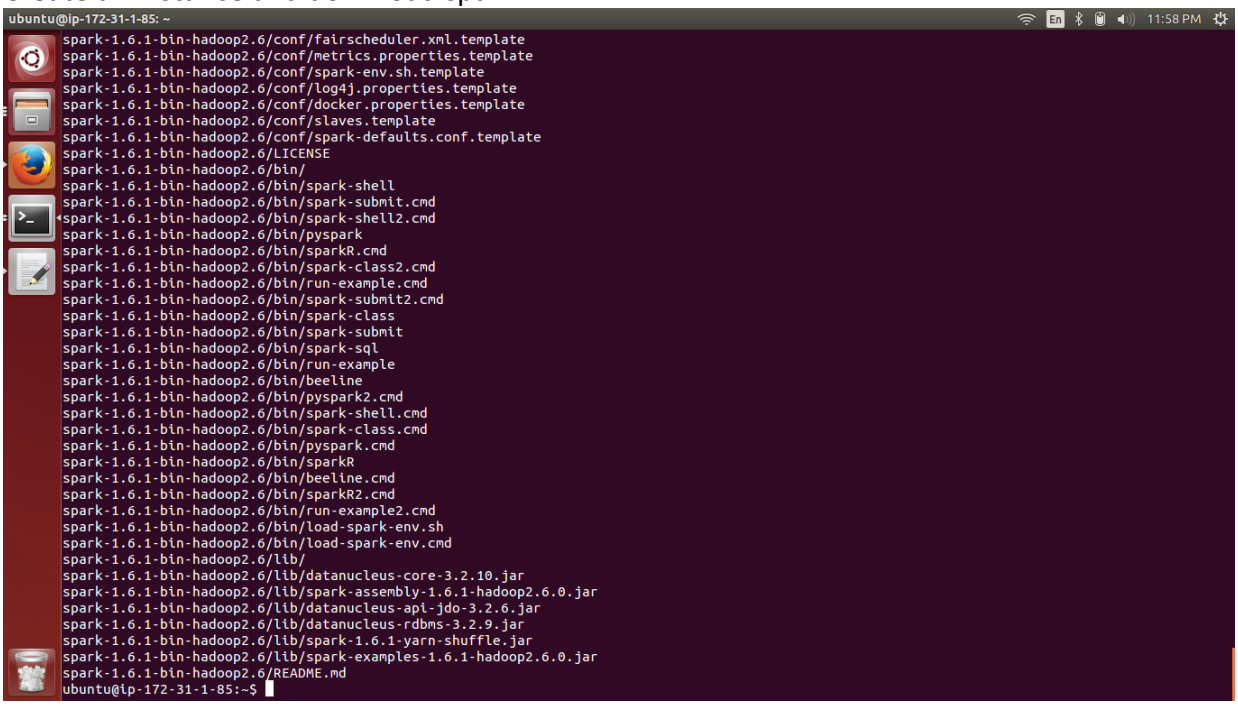
Snehal Sonawane **A20360111**

Valsort



Mulit Node Spark

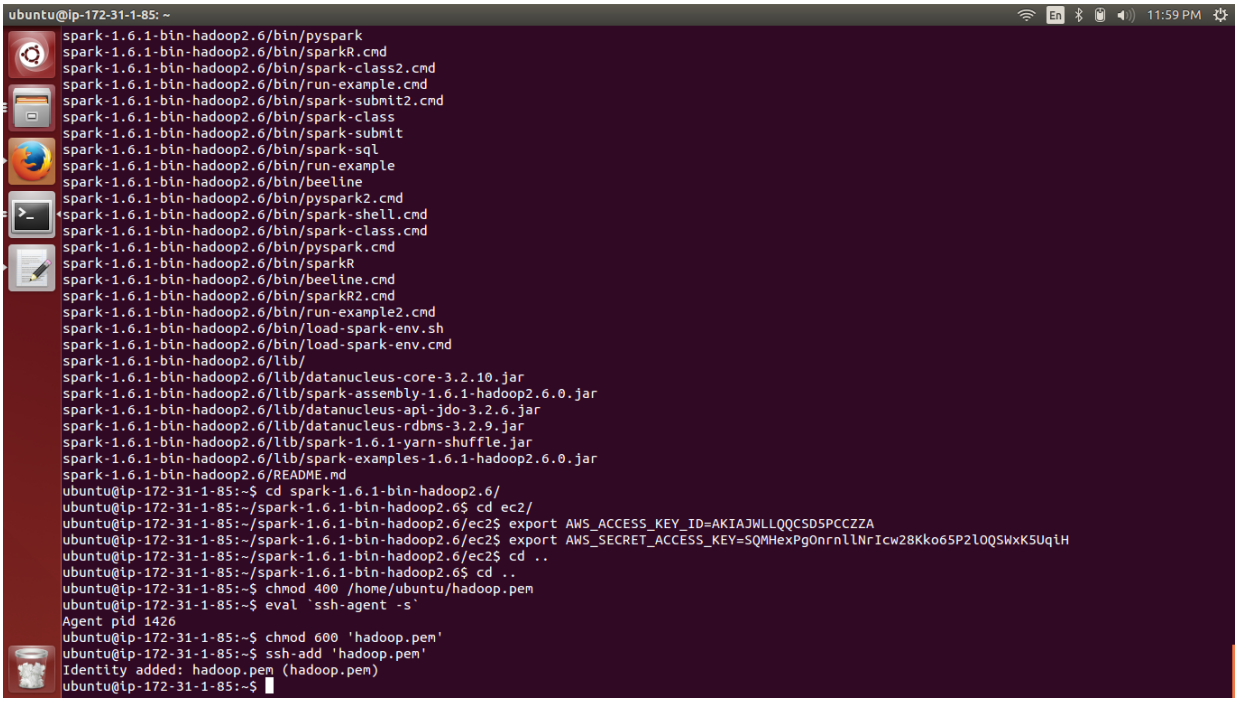
1. Create an instance and download spark



