

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 screenshot shows the AWS EC2 Management Console in Google Chrome. The left sidebar has 'Instances' selected under 'AWS Services'. The main pane displays a table of instances. One instance, named 'Master' with Instance ID i-ba77f821, is highlighted. The details page for this instance shows it is a c3.large type, running in the us-east-1b availability zone, with a Public DNS of ec2-52-207-222-135.compute-1.amazonaws.com. A terminal window at the bottom shows the instance's configuration.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
Master	i-ba77f821	c3.large	us-east-1b	running	Initializing	None	ec2-52-207-222-135.co...

Instance: i-ba77f821 (Master) Public DNS: ec2-52-207-222-135.compute-1.amazonaws.com

Description		Status Checks	Monitoring	Tags
Instance ID	i-ba77f821	Public DNS	ec2-52-207-222-135 compute-1.amazonaws.com	
Instance state	running	Public IP	52.207.222.135	
Instance type	c3.large	Elastic IP	-	
Private DNS	ip-172-31-17-143.ec2.internal	Availability zone	us-east-1b	
Private IPs	172.31.28.195	Security groups	launch-wizard-3, view rules	
Secondary private IPs		Scheduled events	No scheduled events	
VPC ID	vpc-becb1edb	AMI ID	Loading ami-3d50120d...	
Subnet ID	subnet-a1d611d6	Platform	-	

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

This screenshot is identical to the one above, showing the AWS EC2 Management Console with a c3.large instance named 'Master'. However, a terminal window is open at the bottom of the page, showing a command-line session on the instance. The terminal window title is 'ubuntu@ip-172-31-28-195: ~' and the prompt is 'ubuntu@ip-172-31-28-195:~\$'. This indicates that the user has successfully connected to the instance via SSH.

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

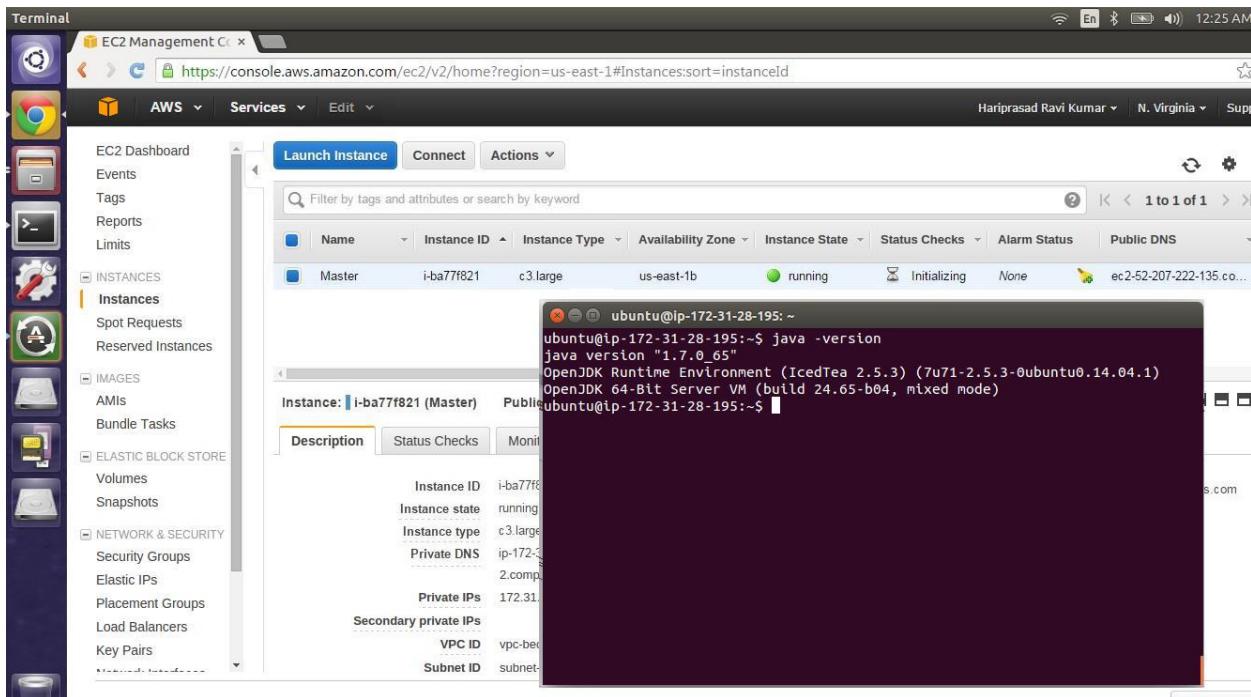
chmod 400 keygiven(This is to change the permission to be read by owner.)

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-jre-headless.
- 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:

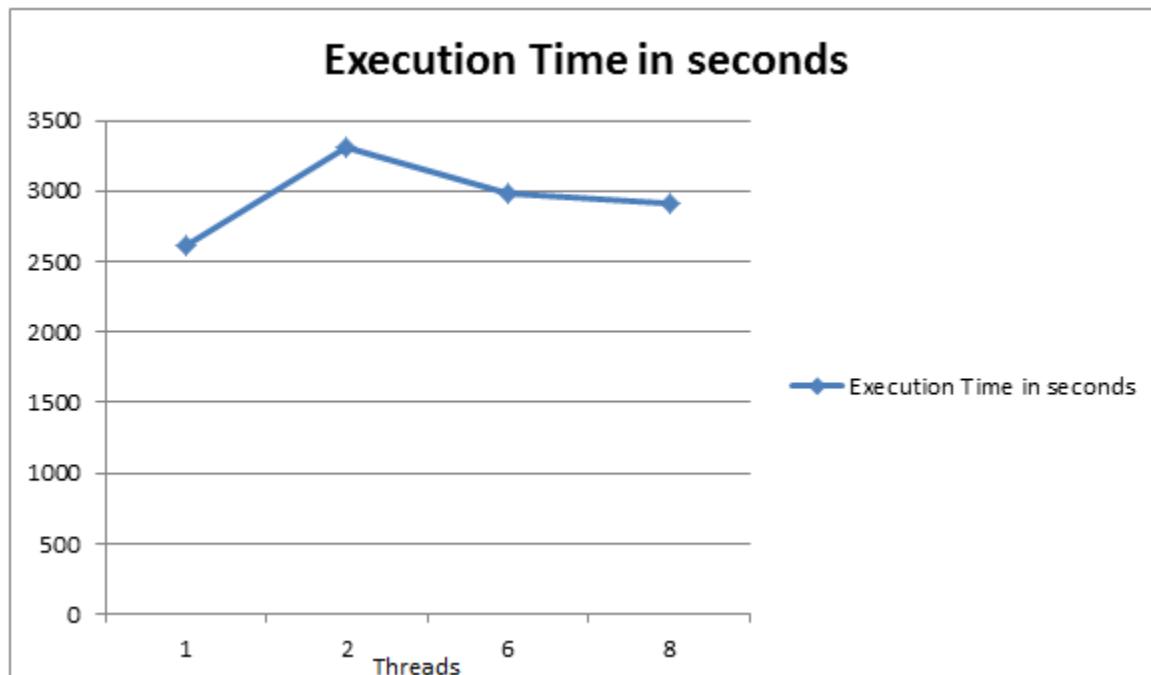
10 GB DATASET ON c3.large instance:

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

EXPLANATION:

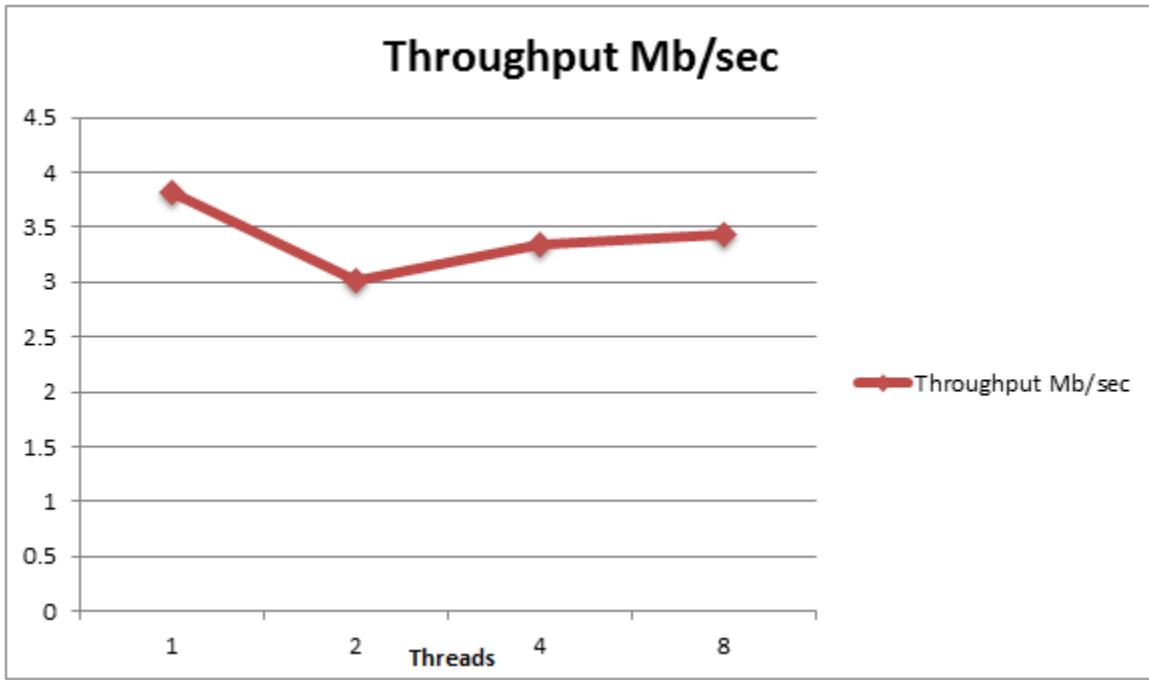
Every thread is responsible for reading a file and processing it achieves the sort functionality

- 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



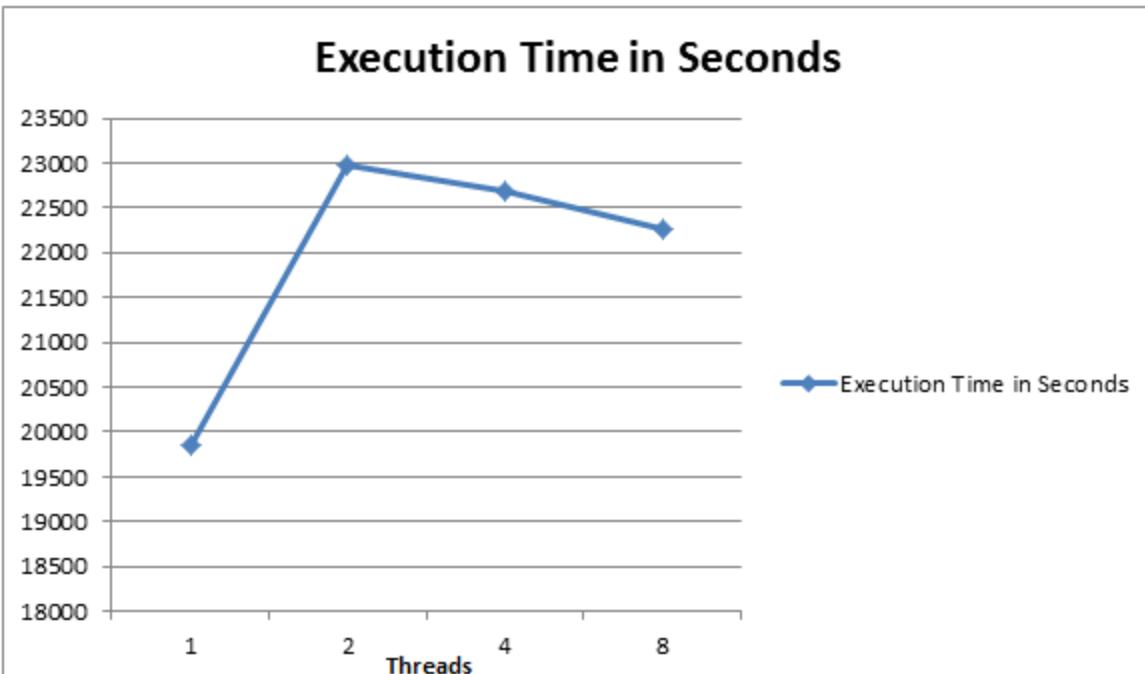
100GB DATASET ON c3.large instance:

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

EXPLANATION:

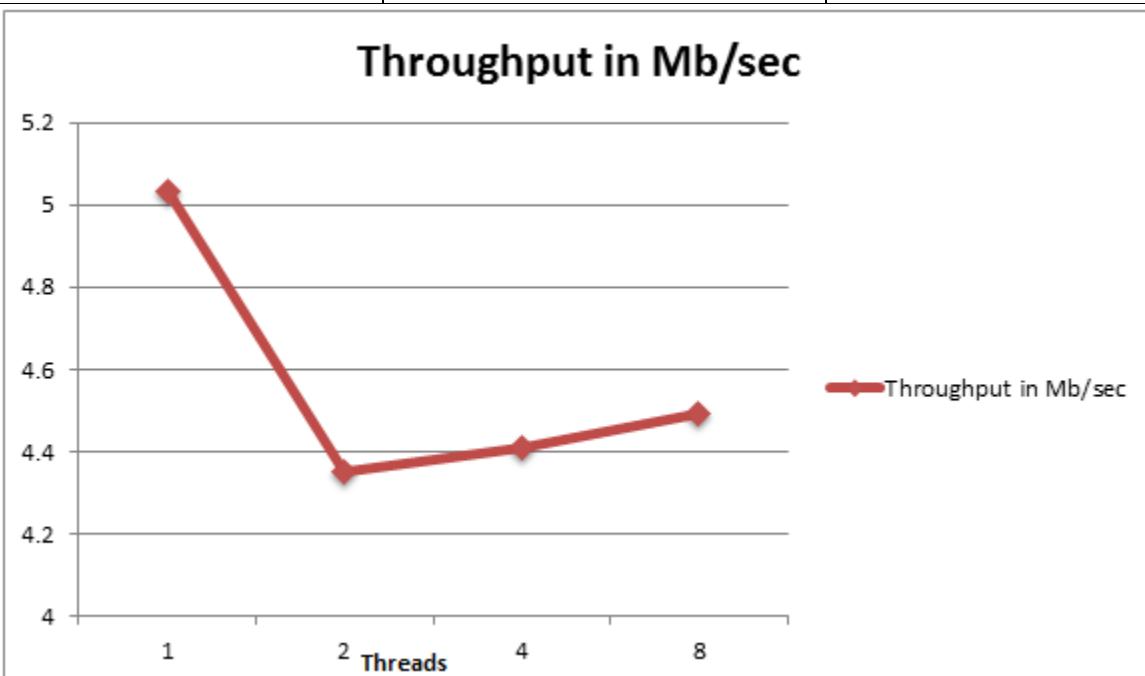
In a multithreaded process on a single processor, the processor can switch execution resources between threads, resulting in concurrent execution.

