CS 553 Cloud Computing

Programming Assignment 2

Report

Hariprasad Ravi Kumar

A20348609

The following are the most important aspects in this assignment:

- Creating a Virtual Cluster using AMAZON WEB SERVICES(AWS)
- Implementing Shared-Memory Sort in JAVA
- Implementing Sort in HADOOP
- Implementing Sort in SPARK
- Implementing Sort using MPI
- Implementing Sorting across 4 approaches (JAVA, HADOOP, SPARK, MPI)
- Measuring the performance of the above

Creating a Virtual Cluster using AMAZON WEB SERVICES (AWS):

First we log in into the Amazon AWS service website (http://aws.amazon.com/) and we select EC2 from the network and services section. We select launch instance here. Then we configure our service to meet our requirements. We choose the AMI we need. Then we choose the instance type which can be micro, small, medium, large Xlarge, 2Xlarge, 4Xlarge or 8Xlarge.
Then we configure the instance i.e we can choose the instance to be either spot instance or on Demand instance. We then add storage to our selected instance and configure the security group. We add TCP, UDP and ICMP protocols from the security group. We then create the instance. We access the instance by connecting to the instance by suing the public IP address of it and later on we can configure the settings of the AMI that we are using.

Implementing Sort in JAVA:
We implement a java program to count the occurrence of each word in a given input file of size 10GB and 100GB. We print the output into a file in the increasing order of the frequency of the word.

Implementing Sort in HADOOP:
After downloading and installing HADOOP and extracting its components and adding the library paths to the bash we connect to the AWS instance using ssh –i key (IP address). The sort application is implemented using HADOOP for 10GB on a single node and 100GB dataset on 16 nodes.

Implementing Sort in SPARK:
The same Sort application is implemented using SPARK programming language for 10GB on a single node and 100GB dataset on 16 nodes.

SORT IN MPI:

The same sort application is implemented using MPI (Message Passing Interface) for 10GB on a single node and 100GB dataset on 16 nodes.

PERFORMANCE CALCULATION:

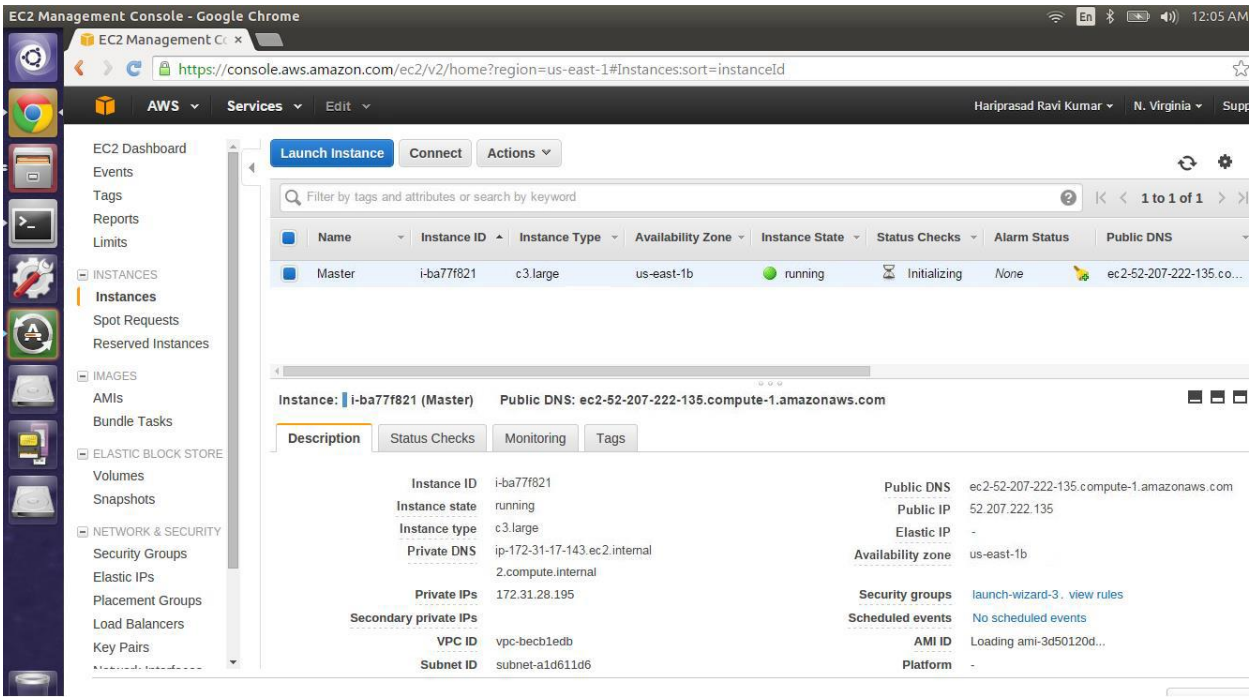
From the given data sets, all the three version of sort i.e. JAVA, HADOOP and SPARK their performances are obtained on 1 node and 16 nodes. These are compared and performance graphs are drawn for them. The performance result of shared memory sort is compared with the performance of Spark and Hadoop sorts which are implemented on 16 nodes. For each of the above separate graphs are drawn to show their performance.

SORTING:

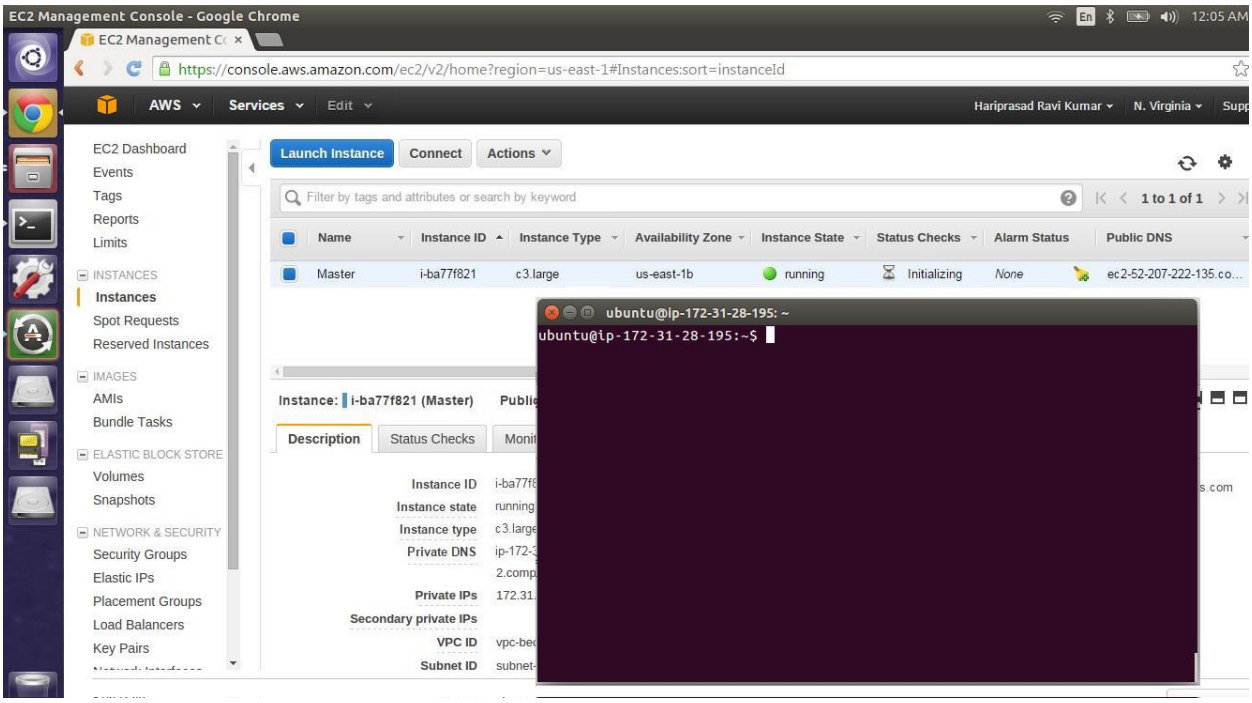
Implementation and evaluation the performance of sorting for all the 4 versions of sorts i.e. for JAVA, HADOOP, SPARK, MPI on a 10GB and 100GB datasets are done. This sorting performance includes reading, sorting and writing data to disks.

Creating an instance in Amazon(AWS):

- Log in into the Amazon AWS service website (http://aws.amazon.com/).
- Select EC2 from the network and services section.
- Select launch instance. Then configure the service and chose the best AMI to meet the requirements.
- Then choose the instance type which is c3.large
- Then we configure the instance i.e we can choose the instance to be either spot instance or on Demand instance.
- Add storage to our selected instance.
- Configure the security group. We add TCP, UDP and ICMP protocols from the security group. We then create the instance. We access the instance by connecting to the instance by suing the public IP address of it and later on we can configure the settings of the AMI that we are using.



The above screen shot is of a c3.large instance.



The above screen shot is of a connected instance via the Ubuntu terminal using the below commands:

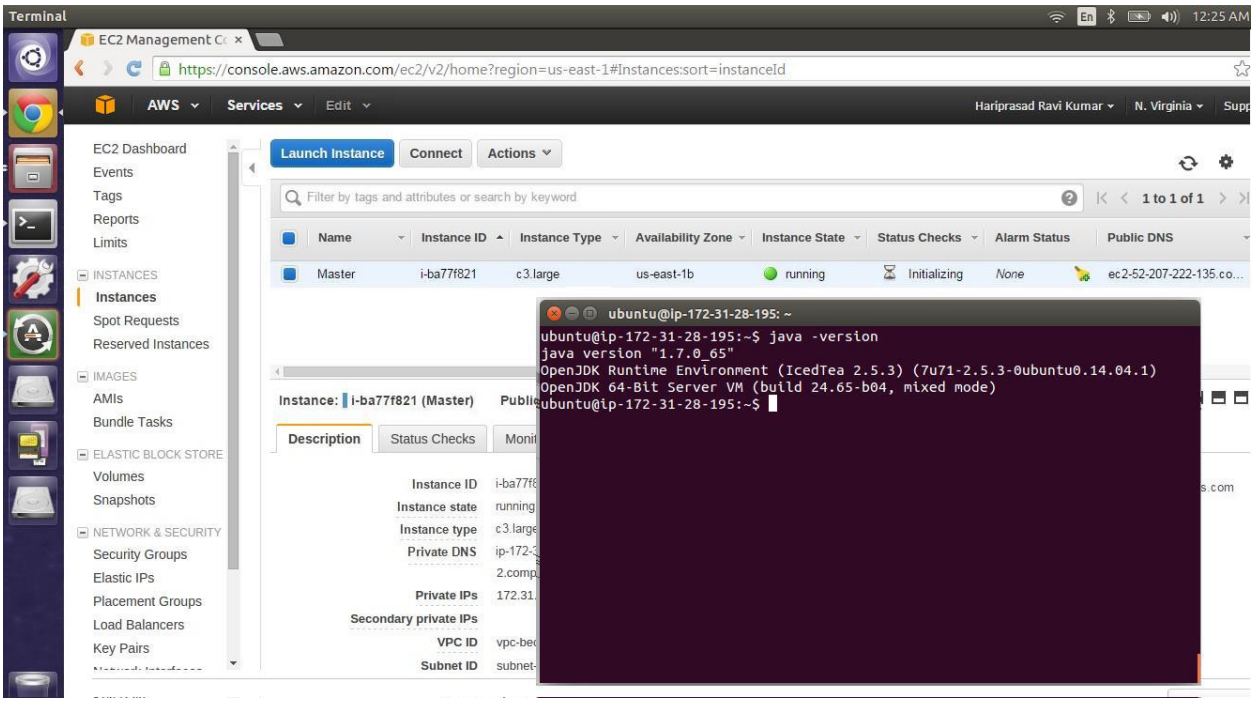
chmod 400 keygiven(This is to change the permission to be read by owner.)

ssh –i key IP address

In our case we use ssh –i newkey.pem Ubuntu@52.207.222.135

After getting connected to the AMI we now have to install java on the virtual node.

- Install java by using sudo apt-get install openjdk-7-jreheadless.
- Install javac by using sudo apt-get install openjdk-7-jdk.
- Now update all the installed applications by using sudo apt-get update
- Now check the java version by using java –version.



java –version is shown in the above screen shot

SHARED MEMORY SORT IN JAVA:

Configuration:

- VERSION: JAVA 1.7
- INSTANCE TYPE: c3. Large UBUNTU
- RAM: 3.75 GB
- NO. OF CORES: 2 virtual cores
- STORAGE: 32 GB

METHODOLOGY:

We take input of total size 10 GB, 100 GB and sort in the file.

- On the basis of number of lines, we further split the data into equal no. of chunks.
- The file name of each chunk is fetched using a user defined function called fileGeneration (). The fileGeneration () has a dual functionality of fetching the file name and removing from the list once it is processed.
- In the function count(), each chunk is opened in the read mode and every line of chunk is read and every line is tokenized and then regular that it reads is stored into the map and it is tokenized into words with delimiters as " " .
- A pattern match is run on each of these tokens to find the leading and trailing special symbols. Once these special symbols are determined the word is stored into the map (m1). If the word has occurred for first time 1 is assigned as key for that particular word. The value of this key is incremented with subsequent occurrence of the token.
- The above mentioned point c and d, are executed under function called run(). The threads are started and execute one file per thread. Numbers of threads are dependent on the number of chunks of a file.
- The point d is executed within the semaphore block to achieve the synchronization and hence avoid deadlocks in threads.
- Finally, we execute sorting. In sorting the hash map is converted into array list and this list is sorted using inbuilt sort function.
- The value of map is stored in the text file name Sort-shared-memory-10GB.txt and Sort-shared-memory-100GB.txt.

Outputs:

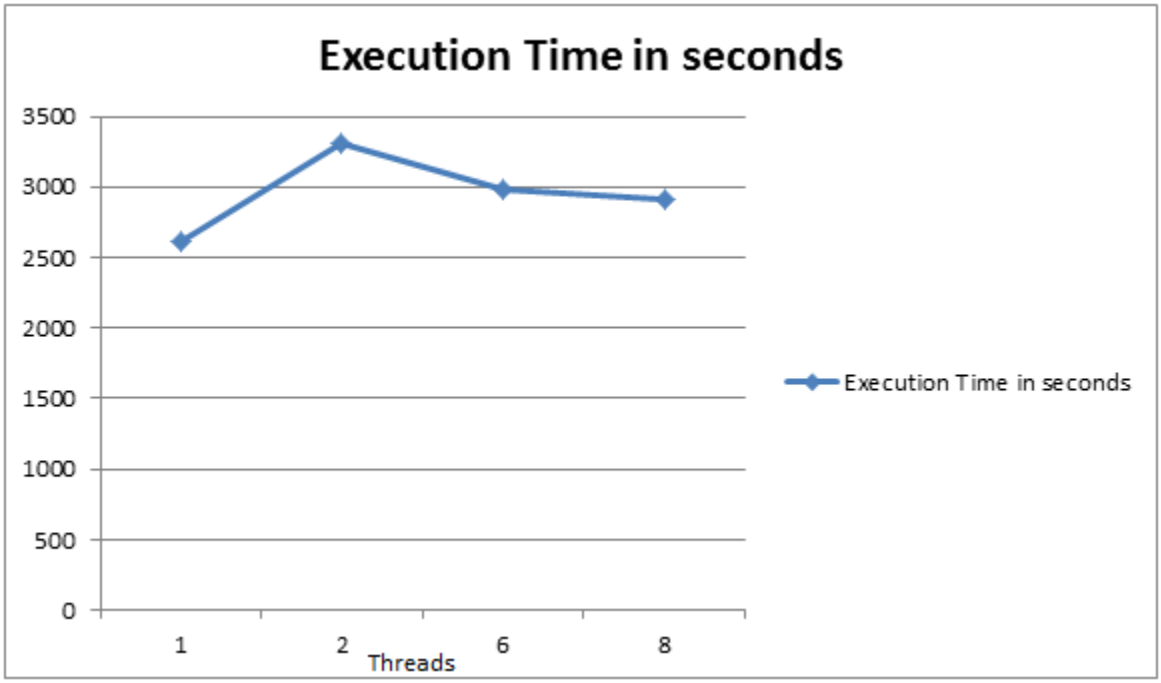
10 GB DATASET ON c3.large instance:

Number of Threads	Execution Time in seconds
1	2618.552
2	3313.751
6	2989.987
8	2917.654

EXPLANATION:

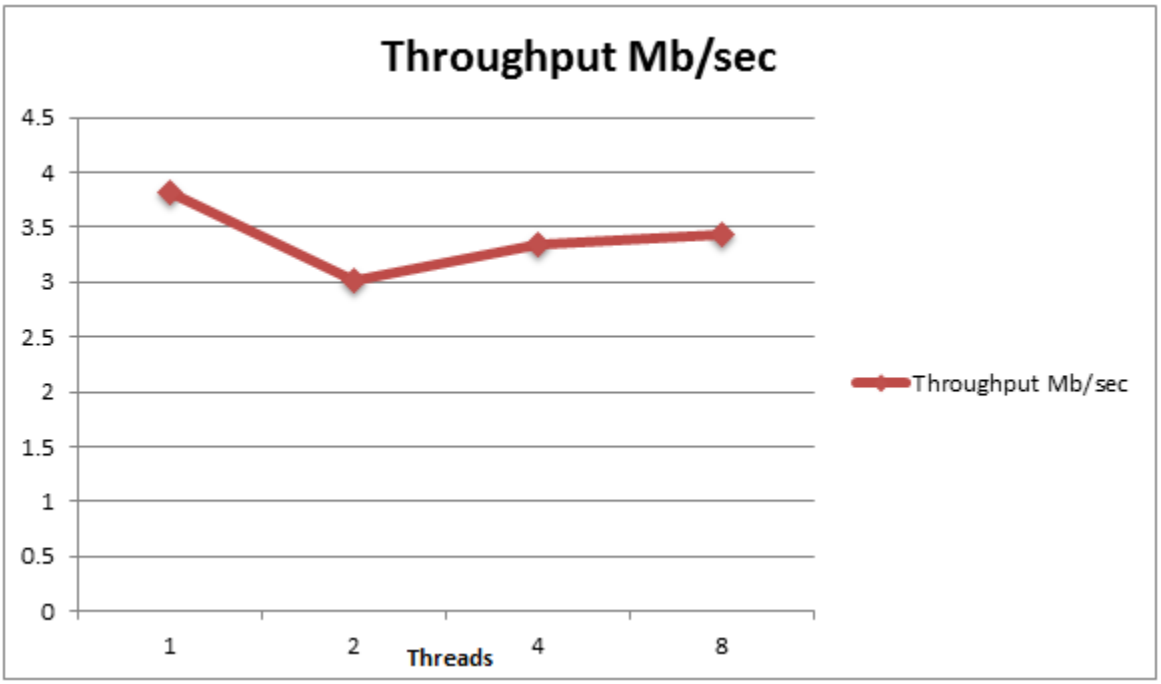
Every thread is responsible for reading a file and processing it achieves the sort functionality

- Best performance was at thread 1



10GB Throughput:

Number of Threads	Execution Time in seconds	Throughput Mb/sec
1	2618.552	3.8189
2	3313.751	3.0177
6	2989.987	3.3444
8	2917.654	3.4274



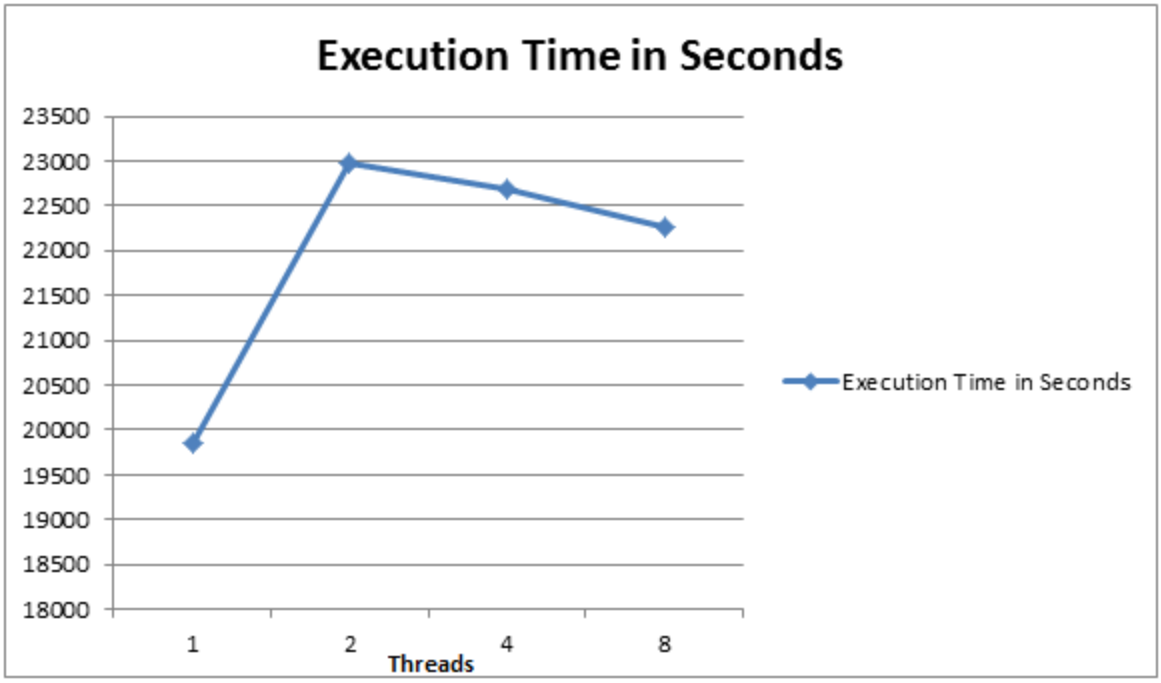
100GB DATASET ON c3.large instance:

Number of threads	Execution Time in Seconds
1	19861.547
2	22979.632
4	22681.487
8	22261.121

EXPLANATION:

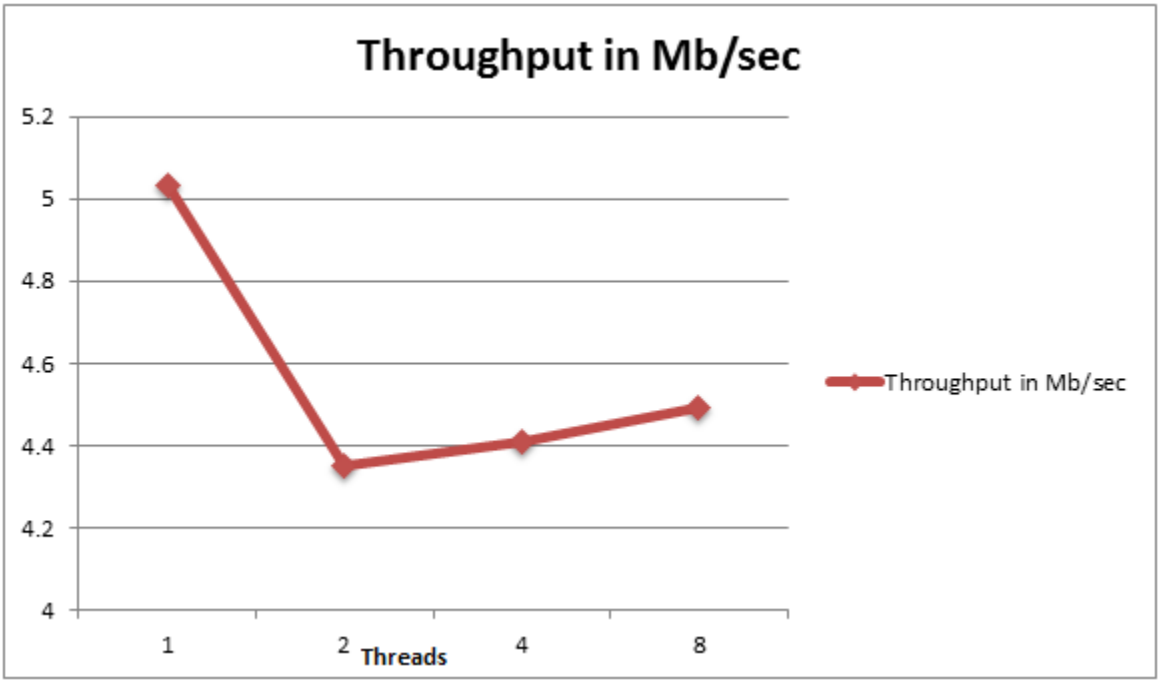
In a multithreaded process on a single processor, the processor can switch execution resources between threads, resulting in concurrent execution.

- Best performance was at thread 1. Thread 2, 4 and 8 were very close in terms of execution time.