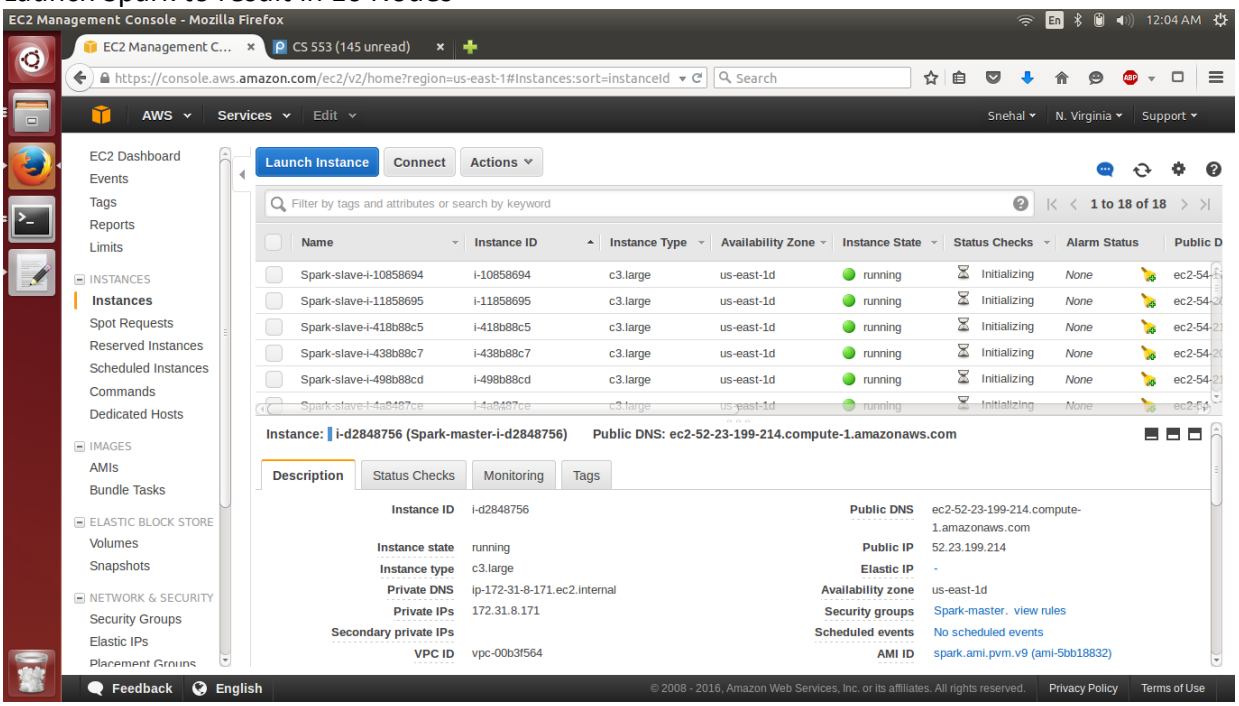
2. Unzip the downloaded file and export ACCESS KEY and SECRET ACCESS KEY