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



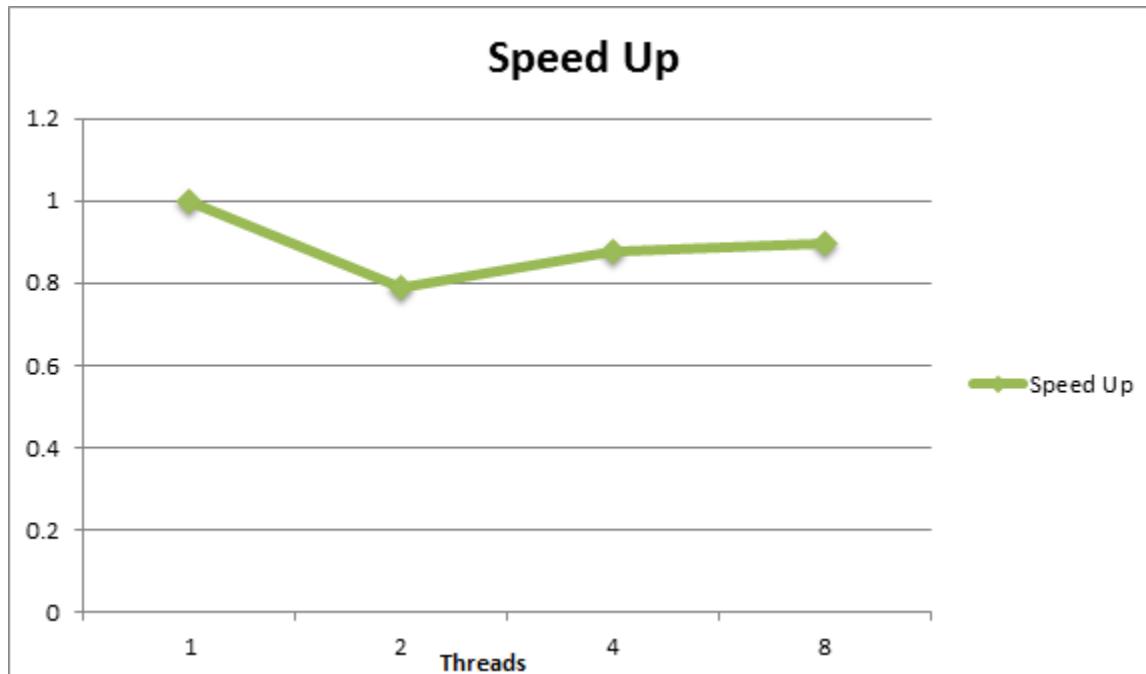
100GB Throughput:

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



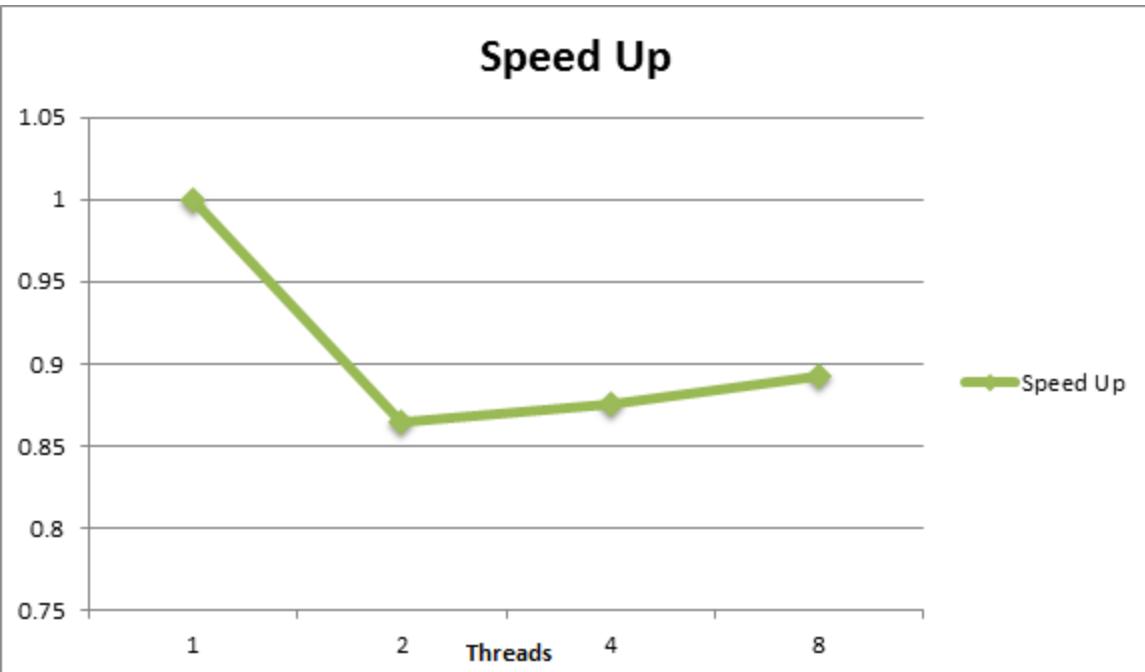
Speed Up data for 1 Node on 10GB dataset

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

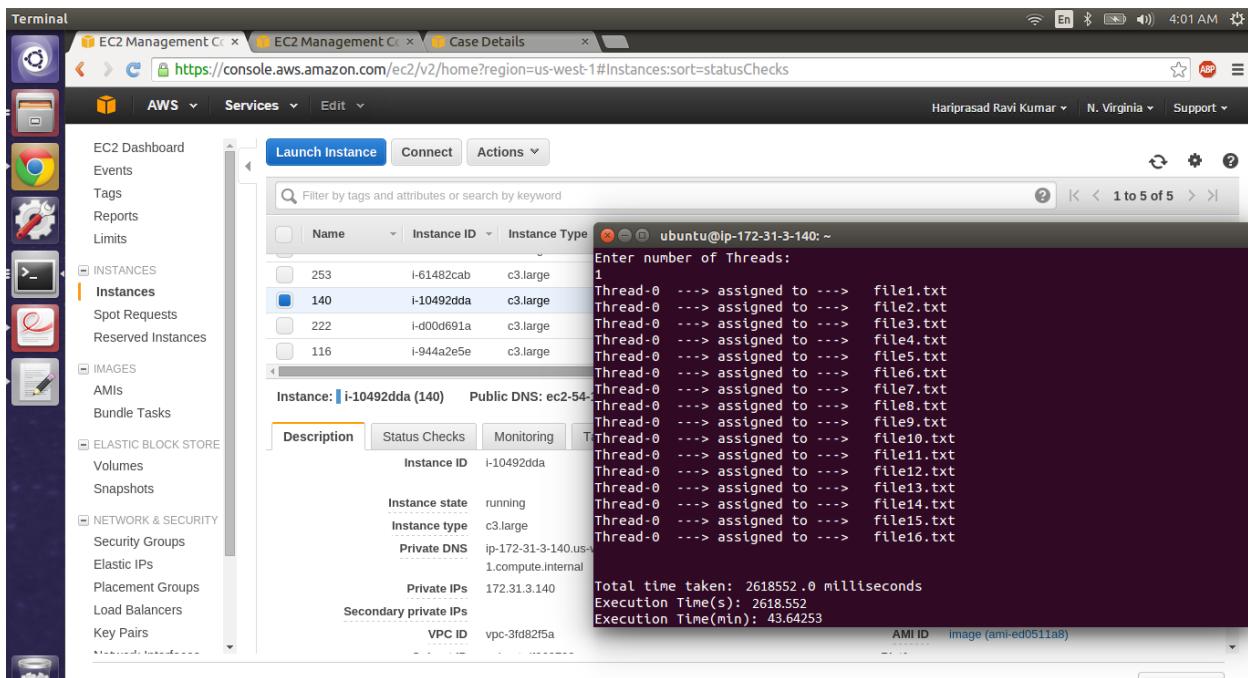


Speed Up data for 1 Node on 100GB dataset

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:



The screenshot shows the AWS EC2 Management Console interface. On the left, there's a sidebar with various AWS services like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, AMIs, and more. The main area shows a list of instances. One instance, i-d00d691a (ID 222), is selected and shown in detail. The instance is running, has a c3.large type, and a private IP of 172.31.14.222. A terminal window is open on this instance, displaying a multi-threaded file processing script. The script creates 16 files (file1.txt to file16.txt) in parallel using threads 0 through 15. The output includes execution times and a total time taken.

```
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-0 -> assigned to -> file5.txt
Thread-1 -> assigned to -> file6.txt
Thread-0 -> assigned to -> file7.txt
Thread-1 -> assigned to -> file8.txt
Thread-0 -> assigned to -> file9.txt
Thread-1 -> assigned to -> file10.txt
Thread-0 -> assigned to -> file11.txt
Thread-1 -> assigned to -> file12.txt
Thread-0 -> assigned to -> file13.txt
Thread-1 -> assigned to -> file14.txt
Thread-0 -> assigned to -> file15.txt
Thread-1 -> assigned to -> file16.txt

Instance: i-d00d691a (222) Public DNS: ec2-54-18-172-31-14-222.us-west-2.compute.internal
Description Status Checks Monitoring Tags
Instance ID i-d00d691a
Instance state running
Instance type c3.large
Private DNS ip-172-31-14-222.us-west-2.compute.internal
Private IPs 172.31.14.222
Secondary private IPs
VPC ID vpc-3fd82f5a

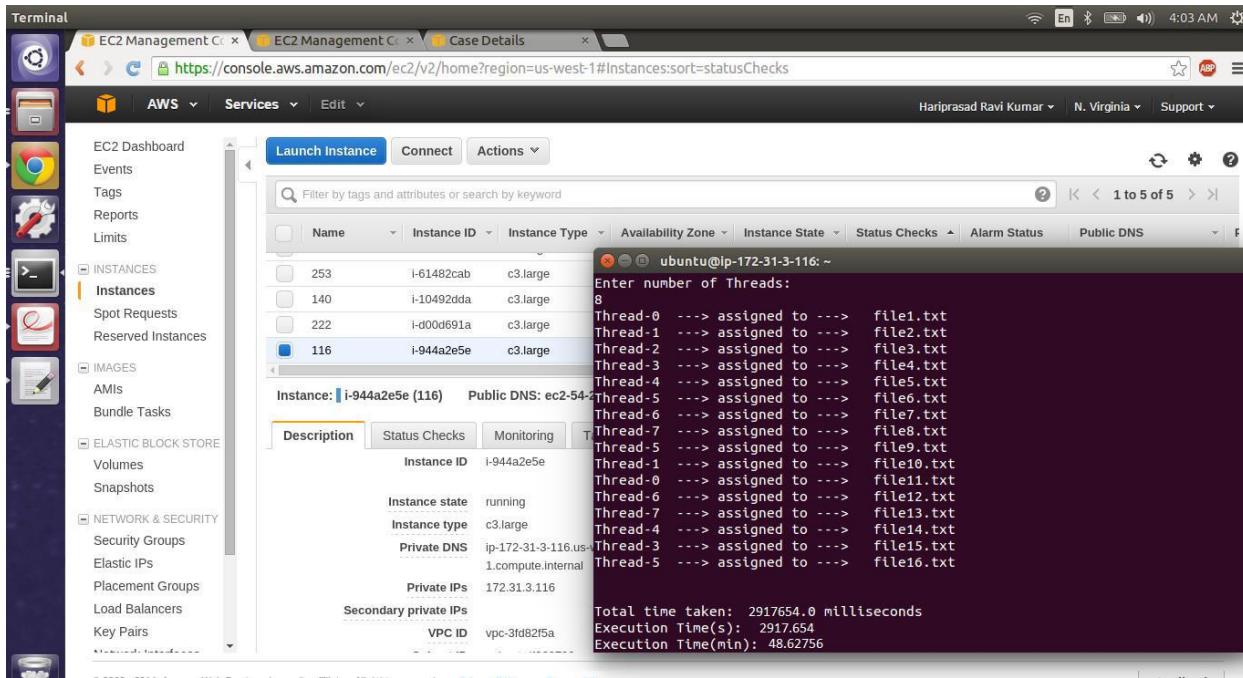
Total time taken: 3313751.0 milliseconds
Execution Time(s): 3313.751
Execution Time(min): 55.22918
```

This screenshot is similar to the first one, showing the AWS EC2 Management Console with a terminal session on instance i-61482cab (ID 253). The instance is running, has a c3.large type, and a private IP of 172.31.13.253. The terminal window displays a multi-threaded file processing script using 4 threads. It creates 16 files (file1.txt to file16.txt) in parallel. The output shows the execution times for each thread and the total time taken.