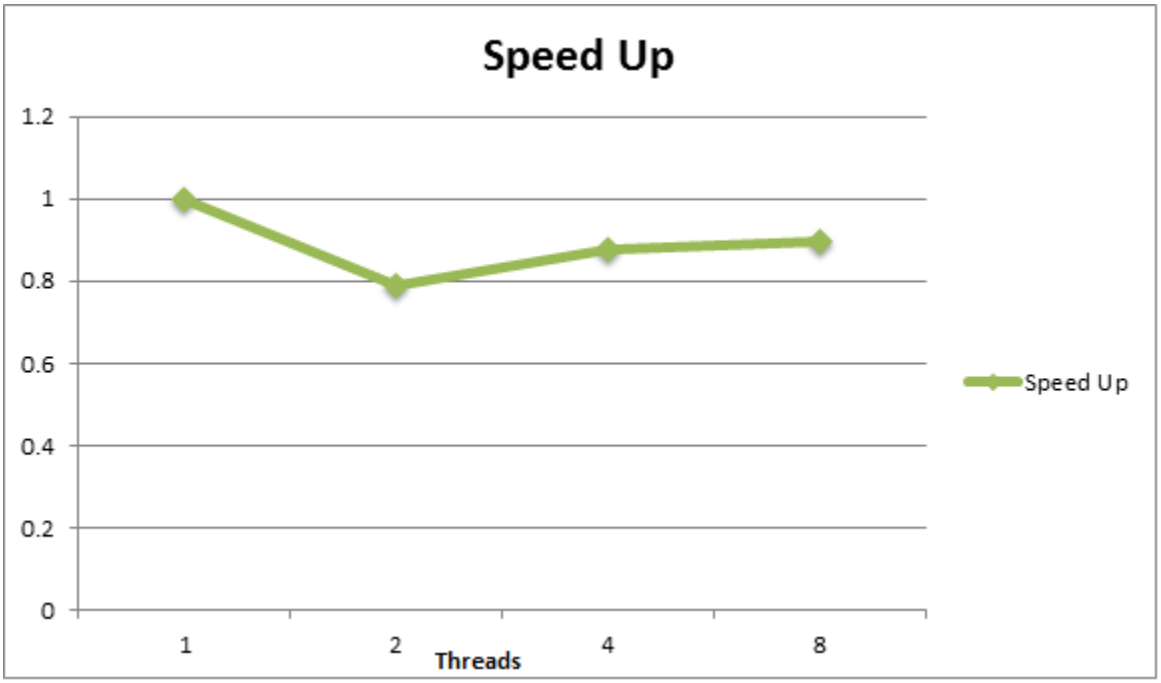
100GB Throughput:

Number of Threads	Execution Time in Seconds	Throughput in Mb/sec
1	19861.547	5.0349
2	22979.632	4.3517
6	22681.487	4.4089
8	22261.121	4.4921



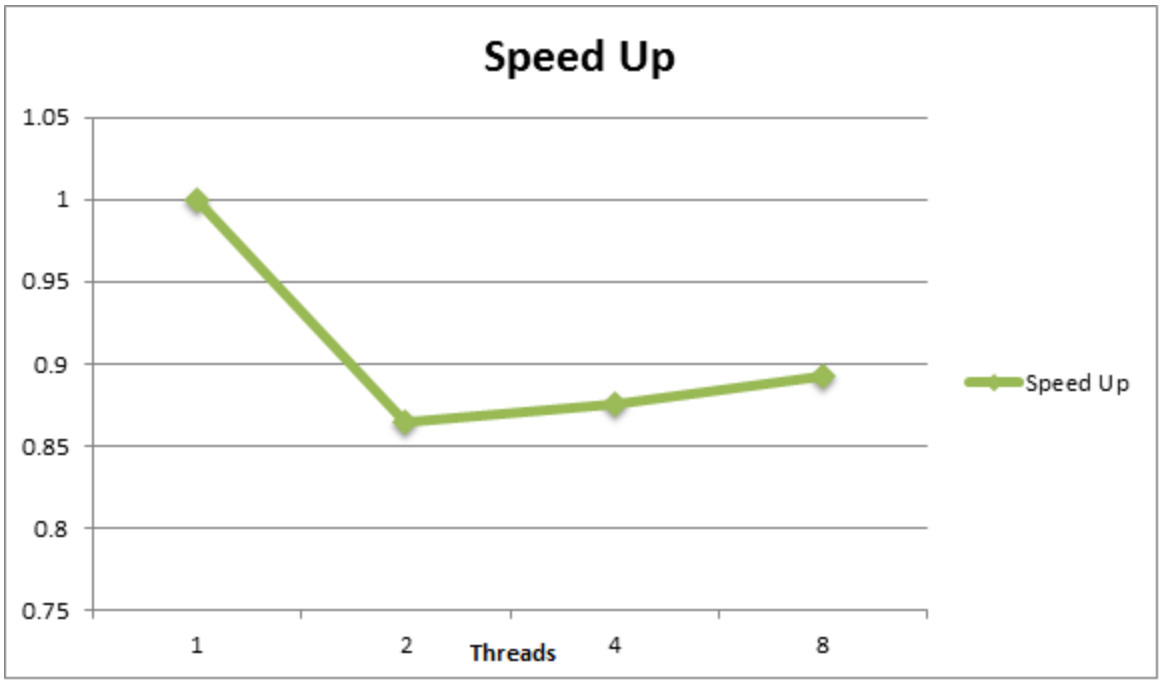
Speed Up data for 1 Node on 10GB dataset

Number of threads	Execution Time in sec	Execution Time in sec for 1 Node	Speed Up
1	2618.552	2618.552	1
2	3313.751	2618.552	0.7902
4	2989.987	2618.552	0.8757
8	2917.654	2618.552	0.8974

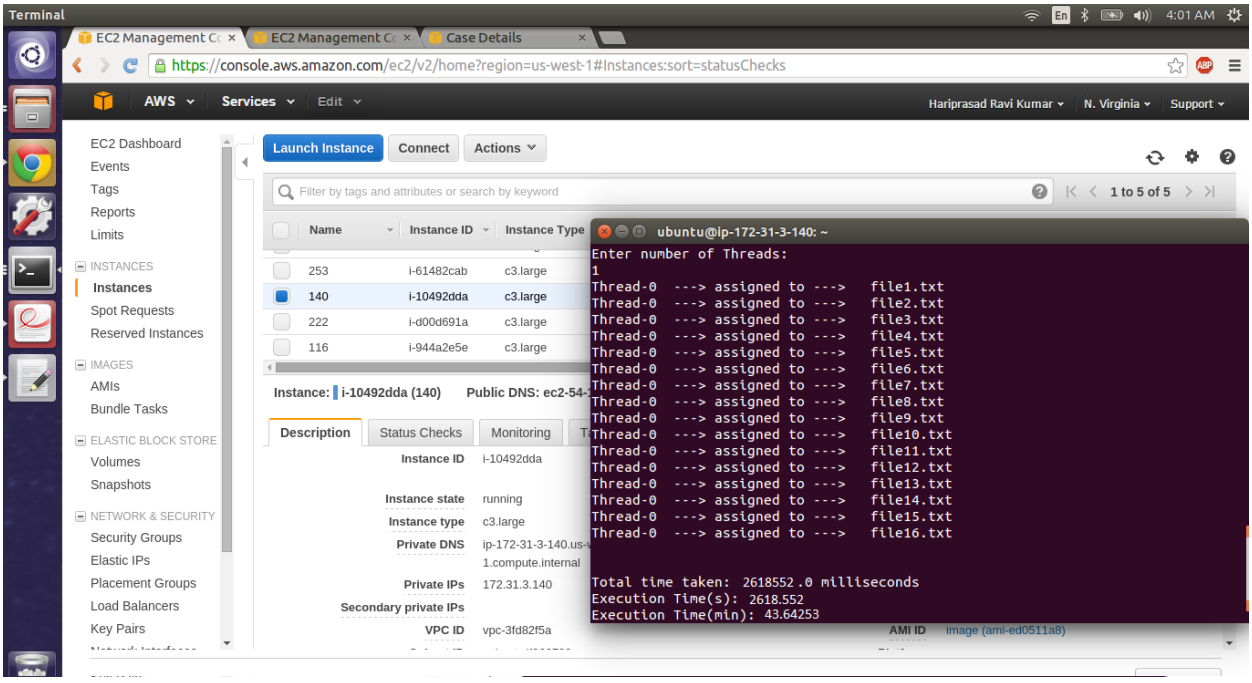


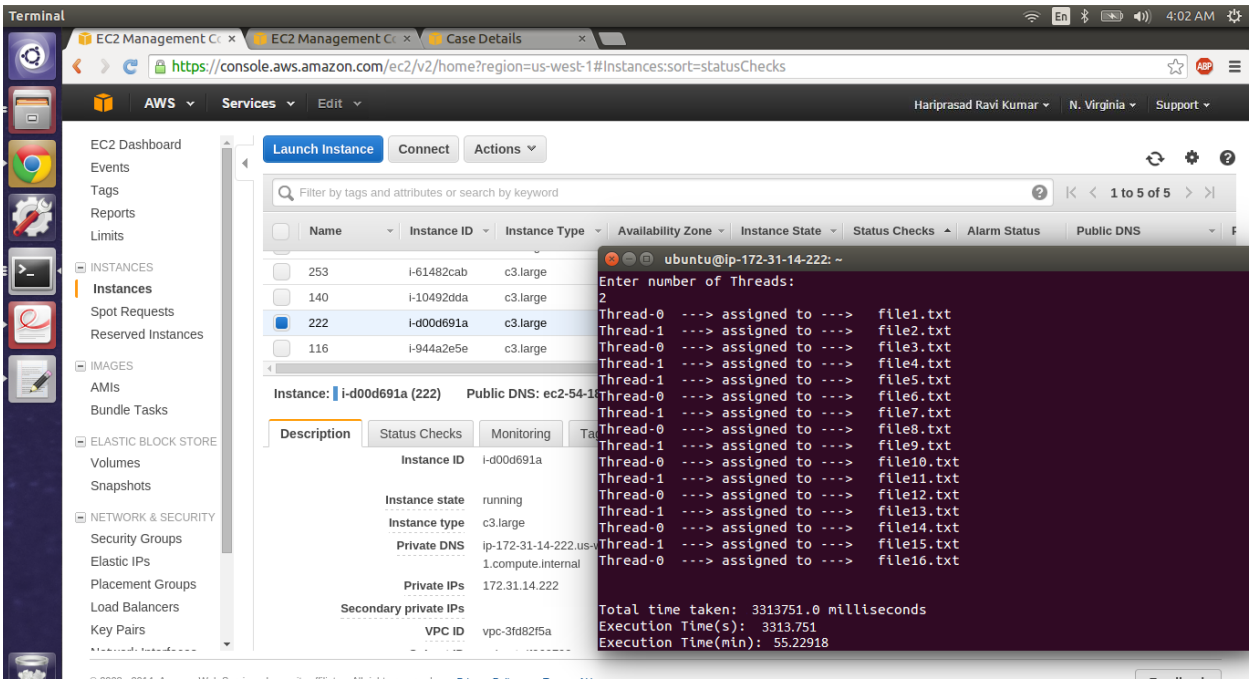
Speed Up data for 1 Node on 100GB dataset

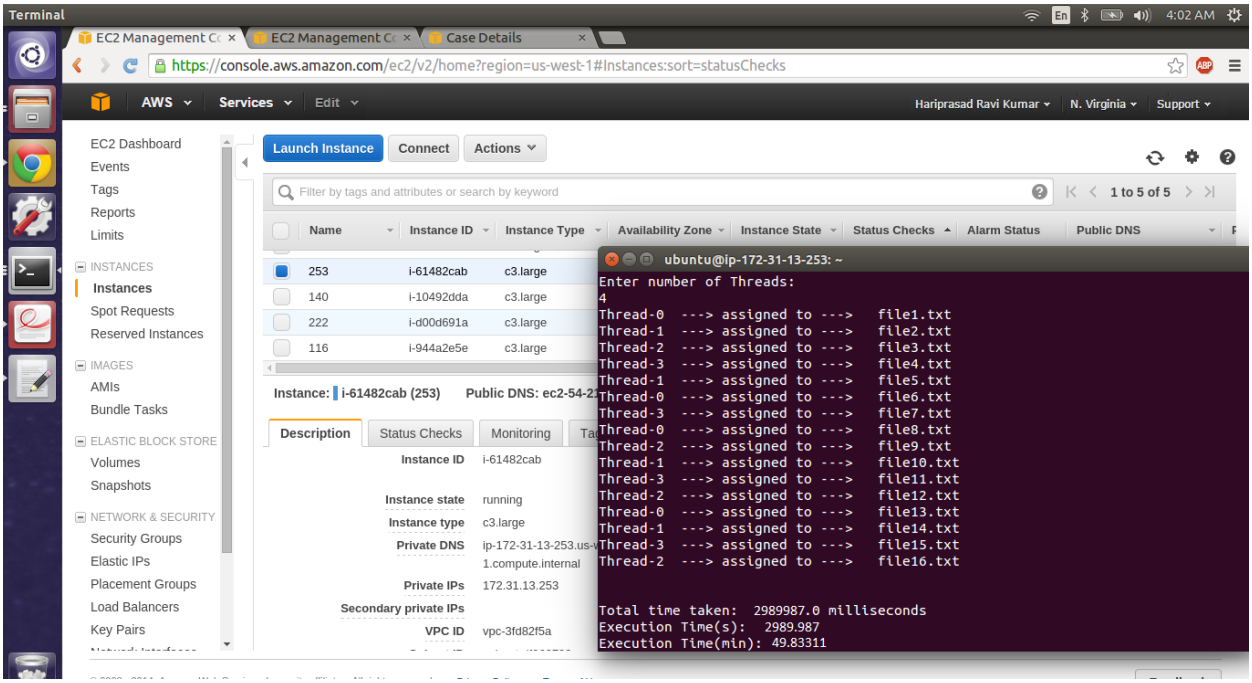
Number of threads	Execution Time in sec	Execution Time in sec for 1 Node	Speed Up
1	19861.547	19861.547	1
2	22979.632	19861.547	0.8643
4	22681.487	19861.547	0.8756
8	22261.121	19861.547	0.8921

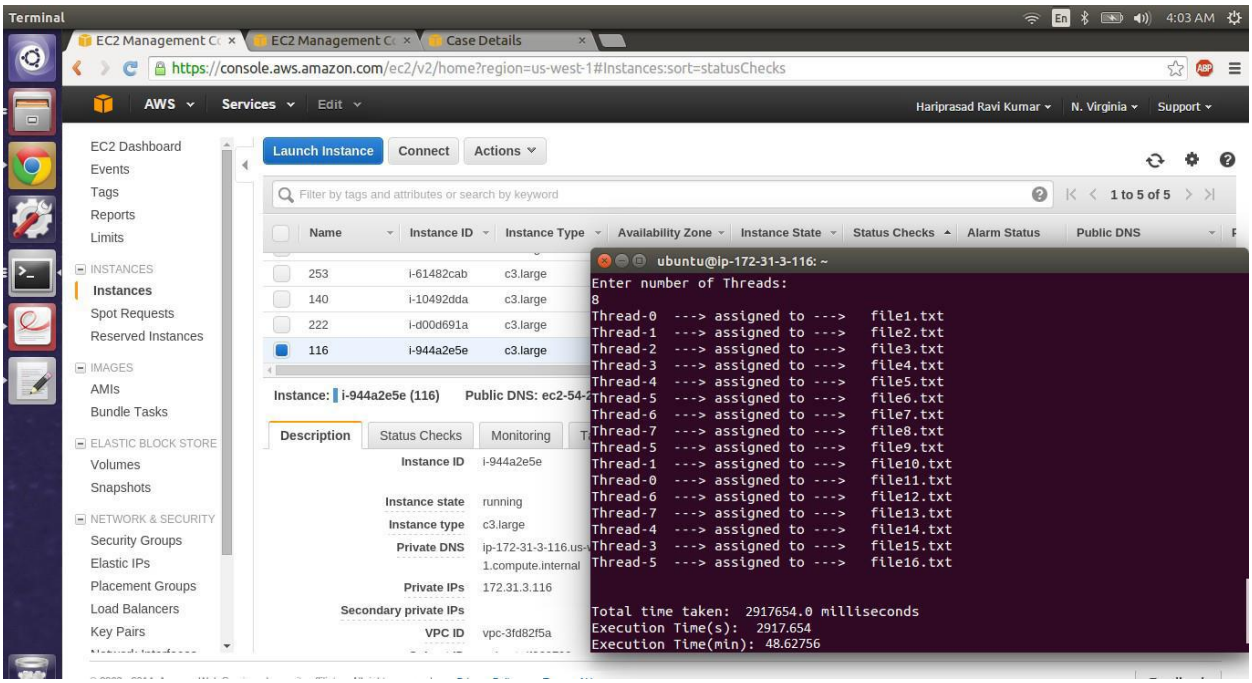


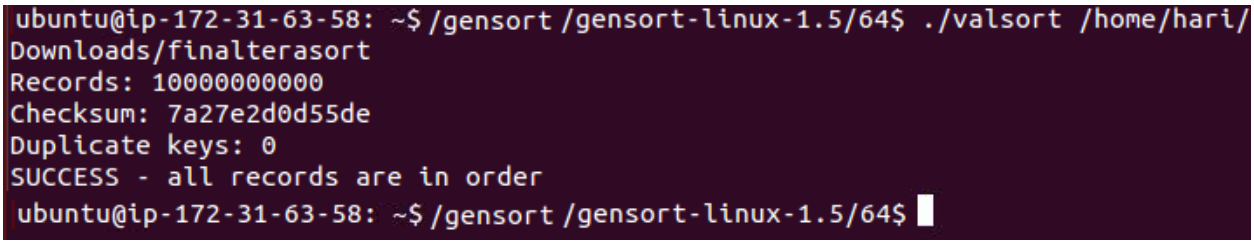
Output Screens Shots for 10GB dataset:



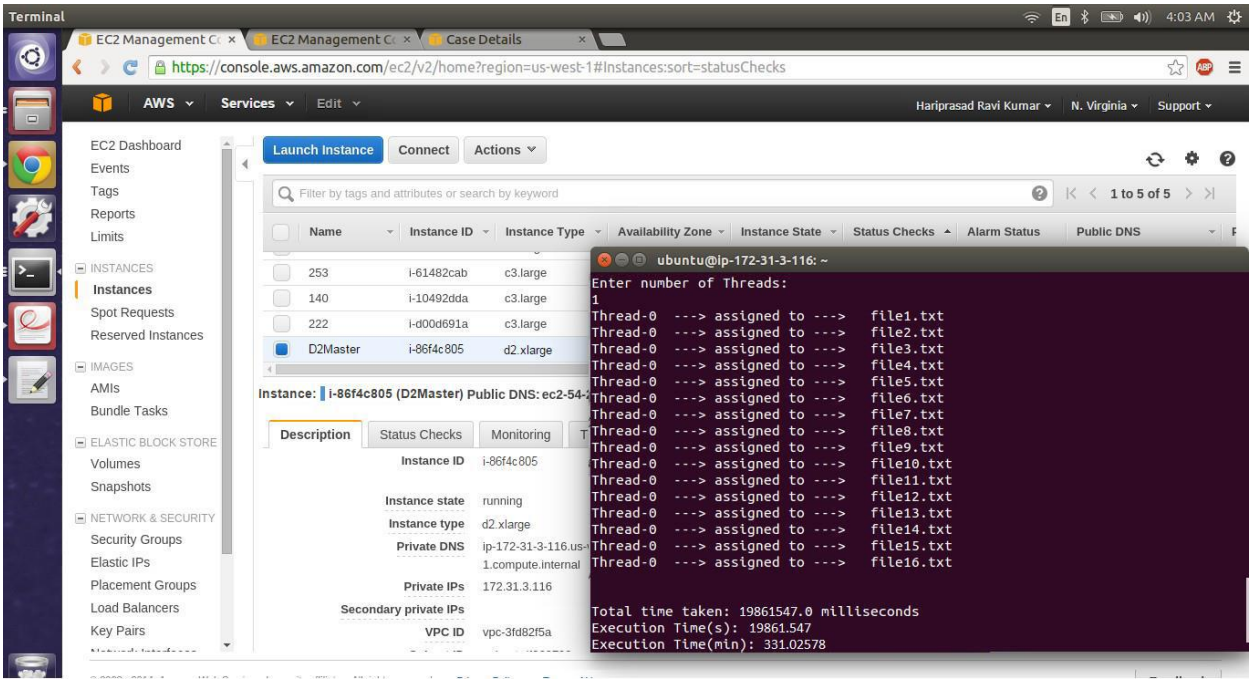

Terminal
4:02 AM
EC2 Management C
EC2 Management C
Case Details
https://console.aws.amazon.com/ec2/v2/home?region=us-west-1#Instances:sort=statusChecks
AWS
Services
Edit
Hariprasad Ravi Kumar
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 5 of 5
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
253 i-61482cab c3.large
140 i-10492dda c3.large
222 i-d00d691a c3.large
116 i-944a2e5e c3.large
Instance: i-d00d691a (222)
Public DNS: ec2-54-1
Description
Status Checks
Monitoring
Instance ID i-d00d691a
Instance state running
Instance type c3.large
Private DNS ip-172-31-14-222.us-
1.compute.internal
Private IPs 172.31.14.222
Secondary private IPs
VPC ID vpc-3fd82f5a
ubuntu@ip-172-31-14-222: ~
Enter number of Threads:
2
Thread-0 ---> assigned to ---> file1.txt
Thread-1 ---> assigned to ---> file2.txt
Thread-0 ---> assigned to ---> file3.txt
Thread-1 ---> assigned to ---> file4.txt
Thread-1 ---> assigned to ---> file5.txt
Thread-0 ---> assigned to ---> file6.txt
Thread-1 ---> assigned to ---> file7.txt
Thread-0 ---> assigned to ---> file8.txt
Thread-1 ---> assigned to ---> file9.txt
Thread-0 ---> assigned to ---> file10.txt
Thread-1 ---> assigned to ---> file11.txt
Thread-0 ---> assigned to ---> file12.txt
Thread-1 ---> assigned to ---> file13.txt
Thread-0 ---> assigned to ---> file14.txt
Thread-1 ---> assigned to ---> file15.txt
Thread-0 ---> assigned to ---> file16.txt
Total time taken: 3313751.0 milliseconds
Execution Time(s): 3313.751
Execution Time(min): 55.22918

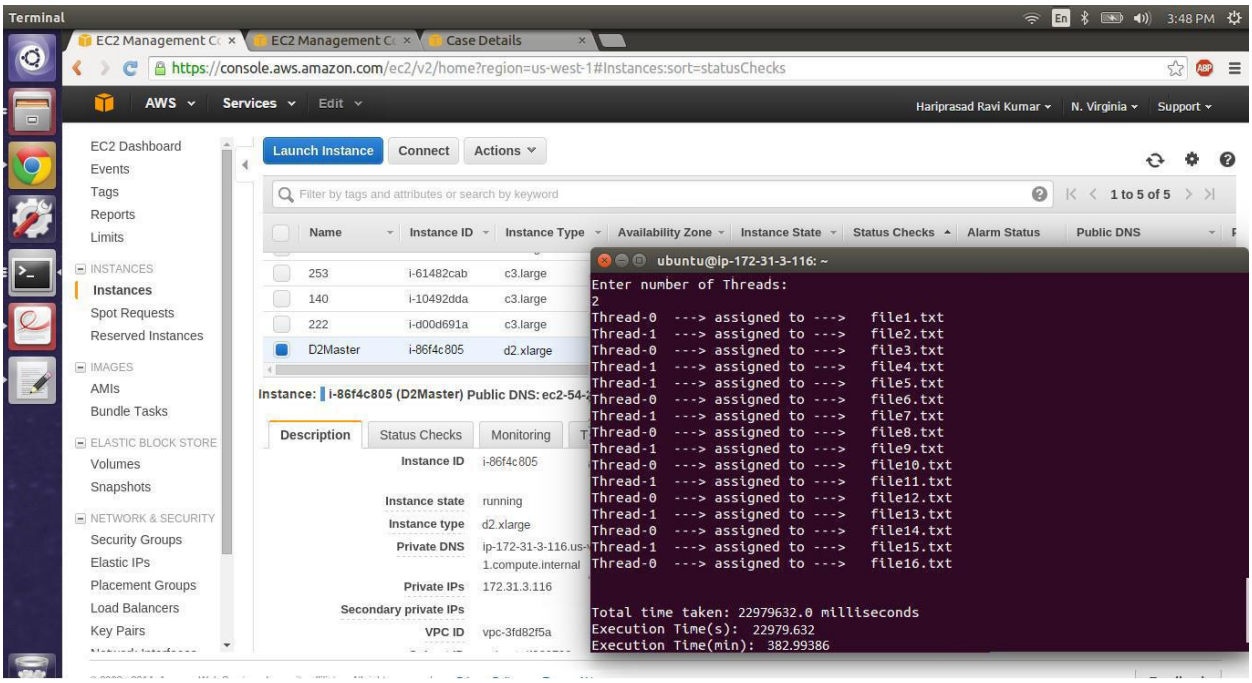

Terminal
4:02 AM
EC2 Management C
EC2 Management C
Case Details
https://console.aws.amazon.com/ec2/v2/home?region=us-west-1#Instances:sort=statusChecks
AWS
Services
Edit
Hariprasad Ravi Kumar
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 5 of 5
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
253 i-61482cab c3.large
140 i-10492dda c3.large
222 i-d00d691a c3.large
116 i-944a2e5e c3.large
Instance: i-61482cab (253)
Public DNS: ec2-54-2
Description
Status Checks
Monitoring
Instance ID i-61482cab
Instance state running
Instance type c3.large
Private DNS ip-172-31-13-253.us-
1.compute.internal
Private IPs 172.31.13.253
Secondary private IPs
VPC ID vpc-3fd82f5a
ubuntu@ip-172-31-13-253: ~
Enter number of Threads:
4
Thread-0 ---> assigned to ---> file1.txt
Thread-1 ---> assigned to ---> file2.txt
Thread-2 ---> assigned to ---> file3.txt
Thread-3 ---> assigned to ---> file4.txt
Thread-1 ---> assigned to ---> file5.txt
Thread-0 ---> assigned to ---> file6.txt
Thread-3 ---> assigned to ---> file7.txt
Thread-0 ---> assigned to ---> file8.txt
Thread-2 ---> assigned to ---> file9.txt
Thread-1 ---> assigned to ---> file10.txt
Thread-3 ---> assigned to ---> file11.txt
Thread-2 ---> assigned to ---> file12.txt
Thread-0 ---> assigned to ---> file13.txt
Thread-1 ---> assigned to ---> file14.txt
Thread-3 ---> assigned to ---> file15.txt
Thread-2 ---> assigned to ---> file16.txt
Total time taken: 2989987.0 milliseconds
Execution Time(s): 2989.987
Execution Time(min): 49.83311