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

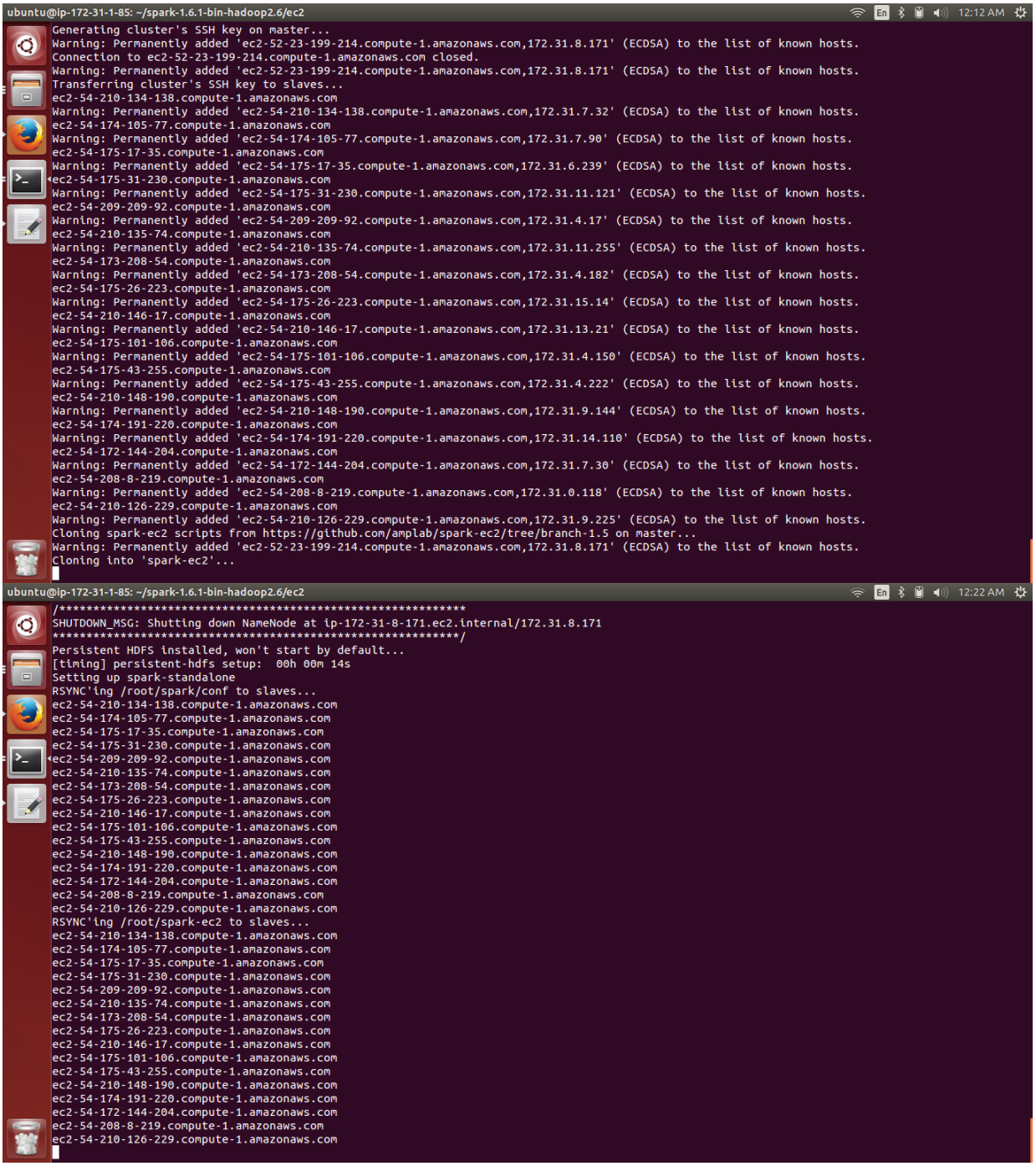


3. Launch Spark to result in 16 Nodes



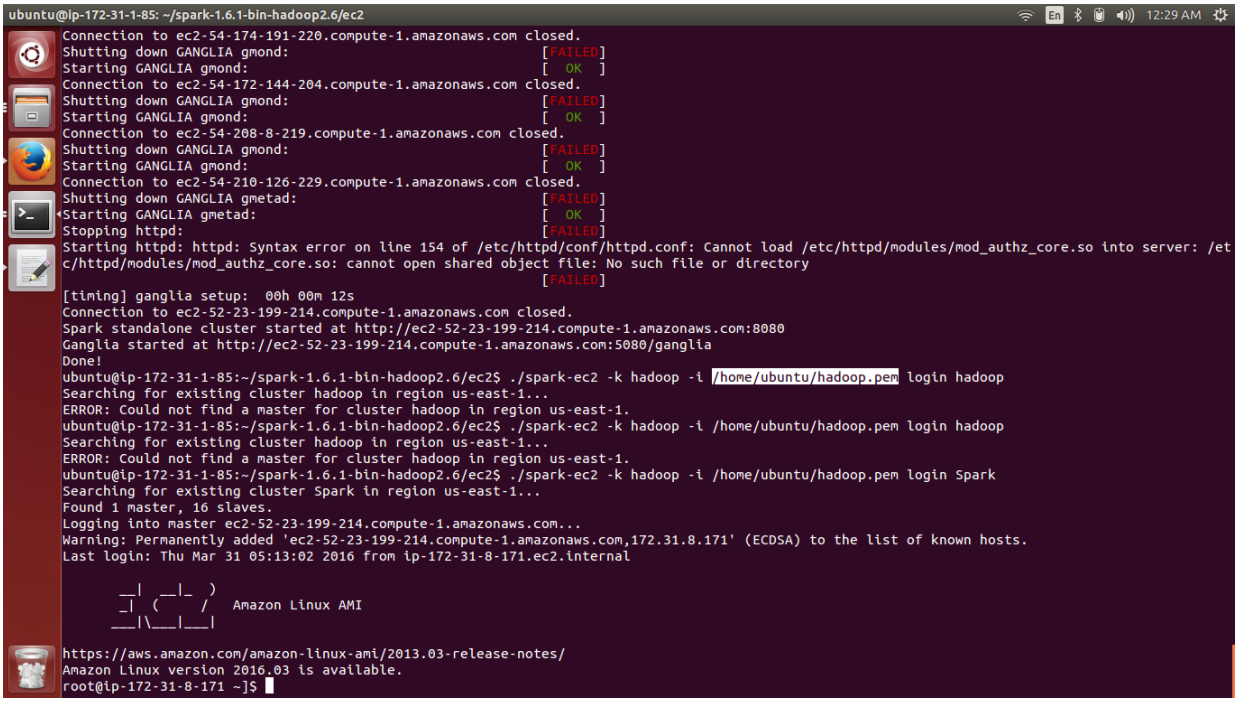
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