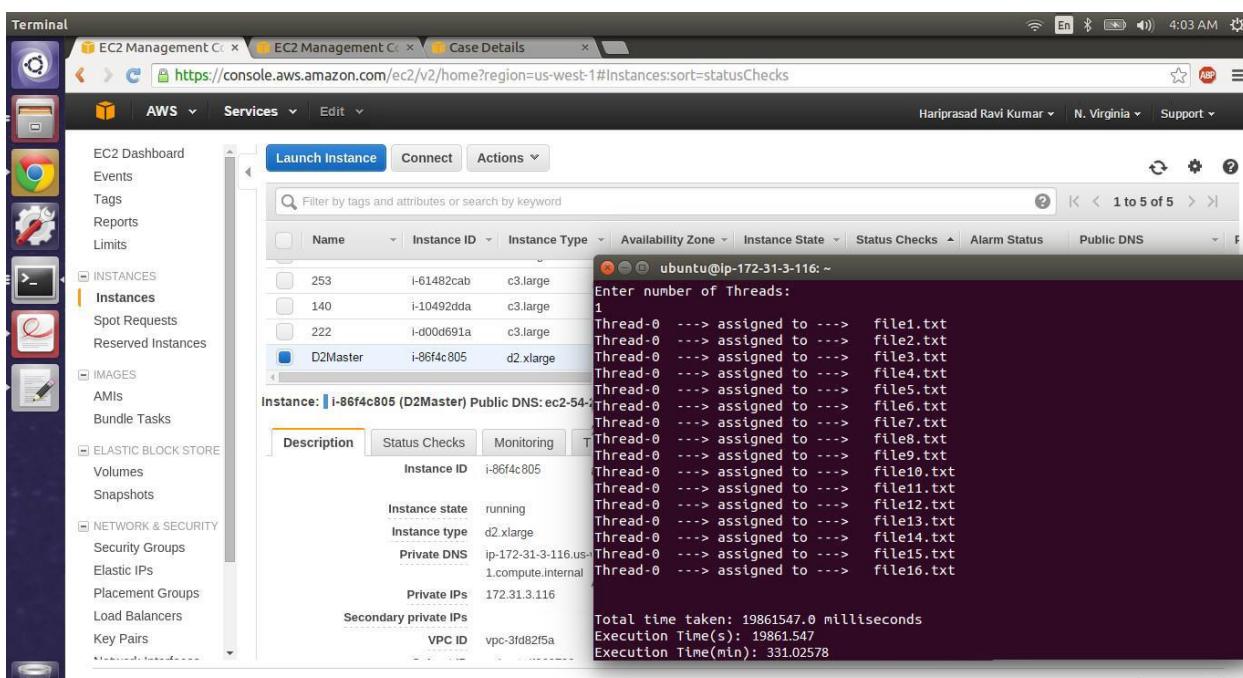
```
ubuntu@ip-172-31-13-253: ~
Enter number of Threads:
4
Thread-0 -> assigned to -> file1.txt
Thread-1 -> assigned to -> file2.txt
Thread-2 -> assigned to -> file3.txt
Thread-3 -> assigned to -> file4.txt
Thread-1 -> assigned to -> file5.txt
Thread-0 -> assigned to -> file6.txt
Thread-3 -> assigned 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

Instance: i-61482cab (253) Public DNS: ec2-54-23-172-31-13-253.us-west-2.compute.internal
Description Status Checks Monitoring Tags
Instance ID i-61482cab
Instance state running
Instance type c3.large
Private DNS ip-172-31-13-253.us-west-2.compute.internal
Private IPs 172.31.13.253
Secondary private IPs
VPC ID vpc-3fd82f5a

Total time taken: 2989987.0 milliseconds
Execution Time(s): 2989.987
Execution Time(min): 49.83311
```



Output Screen Shots for 100GB dataset:



Terminal

EC2 Management C Case Details https://console.aws.amazon.com/ec2/v2/home?region=us-west-1#Instances:sort=statusChecks

Hariprasad Ravi Kumar N. Virginia Support

AWS Services Edit

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

Name	Instance ID	Instance Type
253	i-61482cab	c3.large
140	i-10492dda	c3.large
222	i-d00d691a	c3.large
D2Master	i-86f4c805	d2.xlarge

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

Instance: i-86f4c805 (D2Master) Public DNS: ec2-54-172-31-3-116.us-east-1.compute.internal

Description Status Checks Monitoring

Instance ID: i-86f4c805

Instance state: running

Instance type: d2.xlarge

Private DNS: ip-172-31-3-116.us-east-1.compute.internal

Private IPs: 172.31.3.116

Secondary private IPs:

VPC ID: vpc-3fd82f5a

Total time taken: 22979632.0 milliseconds
Execution Time(s): 22979.632
Execution Time(min): 382.99386

Terminal

EC2 Management C Case Details https://console.aws.amazon.com/ec2/v2/home?region=us-west-1#Instances:sort=statusChecks

Hariprasad Ravi Kumar N. Virginia Support

AWS Services Edit

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

Name	Instance ID	Instance Type
253	i-61482cab	c3.large
140	i-10492dda	c3.large
222	i-d00d691a	c3.large
D2Master	i-86f4c805	d2.xlarge

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-2 -> assigned to -> file11.txt
Thread-0 -> assigned to -> file12.txt
Thread-1 -> assigned to -> file13.txt
Thread-0 -> assigned to -> file14.txt
Thread-3 -> assigned to -> file15.txt
Thread-2 -> assigned to -> file16.txt

Instance: i-86f4c805 (D2Master) Public DNS: ec2-54-172-31-3-116.us-east-1.compute.internal

Description Status Checks Monitoring

Instance ID: i-86f4c805

Instance state: running

Instance type: d2.xlarge

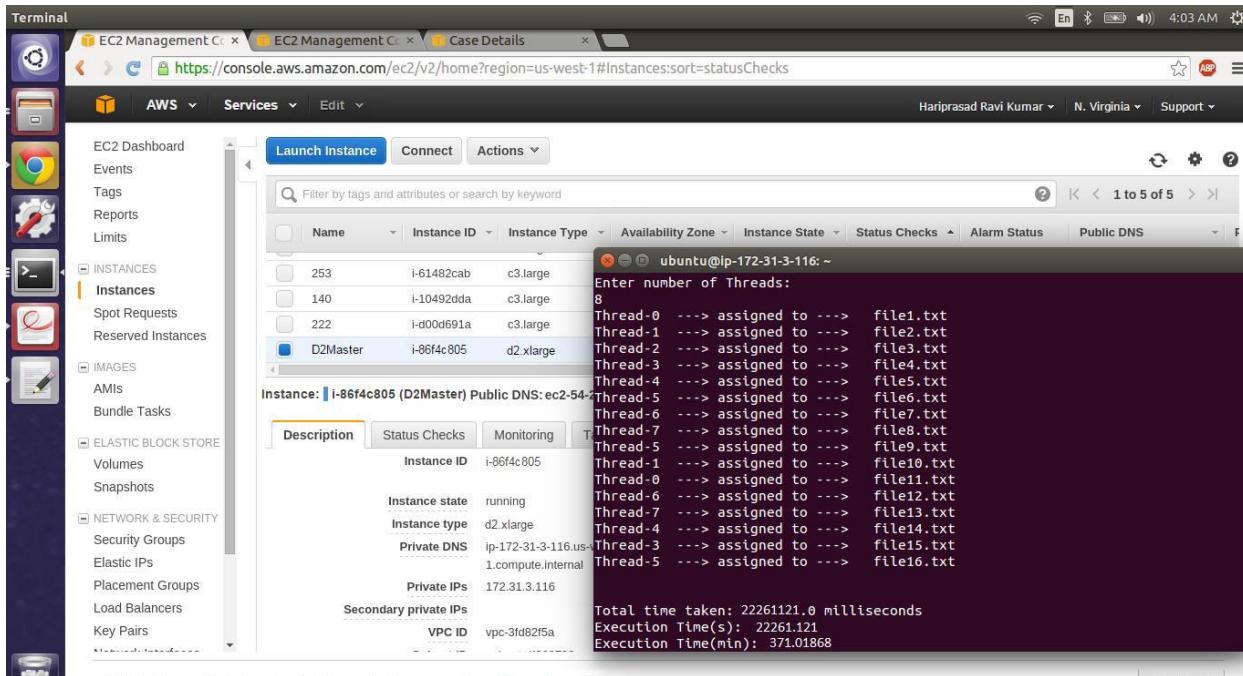
Private DNS: ip-172-31-3-116.us-east-1.compute.internal

Private IPs: 172.31.3.116

Secondary private IPs:

VPC ID: vpc-3fd82f5a

Total time taken: 22681487.0 milliseconds
Execution Time(s): 22681.487
Execution Time(min): 387.02478



```
ubuntu@ip-172-31-3-116: ~$ ./gnsort /gnsort-linux-1.5/64$ ./valsrt /home/hari/Downloads/finalterasort100
Records: 1000000000000
Checksum: 7a27e2d0d55de
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-3-116: ~$ ./gnsort /gnsort-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):

The screenshot shows the AWS EC2 Management Console interface. The left sidebar navigation bar includes links for EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, Elastic Block Store, Network & Security, and more. The main content area displays a table of 17 EC2 instances. The table columns are: Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, and Public DNS. The instances are labeled as follows:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
Master	i-8c789680	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-69-130-130.us...
Slave1	i-784ea074	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-111-103.us...
Slave2	i-ad739da1	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-112-169.us...
Slave3	i-27719f2b	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-187-214-27.us...
Slave4	i-99709e95	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-187-156-49.us...
Slave5	i-ae739da2	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-68-182-246.us...
Slave6	i-98709e94	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-69-241-219.us...
Slave7	i-ac739da0	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-124-168.us...
Slave8	i-87709e8b	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-111-212.us...
Slave9	i-b3739dbf	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-56-160.us...
Slave10	i-b2739dbe	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-132-113.us...
Slave11	i-86709e8a	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-127-242.us...
Slave12	i-24719f2c	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-112-159.us...
Slave13	i-20719f2c	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-187-99-109.us...
Slave14	i-fd719ff1	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-69-7-191.us-we...
Slave15	i-454ea049	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-187-248-111.us...
Slave16	i-85709e89	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-117-106.us...

This screenshot is identical to the one above, showing the AWS EC2 Management Console with 17 instances listed. The difference is that the instance "Slave16" is now highlighted with a blue selection bar at the bottom of the table row. The rest of the interface and data remain the same.

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/install/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/install/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:

The screenshot shows the AWS EC2 Management Console in Google Chrome. The URL is <https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#instances:sort=launchTime>. The left sidebar shows navigation options like EC2 Dashboard, Instances, Images, and Network & Security. The main area displays a table of instances. One instance, named 'Master' with Instance ID i-ebbc55e7, is highlighted. The details pane below shows the instance's configuration: Instance ID i-ebbc55e7, Public DNS ec2-50-112-178-37.us-west-2.compute.amazonaws.com, Instance state running, Instance type c3.large, Private DNS ip-172-31-2-1.us-west-2.compute.internal, and Availability zone us-west-2a.

This screenshot shows the same AWS EC2 Management Console interface. A different slave node, S3, with Instance ID i-74b35a78, is selected. The details pane shows its configuration: Instance ID i-74b35a78, Public DNS ec2-54-186-115-3.us-west-2.compute.amazonaws.com, Instance state running, Instance type c3.large, Private DNS ip-172-31-30-43.us-west-2.compute.internal, and Availability zone us-west-2a.

EC2 Management Console - Google Chrome

EC2 Management Cr 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 Hadoop Tutorial Apache Hadoop Hadoop v2 overv

Hariprasad Ravi Kumar N. Virginia Support

AWS Services Edit

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

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-2bb55cc27	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-2ab55cc26	c3.large	us-west-2a	running	Initializing	None	ec2-50-112-183-85.us...
	i-28b55cc24	c3.large	us-west-2a	running	Initializing	None	ec2-54-186-235-79.us...

Instance: i-2ab55cc26 Public DNS: ec2-50-112-183-85.us-west-2.compute.amazonaws.com

Description Status Checks Monitoring Tags

Instance ID	i-2ab55cc26	Public DNS	ec2-50-112-183-85.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	50.112.183.85
Instance type	c3.large	Elastic IP	-
Private DNS	ip-172-31-21-125.us-west-	Availability zone	us-west-2a

EC2 Management Console - Google Chrome

EC2 Management Cr 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 Hadoop Tutorial Apache Hadoop Hadoop v2 overv

Hariprasad Ravi Kumar N. Virginia Support

AWS Services Edit

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

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-2ab55cc26	c3.large	us-west-2a	running	Initializing	None	ec2-50-112-183-85.us...
S12	i-28b55cc24	c3.large	us-west-2a	running	Initializing	None	ec2-54-186-235-79.us...
S13	i-1b1250fd	c3.large	us-west-2a	running	Initializing	None	ec2-54-186-184-147.us...
S14	i-29b55cc25	c3.large	us-west-2a	running	Initializing	None	ec2-54-69-184-170.us...
S15	i-2fb55cc23	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...

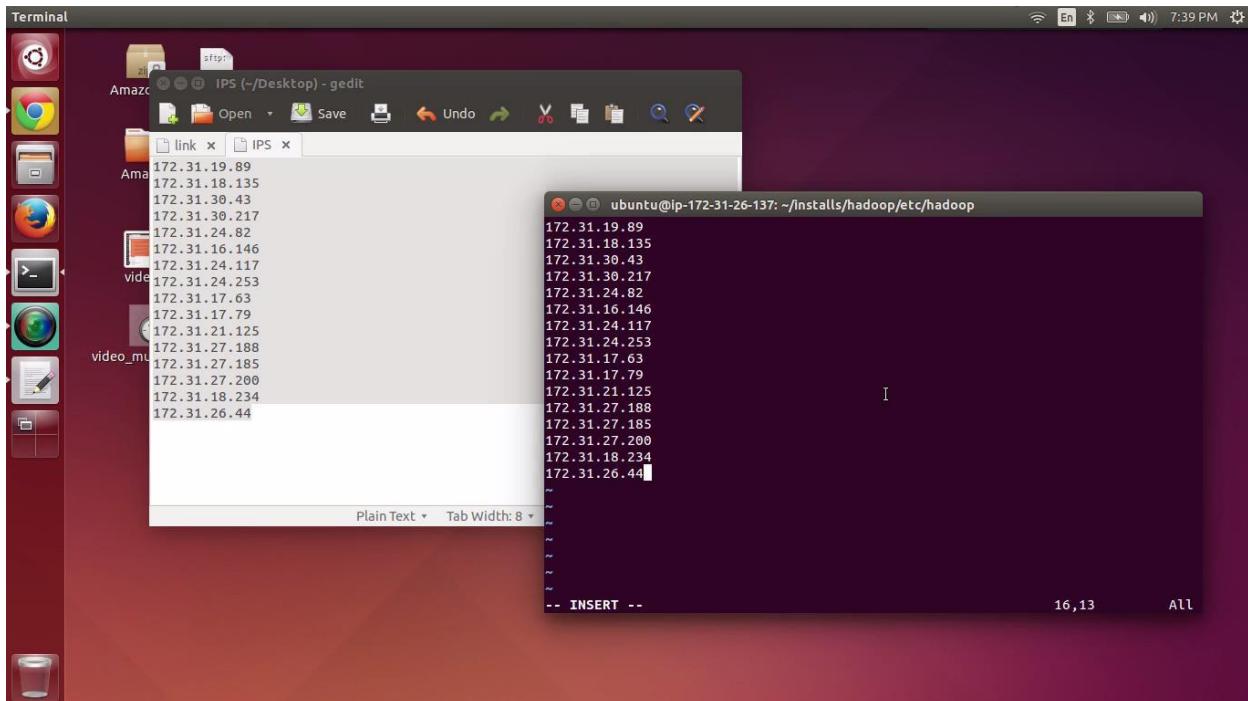
Instance: i-c5b55cc9 (S16) Public DNS: ec2-54-186-248-186.us-west-2.compute.amazonaws.com

Description Status Checks Monitoring Tags

Instance ID	i-c5b55cc9	Public DNS	ec2-54-186-248-186.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	54.186.248.186
Instance type	c3.large	Elastic IP	-
Private DNS	ip-172-31-26-44.us-west-	Availability zone	us-west-2a

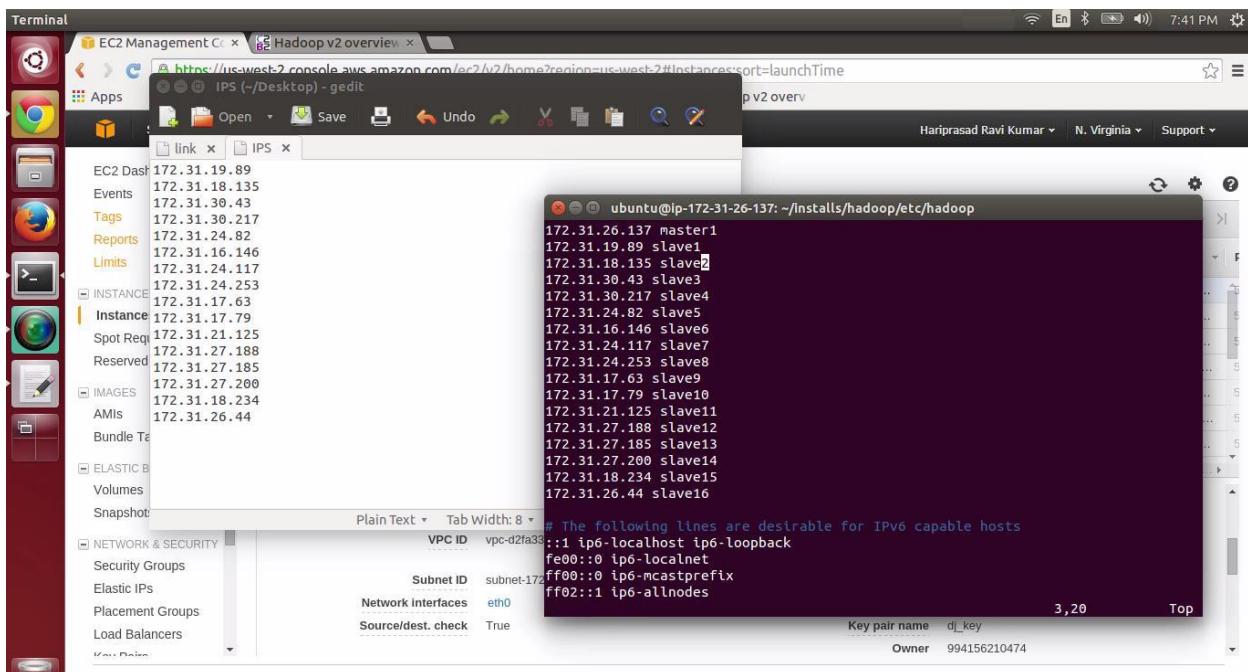
7. / conf /slaves:

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:

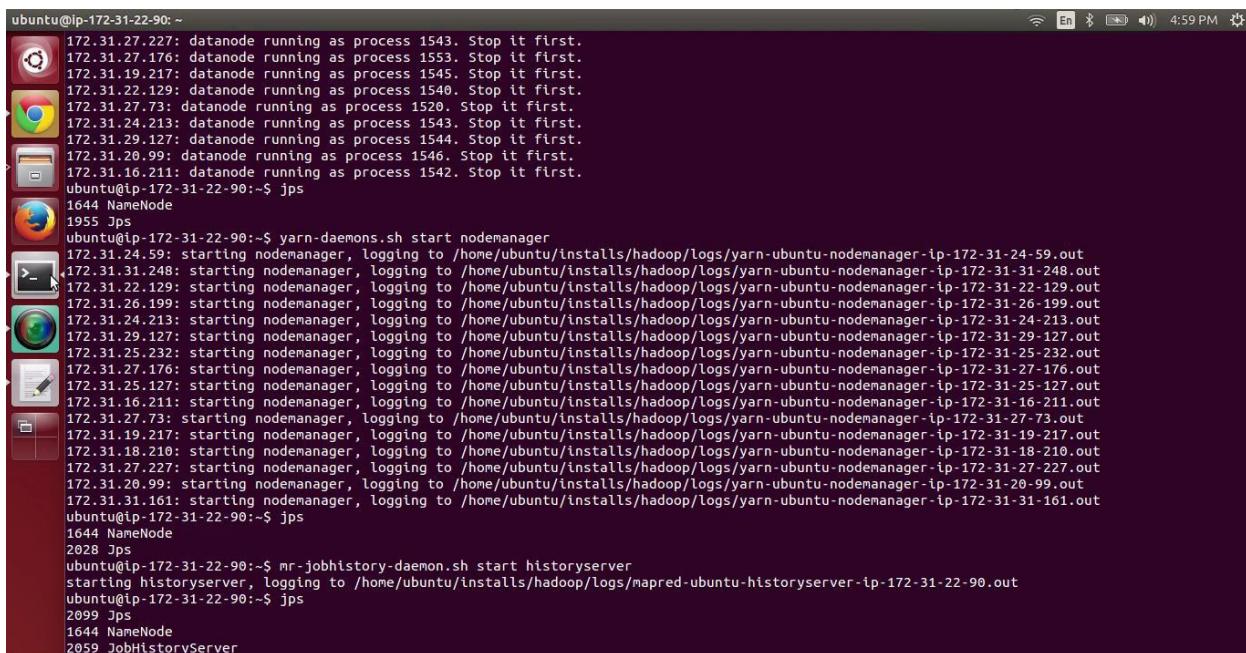
- `$<HADOOP_HOME>/yarn-daemon.sh start resourcemanager`

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:

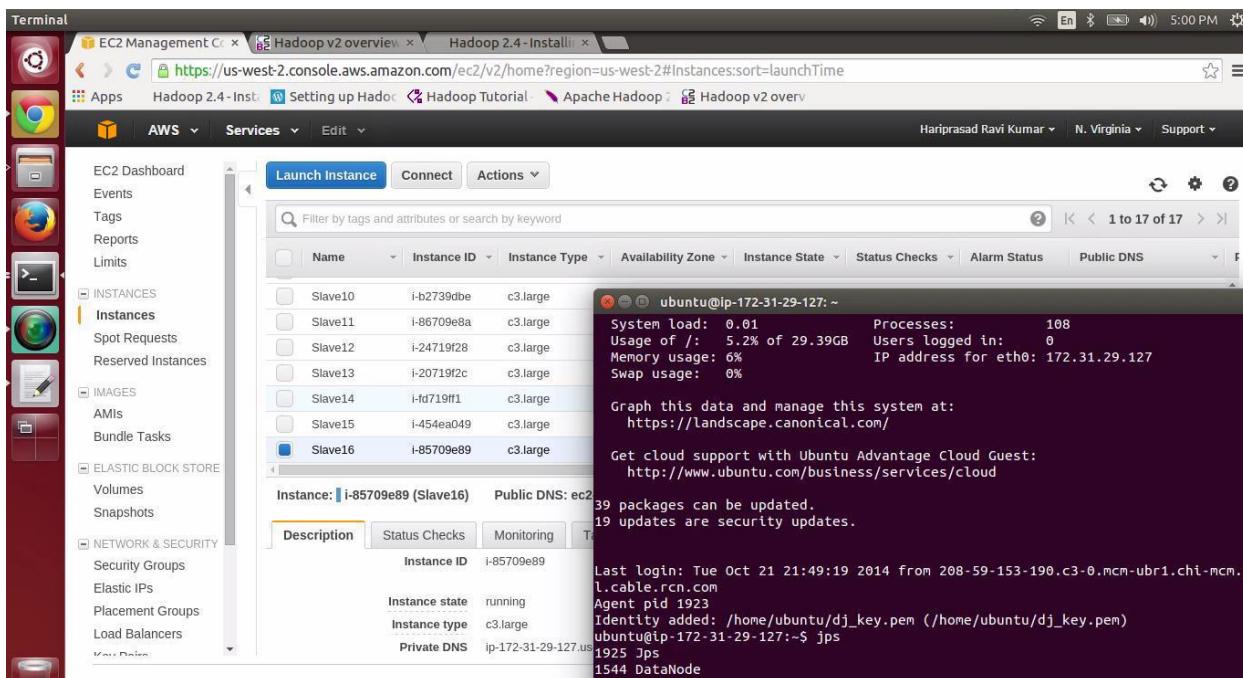
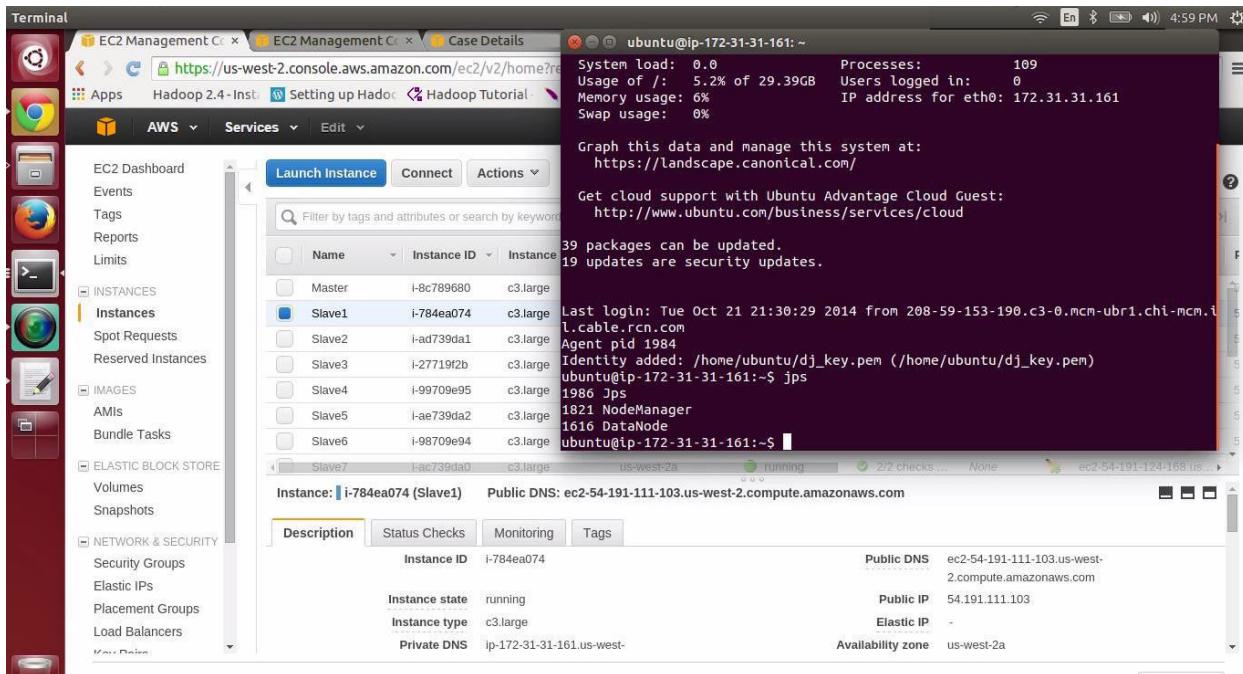
- `$<HADOOP_HOME>/mr-jobhistory-daemon.sh start historyserver`



The screenshot shows a terminal window on an Ubuntu 12.04 desktop environment. The terminal output is as follows:

```
ubuntu@ip-172-31-22-90: ~
172.31.27.227: datanode running as process 1543. Stop it first.
172.31.27.16: datanode running as process 1553. Stop it first.
172.31.19.217: datanode running as process 1545. Stop it first.
172.31.22.129: datanode running as process 1540. Stop it first.
172.31.27.73: datanode running as process 1520. Stop it first.
172.31.24.213: datanode running as process 1543. Stop it first.
172.31.29.127: datanode running as process 1544. Stop it first.
172.31.28.99: datanode running as process 1546. Stop it first.
172.31.16.211: datanode running as process 1542. Stop it first.
ubuntu@ip-172-31-22-90:~$ jps
1644 NameNode
1955 Jps
ubuntu@ip-172-31-22-90:~$ yarn-daemons.sh start nodemanager
172.31.24.59: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-24-59.out
172.31.31.248: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-31-248.out
172.31.22.129: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-22-129.out
172.31.26.199: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-26-199.out
172.31.24.213: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-24-213.out
172.31.29.127: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-29-127.out
172.31.25.232: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-25-232.out
172.31.27.176: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-27-176.out
172.31.25.127: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-25-127.out
172.31.16.211: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-16-211.out
172.31.27.73: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-27-73.out
172.31.19.217: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-19-217.out
172.31.18.210: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-18-210.out
172.31.27.227: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-27-227.out
172.31.31.161: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-31-161.out
ubuntu@ip-172-31-22-90:~$ jps
1644 NameNode
2028 Jps
ubuntu@ip-172-31-22-90:~$ mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/ubuntu/install/hadoop/logs/mapred-ubuntu-historyserver-ip-172-31-22-90.out
ubuntu@ip-172-31-22-90:~$ jps
2099 Jps
1644 NameNode
2059 JobHistoryServer
```

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



Screen Shots of the Configuration FILES:

~/.bashrc: conf

```
ubuntu@ip-172-31-22-90: ~
alias l='ls -CF'
# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "$(($RANDOM % 8))" echo "terminal || echo error" "$($history|tail -n1|sed -e "s/^/\s*([0-9])\+\s*//;s/[;]{2}/;/g")"'*
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/bash_aliases ]; then
  . ~/bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/home/ubuntu/install/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=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
eval $(ssh-agent);
ssh-add /home/ubuntu/dj.key.pem
-- INSERT --
```

conf /hadoop-env.sh

```
ubuntu@ip-172-31-22-90: ~/install/hadoop/etc/hadoop
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# 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.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=${JAVA_HOME}

# The jsvc implementation to use. Jsvc is required to run secure datanodes.
#export JSVC_HOME=${JSVC_HOME}

export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
# Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
  if [ "SHADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
    export HADOOP_CLASSPATH=$f
  fi
done

:wq
```

/conf/core-site.xml:

/conf/yarn-site.xml:

```
ubuntu@ip-172-31-22-90: ~/installs/hadoop/etc/hadoop
<?xml version="1.0"?>
<!--
 Licensed under the Apache License, Version 2.0 (the "License");
 you may not use this file except in compliance with the License.
 You may obtain a copy of the License at

 http://www.apache.org/licenses/LICENSE-2.0

 Unless required by applicable law or agreed to in writing, software
 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>
<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>
```

/conf/hdfs-site.xml:

```
ubuntu@ip-172-31-22-90: ~/installs/hadoop/etc/hadoop
<?xml version="1.0" encoding="UTF-8"?>
<xmlstylesheet type="text/xsl" href="configuration.xsl"?>
<!--
    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    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.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
    <name>dfs.replication</name>
    <value>16</value>
</property>
<property>
    <name>dfs.permissions</name>
    <value>false</value>
</property>
</configuration>
~
```

/conf/mapred-site.xml:

```
ubuntu@ip-172-31-22-90: ~/Installs/hadoop/etc/hadoop
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