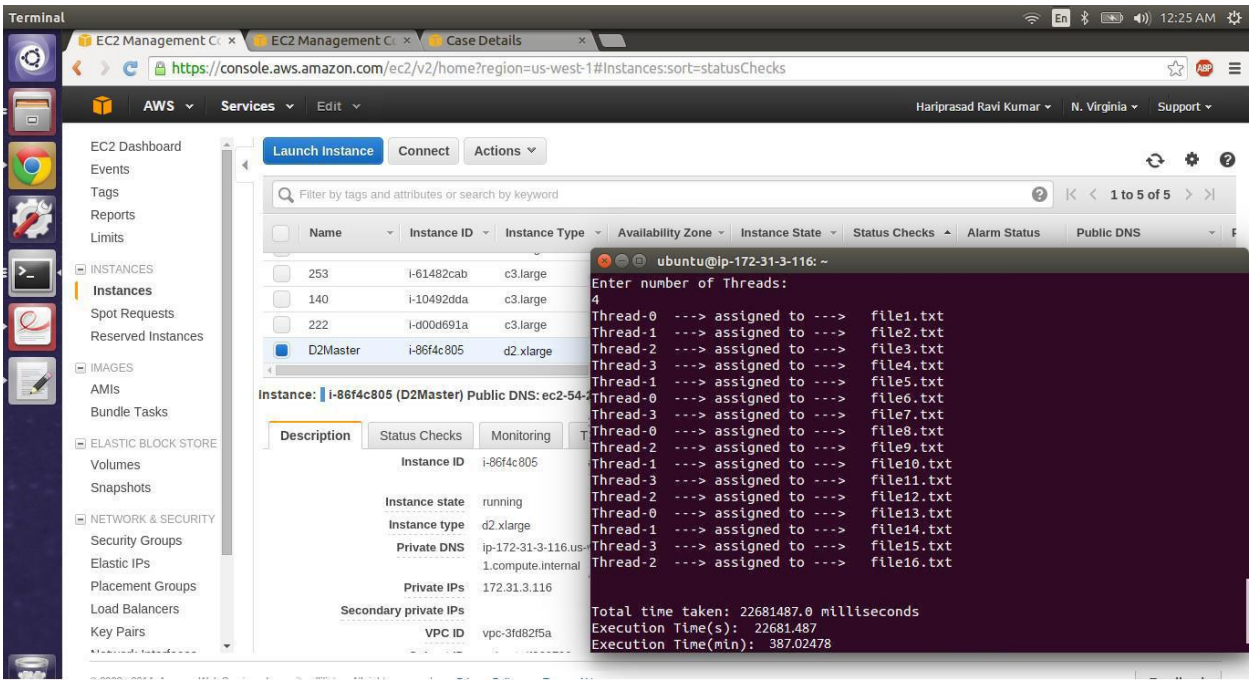


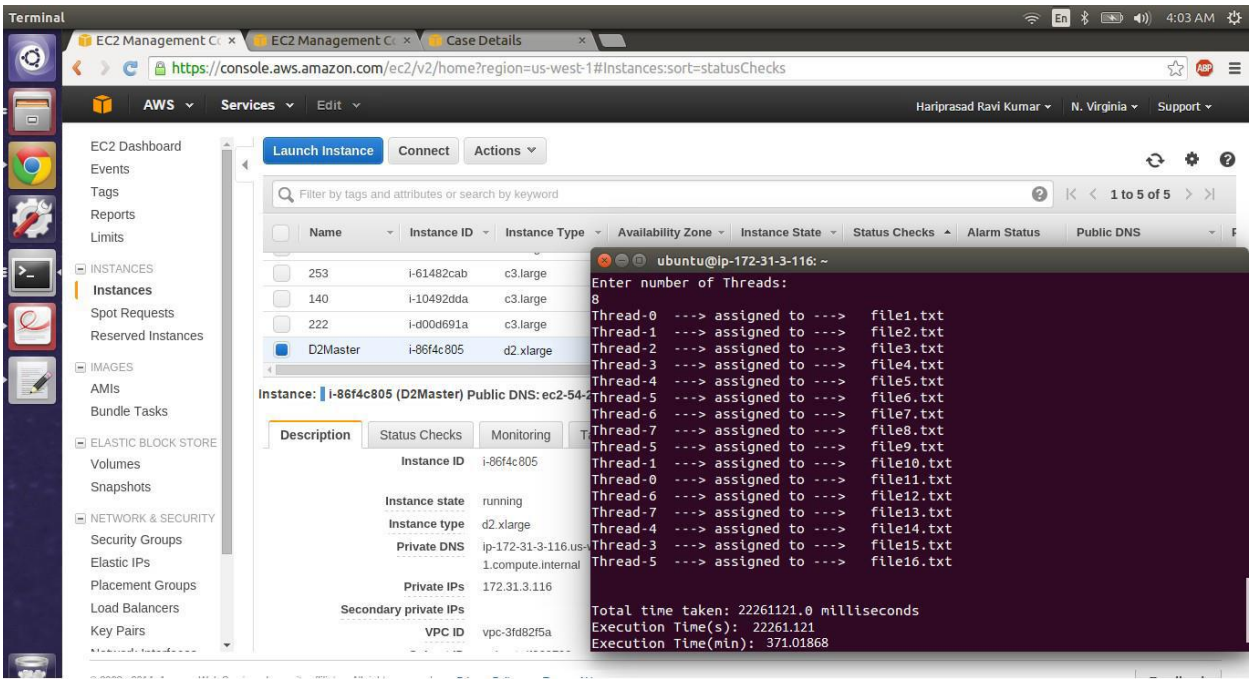


Output Screen Shots for 100GB dataset:




Terminal
3:48 PM
EC2 Management C
EC2 Management C
Case Details
https://console.aws.amazon.com/ec2/v2/home?region=us-west-1#Instances:sort=statusChecks
AWS
Services
Edit
Hariprasad Ravi Kumar
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 5 of 5
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
253 i-61482cab c3.large
140 i-10492dda c3.large
222 i-d00d691a c3.large
D2Master i-86f4c805 d2.xlarge
Instance: i-86f4c805 (D2Master) Public DNS: ec2-54-
Description
Status Checks
Monitoring
Instance ID i-86f4c805
Instance state running
Instance type d2.xlarge
Private DNS ip-172-31-3-116.us-
1.compute.internal
Private IPs 172.31.3.116
Secondary private IPs
VPC ID vpc-3fd82f5a
ubuntu@ip-172-31-3-116: ~
Enter number of Threads:
2
Thread-0 ---> assigned to ---> file1.txt
Thread-1 ---> assigned to ---> file2.txt
Thread-0 ---> assigned to ---> file3.txt
Thread-1 ---> assigned to ---> file4.txt
Thread-1 ---> assigned to ---> file5.txt
Thread-0 ---> assigned to ---> file6.txt
Thread-1 ---> assigned to ---> file7.txt
Thread-0 ---> assigned to ---> file8.txt
Thread-1 ---> assigned to ---> file9.txt
Thread-0 ---> assigned to ---> file10.txt
Thread-1 ---> assigned to ---> file11.txt
Thread-0 ---> assigned to ---> file12.txt
Thread-1 ---> assigned to ---> file13.txt
Thread-0 ---> assigned to ---> file14.txt
Thread-1 ---> assigned to ---> file15.txt
Thread-0 ---> assigned to ---> file16.txt
Total time taken: 22979632.0 milliseconds
Execution Time(s): 22979.632
Execution Time(min): 382.99386


Terminal
12:25 AM
EC2 Management C
EC2 Management C
Case Details
https://console.aws.amazon.com/ec2/v2/home?region=us-west-1#Instances:sort=statusChecks
AWS
Services
Edit
Hariprasad Ravi Kumar
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 5 of 5
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
253 i-61482cab c3.large
140 i-10492dda c3.large
222 i-d00d691a c3.large
D2Master i-86f4c805 d2.xlarge
Instance: i-86f4c805 (D2Master) Public DNS: ec2-54-
Description
Status Checks
Monitoring
Instance ID i-86f4c805
Instance state running
Instance type d2.xlarge
Private DNS ip-172-31-3-116.us-
1.compute.internal
Private IPs 172.31.3.116
Secondary private IPs
VPC ID vpc-3fd82f5a
ubuntu@ip-172-31-3-116: ~
Enter number of Threads:
4
Thread-0 ---> assigned to ---> file1.txt
Thread-1 ---> assigned to ---> file2.txt
Thread-2 ---> assigned to ---> file3.txt
Thread-3 ---> assigned to ---> file4.txt
Thread-1 ---> assigned to ---> file5.txt
Thread-0 ---> assigned to ---> file6.txt
Thread-3 ---> assigned to ---> file7.txt
Thread-0 ---> assigned to ---> file8.txt
Thread-2 ---> assigned to ---> file9.txt
Thread-1 ---> assigned to ---> file10.txt
Thread-3 ---> assigned to ---> file11.txt
Thread-2 ---> assigned to ---> file12.txt
Thread-0 ---> assigned to ---> file13.txt
Thread-1 ---> assigned to ---> file14.txt
Thread-3 ---> assigned to ---> file15.txt
Thread-2 ---> assigned to ---> file16.txt
Total time taken: 22681487.0 milliseconds
Execution Time(s): 22681.487
Execution Time(min): 387.02478




```
ubuntu@ip-172-31-63-58: ~$ /gensort /gensort-linux-1.5/64$ ./valsort /home/hari/
Downloads/finalterasort100
Records: 100000000000
Checksum: 7a27e2d0d55de
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-63-58: ~$ /gensort /gensort-linux-1.5/64$
```

Best Performance:

Shared memory sort for 10GB: 1 Thread

Shared memory sort for 100GB: 1 Thread

Setting Up a Virtual Cluster of 16 Nodes + 1 Master:

Creating an instance in Amazon(AWS):

- Log in into the Amazon AWS service website (http://aws.amazon.com/).
- Select EC2 from the network and services section.
- Select launch instance. Then configure the service and chose the best AMI to meet the requirements.
- Then choose the instance type which can be micro, small, medium, large Xlarge, 2Xlarge, 4Xlarge or 8Xlarge. For our assignment it is c3.large
- Then we configure the instance i.e we can choose the instance to be either spot instance or on Demand instance.
- Add storage to our selected instance.
- Configure the security group. We add TCP, UDP and ICMP protocols from the security group. We then create the instance. We access the instance by connecting to the instance by suing the public IP address of it and later on we can configure the settings of the AMI that we are using.

To Connect to the Node:

chmod 400 keygiven (This is to change the permission to be read by owner.)

ssh –i key IP address

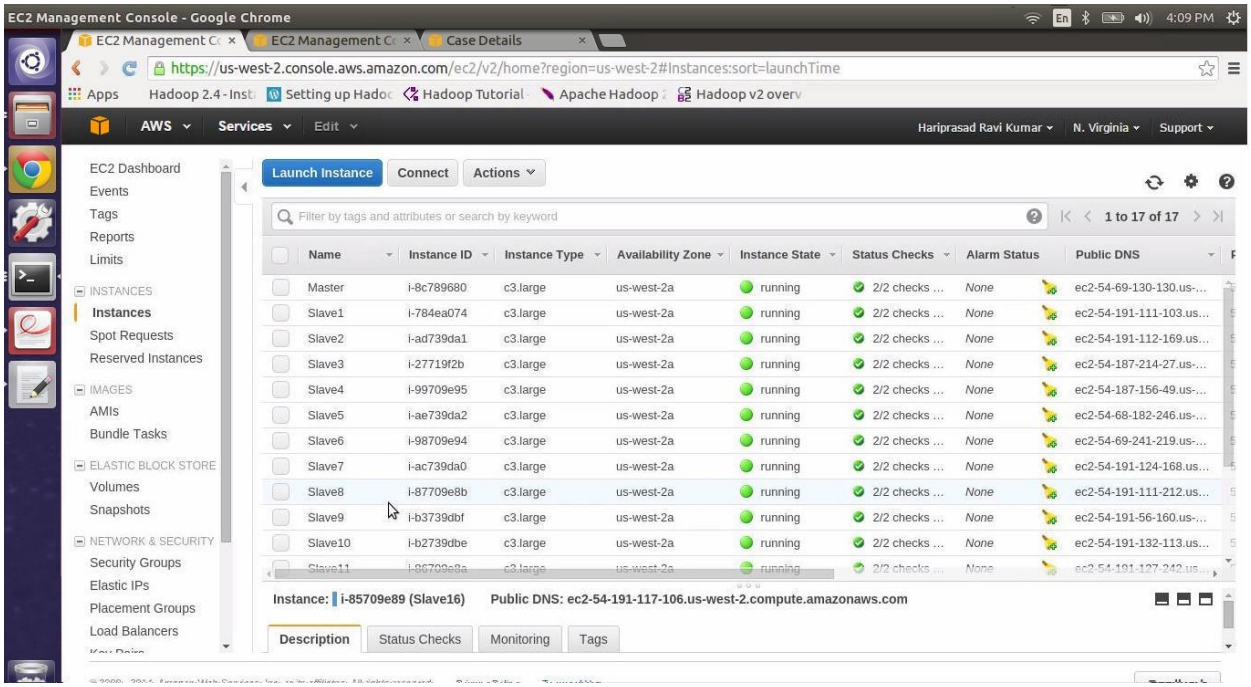
In our case we use:

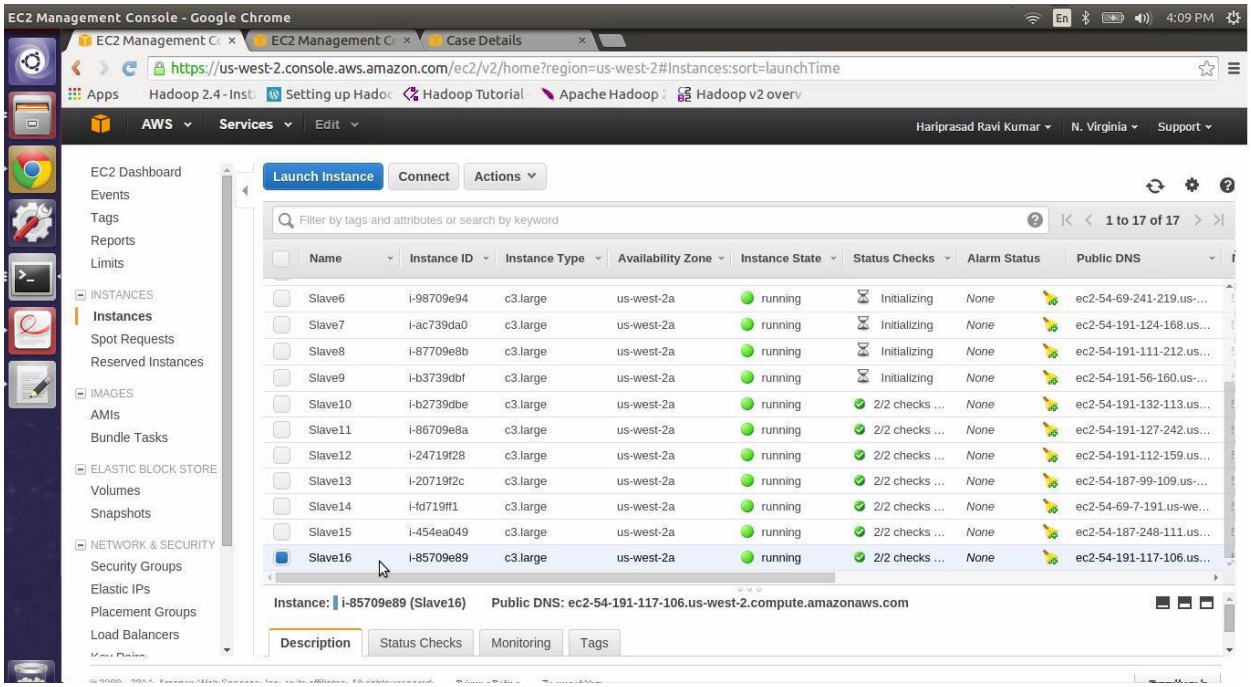
1. chmod 400 newkey.pem
2. ssh –i newkey.pem Ubuntu@54.69.130.130

After the connection is established:

- After getting connected to the AMI we now have to install java on the virtual node.
- Install java by using sudo apt-get install openjdk-7-jreheadless.
- Install javac by using sudo apt-get install openjdk-7-jdk.
- Now update all the installed applications by using sudo apt-get update
- Now check the java version by using java –version.
- Now right click the instance we created and select “create image” option. This will create an image for what all we did till now and reflect the same in the other instances.
- Go to the images tab in the left panel of the AWS webpage and select AMI images. You will see the name of the image we just created. Select that and launch the other 16 nodes.

Screen Shots of 17 nodes (1 Master and the other 16 are slaves):





Hadoop:

Installation:

Creating an instance in Amazon(AWS):

- Log in into the Amazon AWS service website (http://aws.amazon.com/).
- Select EC2 from the network and services section.
- Select launch instance. Then configure the service and chose the best AMI to meet the requirements.
- Then choose the instance type which can be micro, small, medium, large Xlarge, 2Xlarge, 4Xlarge or 8Xlarge. For our assignment it is c3.large
- Then we configure the instance i.e we can choose the instance to be either spot instance or on Demand instance.
- Add storage to our selected instance.
- Configure the security group. We add TCP, UDP and ICMP protocols from the security group. We then create the instance. We access the instance by connecting to the instance by suing the public IP address of it and later on we can configure the settings of the AMI that we are using.

To Connect to the Node:

chmod 400 keygiven (This is to change the permission to be read by owner)

ssh –i key IP address

In our case we use:

3. chmod 400 newkey1.pem

4. ssh –i newkey1.pem Ubuntu@54.69.130.130

After the connection is established:

- After getting connected to the AMI we now have to install java on the virtual node.
- Hadoop Frame Work is written in java, so we need to install java on all the nodes.
- Install java by using sudo apt-get install openjdk-7-jreheadless.
- Install javac by using sudo apt-get install openjdk-7-jdk.
- Now update all the installed applications by using sudo apt-get update
- Now check the java version by using java –version.

To Install Hadoop:

- In the terminal do a wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz
- Unzip it Using "tar xvzf hadoop-2.7.1.tar.gz"
- Create a folder installs in hadoop and then move the contents into this folder.

The following files will have to be modified to complete the Hadoop setup:

- ~/.bashrc
- / conf /hadoop-env.sh
- / conf /core-site.xml
- / conf /yarn-site.xml
- / conf /hdfs-site.xml
- / conf /mapred-site.xml

1. ~/.bashrc:

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/ubuntu/installs/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_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=$HADOOP_INSTALL/lib/native
eval ($ssh-agent);
ssh-add/home/ubuntu/newkey1.pem
#HADOOP VARIABLES END
```

Reboot the connection now for the changes to be reflected.

2. / conf /hadoop-env.sh:

This is to set the JAVA Path by modifying hadoop-env.sh file

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

3. /conf/core-site.xml:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up and can be used to override the default settings.

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://<172.31.22.90>:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/ubuntu/installs/hadoop/hadoop_tmp/tmp</value>
</property>
</configuration>
```

4. /conf/yarn-site.xml:

```
<configuration>
<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><172.31.22.90>:8025</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value><172.31.22.90>:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value><172.31.22.90>:8040</value>
</property>
</configuration>
```

5. /conf/hdfs-site.xml:

<configuration>
<property>
<name>dfs.replication</name>
<value>16</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
</configuration>

6. /conf /mapred-site.xml:

By default, the /usr/local/hadoop/etc/hadoop/ contains the /usr/local/hadoop/etc/hadoop/ mapred-site.xml.template file which has to be renamed/copied with the name mapred-site.xml:

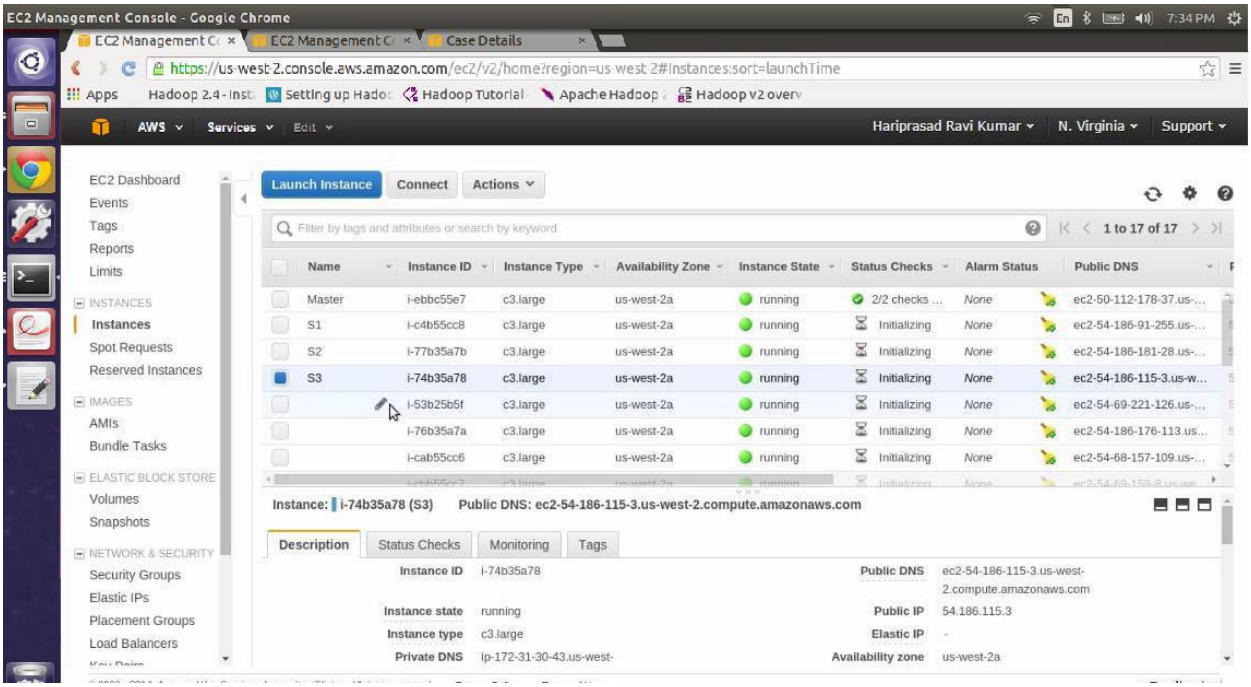
"cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template/usr/local/hadoop/etc/hadoop/mapred-site.xml"

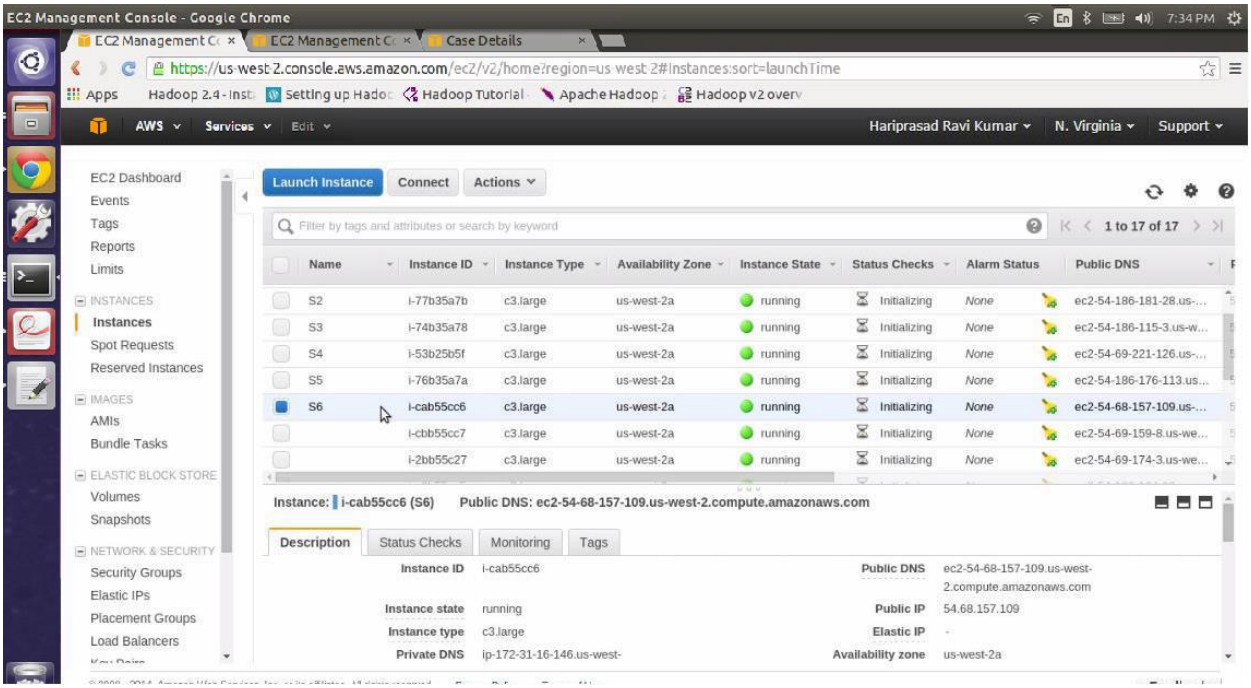
The mapred-site.xml file is used to specify which framework is being used for MapReduce. We need to enter the following content in between the <configuration></configuration> tag:

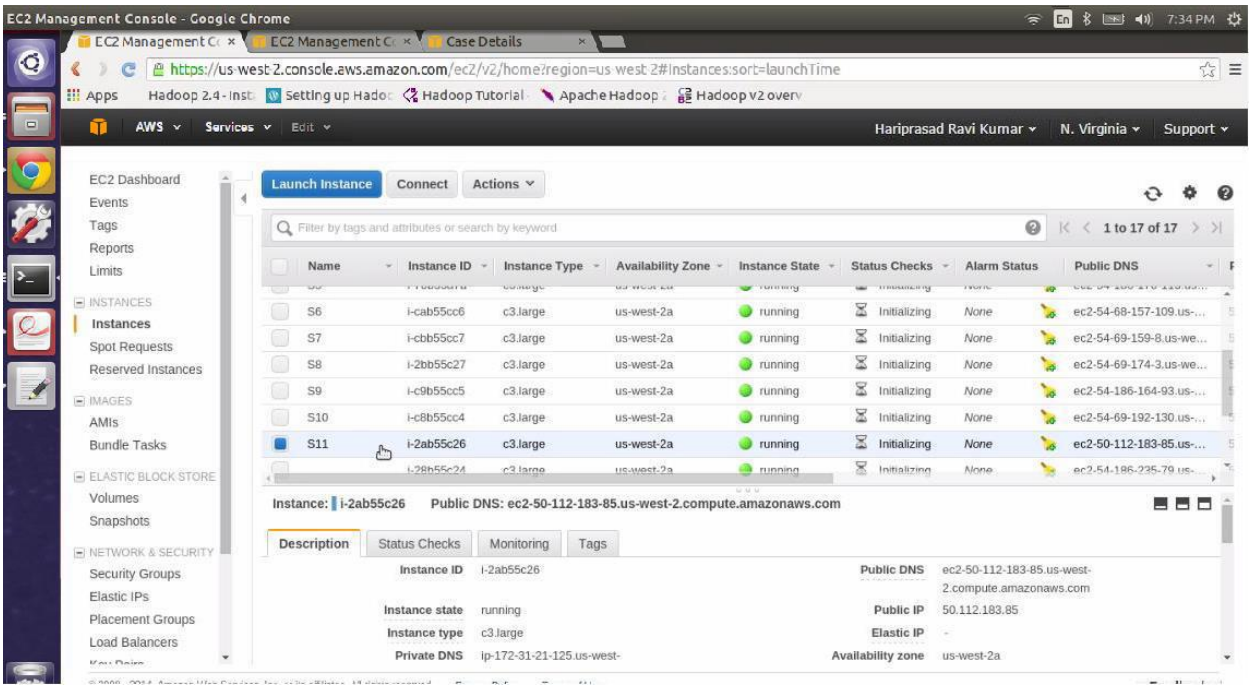
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>

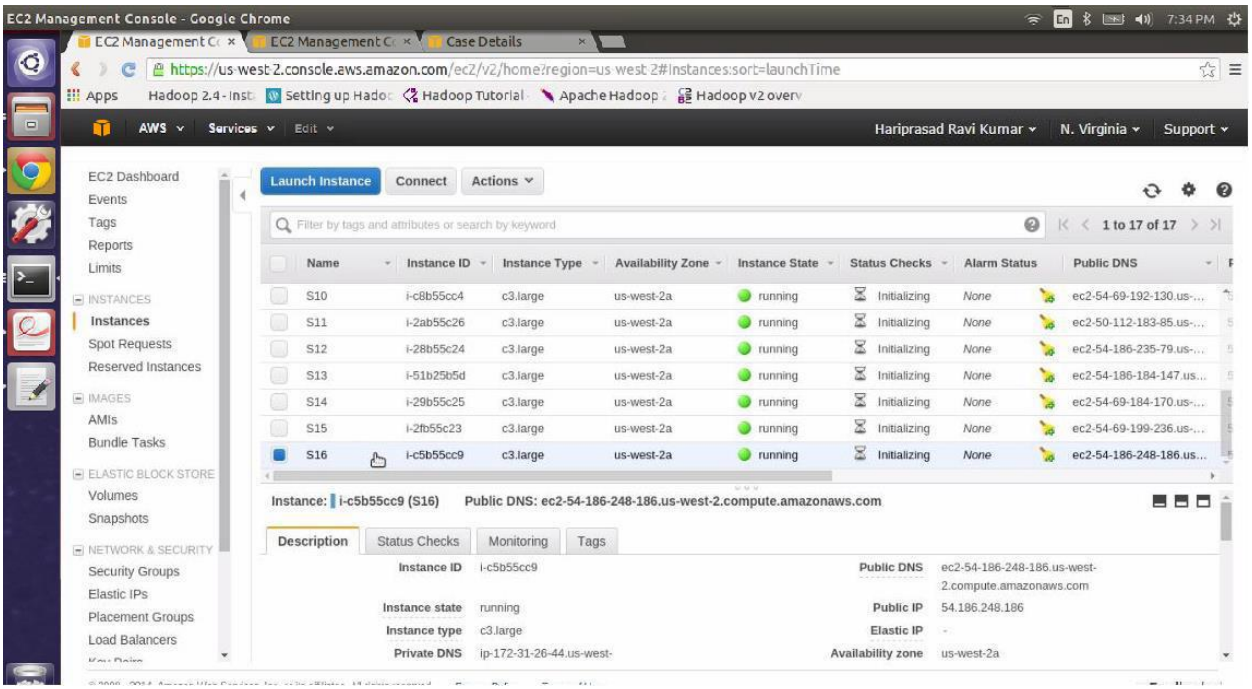
Create an IMAGE for all this so that it can be reflected in all the nodes. This would be the basic configuration for hadoop for 1 node setup. For 16 node setup we have to edit 2 more configuration files the conf/hosts, conf/slaves on the MASTER NODE ONLY.

Master and Slave Nodes Connected:



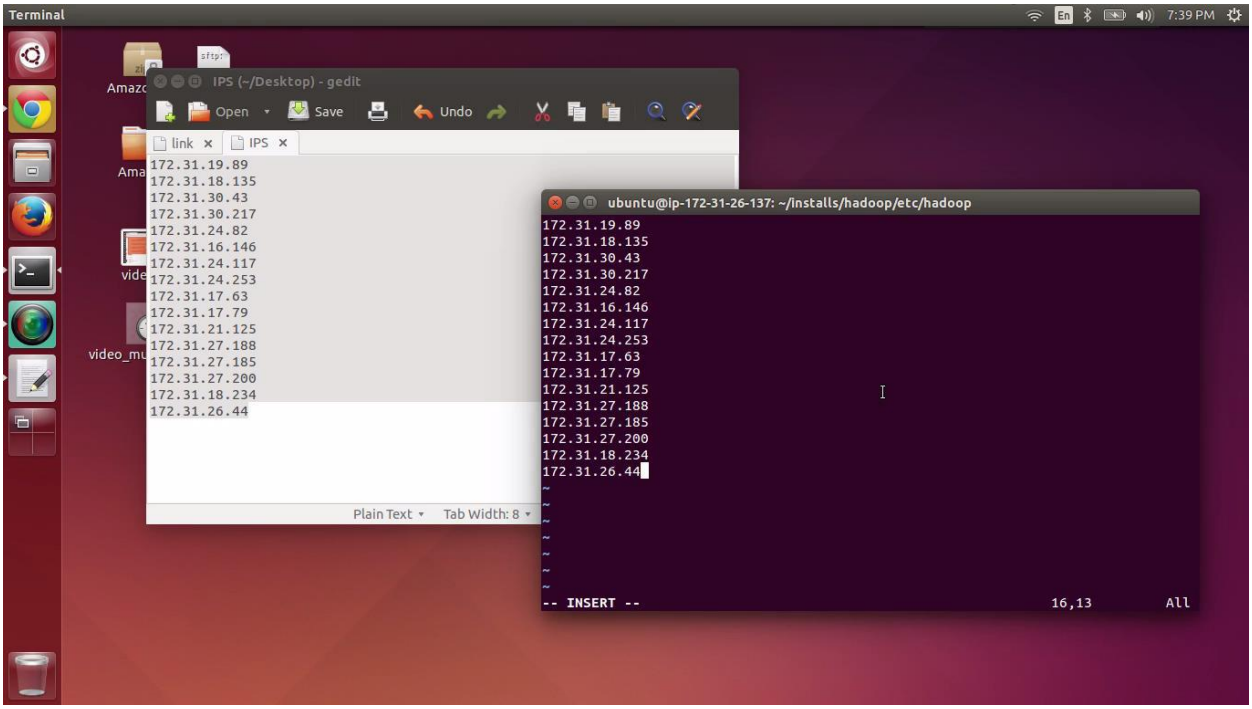



EC2 Management Console - Google Chrome
7:34 PM
EC2 Management Console
EC2 Management Console
Case Details
https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=launchTime
Apps
Hadoop 2.4 - Inst
Setting up Hadoo
Hadoop Tutorial
Apache Hadoop
Hadoop v2 overv
AWS
Services
Edit
Hariprasad Ravi Kumar
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 17 of 17
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
S6 i-cab55cc6 c3.large us-west-2a running Initializing None ec2-54-68-157-109.us-...
S7 i-cbb55cc7 c3.large us-west-2a running Initializing None ec2-54-69-159-8.us-we...
S8 i-2bb55c27 c3.large us-west-2a running Initializing None ec2-54-69-174-3.us-we...
S9 i-c9b55cc5 c3.large us-west-2a running Initializing None ec2-54-186-164-93.us-...
S10 i-c8b55cc4 c3.large us-west-2a running Initializing None ec2-54-69-192-130.us-...
S11 i-2ab55c26 c3.large us-west-2a running Initializing None ec2-50-112-183-85.us-...
Instance: i-2ab55c26 Public DNS: ec2-50-112-183-85.us-west-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID i-2ab55c26
Instance state running
Instance type c3.large
Private DNS ip-172-31-21-125.us-west-
Public DNS ec2-50-112-183-85.us-west-2.compute.amazonaws.com
Public IP 50.112.183.85
Elastic IP -
Availability zone us-west-2a


EC2 Management Console - Google Chrome
7:34 PM
EC2 Management Console
EC2 Management Console
Case Details
https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=launchTime
Apps
Hadoop 2.4 - Inst
Setting up Hadoo
Hadoop Tutorial
Apache Hadoop
Hadoop v2 overv
AWS
Services
Edit
Hariprasad Ravi Kumar
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 17 of 17
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
S10 i-c8b55cc4 c3.large us-west-2a running Initializing None ec2-54-69-192-130.us-...
S11 i-2ab55c26 c3.large us-west-2a running Initializing None ec2-50-112-183-85.us-...
S12 i-28b55c24 c3.large us-west-2a running Initializing None ec2-54-186-235-79.us-...
S13 i-51b25b5d c3.large us-west-2a running Initializing None ec2-54-186-184-147.us...
S14 i-29b55c25 c3.large us-west-2a running Initializing None ec2-54-69-184-170.us-...
S15 i-2fb55c23 c3.large us-west-2a running Initializing None ec2-54-69-199-236.us-...
S16 i-c5b55cc9 c3.large us-west-2a running Initializing None ec2-54-186-248-186.us...
Instance: i-c5b55cc9 (S16) Public DNS: ec2-54-186-248-186.us-west-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID i-c5b55cc9
Instance state running
Instance type c3.large
Private DNS ip-172-31-26-44.us-west-
Public DNS ec2-54-186-248-186.us-west-2.compute.amazonaws.com
Public IP 54.186.248.186
Elastic IP -
Availability zone us-west-2a

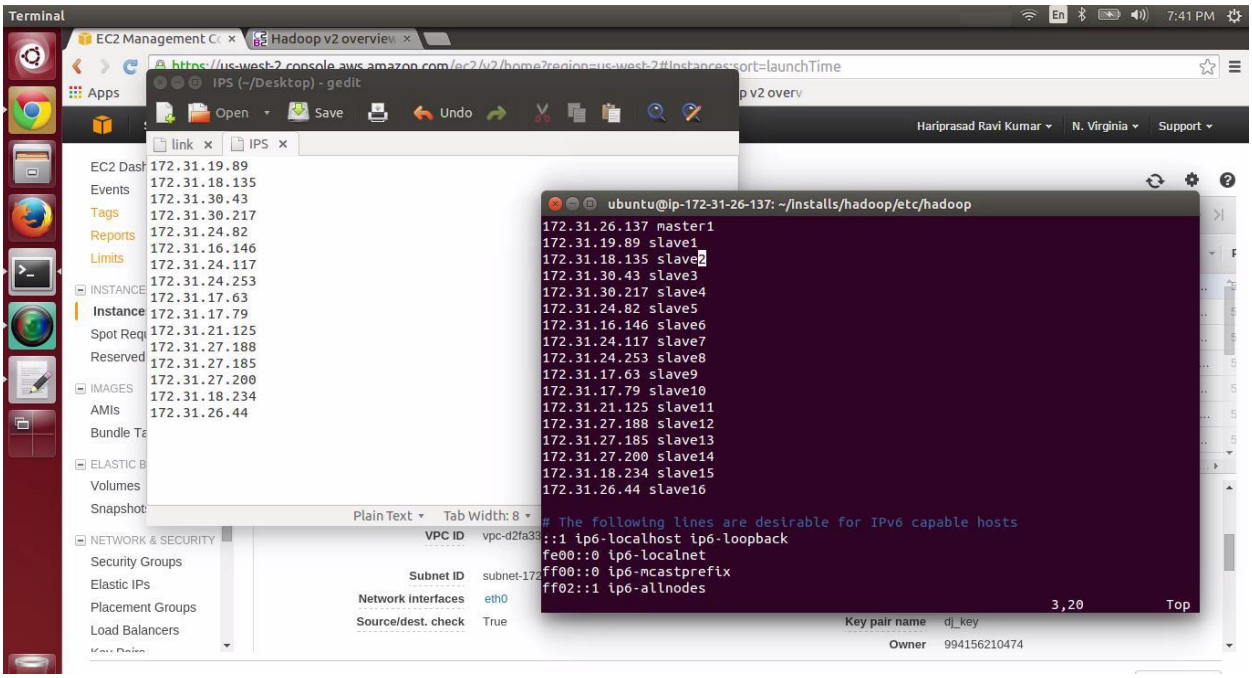
7. / conf /slaves:

Add all the IP address to the slaves file



8. conf/hosts:

Add all the IP's of the the slaves in the file including the master IP.



1. The namenode will run only on the master in our test setup and is started by executing the following command:

- $<HADOOP_HOME>/sbin/hadoop-daemon.sh start namenode

2. Start the datanodes which will run on both the slave nodes:

- $<HADOOP_HOME>/hadoop-daemons.sh start datanode

3. The resource manager process will run only on the master and is started using the following command:

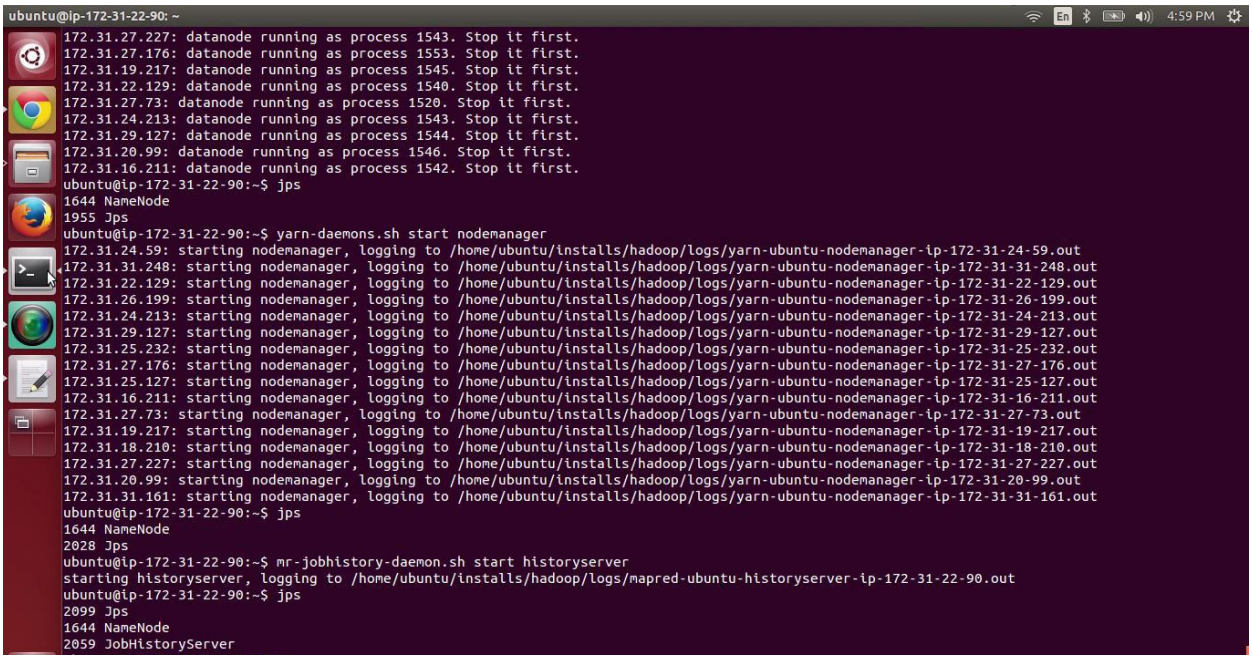
- $<HADOOP_HOME>/yarn-daemon.sh start resourcemanager

4. The node manager process runs on each of the slave nodes and is started using this command:

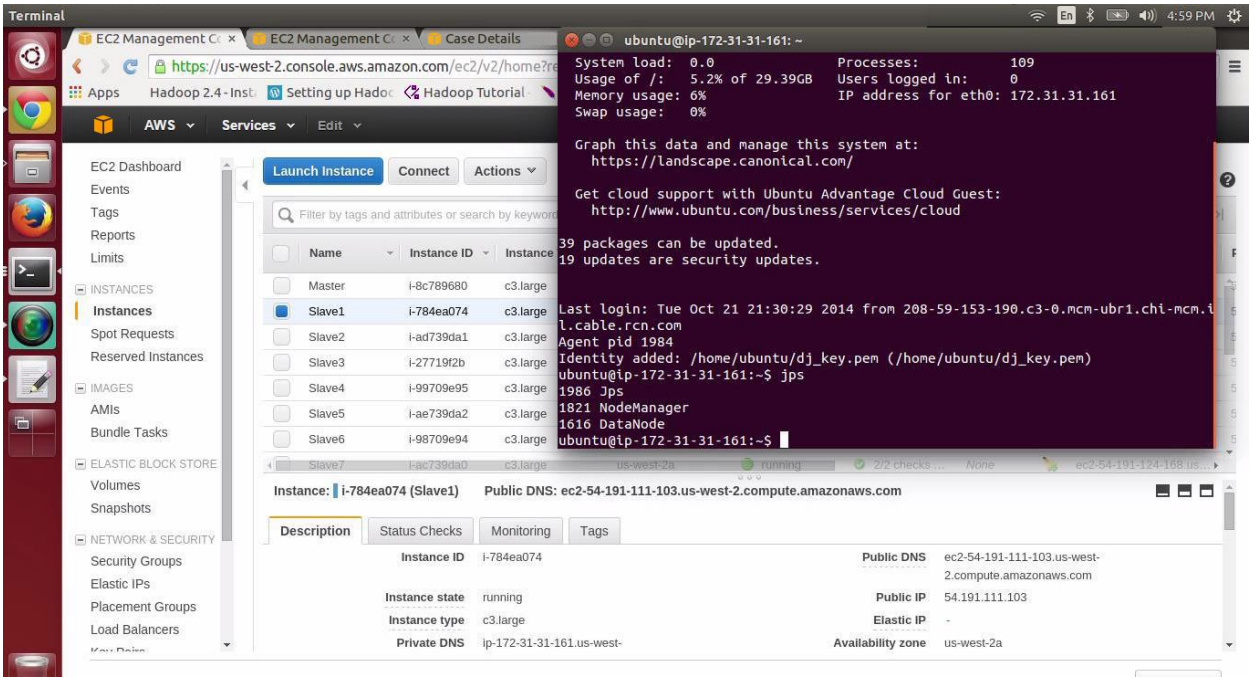
- $<HADOOP_HOME>/yarn-daemons.sh start nodemanager

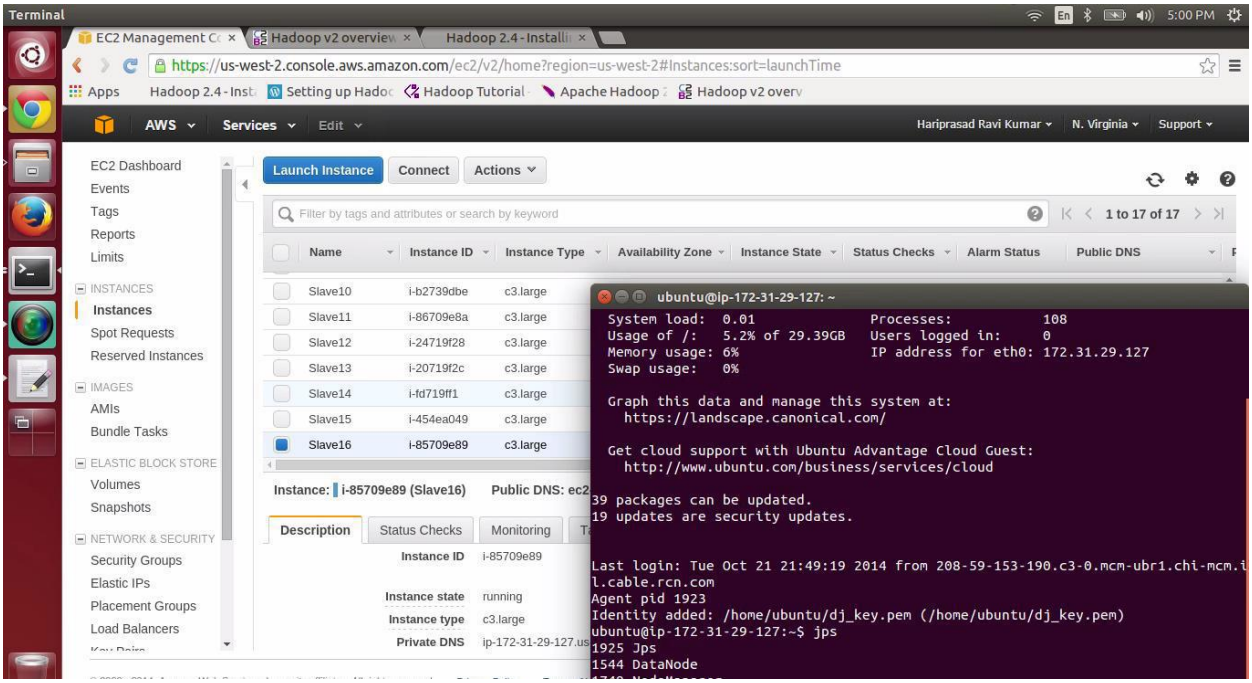
5. The job history server process runs on the master node and is started using the following command:

- $<HADOOP_HOME>/mr-jobhistory-daemon.sh start historyserver



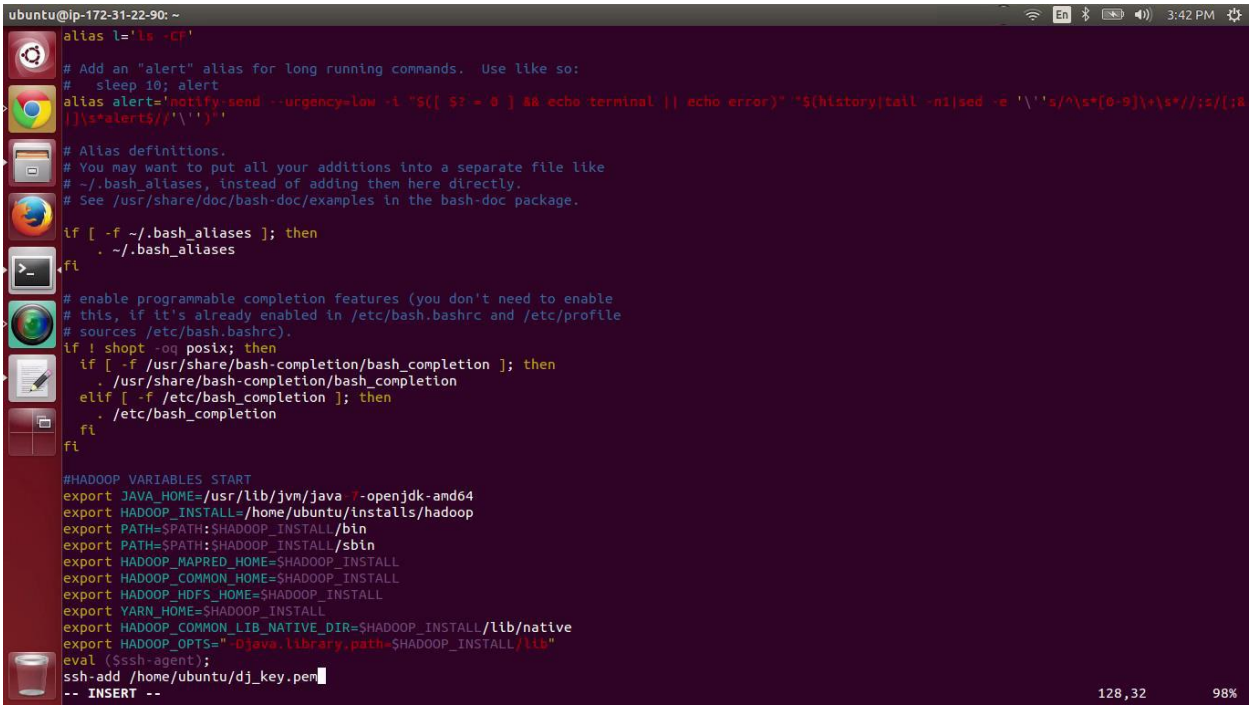
To Check if the nodes are connected we login into another node and do jps:



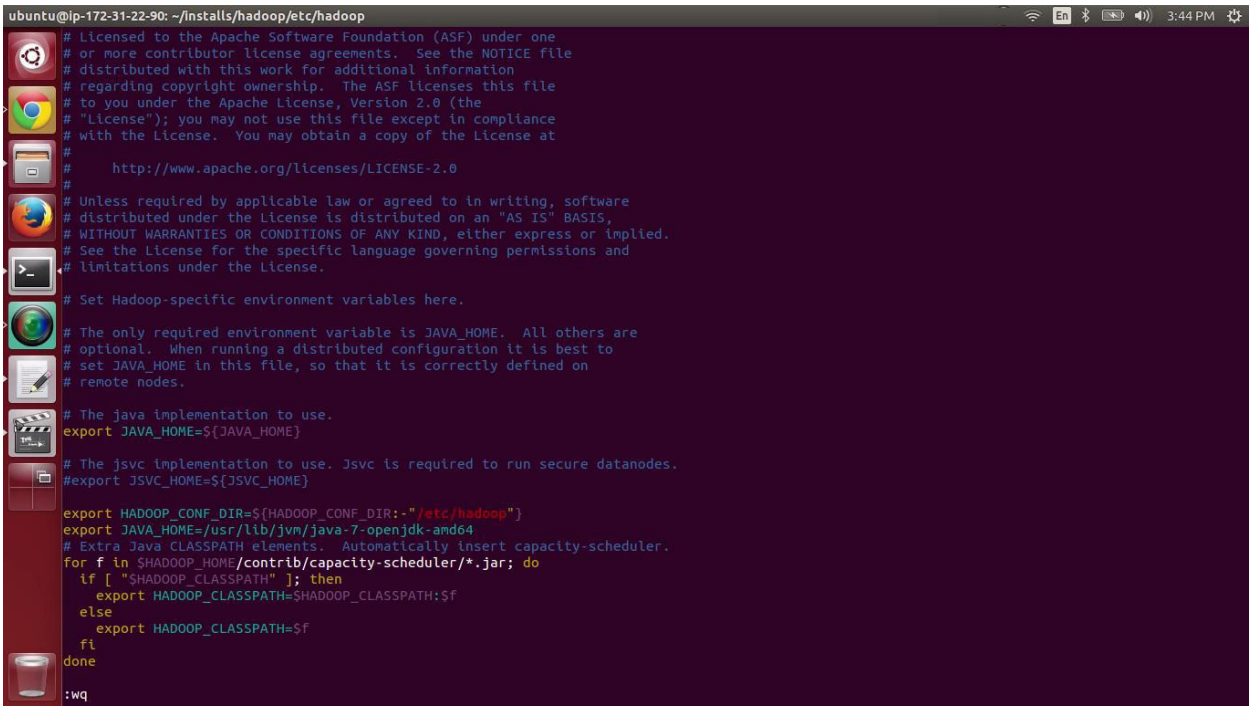


Screen Shots of the Configuration FILES:

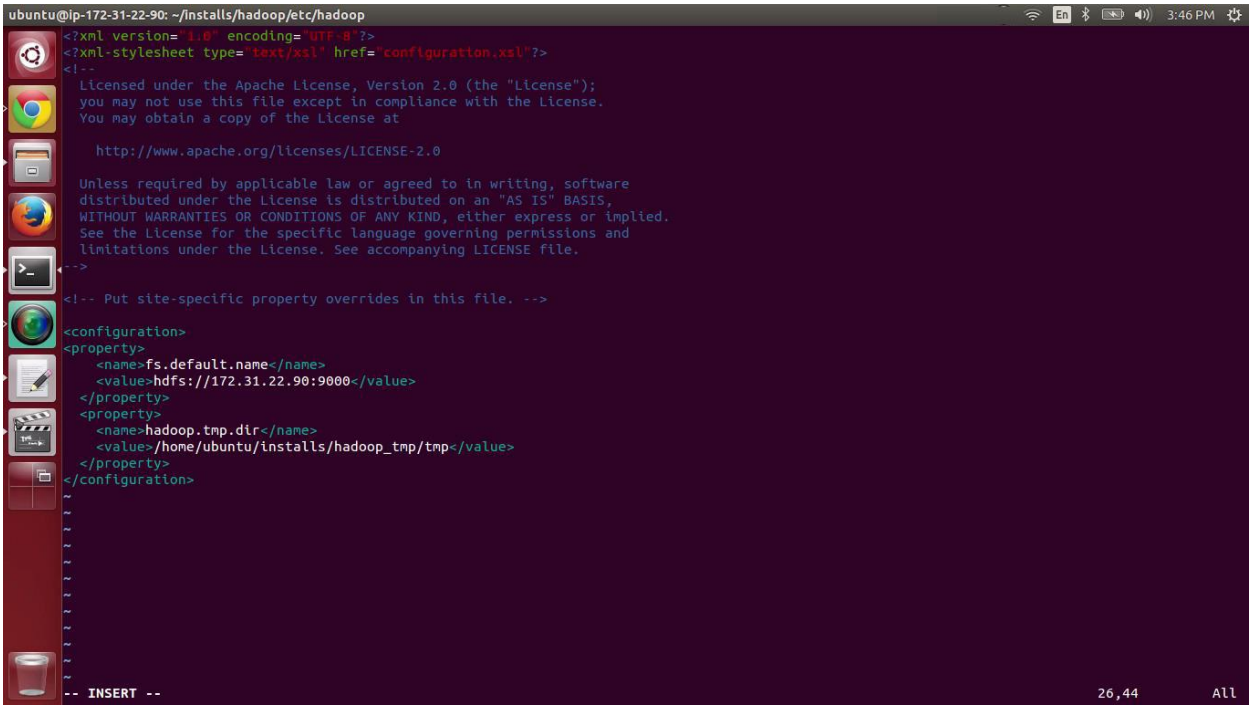
~/.bashrc: conf



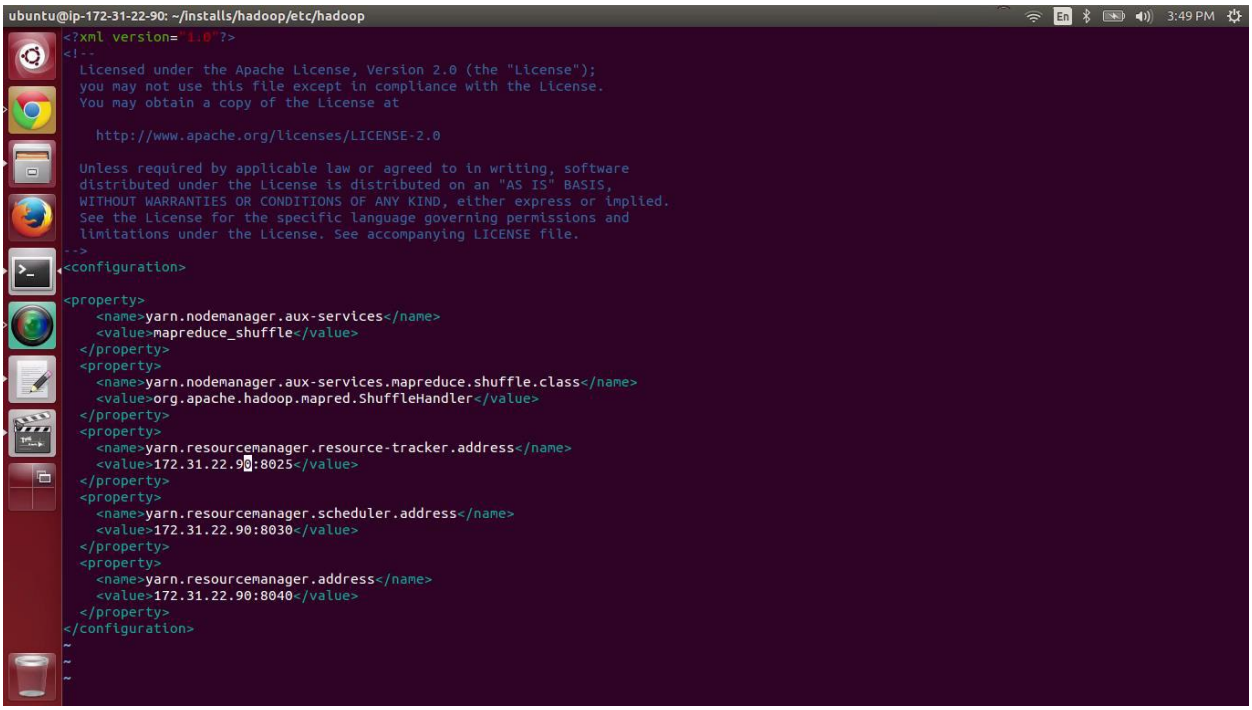
conf /hadoop-env.sh



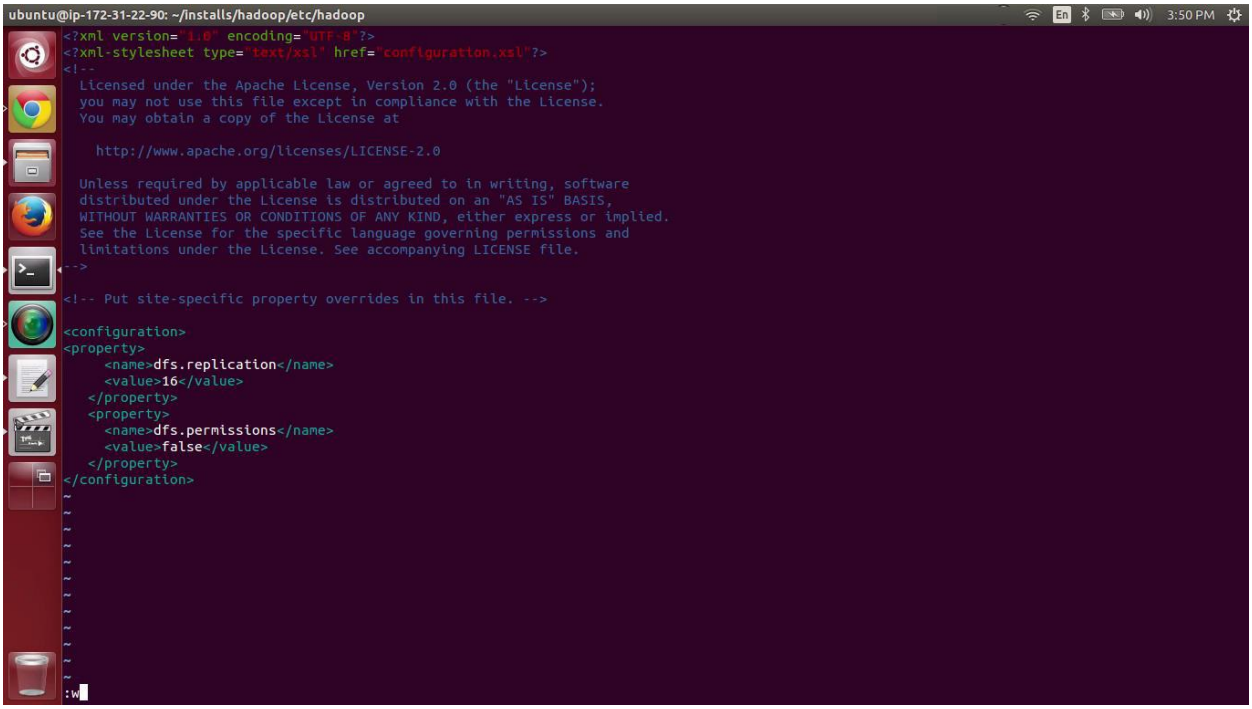
/conf/core-site.xml:



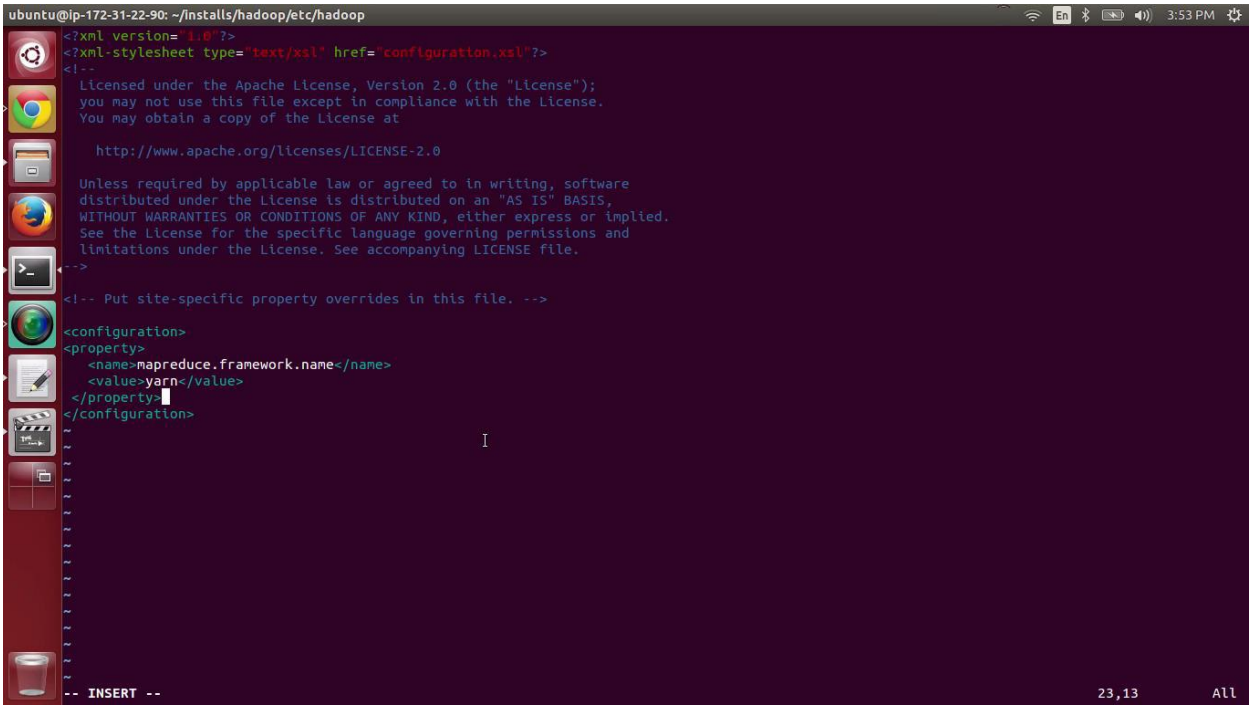
/conf/yarn-site.xml:



/conf/hdfs-site.xml:

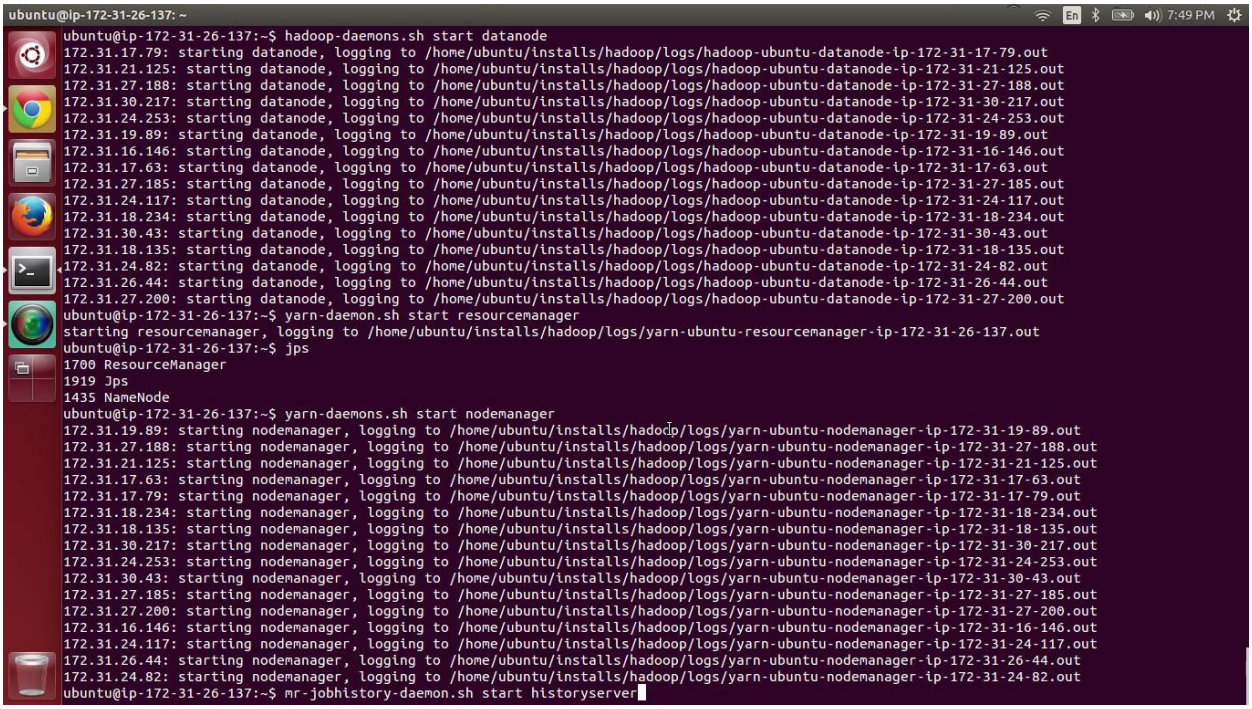


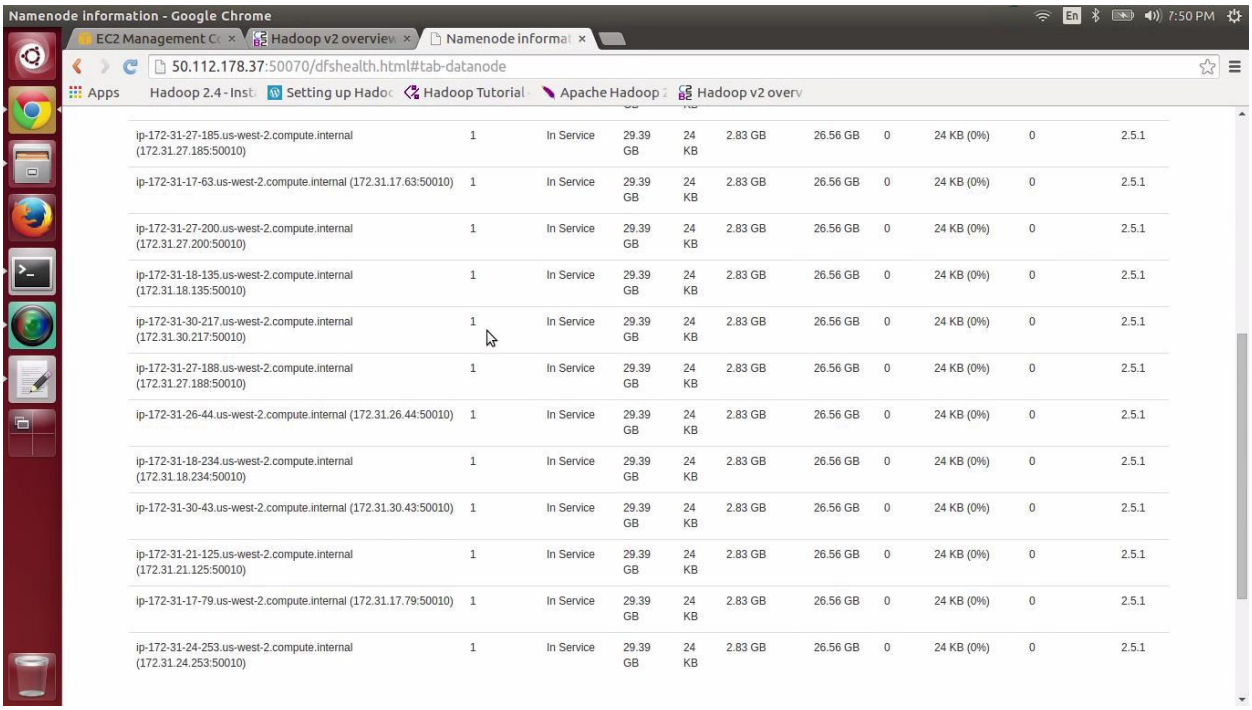
/conf/mapred-site.xml:

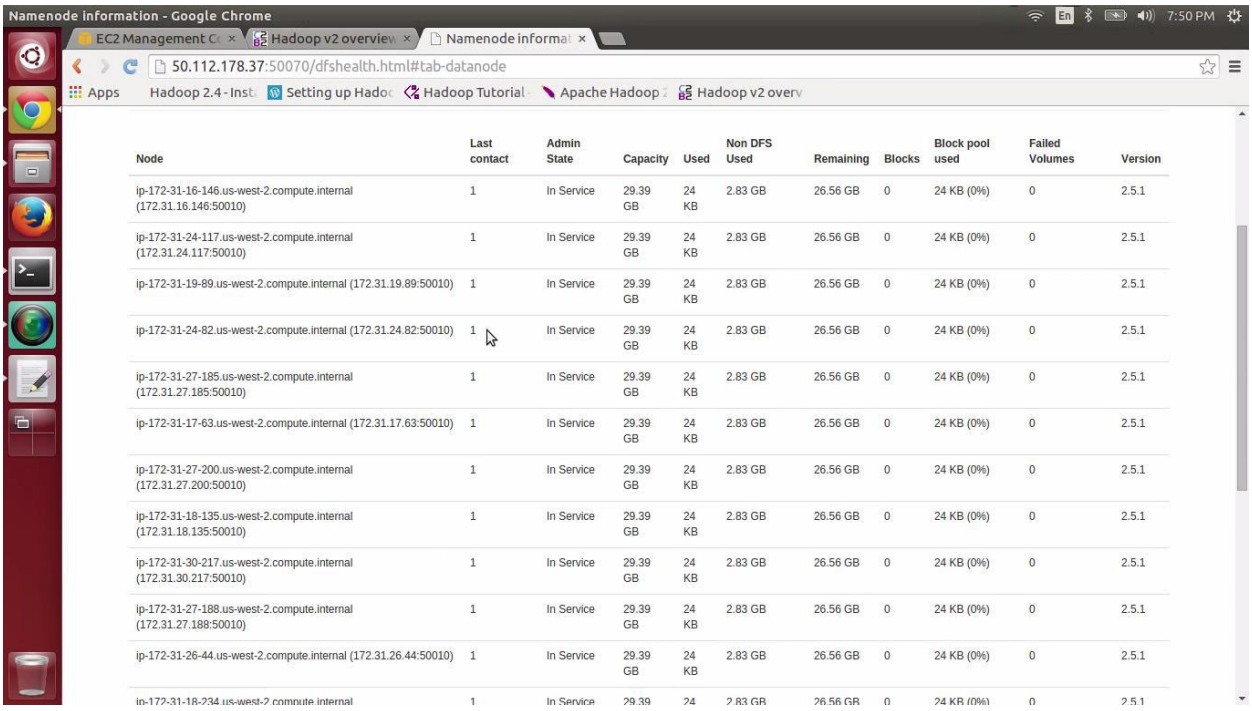


Connecting to Data Nodes:

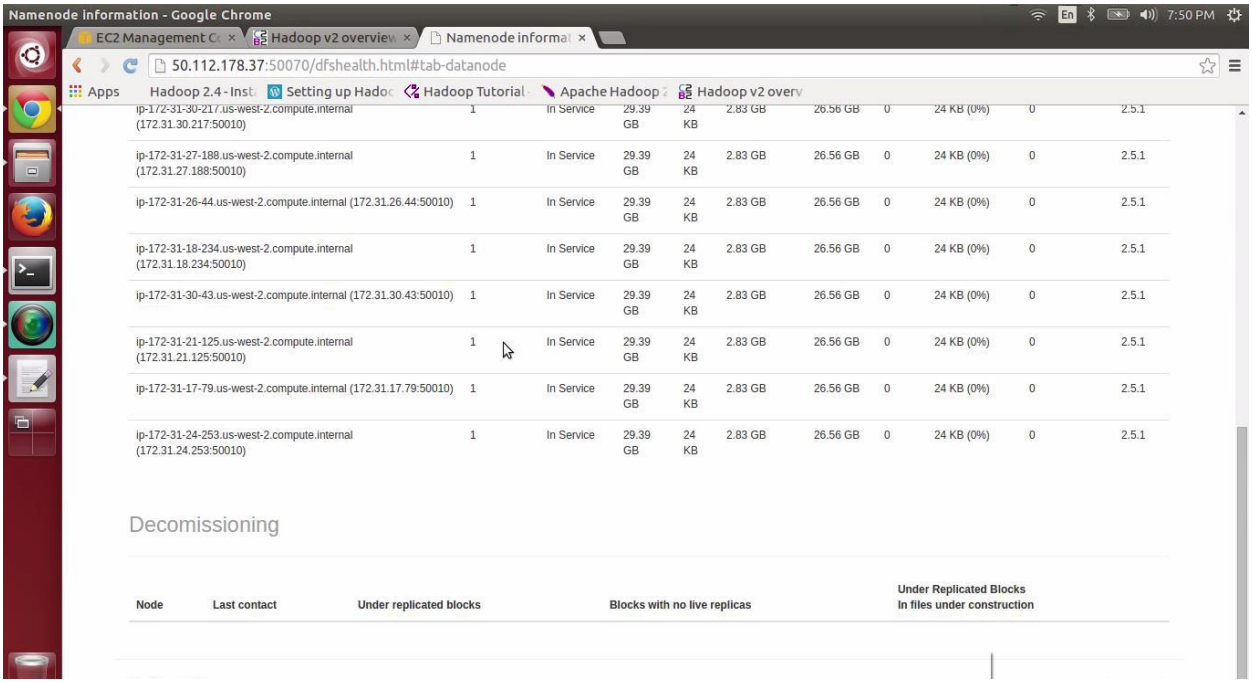
Screen Shots:







Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-16-146.us-west-2.compute.internal (172.31.16.146:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-24-117.us-west-2.compute.internal (172.31.24.117:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-19-89.us-west-2.compute.internal (172.31.19.89:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-24-82.us-west-2.compute.internal (172.31.24.82:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-27-185.us-west-2.compute.internal (172.31.27.185:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-17-63.us-west-2.compute.internal (172.31.17.63:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-27-200.us-west-2.compute.internal (172.31.27.200:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-18-135.us-west-2.compute.internal (172.31.18.135:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-30-217.us-west-2.compute.internal (172.31.30.217:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-27-188.us-west-2.compute.internal (172.31.27.188:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-26-44.us-west-2.compute.internal (172.31.26.44:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1



Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-30-217.us-west-2.compute.internal (172.31.30.217:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-27-188.us-west-2.compute.internal (172.31.27.188:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-26-44.us-west-2.compute.internal (172.31.26.44:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-18-234.us-west-2.compute.internal (172.31.18.234:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-30-43.us-west-2.compute.internal (172.31.30.43:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-21-125.us-west-2.compute.internal (172.31.21.125:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-17-79.us-west-2.compute.internal (172.31.17.79:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1
ip-172-31-24-253.us-west-2.compute.internal (172.31.24.253:50010)	1	In Service	29.39 GB	24 KB	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1

Decomissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks in files under construction

Sort in Hadoop:

SYSTEM CONFIGURATION:

VERSION: Hadoop 1.2.1

INSTANCE TYPE: c3. Large UBUNTU

RAM: 3.75 GB

NO. OF CORES: 2 virtual cores

STORAGE: 32 GB

Methodology:

1. HADOOP framework is written in java.
2. Install java latest version (1.7) and check version by using java –version command.
3. Next is to create and to setup SSH certificates and install HADOOP.
4. Input is obtained from gensort and it is executed using the following command:
   ./gensort –a 10000000000 10GBdata
   ./gensort –a 100000000000 100GBdata
5. Hadoop Sort Map Reduce: MapReduce programs are designed to compute large volumes of data in a parallel fashion i.e. 10GB and 100GB input. This requires dividing the workload across a two or single nodes.
6. List Processing: MapReduce programs transform lists of input data elements into lists of output data elements. A MapReduce program will do this twice, using two different list processing idioms: map, and reduce.
7. Mapping List: In the first phase called mapping, a list of data elements are provided, one at a time, to a function called the Mapper, which transforms each element individually to an output data element. As an example of the utility of map: We have a code to remove all the special characters (symbols) form the beginning and end of the word. We are not modifying the input string here: we are returning a new string that will form part of a new output list.
8. Reducing List: Reducing aggregate values together. A reducer function receives an iterator of input values from an input list. It then combines these values together, returning a single output value.
9. Now set up the configuration files and format the HADOOP file system and start HADOOP.
10. We have to access the output from the HDFS. This can be done using get command.
11. The output files stored are titled as Sort-hadoop-10GB.txt and Sort-hadoop-100GB.txt

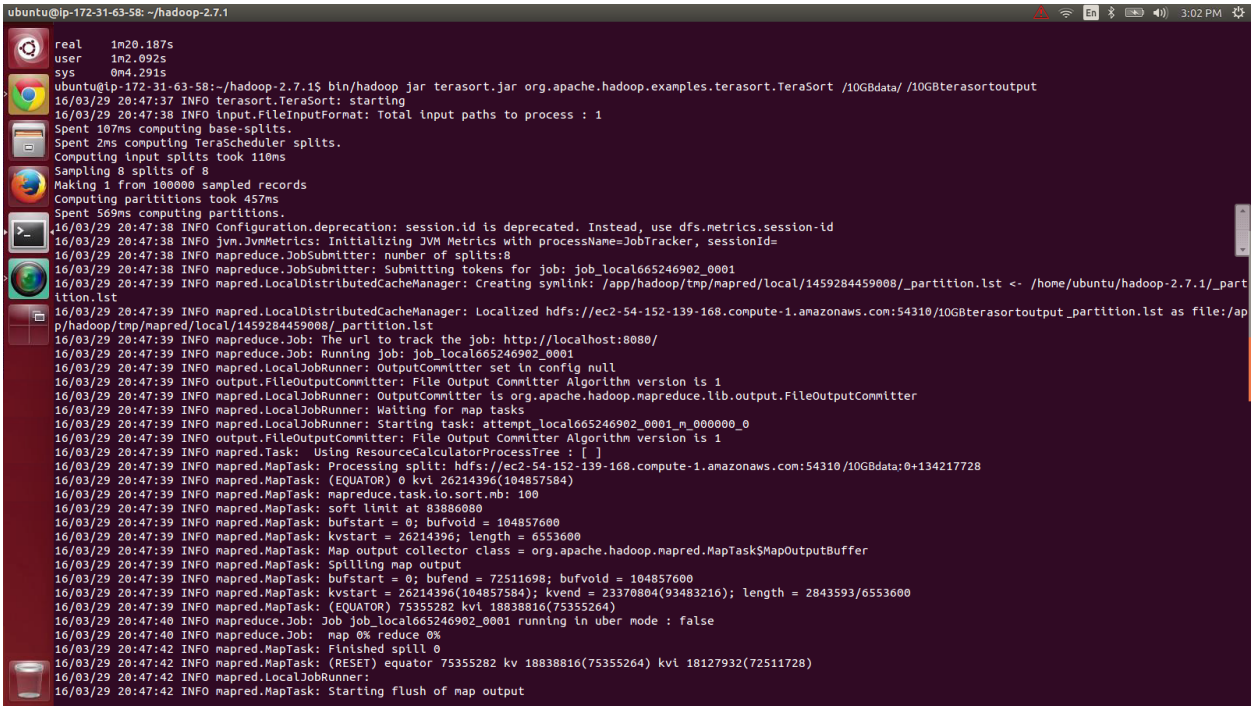
Outputs:

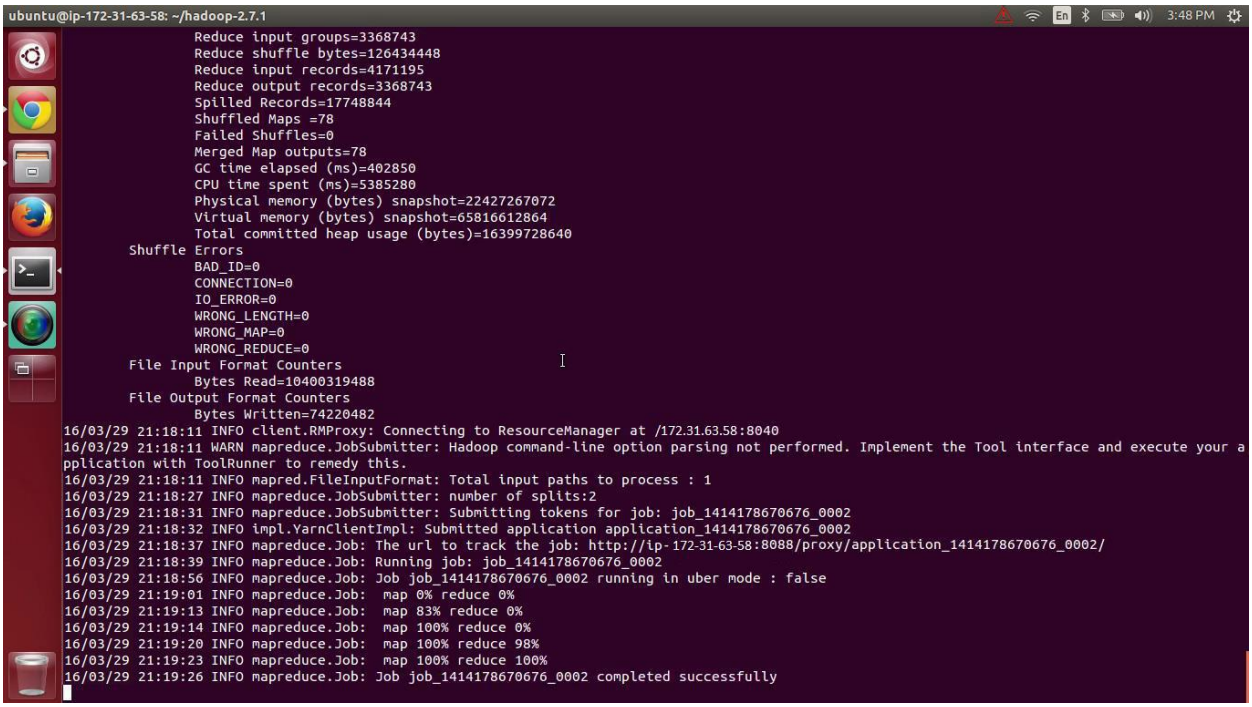
10 GB DATASET ON c3.large instance:

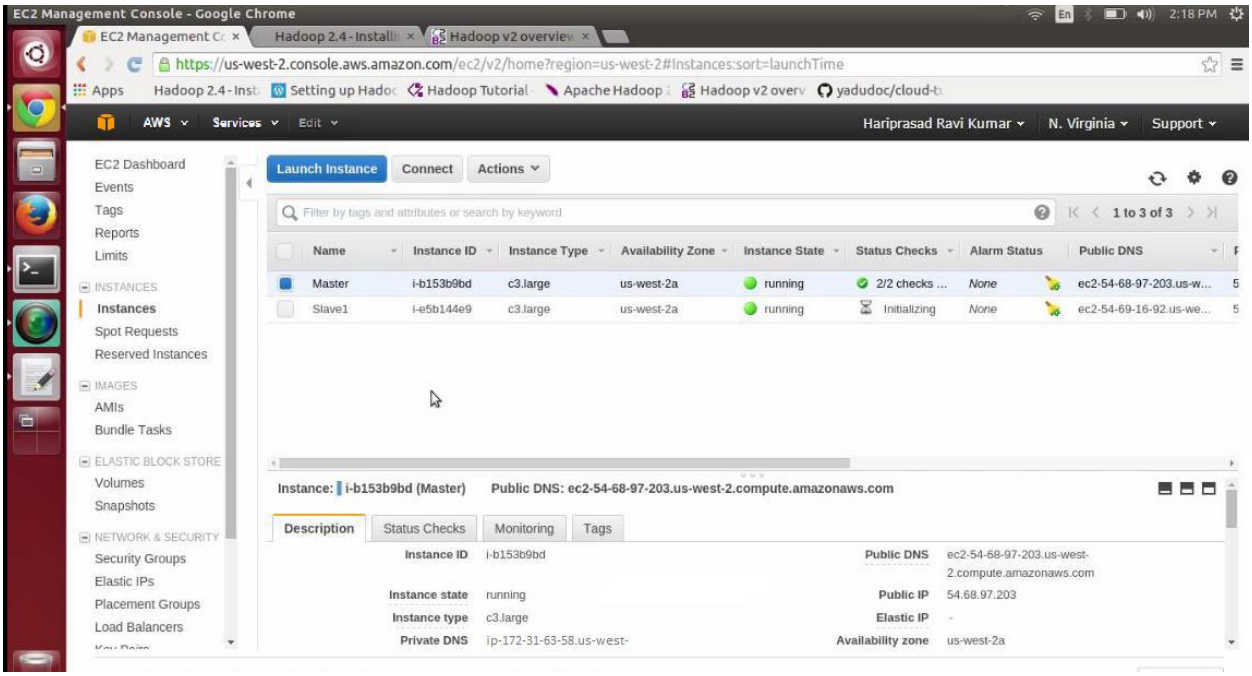
Node: 1

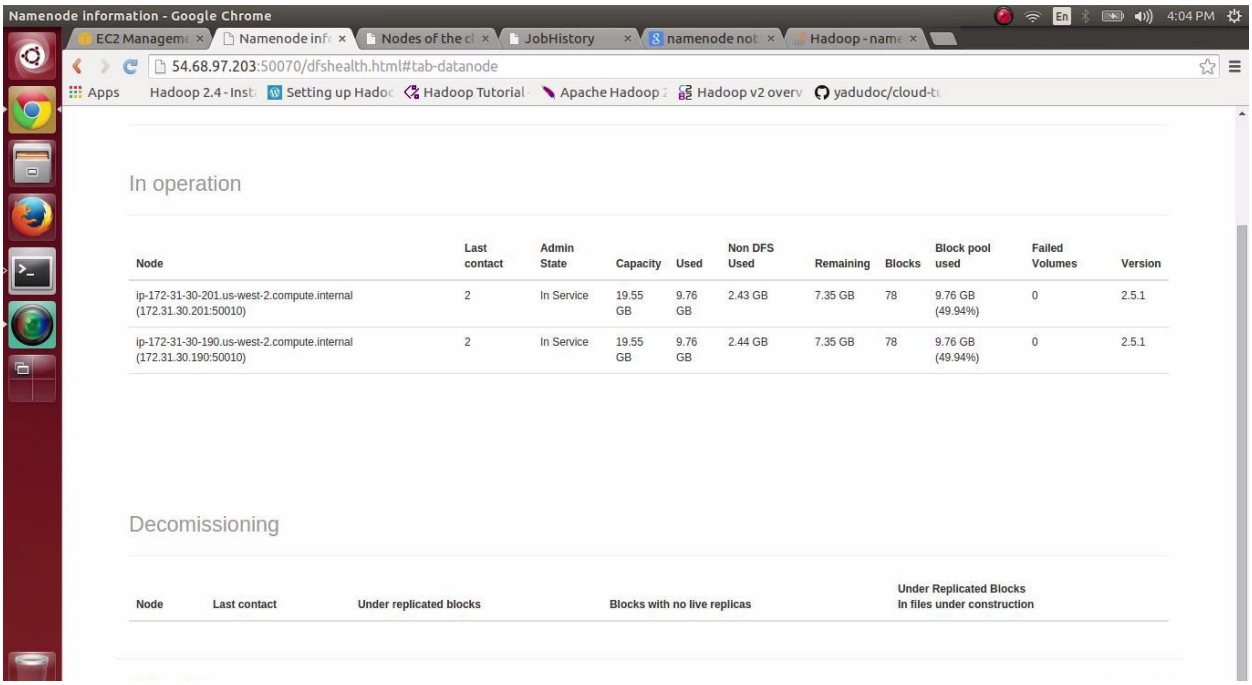
Execution time: 1921 seconds

Screen Shots for setting up 1 Node:




ubuntu@ip-172-31-63-58: ~/hadoop-2.7.1
Reduce input groups=3368743
Reduce shuffle bytes=126434448
Reduce input records=4171195
Reduce output records=3368743
Spilled Records=17748844
Shuffled Maps =78
Failed Shuffles=0
Merged Map outputs=78
GC time elapsed (ms)=402850
CPU time spent (ms)=5385280
Physical memory (bytes) snapshot=22427267072
Virtual memory (bytes) snapshot=65816612864
Total committed heap usage (bytes)=16399728640
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=10400319488
File Output Format Counters
Bytes Written=74220482
16/03/29 21:18:11 INFO client.RMProxy: Connecting to ResourceManager at /172.31.63.58:8040
16/03/29 21:18:11 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/03/29 21:18:11 INFO mapred.FileInputFormat: Total input paths to process : 1
16/03/29 21:18:27 INFO mapreduce.JobSubmitter: number of splits:2
16/03/29 21:18:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1414178670676_0002
16/03/29 21:18:32 INFO impl.YarnClientImpl: Submitted application application_1414178670676_0002
16/03/29 21:18:37 INFO mapreduce.Job: The url to track the job: http://ip-172-31-63-58:8088/proxy/application_1414178670676_0002/
16/03/29 21:18:39 INFO mapreduce.Job: Running job: job_1414178670676_0002
16/03/29 21:18:56 INFO mapreduce.Job: Job job_1414178670676_0002 running in uber mode : false
16/03/29 21:19:01 INFO mapreduce.Job: map 0% reduce 0%
16/03/29 21:19:13 INFO mapreduce.Job: map 83% reduce 0%
16/03/29 21:19:14 INFO mapreduce.Job: map 100% reduce 0%
16/03/29 21:19:20 INFO mapreduce.Job: map 100% reduce 98%
16/03/29 21:19:23 INFO mapreduce.Job: map 100% reduce 100%
16/03/29 21:19:26 INFO mapreduce.Job: Job job_1414178670676_0002 completed successfully


EC2 Management Console - Google Chrome
https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=launchTime
AWS
Services
Edit
Hariprasad Ravi Kumar
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Launch Instance
Connect
Actions
1 to 3 of 3
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
Master
i-b153b9bd
c3.large
us-west-2a
running
2/2 checks ...
None
ec2-54-68-97-203.us-w...
Slave1
i-e5b144e9
c3.large
us-west-2a
running
Initializing
None
ec2-54-69-16-92.us-we...
Instance: i-b153b9bd (Master)
Public DNS: ec2-54-68-97-203.us-west-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID
i-b153b9bd
Instance state
running
Instance type
c3.large
Private DNS
ip-172-31-63-58.us-west-
Public DNS
ec2-54-68-97-203.us-west-2.compute.amazonaws.com
Public IP
54.68.97.203
Elastic IP
-
Availability zone
us-west-2a

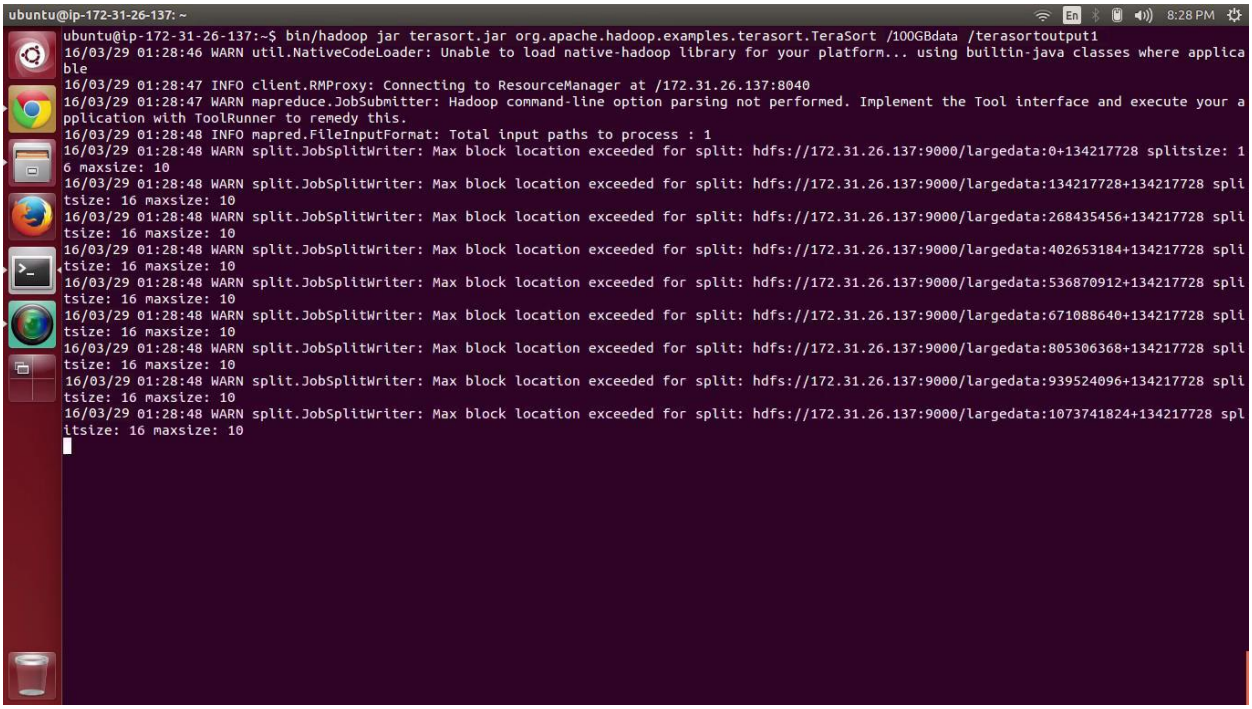


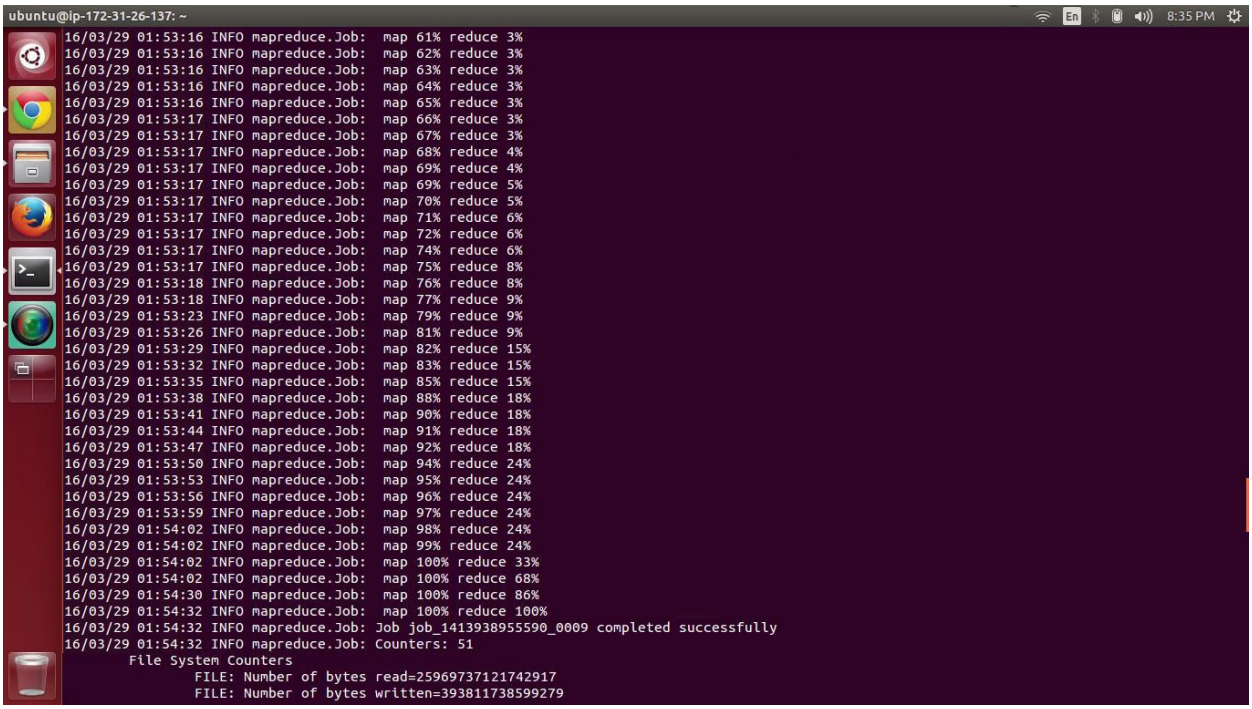
100 GB DATASET ON c3.large instance:

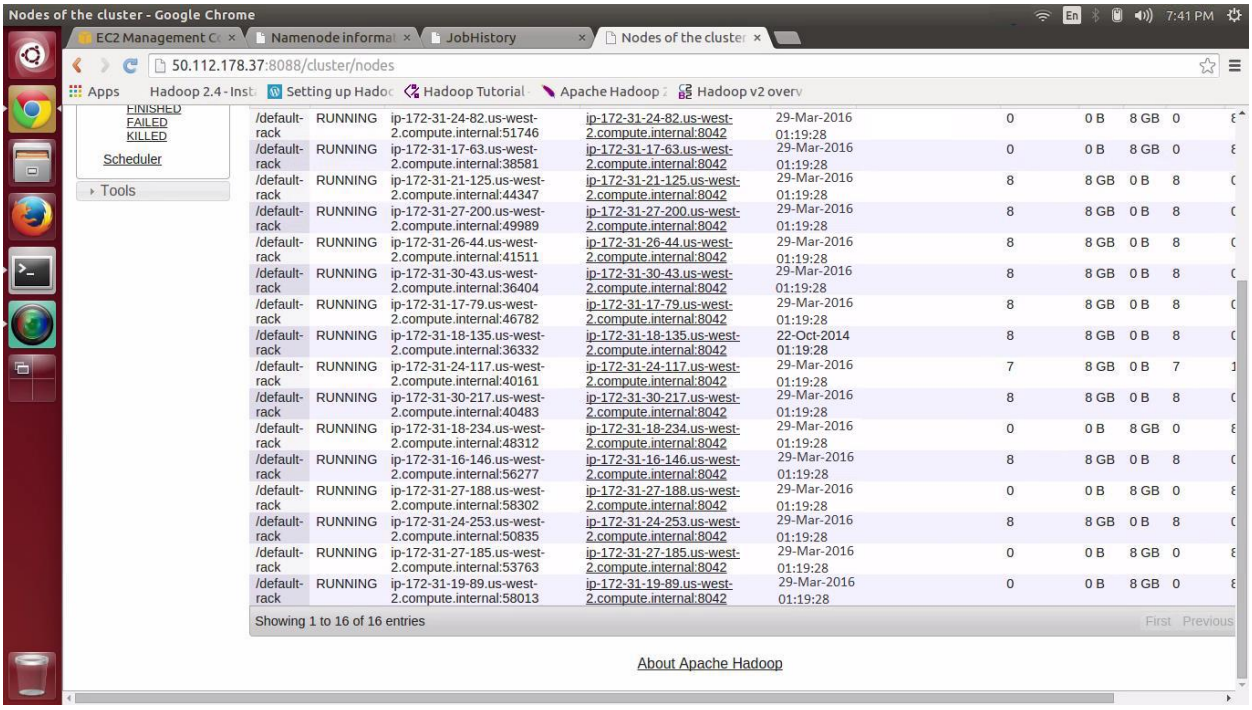
Nodes: 16

Execution time: 1562 seconds

Screen Shots for setting up 16 Nodes:

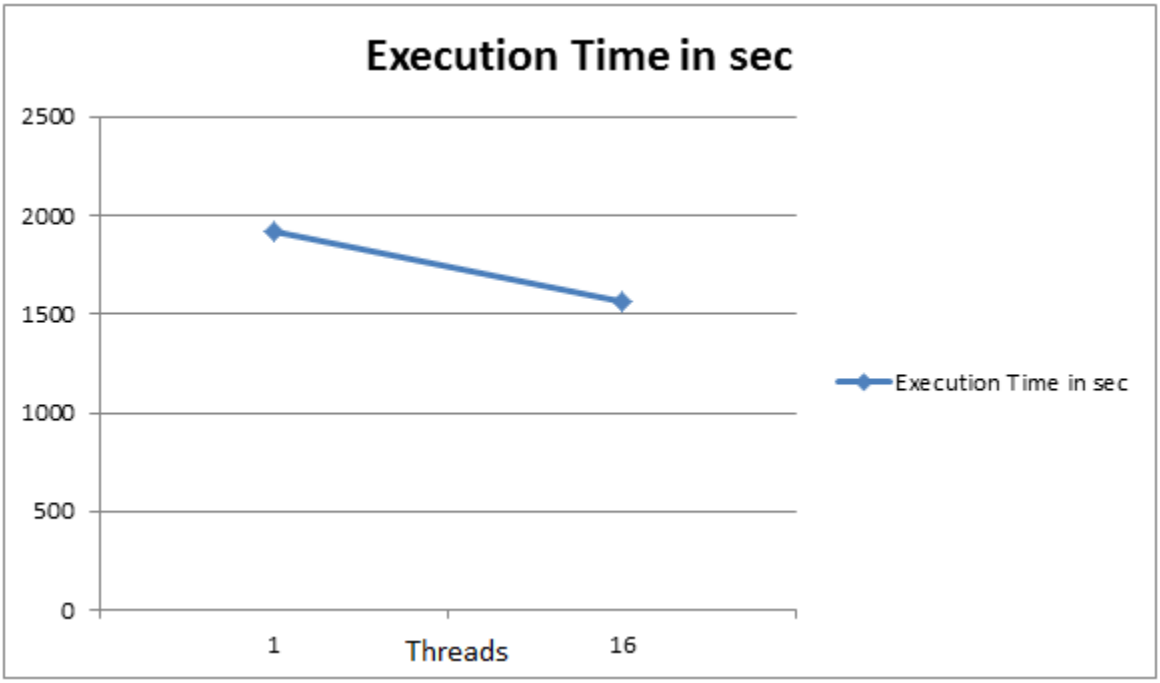


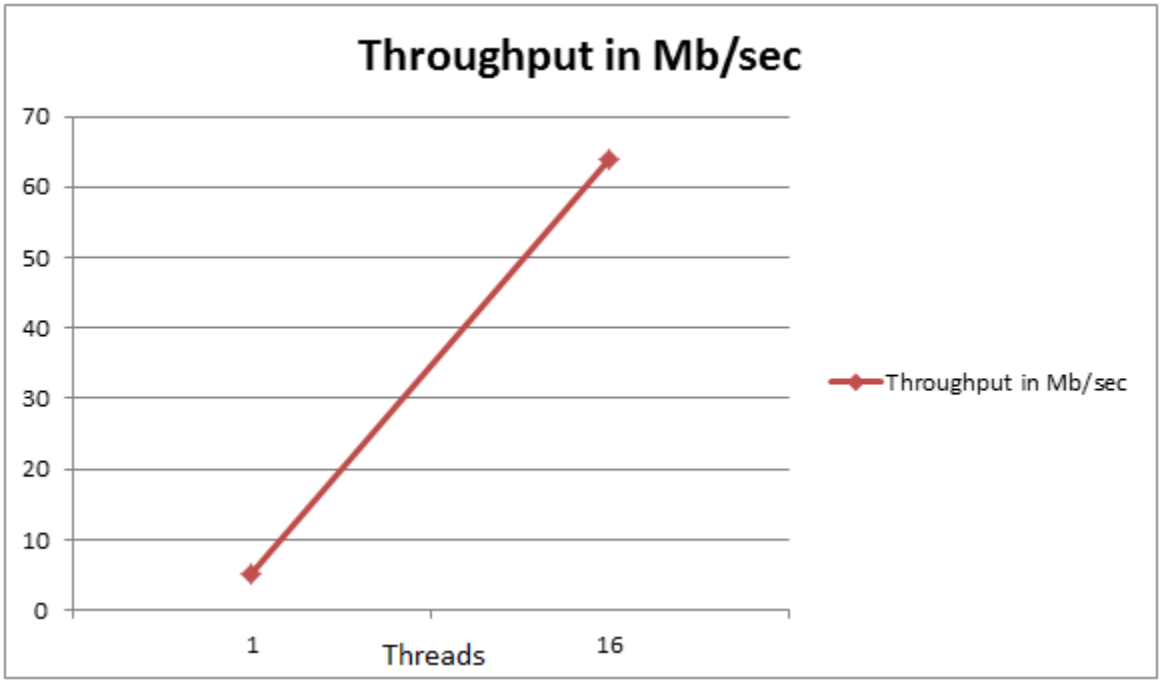

ubuntu@ip-172-31-26-137: ~
8:35 PM
16/03/29 01:53:16 INFO mapreduce.Job: map 61% reduce 3%
16/03/29 01:53:16 INFO mapreduce.Job: map 62% reduce 3%
16/03/29 01:53:16 INFO mapreduce.Job: map 63% reduce 3%
16/03/29 01:53:16 INFO mapreduce.Job: map 64% reduce 3%
16/03/29 01:53:16 INFO mapreduce.Job: map 65% reduce 3%
16/03/29 01:53:17 INFO mapreduce.Job: map 66% reduce 3%
16/03/29 01:53:17 INFO mapreduce.Job: map 67% reduce 3%
16/03/29 01:53:17 INFO mapreduce.Job: map 68% reduce 4%
16/03/29 01:53:17 INFO mapreduce.Job: map 69% reduce 4%
16/03/29 01:53:17 INFO mapreduce.Job: map 69% reduce 5%
16/03/29 01:53:17 INFO mapreduce.Job: map 70% reduce 5%
16/03/29 01:53:17 INFO mapreduce.Job: map 71% reduce 6%
16/03/29 01:53:17 INFO mapreduce.Job: map 72% reduce 6%
16/03/29 01:53:17 INFO mapreduce.Job: map 74% reduce 6%
16/03/29 01:53:17 INFO mapreduce.Job: map 75% reduce 8%
16/03/29 01:53:18 INFO mapreduce.Job: map 76% reduce 8%
16/03/29 01:53:18 INFO mapreduce.Job: map 77% reduce 9%
16/03/29 01:53:23 INFO mapreduce.Job: map 79% reduce 9%
16/03/29 01:53:26 INFO mapreduce.Job: map 81% reduce 9%
16/03/29 01:53:29 INFO mapreduce.Job: map 82% reduce 15%
16/03/29 01:53:32 INFO mapreduce.Job: map 83% reduce 15%
16/03/29 01:53:35 INFO mapreduce.Job: map 85% reduce 15%
16/03/29 01:53:38 INFO mapreduce.Job: map 88% reduce 18%
16/03/29 01:53:41 INFO mapreduce.Job: map 90% reduce 18%
16/03/29 01:53:44 INFO mapreduce.Job: map 91% reduce 18%
16/03/29 01:53:47 INFO mapreduce.Job: map 92% reduce 18%
16/03/29 01:53:50 INFO mapreduce.Job: map 94% reduce 24%
16/03/29 01:53:53 INFO mapreduce.Job: map 95% reduce 24%
16/03/29 01:53:56 INFO mapreduce.Job: map 96% reduce 24%
16/03/29 01:53:59 INFO mapreduce.Job: map 97% reduce 24%
16/03/29 01:54:02 INFO mapreduce.Job: map 98% reduce 24%
16/03/29 01:54:02 INFO mapreduce.Job: map 99% reduce 24%
16/03/29 01:54:02 INFO mapreduce.Job: map 100% reduce 33%
16/03/29 01:54:02 INFO mapreduce.Job: map 100% reduce 68%
16/03/29 01:54:30 INFO mapreduce.Job: map 100% reduce 86%
16/03/29 01:54:32 INFO mapreduce.Job: map 100% reduce 100%
16/03/29 01:54:32 INFO mapreduce.Job: Job job_1413938955590_0009 completed successfully
16/03/29 01:54:32 INFO mapreduce.Job: Counters: 51
File System Counters
FILE: Number of bytes read=25969737121742917
FILE: Number of bytes written=393811738599279


Nodes of the cluster - Google Chrome
7:41 PM
EC2 Management C
Namenode informal
JobHistory
Nodes of the cluster
50.112.178.37:8088/cluster/nodes
Apps
Hadoop 2.4 - Inst
Setting up Hadoc
Hadoop Tutorial
Apache Hadoop
Hadoop v2 overv
FINISHED
FAILED
KILLED
Scheduler
Tools
/default-rack RUNNING ip-172-31-24-82.us-west-2.compute.internal:51746 ip-172-31-24-82.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 0 0 B 8 GB 0
/default-rack RUNNING ip-172-31-17-63.us-west-2.compute.internal:38581 ip-172-31-17-63.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 0 0 B 8 GB 0
/default-rack RUNNING ip-172-31-21-125.us-west-2.compute.internal:44347 ip-172-31-21-125.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-27-200.us-west-2.compute.internal:49989 ip-172-31-27-200.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-26-44.us-west-2.compute.internal:41511 ip-172-31-26-44.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-30-43.us-west-2.compute.internal:36404 ip-172-31-30-43.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-17-79.us-west-2.compute.internal:46782 ip-172-31-17-79.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-18-135.us-west-2.compute.internal:36332 ip-172-31-18-135.us-west-2.compute.internal:8042 22-Oct-2014 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-24-117.us-west-2.compute.internal:40161 ip-172-31-24-117.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 7 8 GB 0 B 7
/default-rack RUNNING ip-172-31-30-217.us-west-2.compute.internal:40483 ip-172-31-30-217.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-18-234.us-west-2.compute.internal:48312 ip-172-31-18-234.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 0 0 B 8 GB 0
/default-rack RUNNING ip-172-31-16-146.us-west-2.compute.internal:56277 ip-172-31-16-146.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-27-188.us-west-2.compute.internal:58302 ip-172-31-27-188.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 0 0 B 8 GB 0
/default-rack RUNNING ip-172-31-24-253.us-west-2.compute.internal:50835 ip-172-31-24-253.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 8 8 GB 0 B 8
/default-rack RUNNING ip-172-31-27-185.us-west-2.compute.internal:53763 ip-172-31-27-185.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 0 0 B 8 GB 0
/default-rack RUNNING ip-172-31-19-89.us-west-2.compute.internal:58013 ip-172-31-19-89.us-west-2.compute.internal:8042 29-Mar-2016 01:19:28 0 0 B 8 GB 0
Showing 1 to 16 of 16 entries
First Previous
About Apache Hadoop

Number of nodes, the execution time and throughput for 10GB and 100GB data:

Number of nodes	Execution Time in sec	Throughput in Mb/sec
1 [10GB]	1921	5.2056
16 [100GB]	1562	64.0204

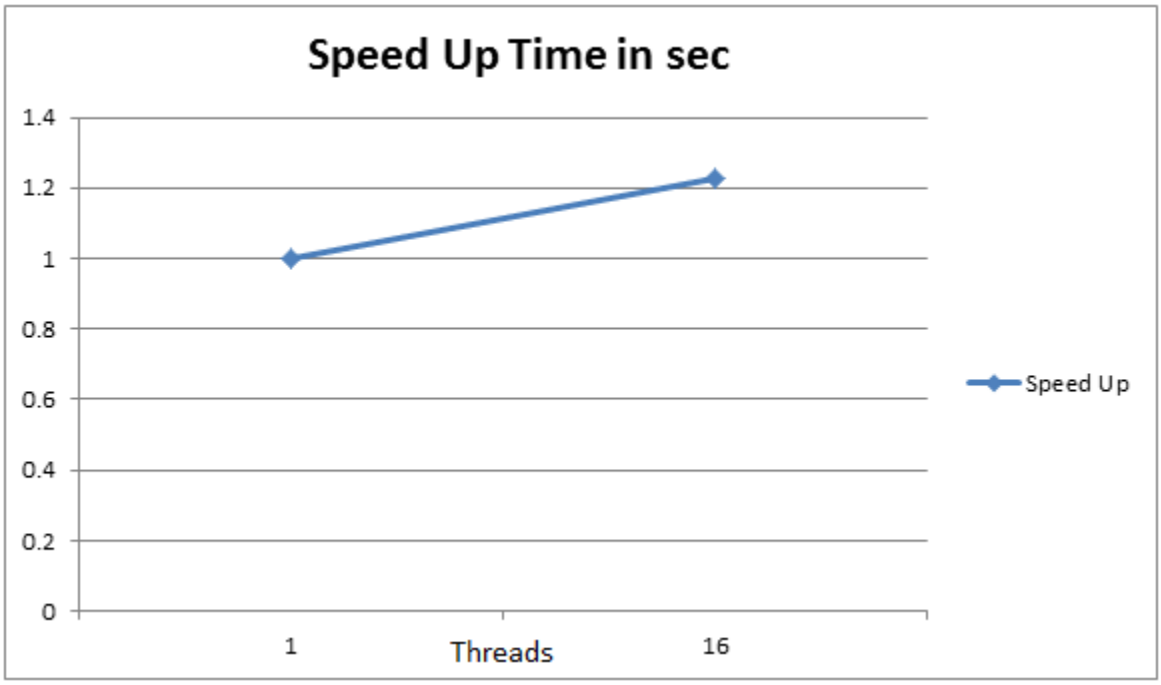




Number of nodes and the speed up time for 10GB and 100GB data:

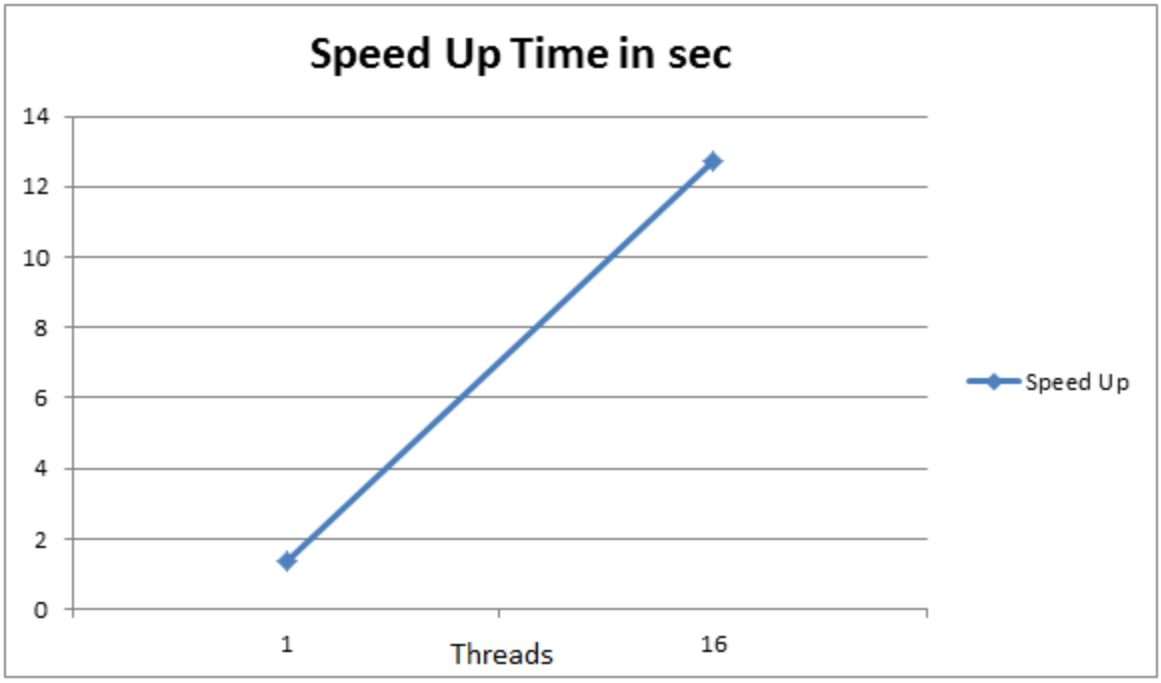
Base: 1 Node

Nodes	Execution time in seconds	Execution time for 1 Node	Speed Up
1[10GB]	1921	1921	1
16[100GB]	1562	1921	1.2298



<u>Speed Up comparison for Shared memory Java 1 Thread Baseline:</u>

Nodes	Execution time in seconds	Execution time for 1 Node Shared memory for 10GB and 100GB respectively	Speed Up
1[10GB]	1921	2618.552	1.36311
16[100GB]	1562	19861.547	12.71545



Best Performance is achieved at 16 nodes.

SPARK INSTALLATION:

Steps to get you IAM access keys:

1. Go to the IAM console.

2. From the navigation menu, click Users.

3. Select your IAM user name, or create a new one.

Access Key ID: AKIAIGHJ4UJMAVZO3SYA

Secret Access Key: 9CQTXf3b9HSm3iH++2cqY7eFe2O2JCagpAyUijeb

4. Click User Actions, and then click Manage Access Keys.

5. Click Create Access Key.

6. Click Download Credentials, and store the keys (a .csv file) in a secure location.

Create a Keypair to ssh to the ec2 instances:

1. Open the Amazon EC2 console.

2. In the top navigation bar, in the region selector, click US East (N. Virginia)

3. In the left navigation pane, under Network and Security, click Key Pairs.

4. Click Create Key Pair. Type awskey in the new Key pair name box.

5. Download the private key file, which is named awskey.pem, and keep it in a safe place. You will need it to access any instances that you launch with this key pair.

Launch an EC2 instance with AMI as Ubuntu Server 14.04 LTS (HVM)

Do shh-add mykey.pem. Then connect to the instance by using ssh –i @ubuntu:ipaddress Then do a scp to send mykeypair.pem and credentials to the instance location.

Install the following:

1. sudo apt-get install -y python python-pip git

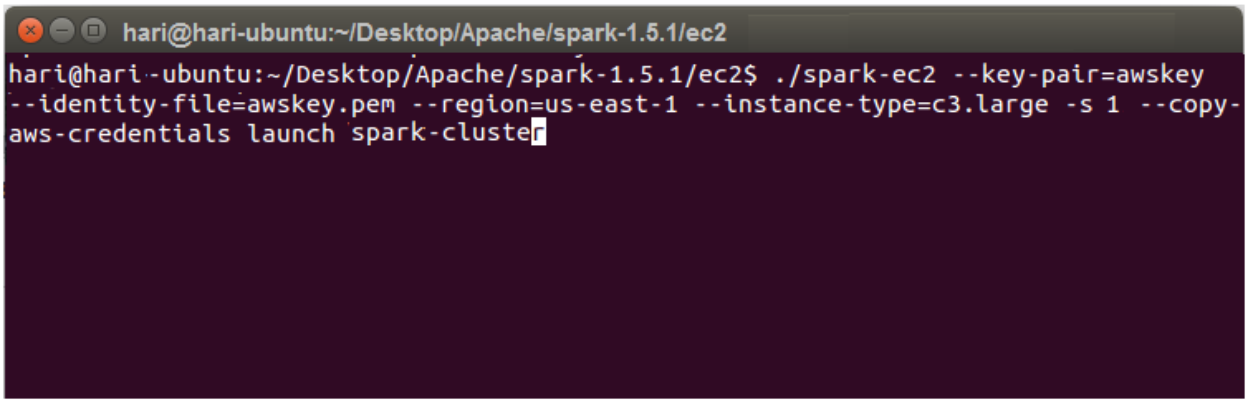
2. sudo pip install apache-libcloud

3. sudo apt-get install default-java

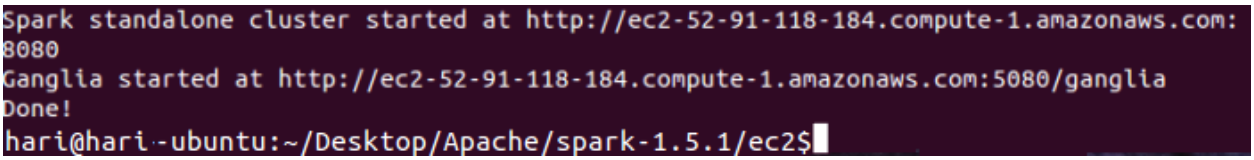
4. git clone git://github.com/apache/spark.git –b branch-1.5

5. In the Spark folder you had downloaded, navigate to the directory named “spark-1.5.0/ec2”.

6. Run the “spark-ec2” file with these arguments:



./spark-ec2 -k <keypair> -i <key-file> -s <num-slaves> --instancetype=<INSTANCE_TYPE> launch <cluster-name>



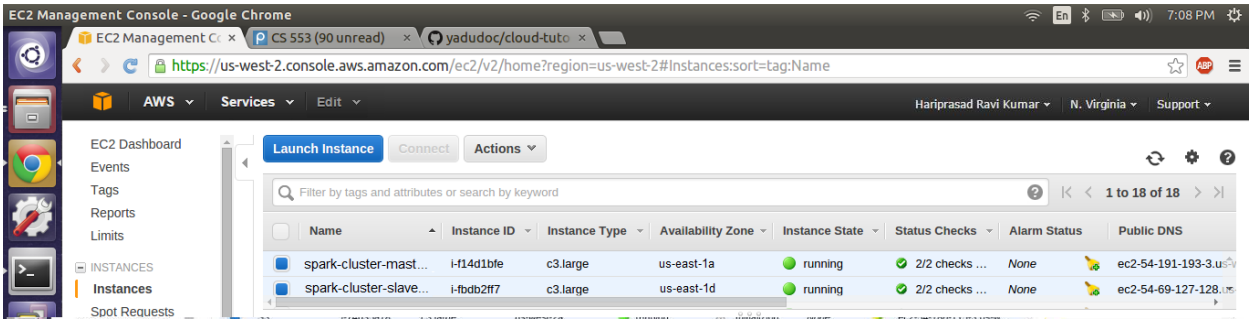
7. After the cluster has finished initializing, you can ssh into its root directory with:

./spark-ec2 -k <keypair> -i <key-file> login <cluster-name>

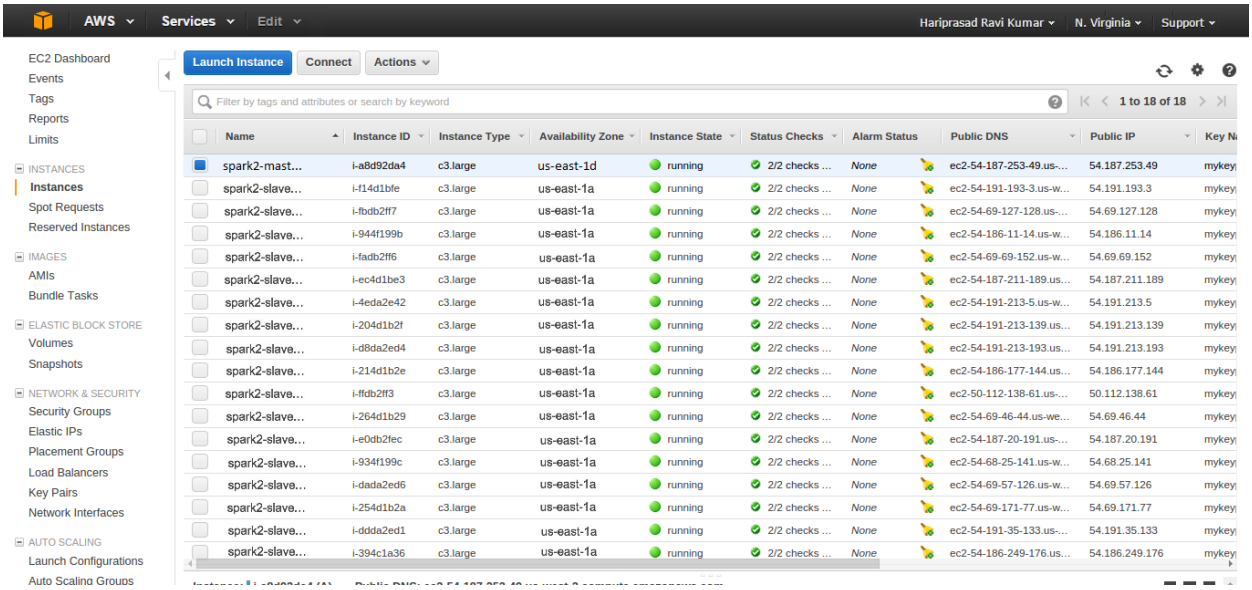
8. To end the EC2 Spark cluster, ssh out of the cluster and use:

./spark-ec2 destroy <cluster-name>

Screenshot of Head node and 1 Worker:



Head Node and 16 Workers:



Spark Sort:

SYSTEM CONFIGURATION:

VERSION : Spark-1.5.1

INSTANCE TYPE : c3. Large UBUNTU

RAM : 3.75 GB

NO. OF CORES : 2 virtual cores

STORAGE : 32 GB

Methodology:

1. Install spark on 1 and 17 nodes

2. Install the following in the node:

- Install python-pip, apache-libcloud
- git clone git://github.com/apache/spark.git –b branch-1.5
- Configure and start the cluster

3. Create an Access key credentials.csv and store it in an secure location.

4. Connect to instance by ssh and save the credentials and key file in it.

5. Navigate to ec2 and do a source setup.sh

6. Connect to masternode

7. Now run the program for 10GBdata:

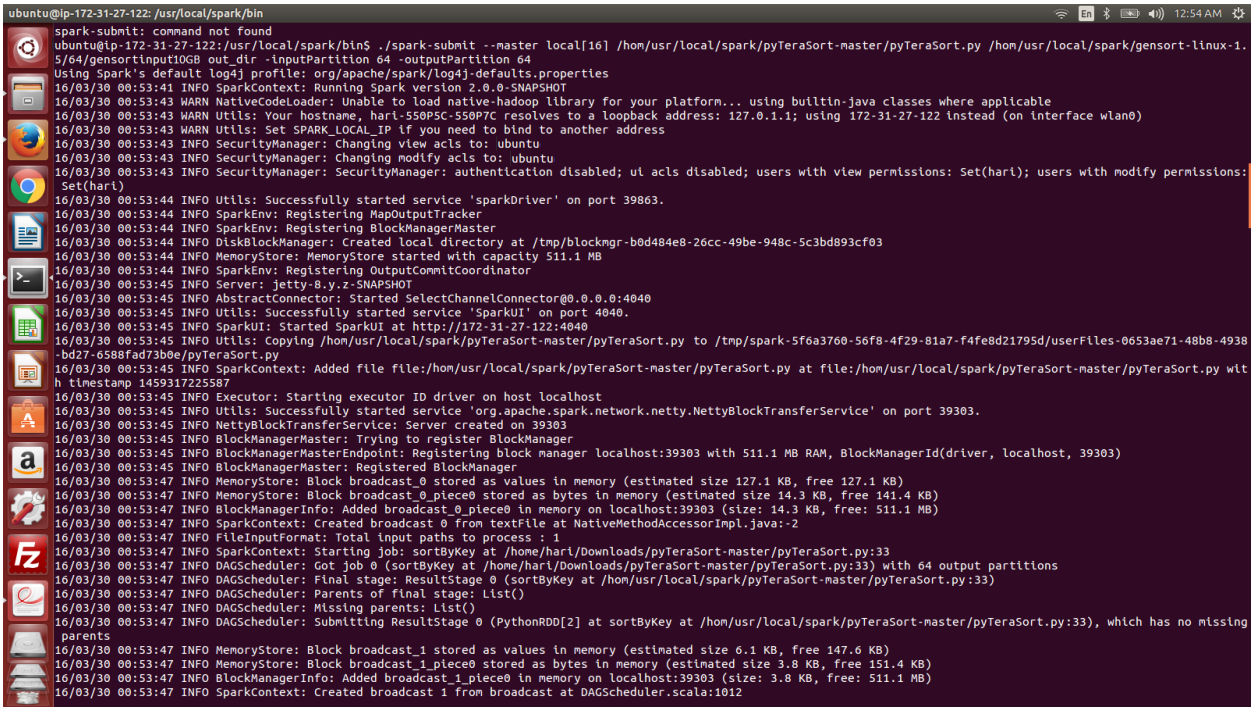
./bin/spark-submit --master spark://MASTER IP:7077 /terasort/terasort.py /gensort/gensortinput10GB out_dir

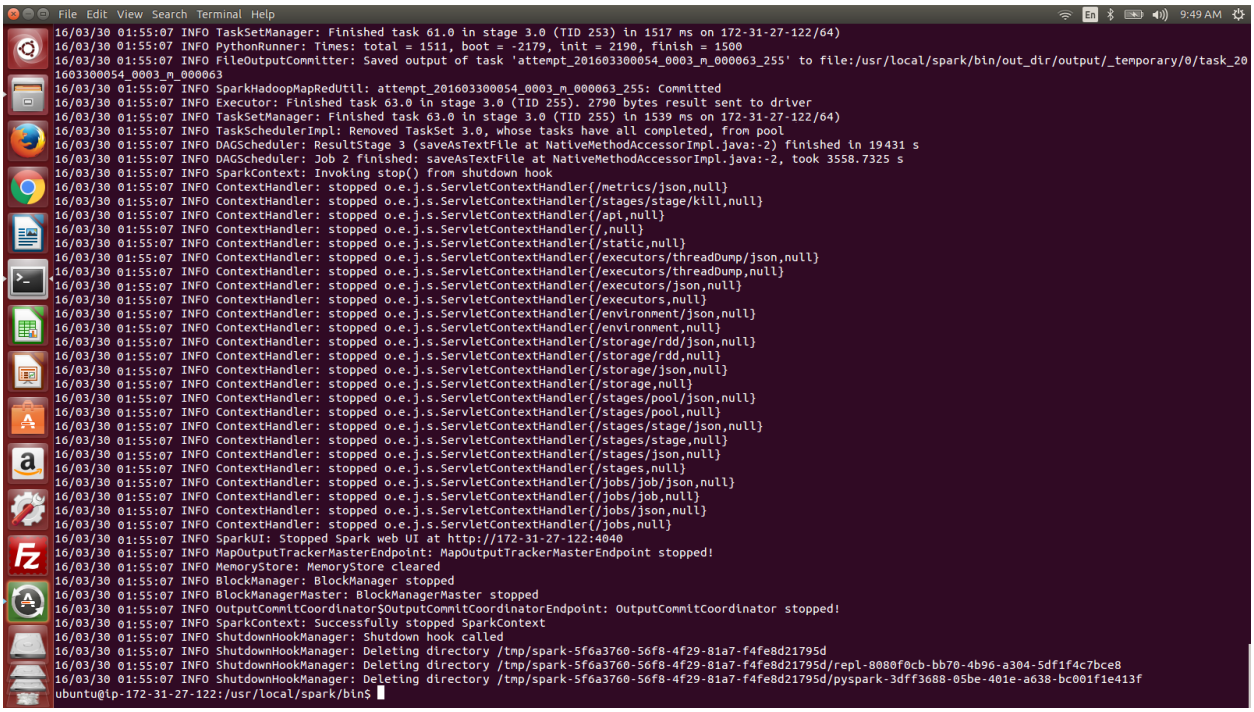
Execution results:

10 GB DATASET ON c3.large instance:

Node: 1

Execution time: 3558.7325 seconds

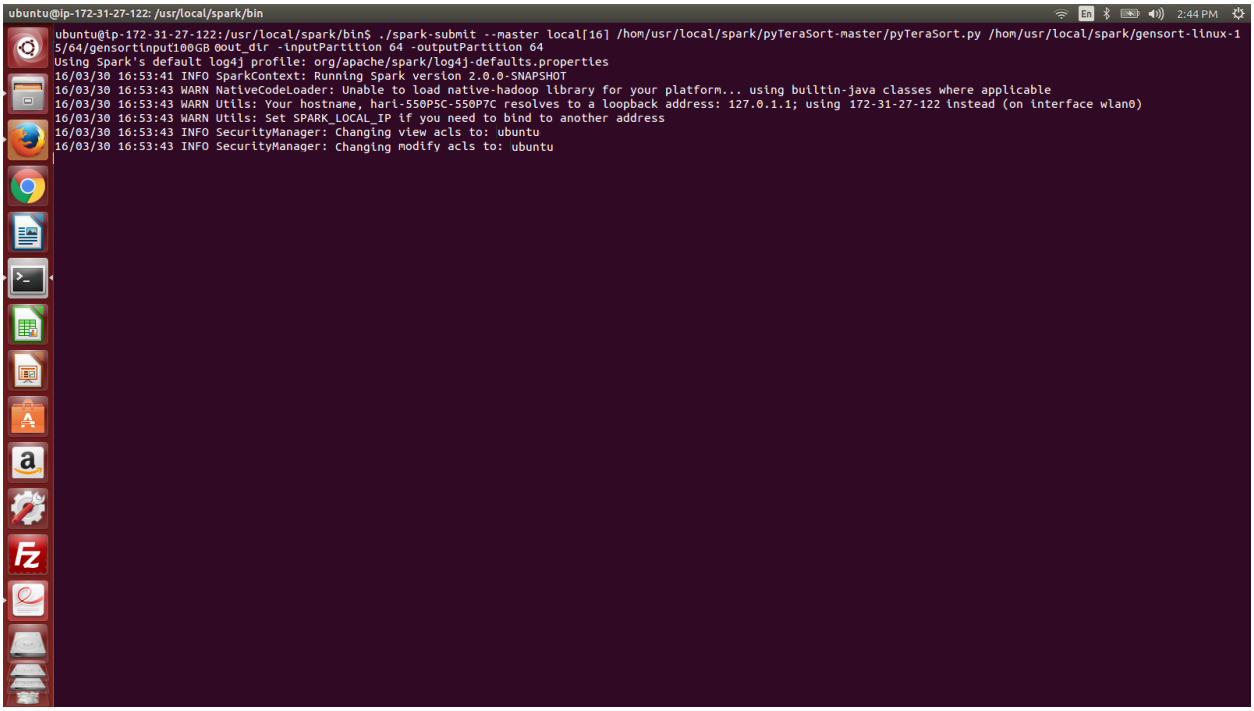


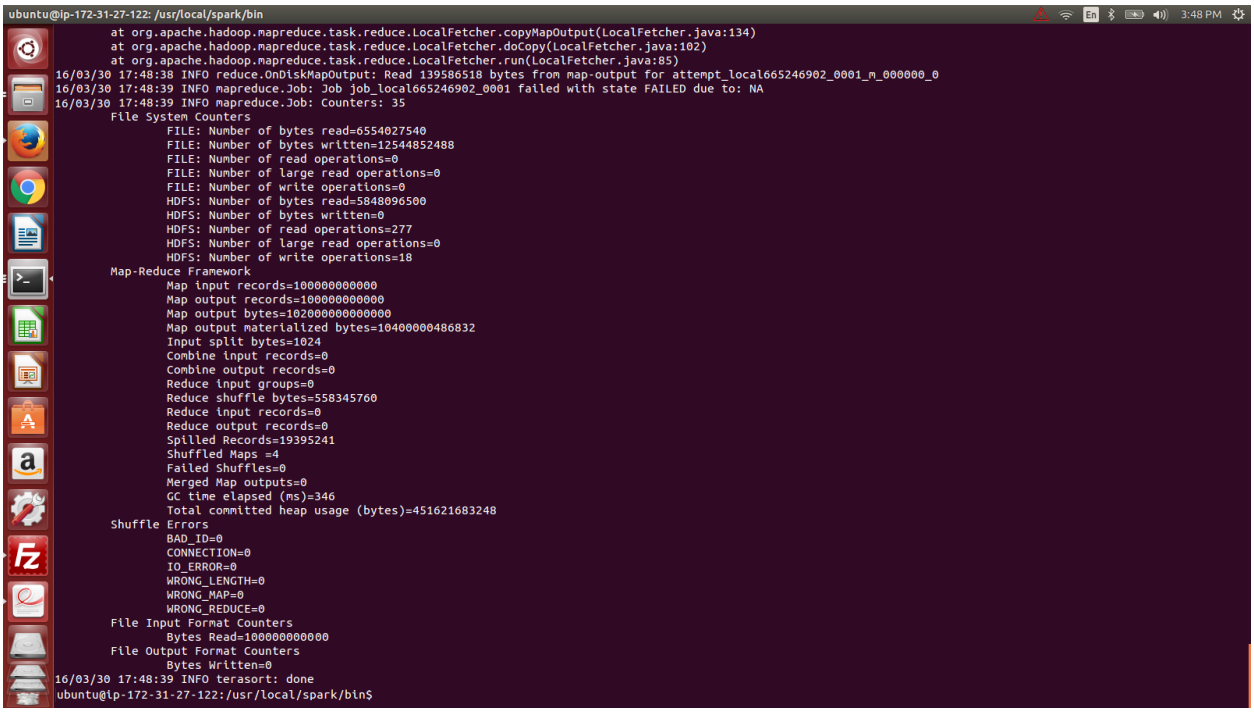


100 GB DATASET ON c3.large instance:

Nodes: 16

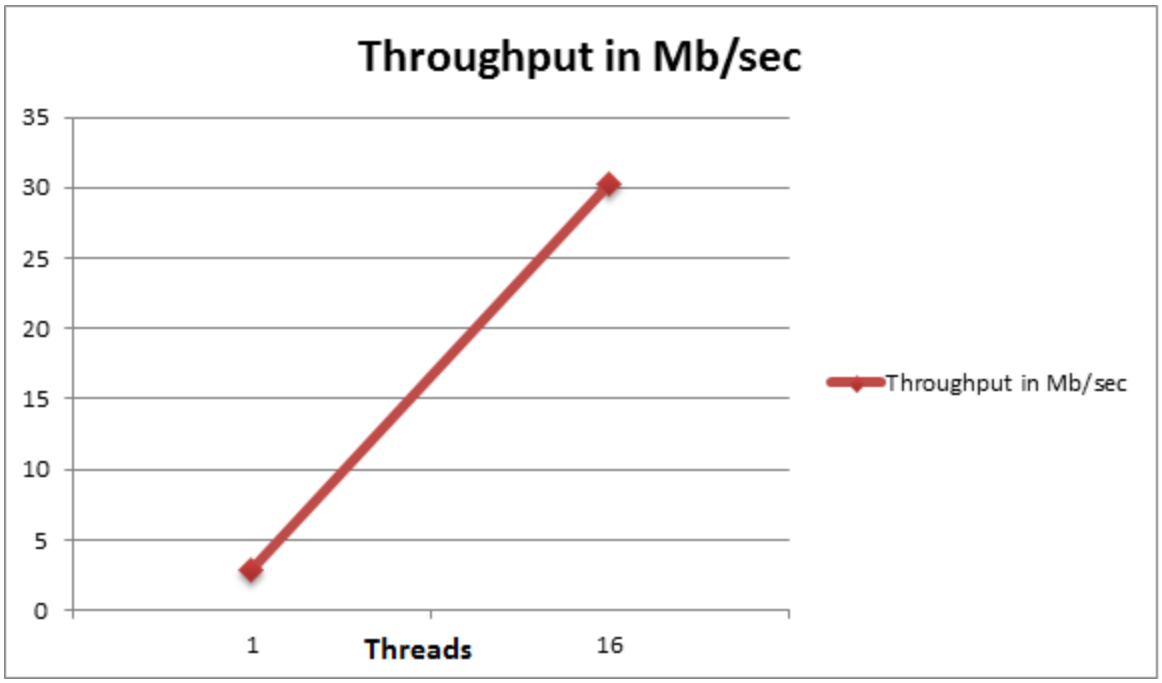
Execution time: 3303.6 seconds





Number of nodes, the execution time and throughput for 10GB and 100GB data:

Number of nodes	Execution Time in sec	Throughput in Mb/sec
1 [10GB]	3558.7325	2.8099
16 [100GB]	3303.6	30.27



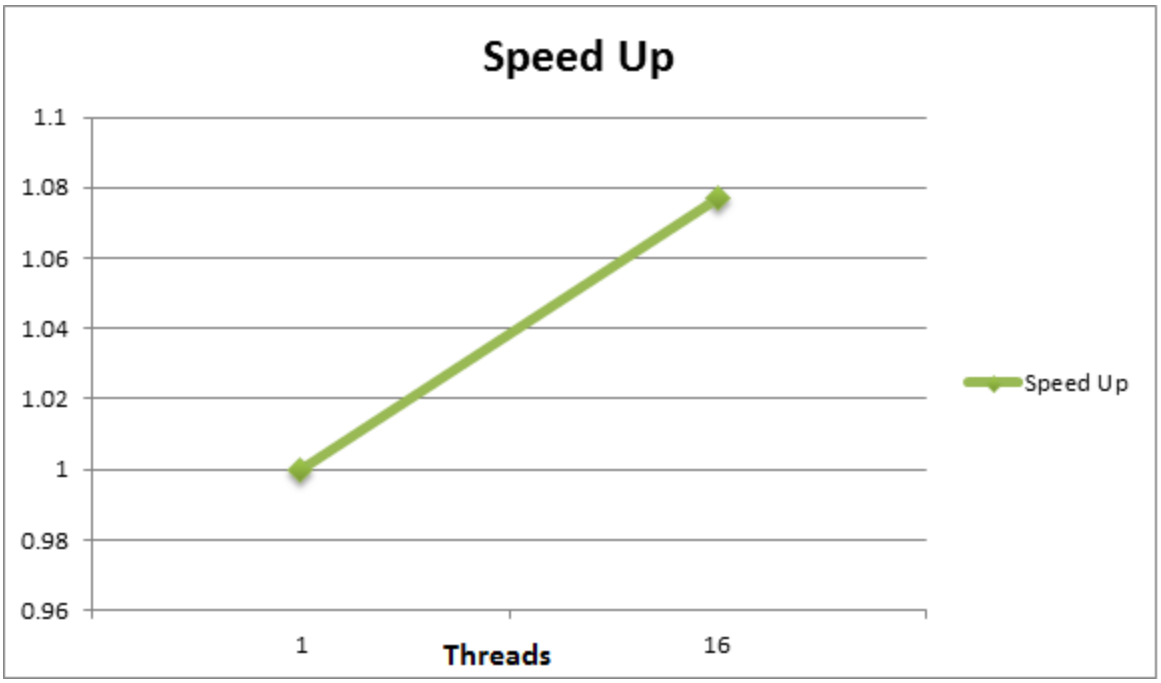
Number of nodes and the speed up time for 10GB and 100GB data:

Base: 1 Node

Nodes	Execution Time in sec	Execution time in 1 Node	Speed Up
1[10GB]	3558.7325	3558.7325	1
16[100GB]	3303.6	3558.7325	1.0772

GRAPH:

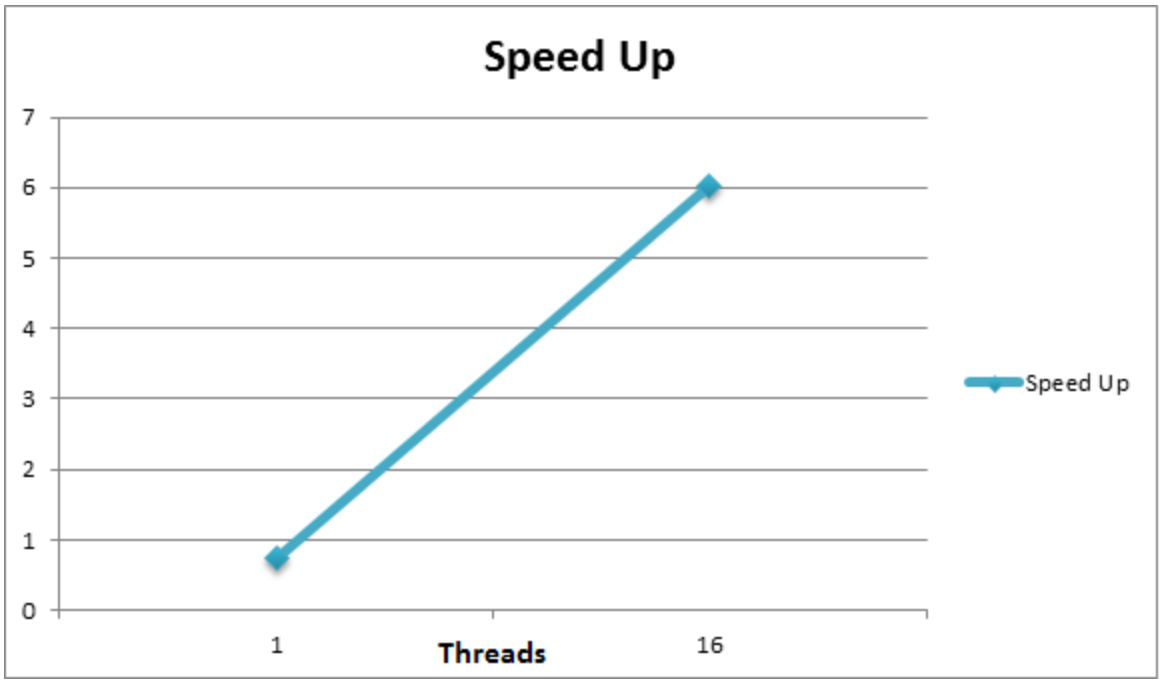
Using the information from the table above a graph is plotted between number of nodes and speed up time. The baseline of this graph is value obtained at node 1.



Best Performance is obtained at 16 nodes.

<u>Speed Up comparison for Shared memory Java 1 Thread Baseline:</u>

Nodes	Execution time in seconds	Execution time for 1 Node Shared memory for 10GB and 100GB respectively	Speed Up
1[10GB]	3558.7325	2618.552	0.73581
16[100GB]	3303.6	19861.547	6.01209


Speed Up
7
6
5
4
3
2
1
0
1
16
Threads
Speed Up

Performance:

PLATFORM	NUMBER OF NODES	EXECUTION TIME IN SEC
JAVA [10GB data]	1	2618.552
HADOOP [10GB data]	1	1921
SPARK [10GB data]	1	3558.7325
JAVA [100GB data]	1	19861.547
HADOOP [100GB data]	16	1562
SPARK [100GB data]	16	3303.6

Out of all the three **HADOOP** is said to give the best performance.

From the above table we can say that best performance at 16 nodes is offered by HADOOP and at 1 node is offered by JAVA. Compared to 1 node execution time of JAVA execution time of 16 nodes of SPARK and HADOOP is faster. Let it be 100 nodes or 1000 nodes it doesn't make any difference as you scale from 16 nodes to higher because the execution time stays constant as there is resource saturation.

While considering sorting we can see that spark gives the best performance. It gives almost 1/4th times that of hadoop. So while considering sorting we can say that SPARK has the best performance.

MPI Sort:

Configuration:

VERSION: MPI 3.1.3

INSTANCE TYPE: c3. Large UBUNTU

RAM: 3.75 GB

NO. OF CORES: 2 virtual cores

STORAGE: 32 GB

Methodology:

MPI INSTALL VERSION: mpich3.1.3

1. install mpich3.1.3 and configure it with command ./configure disable --fortran

2. build $make

after the installation of mpich3.1.3 on the local node

3. run MPIsort.cpp using

mpicc MPIsort.cpp -o MPITsort

after compilation the resulting file is an executable file MPIsort

4. mpirun -np 2 /home/ubuntu/MPIsort 10gb.txt

mpirun -np 2 /home/ubuntu/MPIsort 100gb.txt

mpirun executes the executable file with 2 processes sorting the 10gb.txt and 100gb.txt files.

Running MPI ON AMAZON EC2

run the below commands

sudo apt-get install pythonsetuptools
sudo apt-get install python-pip
sudo apt-get update
sudo easy_install -upgrade pip
sudo easy_install StarCluster0.95.5

and install pycrypto-2.6.1

after installing Starcluster0.95.5
configure starcluster in
./starcluster/config

and after configuring the start cluster is installed
now start the cluster on 16 nodes.