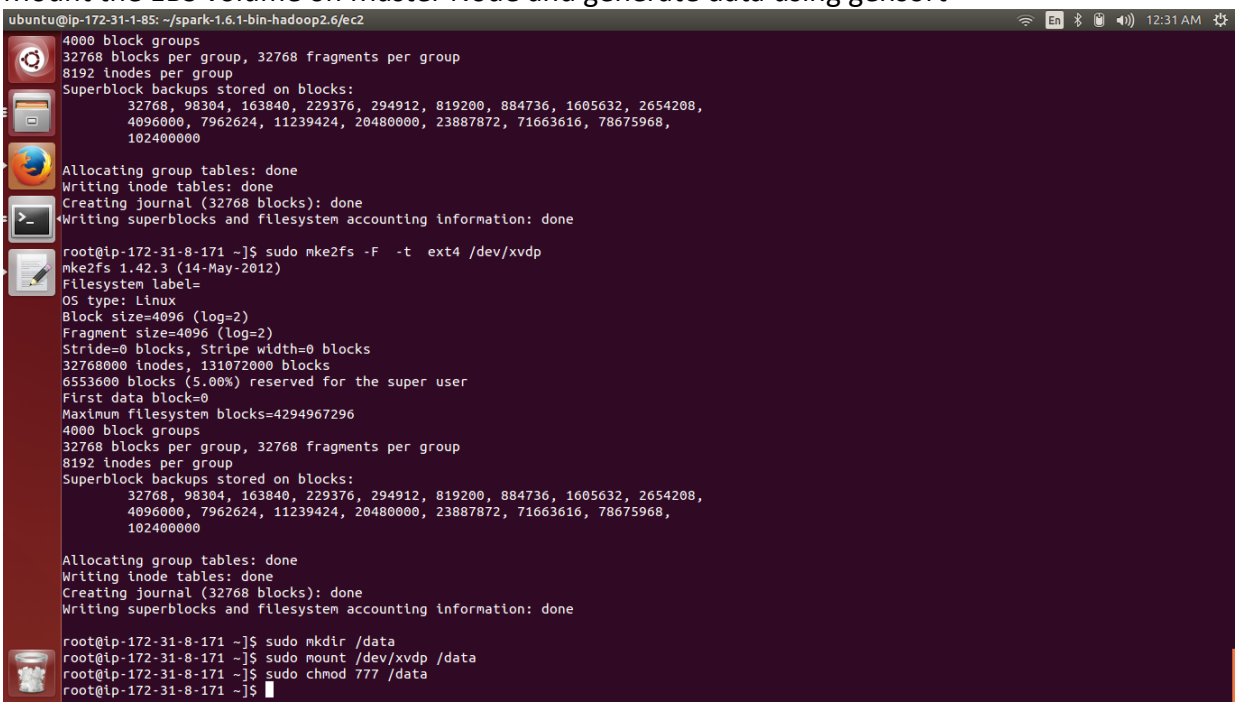


PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**



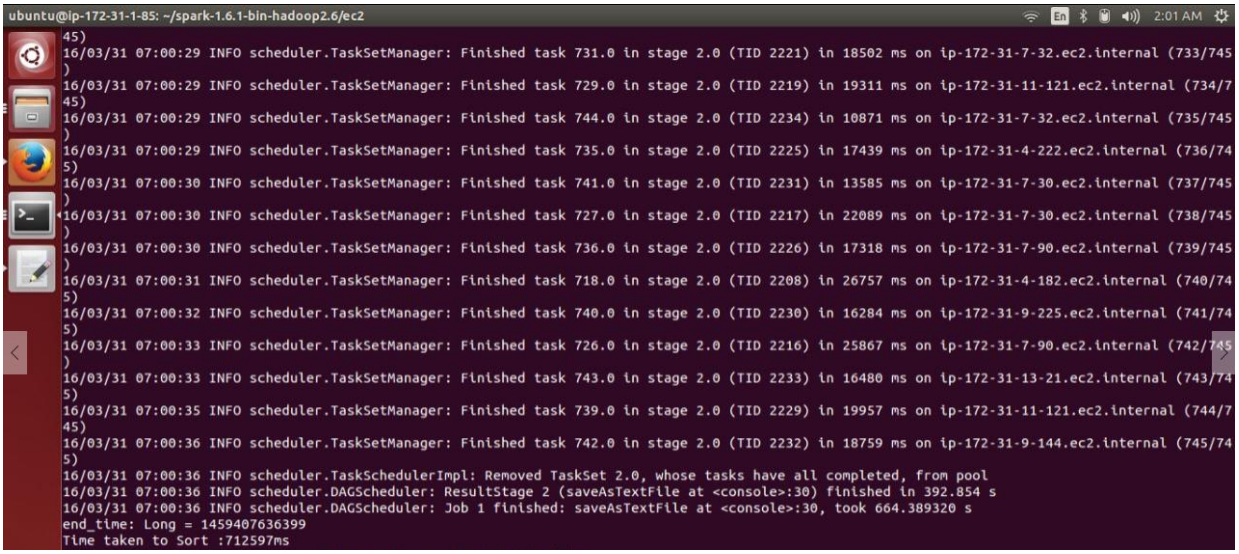
4. Mount the EBS volume on Master Node and generate data using gensort



