CSS553 Programming Assignment 2

By

George Mathew

(A20352131)

The programming assignment involved sorting of data using the following methodologies

1)Simple Merge Sort

2)Shared Memory Sort

3)Hadoop Pseudo Mode Sorting

4)Hadoop 16 node cluster Sorting

5)Spark Pseudo Mode Sorting

6)Spark 16 node cluster Sorting

The assignment had 8 parts as explained below

1)Virtual Cluster One Node:

The following kinds of virtual instances were used.

1)c3.large

2)d2.xlarge

Ubuntu instances were used. SSH, All TCP, All UDP, All ICMP security groups were added to the instances

All instances were updated and java was installed. Raid was used to combine the available hard disk spaces.

Java Version :1.7.0_95

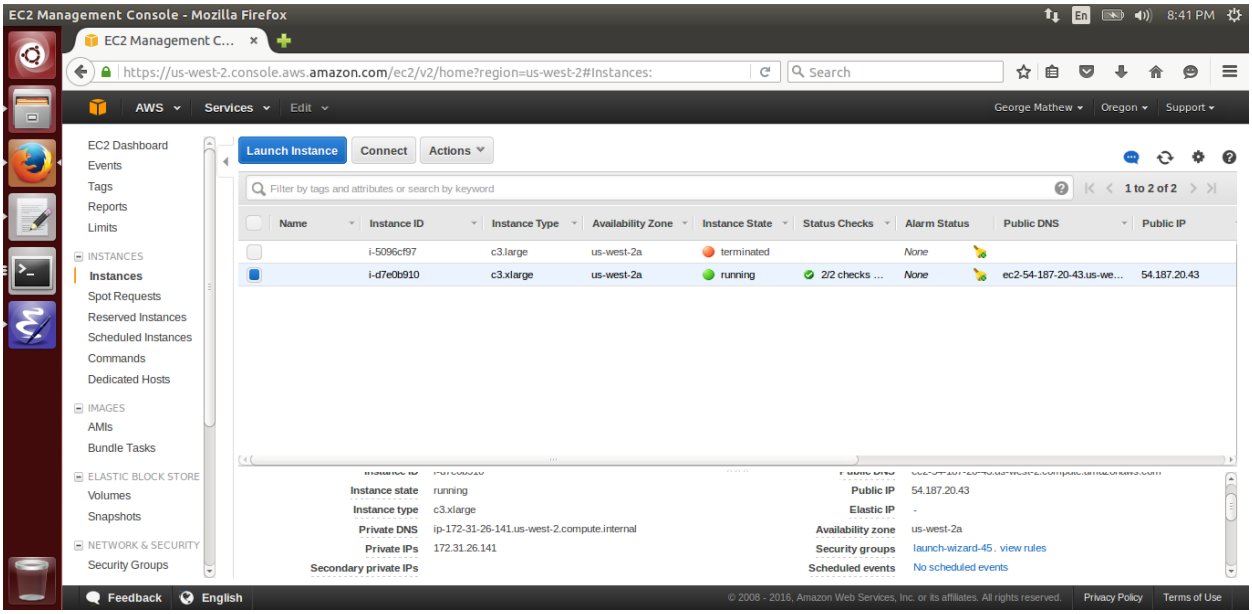
Spot instances were used. c3.large was bid at $0.09 and d2.xlarge was bid at $0.5