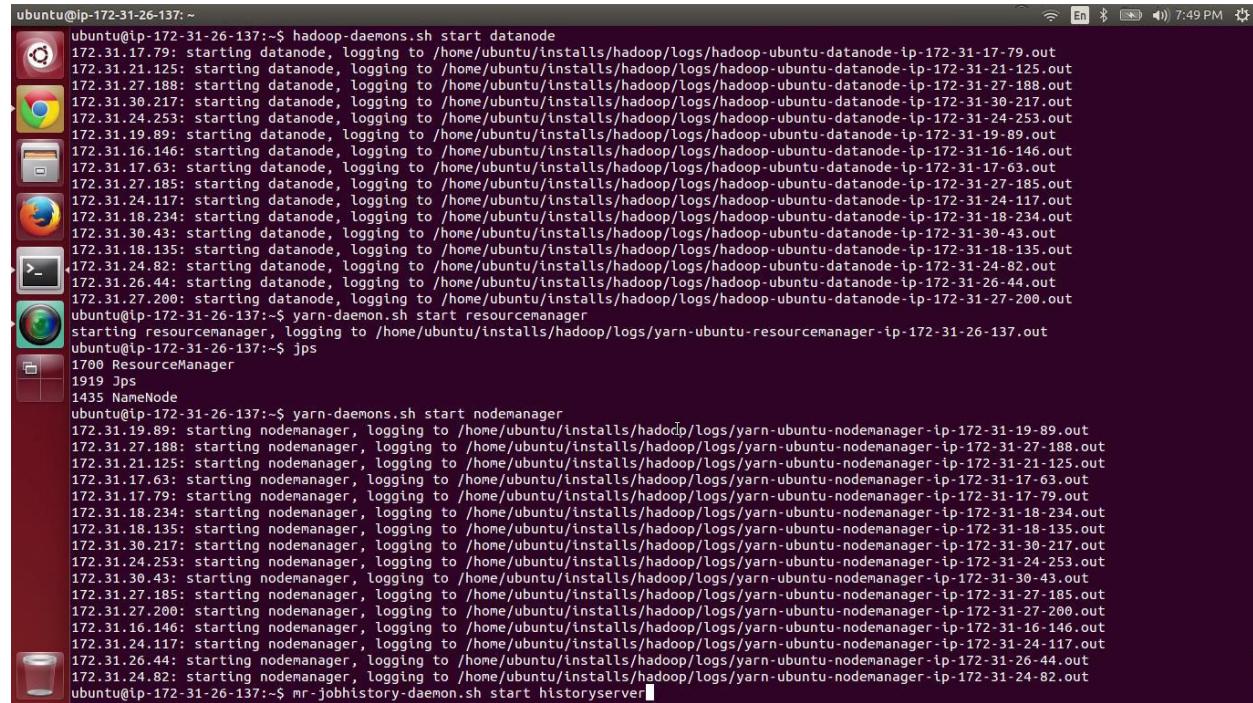
        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    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.
-->
<!-- Put site-specific property overrides in this file. -->

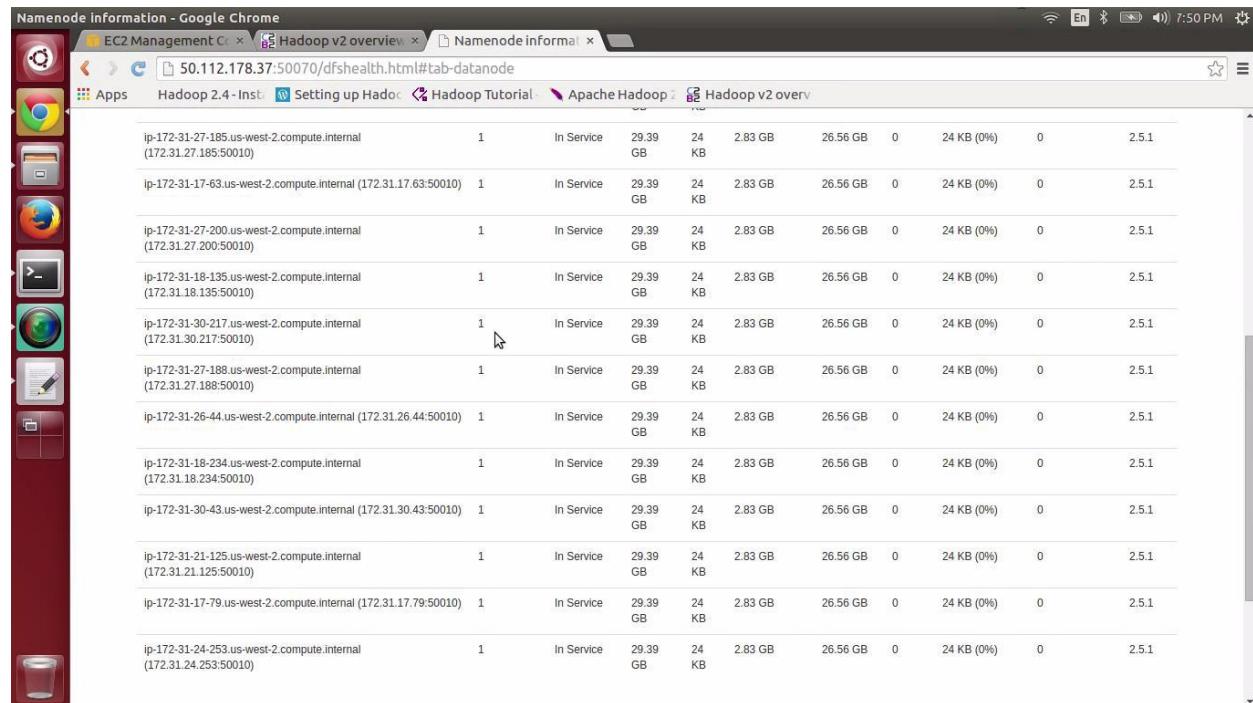
<configuration>
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
</configuration>
```

Connecting to Data Nodes:

Screen Shots:



```
ubuntu@ip-172-31-26-137:~$ hadoop-daemons.sh start datanode
172.31.17.79: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-17-79.out
172.31.21.125: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-21-125.out
172.31.27.188: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-27-188.out
172.31.30.217: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-30-217.out
172.31.24.253: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-24-253.out
172.31.19.89: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-19-89.out
172.31.16.146: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-16-146.out
172.31.17.63: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-17-63.out
172.31.27.185: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-27-185.out
172.31.24.117: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-24-117.out
172.31.18.234: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-18-234.out
172.31.30.43: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-30-43.out
172.31.18.135: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-18-135.out
172.31.24.82: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-24-82.out
172.31.26.44: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-26-44.out
172.31.27.200: starting datanode, logging to /home/ubuntu/install/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-27-200.out
ubuntu@ip-172-31-26-137:~$ yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-resourcemanager-ip-172-31-26-137.out
ubuntu@ip-172-31-26-137:~$ jps
1700 ResourceManager
1919 Jps
1435 NameNode
ubuntu@ip-172-31-26-137:~$ yarn-daemons.sh start nodemanager
172.31.19.89: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-19-89.out
172.31.27.188: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-27-188.out
172.31.21.125: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-21-125.out
172.31.17.63: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-17-63.out
172.31.17.79: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-17-79.out
172.31.18.234: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-18-234.out
172.31.18.135: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-18-135.out
172.31.30.217: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-30-217.out
172.31.24.253: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-24-253.out
172.31.30.43: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-30-43.out
172.31.27.185: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-27-185.out
172.31.27.200: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-27-200.out
172.31.16.146: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-16-146.out
172.31.24.117: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-24-117.out
172.31.26.44: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-26-44.out
172.31.24.82: starting nodemanager, logging to /home/ubuntu/install/hadoop/logs/yarn-ubuntu-nodemanager-ip-172-31-24-82.out
ubuntu@ip-172-31-26-137:~$ mr-jobhistory-daemon.sh start historyserver
```



Namenode Information - Google Chrome										
EC2 Management C... Hadoop v2 overview X Namenode informa... X										
50.112.178.37:50070/dshealth.html#tab-datanode										
Apps Hadoop 2.4-Inst. Setting up Hadoop Hadoop Tutorial Apache Hadoop Hadoop v2 overv...										
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
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

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
in-172-31-18-234.us-west-2.compute.internal	1	In Service	29.39	24	2.83 GB	26.56 GB	0	24 KB (0%)	0	2.5.1

Apps	Hadoop 2.4 - Inst: ip-172-31-30-21.us-west-2.compute.internal (172.31.30.21:50010)	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

Decommissioning

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 100000000000 10GBdata
./gensort -a 1000000000000 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

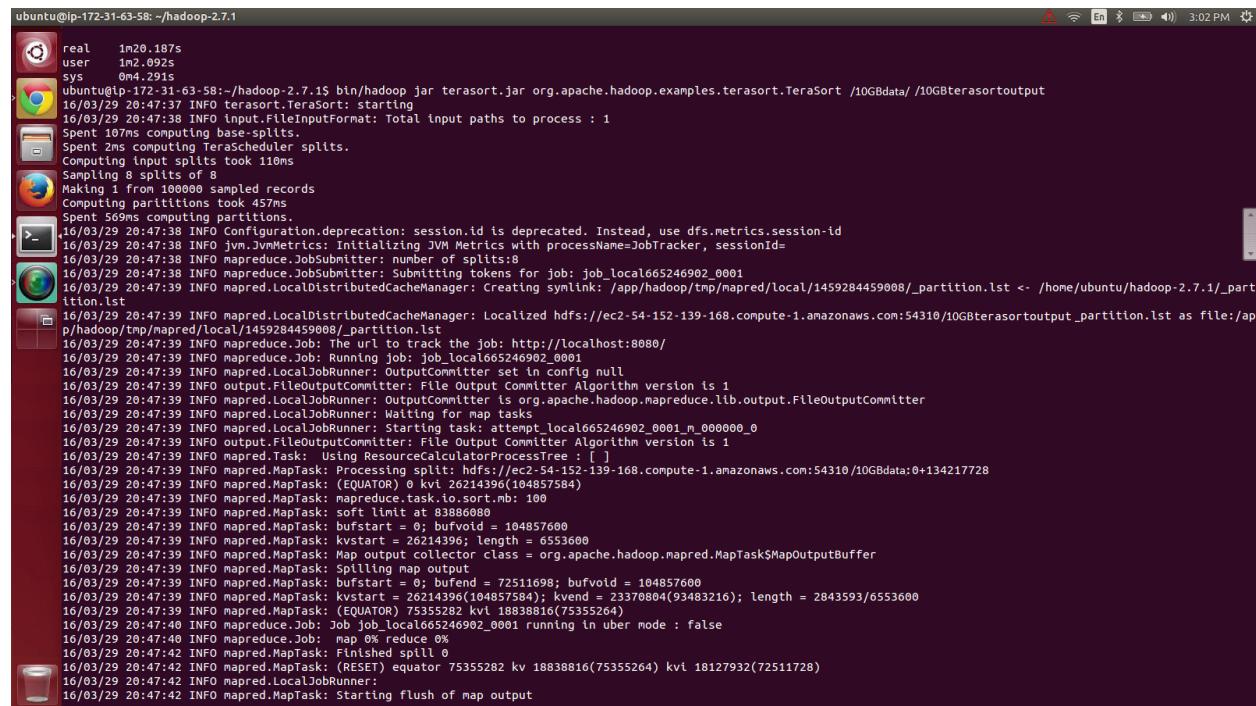
Outputs:

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
real    1m20.187s
user    1m2.092s
sys     0m4.291s
ubuntu@ip-172-31-63-58:~/hadoop jar terasort.jar org.apache.hadoop.examples.terasort.TeraSort /10GBdata/ /10GBterasortoutput
16/03/29 20:47:37 INFO terasort.TeraSort: starting
16/03/29 20:47:38 INFO InputFileInputFormat: Total input paths to process : 1
Spent 107ms computing base-splits
Spent 2ms computing Terascheduler splits.
Computing input splits took 110ms
Sampling 8 splits of 8
Making 1 from 100000 sampled records
Computing partitions took <57ms
Spent 569ms computing partitions.
16/03/29 20:47:38 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
16/03/29 20:47:38 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
16/03/29 20:47:38 INFO mapred.JobSubmitter: number of splits:8
16/03/29 20:47:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local655246902_0001
16/03/29 20:47:39 INFO mapred.LocalDistributedCacheManager: Creating symlink: /app/hadoop/tmp/mapred/local/1459284459008/_partition.lst <- /home/ubuntu/hadoop-2.7.1/partition.lst
16/03/29 20:47:39 INFO mapred.LocalDistributedCacheManager: Localized hdfs://ec2-54-152-139-168.compute-1.amazonaws.com:54310/10GBterasortoutput_partition.lst as file:/app/hadoop/tmp/mapred/local/1459284459008/_partition.lst
16/03/29 20:47:39 INFO mapred.Job: The url to track the job: http://Localhost:8080/
16/03/29 20:47:39 INFO mapred.Job: Running job: job_local655246902_0001
16/03/29 20:47:39 INFO mapred.LocalJobRunner: OutputCommitter set in config null
16/03/29 20:47:39 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
16/03/29 20:47:39 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
16/03/29 20:47:39 INFO mapred.LocalJobRunner: Waiting for map tasks
16/03/29 20:47:39 INFO mapred.LocalJobRunner: Starting task: attempt_local655246902_0001_m_000000_0
16/03/29 20:47:39 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
16/03/29 20:47:39 INFO mapred.Task: Using ResourceCalculatorProcessTree : []
16/03/29 20:47:39 INFO mapred.MapTask: Processing split: hdfs://ec2-54-152-139-168.compute-1.amazonaws.com:54310/10GBdata:0+134217728
16/03/29 20:47:39 INFO mapred.MapTask: (EQUATOR) 0 kvl 26214396(104857584)
16/03/29 20:47:39 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
16/03/29 20:47:39 INFO mapred.MapTask: soft limit at 83886080
16/03/29 20:47:39 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
16/03/29 20:47:39 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
16/03/29 20:47:39 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
16/03/29 20:47:39 INFO mapred.MapTask: Spilling map output
16/03/29 20:47:39 INFO mapred.MapTask: bufstart = 0; bufend = 72511698; bufvoid = 104857600
16/03/29 20:47:39 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 23370804(93483216); length = 2843593/6553600
16/03/29 20:47:39 INFO mapred.MapTask: (EQUATOR) 75355282 kvl 18838816(75355264)
16/03/29 20:47:40 INFO mapreduce.Job: Job job_local655246902_0001 running in uber mode : false
16/03/29 20:47:40 INFO mapreduce.Job: map 0% reduce 0%
16/03/29 20:47:42 INFO mapred.MapTask: Finished spill 0
16/03/29 20:47:42 INFO mapred.MapTask: (RESET) equator 75355282 kv 18838816(75355264) kvl 18127932(72511728)
16/03/29 20:47:42 INFO mapred.LocalJobRunner:
16/03/29 20:47:42 INFO mapred.MapTask: Starting flush of map output
```

```

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

EC2 Management > Hadoop 2.4-Inst... > Hadoop v2 overview

<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 Hadoop Hadoop Tutorial Apache Hadoop Hadoop v2 over... yadudoc/cloud-t...

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

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-west-2.compute.amazonaws.com
Slave1	i-e5b144e9	c3.large	us-west-2a	running	Initializing	None	ec2-54-69-16-92.us-west-2.compute.amazonaws.com

1 to 3 of 3

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

Public DNS: ec2-54-68-97-203.us-west-2.compute.amazonaws.com

Instance state: running

Public IP: 54.68.97.203

Instance type: c3.large

Elastic IP: -

Private DNS: ip-172-31-63-58.us-west-2.compute.amazonaws.com

Availability zone: us-west-2a

Namenode Information - Google Chrome

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-30-201.us-west-2.compute.internal (172.31.30.201:50010)	2	In Service	19.55 GB	9.76 GB	2.43 GB	7.35 GB	78	9.76 GB (49.94%)	0	2.5.1
ip-172-31-30-190.us-west-2.compute.internal (172.31.30.190:50010)	2	In Service	19.55 GB	9.76 GB	2.44 GB	7.35 GB	78	9.76 GB (49.94%)	0	2.5.1

Decommissioning

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

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:~
```