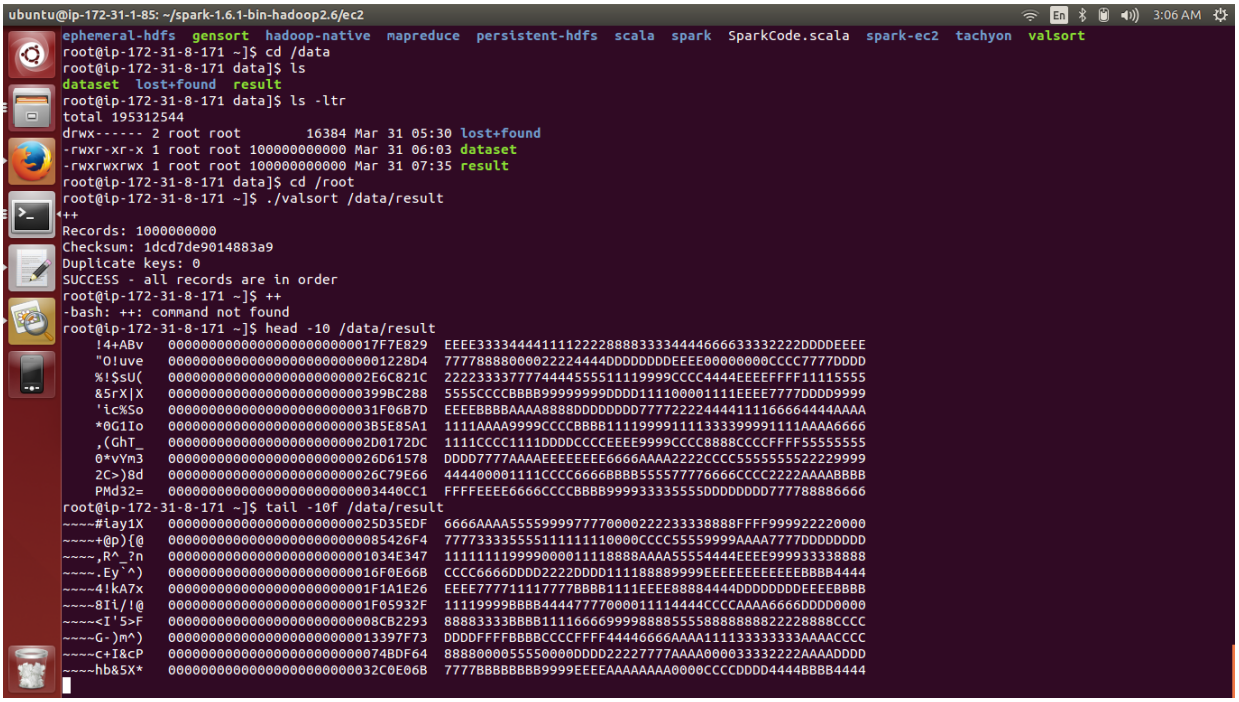
PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

5. Output



Valsort



PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

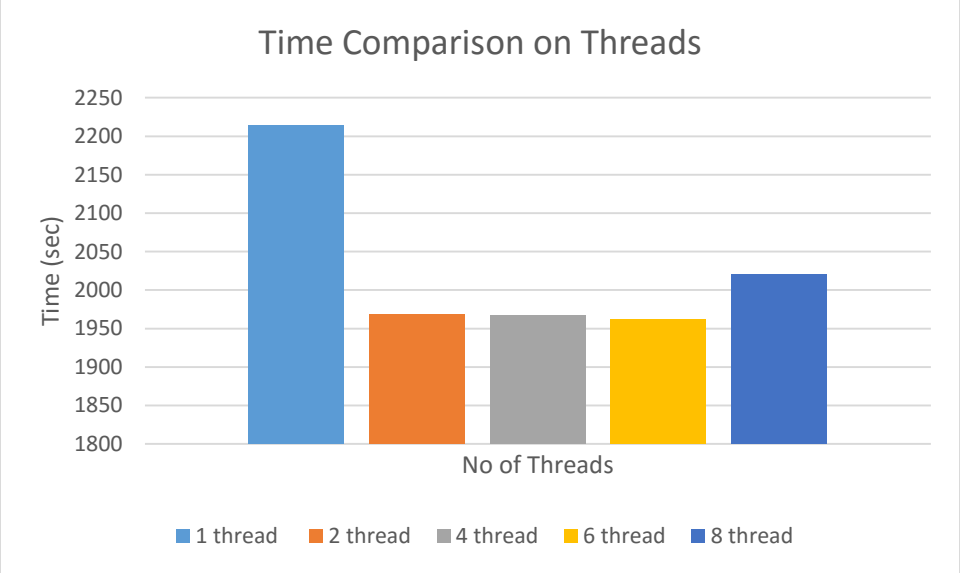
Snehal Sonawane **A20360111**

PERFORMANCE

1. Shared Memory

   The time required to sort 10 GB dataset on 1 –node using multiple threads is as follows:

Number of Threads	Time Required (sec)
1 Thread	2214
2 Thread	1969.2
4 Thread	1966.8
6 Thread	1961.4
8 Thread	2020.8