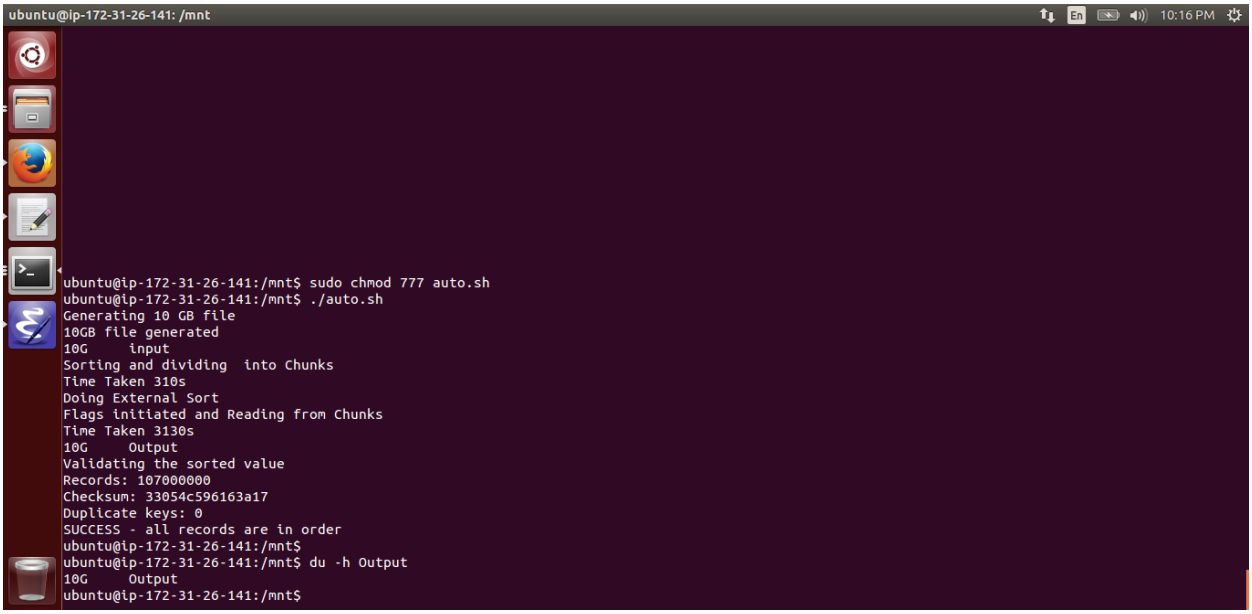
The initial Instance was made using d2.xlarge and an AMI image of this was made for future use

2) Shared Memory Sort

The problem with shared memory is that the data to be sorted is bigger than the available memory. So external sort algorithm was used to do this

10 gb of data was sorted using 3.5 gb of RAM





The sorting was done in 57.3 minutes

Valsort was done and the output was verified

3)Virtual Cluster

A Virtual Cluster of 17 nodes was setup using 1 d2.xlarge and 16 c3.large instances using the images made from step 1.

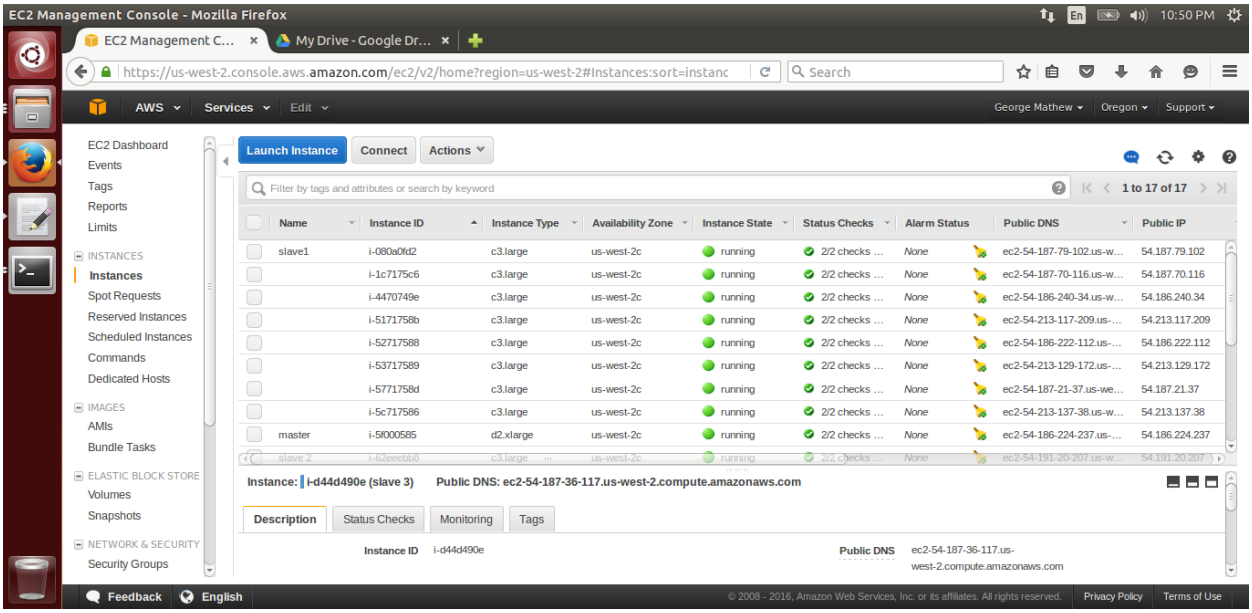
An EBS volumes were also added to the c3.large systems. RAID0 was used to configure the disks of the instances