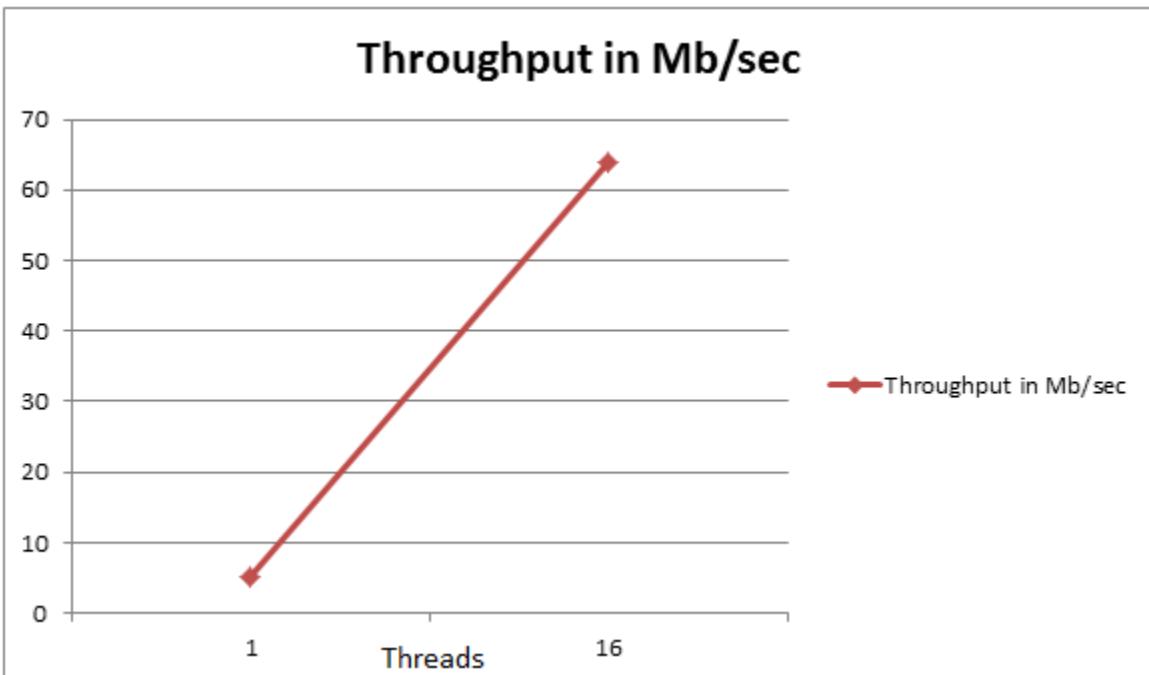
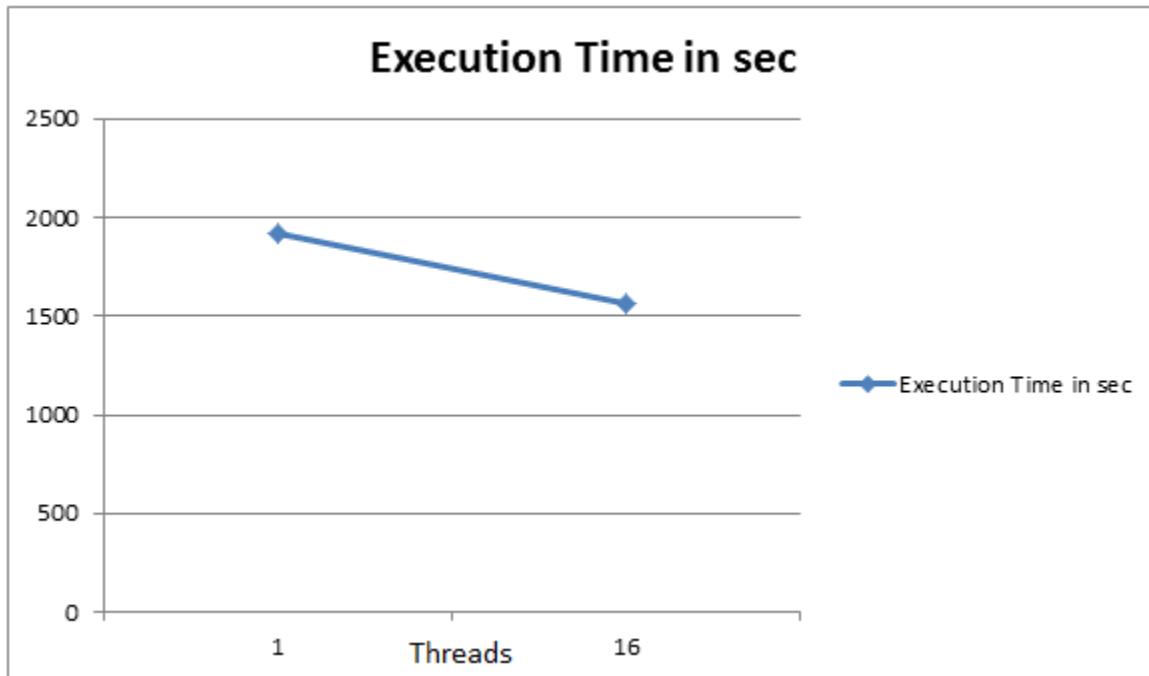
```
16/03/29 01:28:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
16/03/29 01:28:47 INFO client.RMProxy: Connecting to ResourceManager at /172.31.26.137:8040
16/03/29 01:28:47 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/03/29 01:28:48 INFO mapred.FileInputFormat: Total input paths to process : 1
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:0+134217728 splitsize: 1
6 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:134217728+134217728 splitsize: 16 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:268435456+134217728 splitsize: 16 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:402653184+134217728 splitsize: 16 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:536870912+134217728 splitsize: 16 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:671088640+134217728 splitsize: 16 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:805306368+134217728 splitsize: 16 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:939524096+134217728 splitsize: 16 maxsize: 10
16/03/29 01:28:48 WARN split.JobSplitWriter: Max block location exceeded for split: hdfs://172.31.26.137:9000/largedata:1073741824+134217728 splitsize: 16 maxsize: 10
```

```
ubuntu@ip-172-31-26-137: ~
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 01:54:05 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_141393895590_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		EC2 Management C		Namenode informa		JobHistory		Nodes of the cluster			
Apps		Hadoop 2.4-Inst		Setting up Hado		Hadoop Tutorial		Apache Hadoop		Hadoop v2 overv	
FINISHED FAILED KILLED		Scheduler		/default- rack		/default- rack		ip-172-31-24-82.us-west- 2.compute.internal:51746		29-Mar-2016 0 0 B 8 GB 0 E	
Tools											
								ip-172-31-24-82.us-west- 2.compute.internal:8042	01:19:28	0 0 B 8 GB 0 E	
								ip-172-31-17-63.us-west- 2.compute.internal:38581	01:19:28	29-Mar-2016 0 0 B 8 GB 0 E	
								ip-172-31-21-125.us-west- 2.compute.internal:44347	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-27-200.us-west- 2.compute.internal:49988	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-26-44.us-west- 2.compute.internal:41511	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-30-43.us-west- 2.compute.internal:36404	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-17-79.us-west- 2.compute.internal:46782	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-18-135.us-west- 2.compute.internal:36332	22-Oct-2014 01:19:28	8 8 GB 0 B 8 C	
								ip-172-31-24-117.us-west- 2.compute.internal:40161	29-Mar-2016 7 8 GB 0 B 7 I	8 8 GB 0 B 8 C	
								ip-172-31-30-217.us-west- 2.compute.internal:40483	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-18-234.us-west- 2.compute.internal:48312	01:19:28	29-Mar-2016 0 0 B 8 GB 0 E	
								ip-172-31-16-146.us-west- 2.compute.internal:56277	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-27-188.us-west- 2.compute.internal:58302	01:19:28	29-Mar-2016 0 0 B 8 GB 0 E	
								ip-172-31-24-253.us-west- 2.compute.internal:50833	01:19:28	29-Mar-2016 8 8 GB 0 B 8 C	
								ip-172-31-27-185.us-west- 2.compute.internal:53763	01:19:28	29-Mar-2016 0 0 B 8 GB 0 E	
								ip-172-31-19-89.us-west- 2.compute.internal:58013	01:19:28	29-Mar-2016 0 0 B 8 GB 0 E	

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

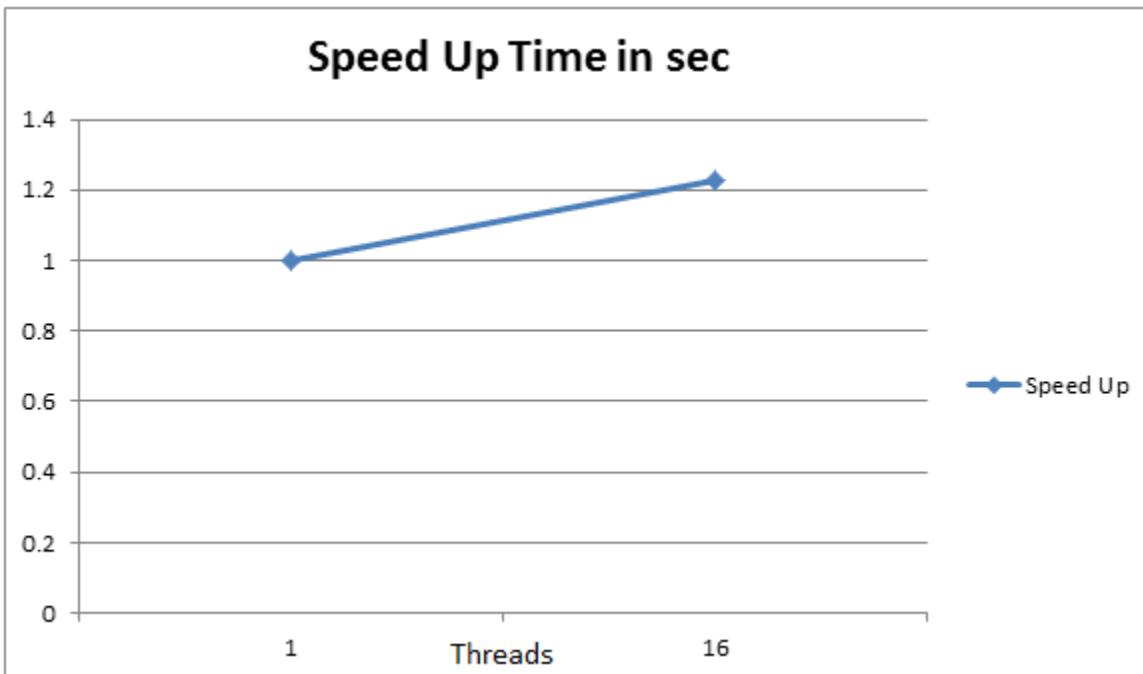
Number of nodes	Execution Time in sec	Throughput in Mb/sec
1 [10GB]	1921	5.2056
16 [100GB]	1562	64.0204



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

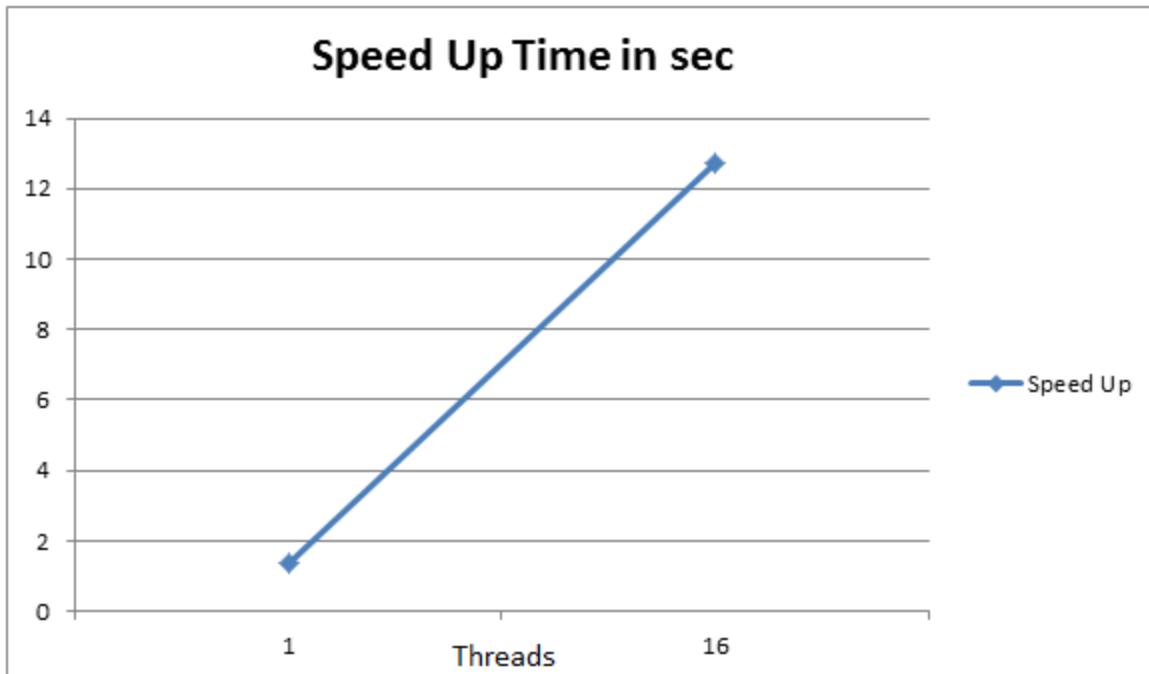
Base: 1 Node

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



Speed Up comparison for Shared memory Java 1 Thread Baseline:

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)

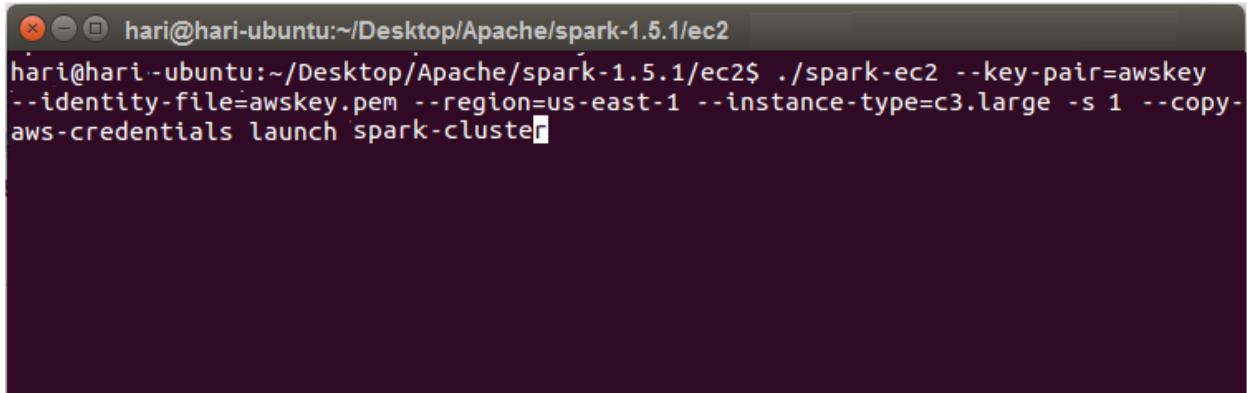
Do ssh-add mykey.pem. Then connect to the instance by using ssh -i @ubuntu:ipaddress Then do a scp to send mykeypair.pem and credentials to the instance location.

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-jdk
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:



```
hari@hari-ubuntu:~/Desktop/Apache/spark-1.5.1/ec2$ ./spark-ec2 --key-pair=awskey --identity-file=awskey.pem --region=us-east-1 --instance-type=c3.large -s 1 --copy-aws-credentials launch spark-cluster
```

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

```
Spark standalone cluster started at http://ec2-52-91-118-184.compute-1.amazonaws.com:  
8080  
Ganglia started at http://ec2-52-91-118-184.compute-1.amazonaws.com:5080/ganglia  
Done!  
hari@hari-ubuntu:~/Desktop/Apache/spark-1.5.1/ec2$
```

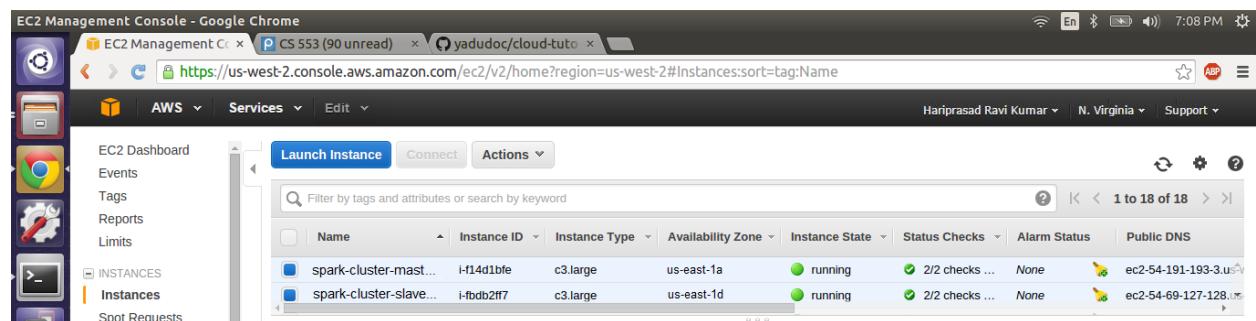
7. After the cluster has finished initializing, you can ssh into its root directory with:

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

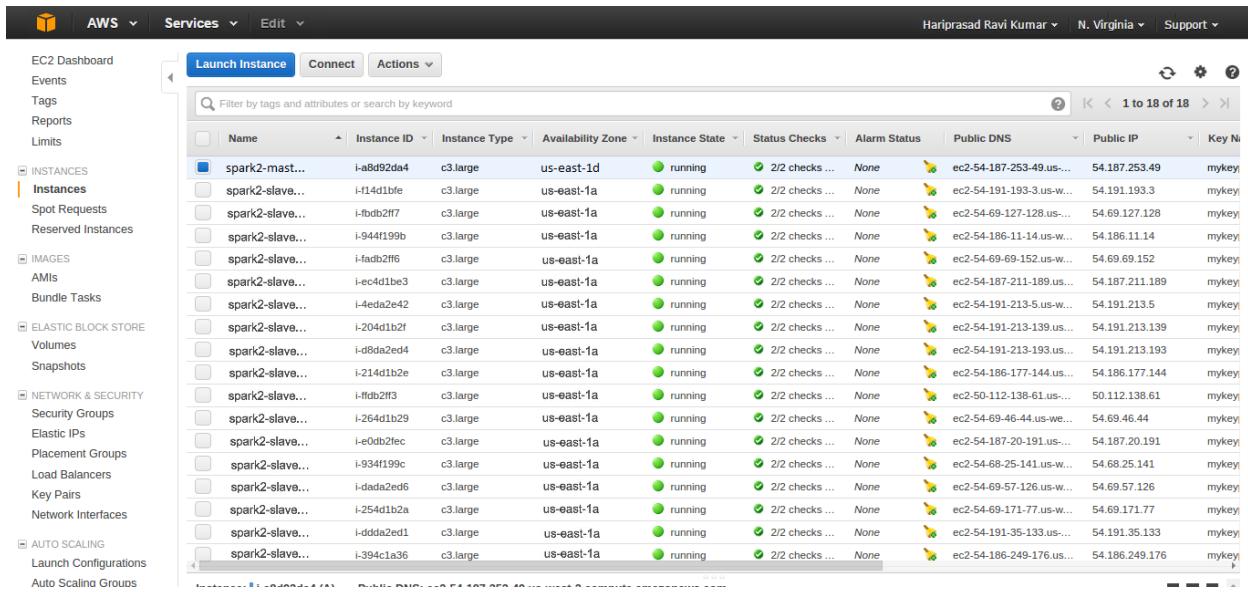
8. To end the EC2 Spark cluster, ssh out of the cluster and use:

```
./spark-ec2 destroy <cluster-name>
```

Screenshot of Head node and 1 Worker:



Head Node and 16 Workers:



The screenshot shows the AWS EC2 Instances page. The left sidebar navigation includes EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES (with Instances selected), IMAGES, AMIs, Bundle Tasks, ELASTIC BLOCK STORE, Volumes, Snapshots, NETWORK & SECURITY, Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces, and AUTO SCALING. The main content area displays a table of 17 EC2 instances. The columns are: Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS, Public IP, and Key Name. The table shows 1 instance named 'spark2-mast...' and 16 instances named 'spark2-slave...'. All instances are in the 'running' state, located in 'us-east-1a' or 'us-east-1d' availability zones, and have 2/2 status checks. The Public DNS column lists various domain names starting with 'ec2-54-' followed by a unique identifier. The Public IP column shows IP addresses like '54.187.253.49' and '54.191.193.3'. The Key Name column shows 'mykey' for all instances.

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name
	spark2-mast...	i-a8d92da4	c3.large	us-east-1d	running	2/2 checks ...	None	ec2-54-187-253-49.us...	54.187.253.49	mykey
	spark2-slave...	i-14d1bfe	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-191-193-3.us.w...	54.191.193.3	mykey
	spark2-slave...	i-fdb2f7f	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-69-127-128.us...	54.69.127.128	mykey
	spark2-slave...	i-944f199b	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-186-11-14.us.w...	54.186.11.14	mykey
	spark2-slave...	i-fadb2ff6	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-69-69-152.us.w...	54.69.69.152	mykey
	spark2-slave...	i-ec4d1be3	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-187-211-189.us...	54.187.211.189	mykey
	spark2-slave...	i-4e0a2e42	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-191-213-5.us.w...	54.191.213.5	mykey
	spark2-slave...	i-204d1b2f	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-191-213-139.us...	54.191.213.139	mykey
	spark2-slave...	i-d8da2ed4	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-191-213-193.us...	54.191.213.193	mykey
	spark2-slave...	i-214d1b2e	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-186-177-144.us...	54.186.177.144	mykey
	spark2-slave...	i-ftdb2ff3	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-50-112-138-61.us...	50.112.138.61	mykey
	spark2-slave...	i-264d1b29	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-69-46-44.us.w...	54.69.46.44	mykey
	spark2-slave...	i-e0db2fec	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-187-20-191.us...	54.187.20.191	mykey
	spark2-slave...	i-934f199c	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-68-25-141.us.w...	54.68.25.141	mykey
	spark2-slave...	i-dada2ed6	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-69-57-126.us.w...	54.69.57.126	mykey
	spark2-slave...	i-254d1b2b	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-69-171-77.us.w...	54.69.171.77	mykey
	spark2-slave...	i-ddda2ed1	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-191-35-133.us...	54.191.35.133	mykey
	spark2-slave...	i-394c1a36	c3.large	us-east-1a	running	2/2 checks ...	None	ec2-54-186-249-176.us...	54.186.249.176	mykey

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

```
ubuntu@ip-172-31-27-122:/usr/local/spark/bin
spark-submit: command not found
ubuntu@ip-172-31-27-122:/usr/local/spark/bin$ ./spark-submit --master local[16] /hom/usr/local/spark/pyTeraSort-master/pyTeraSort.py /hom/usr/local/spark/gensort-linux-1.5/64/gensortinput10GB.out.dir -inputPartition 64 -outputPartition 64
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
16/03/30 00:53:43 INFO SparkContext: Running Spark version 2.0.0-SNAPSHOT
16/03/30 00:53:43 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
16/03/30 00:53:43 WARN Utils: Your hostname, harl-550PSC-550P7C resolves to a loopback address: 127.0.1.1; using 172-31-27-122 instead (on interface wlan0)
16/03/30 00:53:43 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
16/03/30 00:53:43 INFO SecurityManager: Changing view acls to: ubuntu
16/03/30 00:53:43 INFO SecurityManager: Changing modify acls to: ubuntu
16/03/30 00:53:43 INFO SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(hari); users with modify permissions: Set(hari)
16/03/30 00:53:44 INFO Utils: Successfully started service 'sparkDriver' on port 39863.
16/03/30 00:53:44 INFO SparkEnv: Registering MapOutputTracker
16/03/30 00:53:44 INFO BlockManagerMaster: Created local directory at /tmp/blockmgr-b0d484e8-26cc-49be-948c-5c3bd893cf03
16/03/30 00:53:44 INFO MemoryStore: MemoryStore started with capacity 511.1 MB
16/03/30 00:53:44 INFO SparkEnv: Registering OutputCommitCoordinator
16/03/30 00:53:45 INFO Server: Jetty-8.y.z-SNAPSHOT
16/03/30 00:53:45 INFO AbstractConnector: Started SelectChannelConnector@0.0.0.0:4040
16/03/30 00:53:45 INFO Utils: Successfully started service 'SparkUI' on port 4040.
16/03/30 00:53:45 INFO SparkUI: Started SparkUI at http://172-31-27-122:4040
16/03/30 00:53:45 INFO Utils: Copying /hom/usr/local/spark/pyTeraSort-master/pyTeraSort.py to /tmp/spark-5f6a3760-56f8-4f29-81a7-f4fe8d21795d/userFiles-0653ae71-48b8-4938-bd27-6588fad73b0e/pyTeraSort.py
16/03/30 00:53:45 INFO SparkContext: Added file file:///hom/usr/local/spark/pyTeraSort-master/pyTeraSort.py at file:///hom/usr/local/spark/pyTeraSort-master/pyTeraSort.py with timestamp 1459317225587
16/03/30 00:53:45 INFO Executor: Starting executor ID driver on host localhost
16/03/30 00:53:45 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 39383.
16/03/30 00:53:45 INFO NettyBlockTransferService: Server created on 39383
16/03/30 00:53:45 INFO BlockManagerMasterEndpoint: Registering block manager localhost:39383 with 511.1 MB RAM, BlockManagerId(driver, localhost, 39383)
16/03/30 00:53:45 INFO BlockManagerMaster: Trying to register BlockManager
16/03/30 00:53:45 INFO BlockManagerMaster: Registered BlockManager
16/03/30 00:53:47 INFO MemoryStore: Block broadcast_0 stored as values in memory (estimated size 127.1 KB, free 127.1 KB)
16/03/30 00:53:47 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 14.3 KB, free 141.4 KB)
16/03/30 00:53:47 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on localhost:39383 (size: 14.3 KB, free: 511.1 MB)
16/03/30 00:53:47 INFO SparkContext: Created broadcast 0 from textFile at NativeMethodAccessorImpl.java:-2
16/03/30 00:53:47 INFO FileInputFormat: Total input paths to process : 1
16/03/30 00:53:47 INFO SparkContext: Starting job: sortByKey at /home/harl/Downloads/pyTeraSort-master/pyTeraSort.py:33
16/03/30 00:53:47 INFO DAGScheduler: Got job 0 (sortByKey at /home/harl/Downloads/pyTeraSort-master/pyTeraSort.py:33) with 64 output partitions
16/03/30 00:53:47 INFO DAGScheduler: Final stage: ResultStage 0 (sortByKey at /hom/usr/local/spark/pyTeraSort-master/pyTeraSort.py:33)
16/03/30 00:53:47 INFO DAGScheduler: Parents of final stage: List()
16/03/30 00:53:47 INFO DAGScheduler: Missing parents: List()
16/03/30 00:53:47 INFO DAGScheduler: Submitting ResultStage 0 (PythonRDD[2] at sortByKey at /hom/usr/local/spark/pyTeraSort-master/pyTeraSort.py:33), which has no missing parents
16/03/30 00:53:47 INFO MemoryStore: Block broadcast_1 stored as values in memory (estimated size 6.1 KB, free 147.6 KB)
16/03/30 00:53:47 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 3.8 KB, free 151.4 KB)
16/03/30 00:53:47 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on localhost:39383 (size: 3.8 KB, free: 511.1 MB)
16/03/30 00:53:47 INFO SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1012
```