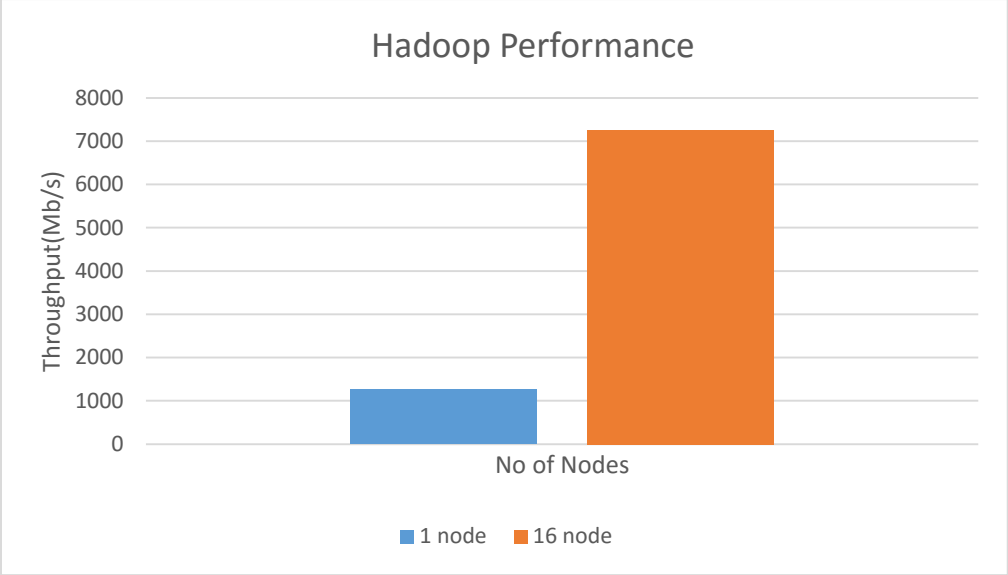
Observation:

It is observed that the Shared Memory Sort runs the fastest for 6 threads. Thus, concluding 6 threads as the best performance.

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

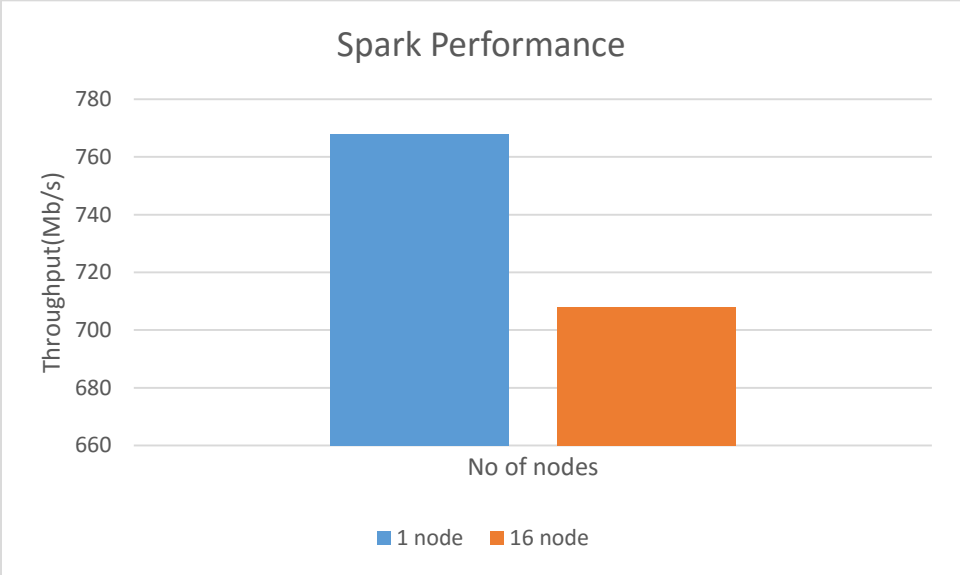
2. Hadoop
   Performance on Single Node and Multi Node



Observation:

1. It can be observed that throughput is higher for 16 nodes as we scale up from 1-node.

3. Spark
   Performance on Single Node and Multi Node

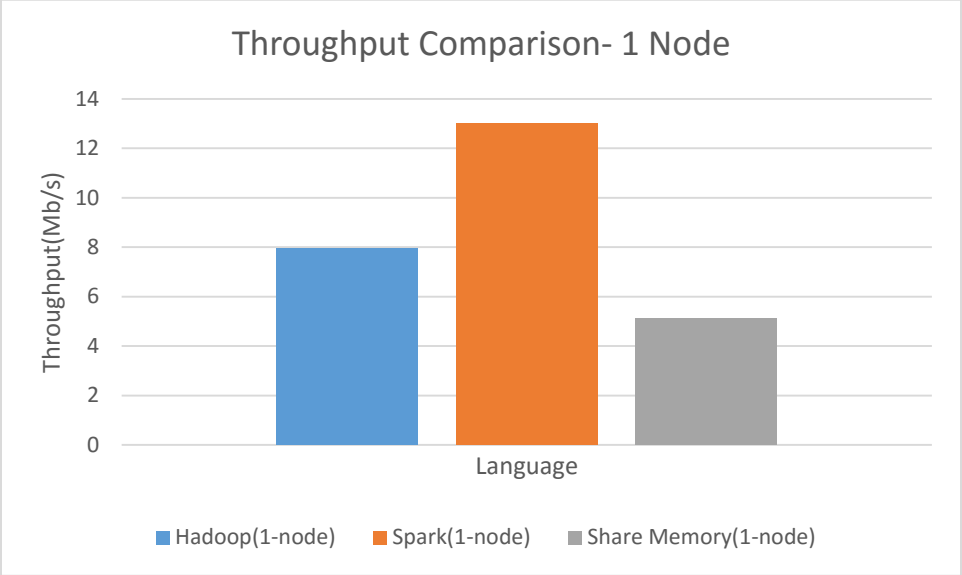


Observation:

1. It can be observed that the throughput is higher on 1- node. But here is no

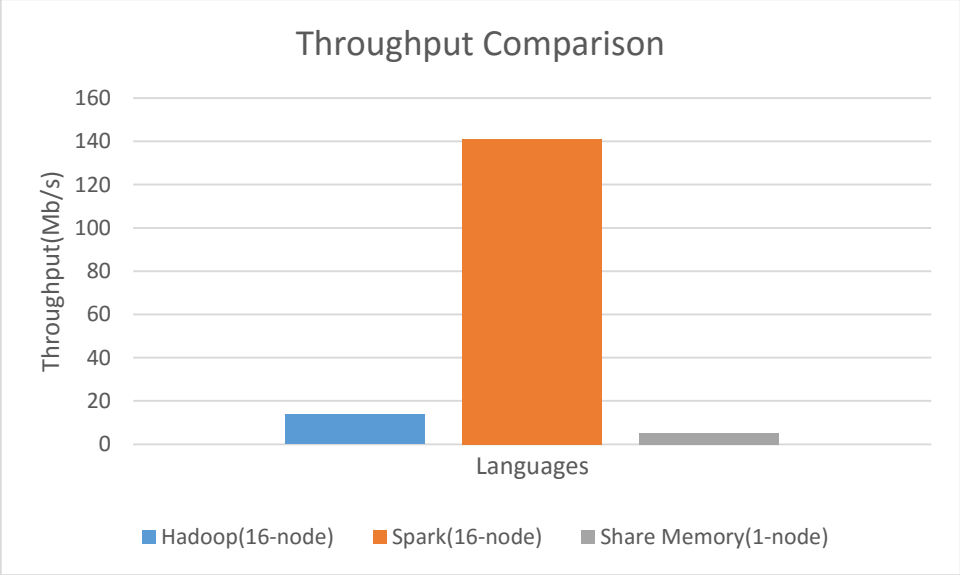
drastic change in the values.

4. Performance of Shared Memory, Hadoop, Spark



Observation:

1. Throughput is highest for Spark in comparison to Hadoop and Shared Memory on a 1 node scale.



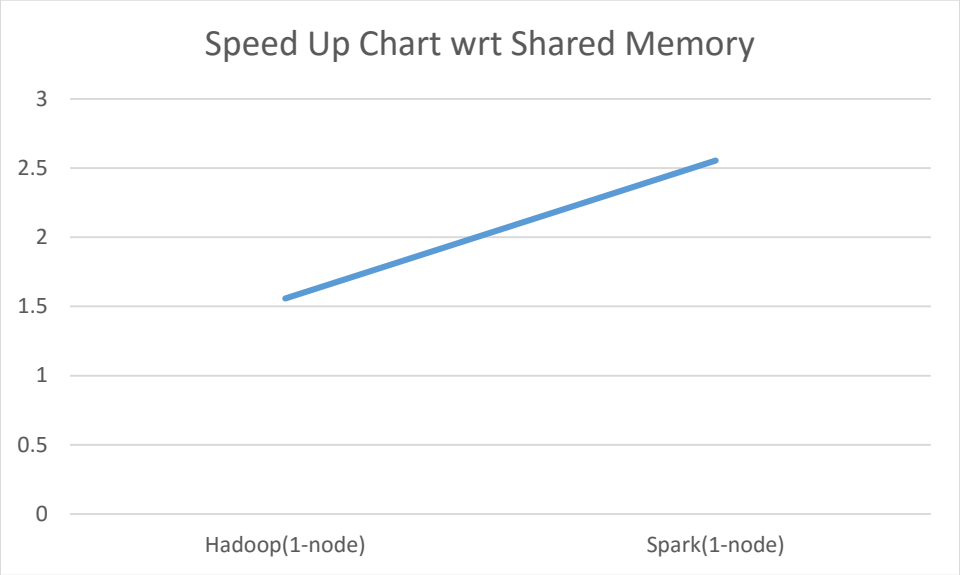
Observation:

1. Throughput is highest for 16 node Spark in comparison to 16 Node Hadoop and 1 node Shared Memory
2. Throughput of Spark is high in spite of the scalability from 1-node to 16-node.