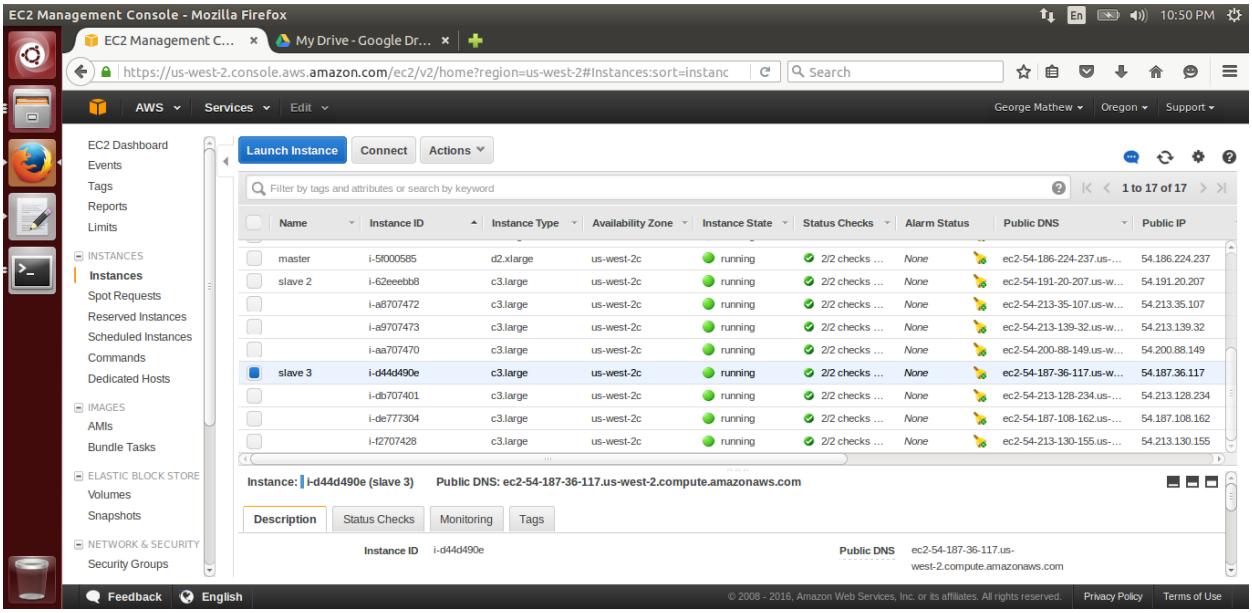
Password less SSH was enabled and the keys of all the instances were added to the master so that all the instances can be accessed from the Master

A user ‘me’ was made in all the instances and was granted super user privileges

Pssh was used to push commands from the master to all the nodes

Scp was used to push data from the master to all the 16 nodes





4)Hadoop

Hadoop was installed and used in pseudo mode and 16 cluster

Hadoop 2.7.2 was downloaded and moved to my mount point /mnt

The bashrc file for the user 'me' was edited and the following lines were added

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_HOME=/mnt/hadoop
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