```
File Edit View Search Terminal Help
16/03/30 01:55:07 INFO TaskSetManager: Finished task 61.0 in stage 3.0 (TID 253) in 1517 ms on 172-31-27-122(64)
16/03/30 01:55:07 INFO PythonRunner: Times: total = 1511, boot = 2179, init = 2190, finish = 1500
16/03/30 01:55:07 INFO OutputCommitCoordinator: Saved output of task attempt_201603300054_0003_m_000063_255 to file:/usr/local/spark/bin/out_dir/output/_temporary/0/task_201603300054_0003_m_000063
16/03/30 01:55:07 INFO SparkHadoopMapRedUtil: attempt_201603300054_0003_m_000063_255: Committed
16/03/30 01:55:07 INFO Executor: Finished task 63.0 in stage 3.0 (TID 255) - 2790 bytes result sent to driver
16/03/30 01:55:07 INFO TaskSetManager: Finished task 63.0 in stage 3.0 (TID 255) in 1539 ms on 172-31-27-122(64)
16/03/30 01:55:07 INFO DAGScheduler: Removed TaskSet 3.0, whose tasks have all completed from pool
16/03/30 01:55:07 INFO DAGScheduler: ResultStage 3 (SaveAsTextFile at NativeMethodAccessorImpl.java:-2) finished in 19431 s
16/03/30 01:55:07 INFO DAGScheduler: Job 2 (finished) saveAsTextFile at NativeMethodAccessorImpl.java:-2, took 3558.7325 s
16/03/30 01:55:07 INFO SparkContext: Invoking stop() from shutdown hook
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/metrics/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/stage/kill,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/api/null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/static,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/executors/threadDump/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/executors/threadDump,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/executors/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/environment/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/environment,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/storage/rdd/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/storage/rdd,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/storage/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/json}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/pool/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/pool,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/stage/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/stage,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServletContextHandler{/stages/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServiceContextHandler{/jobs/job/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServiceContextHandler{/jobs/job,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServiceContextHandler{/jobs/json,null}
16/03/30 01:55:07 INFO ContextHandler: stopped o.e.j.s.ServiceContextHandler{/jobs,null}
16/03/30 01:55:07 INFO SparkUI: Stopped Spark web UI at http://172-31-27-122:4040
16/03/30 01:55:07 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/03/30 01:55:07 INFO MemoryStore: MemoryStore cleared
16/03/30 01:55:07 INFO BlockManagerMaster: BlockManager stopped
16/03/30 01:55:07 INFO BlockManagerMaster: BlockManager stopped
16/03/30 01:55:07 INFO OutputCommitCoordinator: OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/03/30 01:55:07 INFO SparkContext: Successfully stopped SparkContext
16/03/30 01:55:07 INFO ShutdownHookManager: Shutdown hook called
16/03/30 01:55:07 INFO ShutdownHookManager: Deleting directory /tmp/spark-5f6a3760-56f8-4f29-81a7-f4fe8d21795d/repl-8080f0cb-bb70-4b96-a304-5df1f4c7bc8
16/03/30 01:55:07 INFO ShutdownHookManager: Deleting directory /tmp/spark-5f6a3760-56f8-4f29-81a7-f4fe8d21795d/pyspark-3df3688-05be-401e-a638-bc001f1e413f
ubuntu@ip-172-31-27-122:/usr/local/spark/bin$
```

100 GB DATASET ON c3.large instance:

Nodes: 16

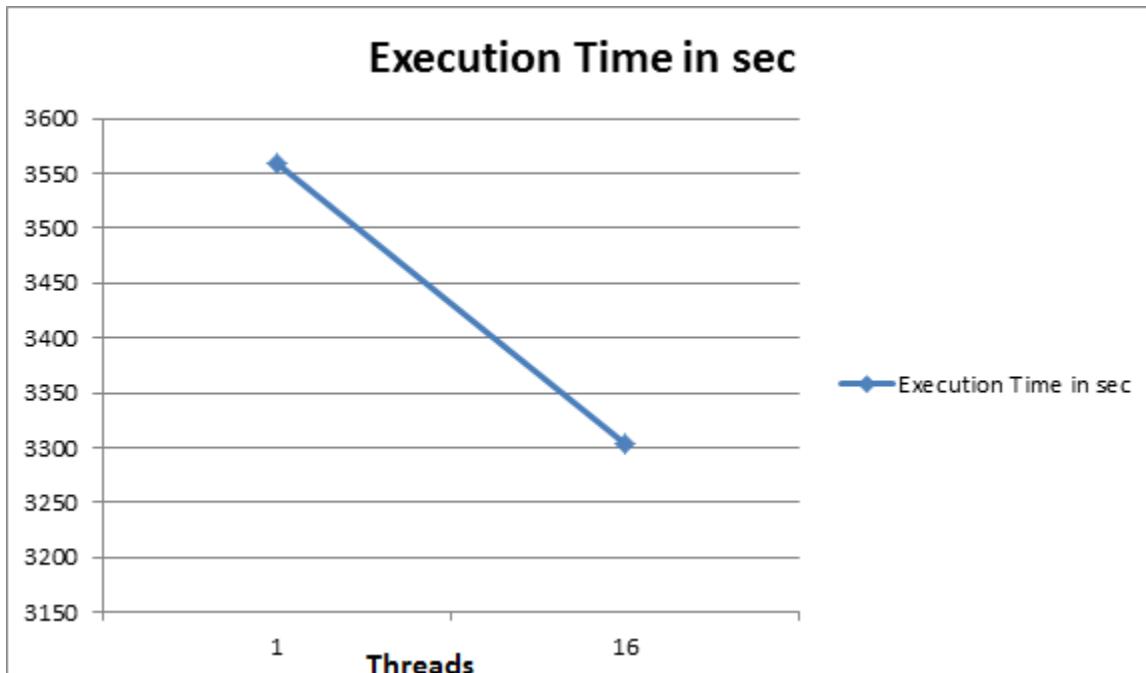
Execution time: 3303.6 seconds

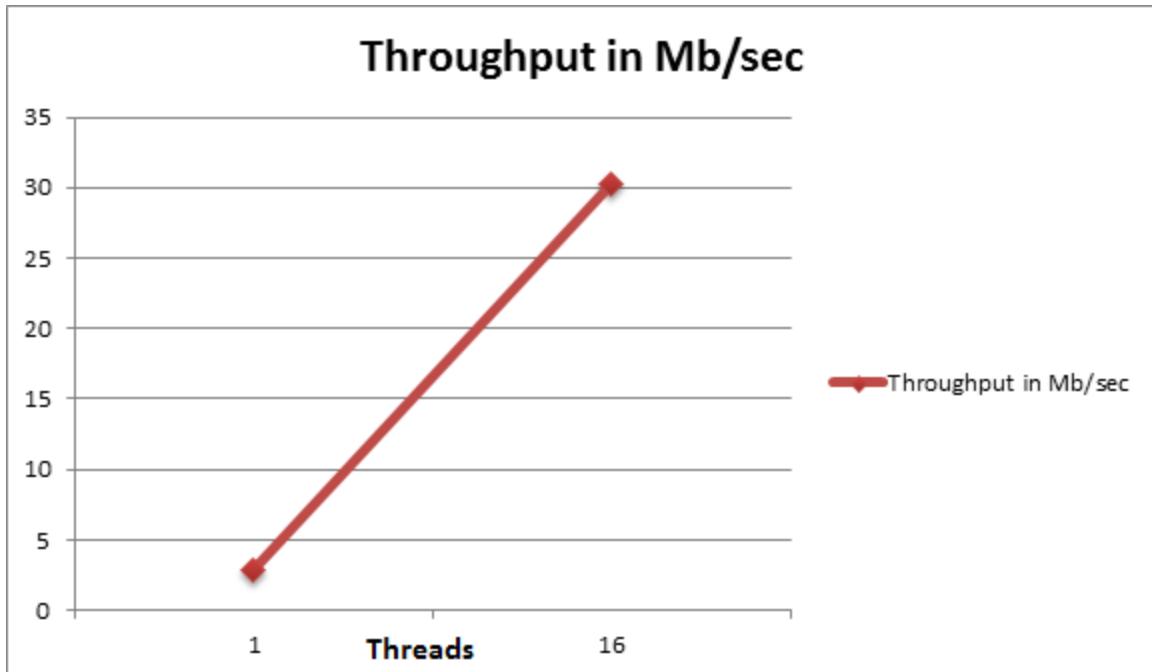
```
ubuntu@ip-172-31-27-122: /usr/local/spark/bin
ubuntu@ip-172-31-27-122:/usr/local/spark/bin$ ./spark-submit --master local[16] /hom/usr/local/spark/pyTeraSort-master/pyTeraSort.py /hom/usr/local/spark/gensort-linux-1
5/64/gensortinput100GB.out_dir -inputPartition 64 -outputPartition 64
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
16/03/30 16:53:41 INFO SparkContext: Running Spark version 2.0.0-SNAPSHOT
16/03/30 16:53:43 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
16/03/30 16:53:43 WARN Utils: Your hostname harl-550PSC-550P7C resolves to a loopback address: 127.0.1.1; using 172-31-27-122 instead (on interface wlan0)
16/03/30 16:53:43 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
16/03/30 16:53:43 INFO SecurityManager: Changing view acls to: ubuntu
16/03/30 16:53:43 INFO SecurityManager: Changing modify acls to: ubuntu
```

```
ubuntu@ip-172-31-27-122: /usr/local/spark/bin
at org.apache.hadoop.mapreduce.task.reduce.LocalFetcher.copyMapOutput(LocalFetcher.java:134)
at org.apache.hadoop.mapreduce.task.reduce.LocalFetcher.doCopy(LocalFetcher.java:102)
16/03/30 17:48:38 INFO reduce.OnDiskKapOutput: Read 139586518 bytes from map-output for attempt_local665246902_0001_m_000000_0
16/03/30 17:48:39 INFO mapreduce.Job: Job job_local665246902_0001 failed with state FAILED due to: NA
16/03/30 17:48:39 INFO mapreduce.Job: Counters: 35
File System Counters
    FILE: Number of bytes read=6554027540
    FILE: Number of bytes written=12544852488
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=5848096500
    HDFS: Number of bytes written=0
    HDFS: Number of read operations=277
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=18
Map-Reduce Framework
    Map input records=1000000000000
    Map output records=1000000000000
    Map output bytes=1020000000000000
    Map output materialized bytes=10400000486832
    Input split bytes=1024
    Combine input records=0
    Combine output records=0
    Reduce input groups=0
    Reduce shuffle bytes=5583457600
    Reduce input records=0
    Reduce output records=0
    Spilled Records=19395241
    Shuffled Maps =4
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ns)=346
    Total committed heap usage (bytes)=451621683248
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=1000000000000
File Output Format Counters
    Bytes Written=0
16/03/30 17:48:39 INFO terasort: done
ubuntu@ip-172-31-27-122:/usr/local/spark/bin$
```

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





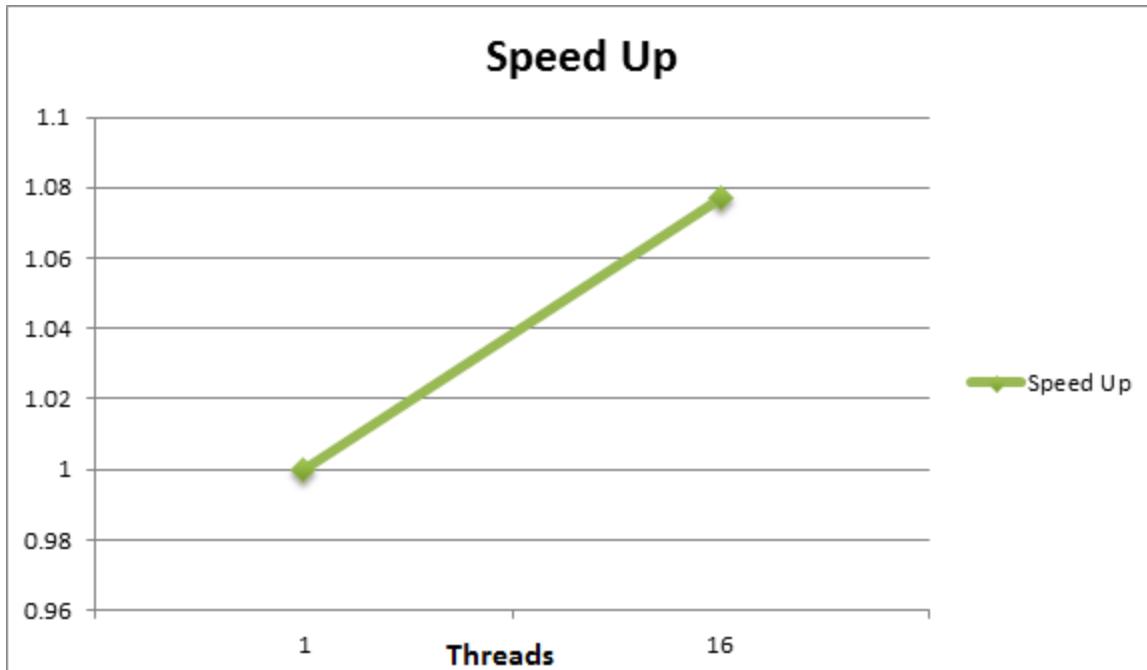
Number of nodes and the speed up time for 10GB and 100GB data:

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:

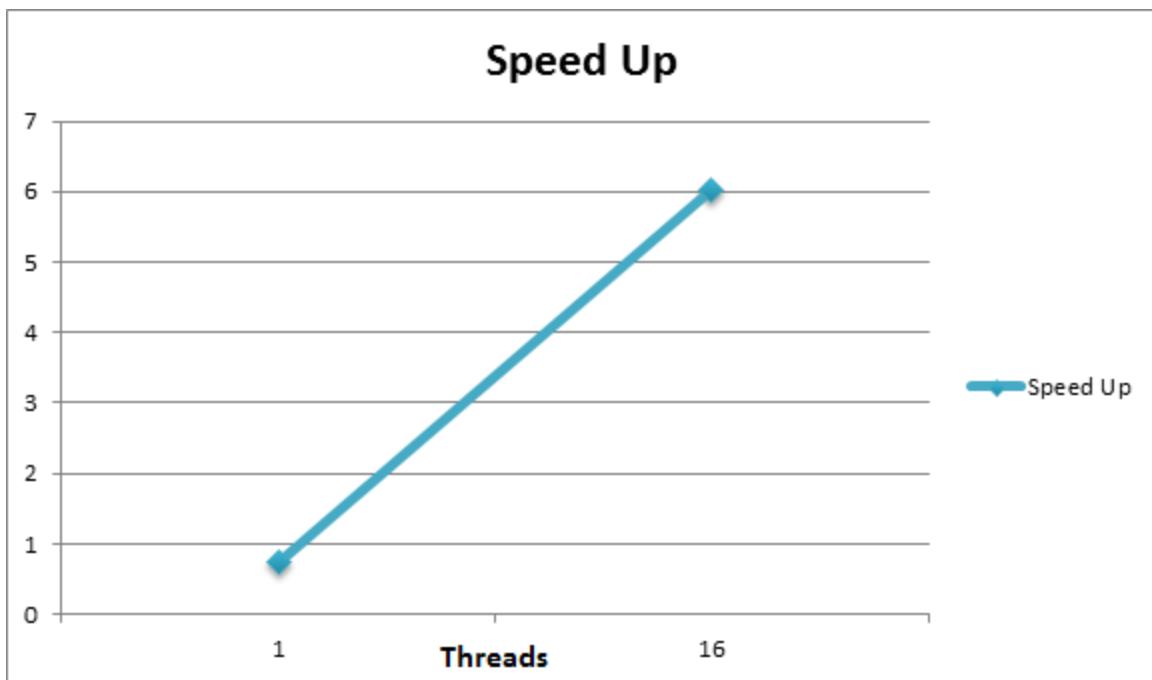
Using the information from the table above a graph is plotted between number of nodes and speed up time. The baseline of this graph is value obtained at node 1.



Best Performance is obtained at 16 nodes.

Speed Up comparison for Shared memory Java 1 Thread Baseline:

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



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.