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

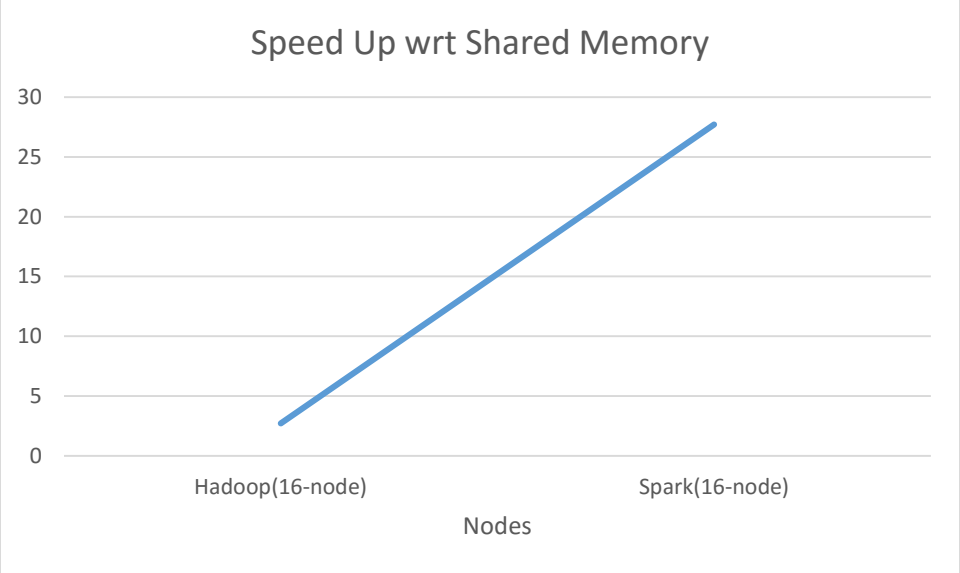
Snehal Sonawane **A20360111**

5. Speed Up Line Chart with respect to Shared Memory on 1- node



Observation:

1. It is observed that the speed up is greater than 1 for both Hadoop and Spark with respect to Shared Memory.
2. It can be concluded that the code is not optimized to give the best solution.



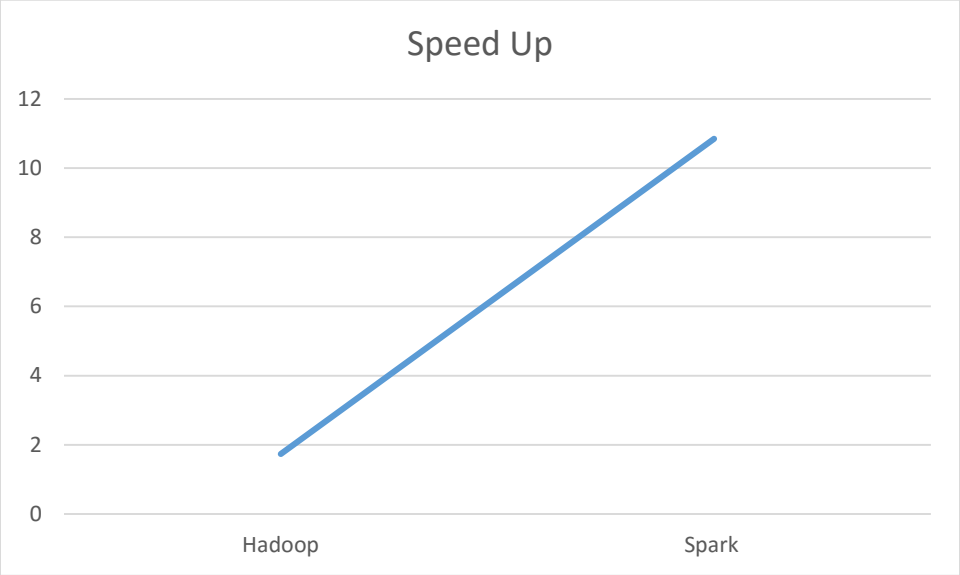
Observation:

1. It is observed that the speed up is greater than 1 for both Hadoop and Spark with respect to Shared Memory.
2. For Spark it is immensely high.

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

3. It can be concluded that the code is not optimized to give the best solution.



Observation:

1. It is observed that the speed up is greater than 1 for both Hadoop 16-node and Spark 16 node with respect to Hadoop 1-node and Spark 1-node.
2. It follows the expected trend.

Questions:

1. What conclusions can you draw? Which seems to be best at 1 node scale? How about 16 nodes? Can you predict which would be best at 100 node scale? How about 1000 node scales?
   Answer: It can be concluded that Spark is the best for 1-node and 16-node scale and 100 node scale based on the statistics. Spark is relatively new so the performance of Spark cannot be determined to be the best on a 1000 node scale.

2. Compare your results with those from the Sort Benchmark [9], specifically the winners in 2013 and 2014 who used Hadoop and Spark.
   Answer:

	Spark	Hadoop
2014 & 2013	100 TB in 1,406 seconds 207 Amazon EC2 i2.8xlarge nodes	102.5 TB in 4,328 seconds 2100 nodes
My Results	10 GB in 768 seconds 1 node	10 GB in 1260 seconds on 1 node

   100 TB in how many seconds using my code?

PROGRAMMING ASSIGNMENT 2
Sort on HADOOP/SPARK

Snehal Sonawane **A20360111**

10 gb→ 768 sec
100000 gb → 7680000 sec on 1 –node
Thus, with the increase in the number of nodes the throughput increases for both Hadoop and Spark. As my code won't be able to give the best results in comparison with the sort benchmark for both spark and Hadoop.

3. Also, what can you learn from the CloudSort benchmark.
   Cloudsort benchmark sorts records at a cheap cost on public clouds. Since the clouds are not well supported to work for IO intensive workloads and also since we face total –cost of ownership. Cloudsort benchmark is innovative and supports the total cost of ownership