The following xml files inside Hadoop/etc/Hadoop was configured for using Hadoop

Core-site.xml

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/mnt/raid/tmp/</value>
</property>
</configuration>
```

A tmp file was made in the location /mnt/raid/tmp so that that the Hadoop can use this location for its temporary files. This was pointed to the /mnt so that EBS volumes and s3 volumes can be mounted here if required

Hdfs-site.xml

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/mnt/hadoop/hadoop_data/hdfs/namenode</value>
</property>
```

```
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/mnt/hadoop/hadoop_data/hdfs/datanode</value>
</property>
</configuration>
```

Replication was turned off so that data won't be replicated in other nodes and space can be conserved
The locations for the data node and namenode was pointed to the /mnt so that additional devices can be mounted if necessary

Mapred-site.xml

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.jobtracker.address</name>
<value>hdfs://localhost:9001</value>
</property>
<property>
<name>mapred.tasktracker.map.tasks</name>
<value>2</value>
</property>
<property>
<name>mapred.tasktracker.reduce.tasks</name>
<value>2</value>
</property>
</configuration>
```

The reducer and map tasks were set as two as there were only 2 virtual cores on the c3.large systems which were the slaves

Yarn-site.xml

```
<configuration>
<property>
<name>yarn.resourcemanager.hostname></name>
<value>localhost</value>
</property>
<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.scheduler.address</name>
```

```
<value>localhost:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>localhost:8032</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>localhost:8088</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>localhost:8031</value>
</property>
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>localhost:8033</value>
</property>
</configuration>
```

All the processes of yarns are pointed to different ports

Slaves

Localhost is added to the slave file

In the pseudo mode the master and the slave both are in one system and all the processes happen in the same node

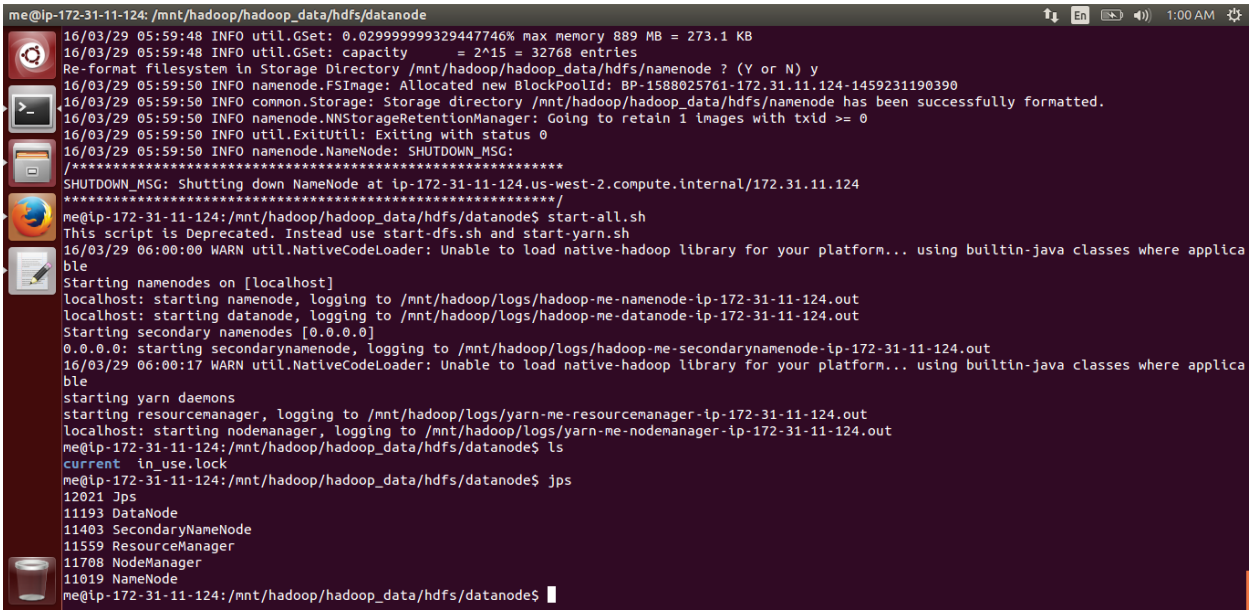
Then namenode is formatted using the command

Hdfs namenode -format

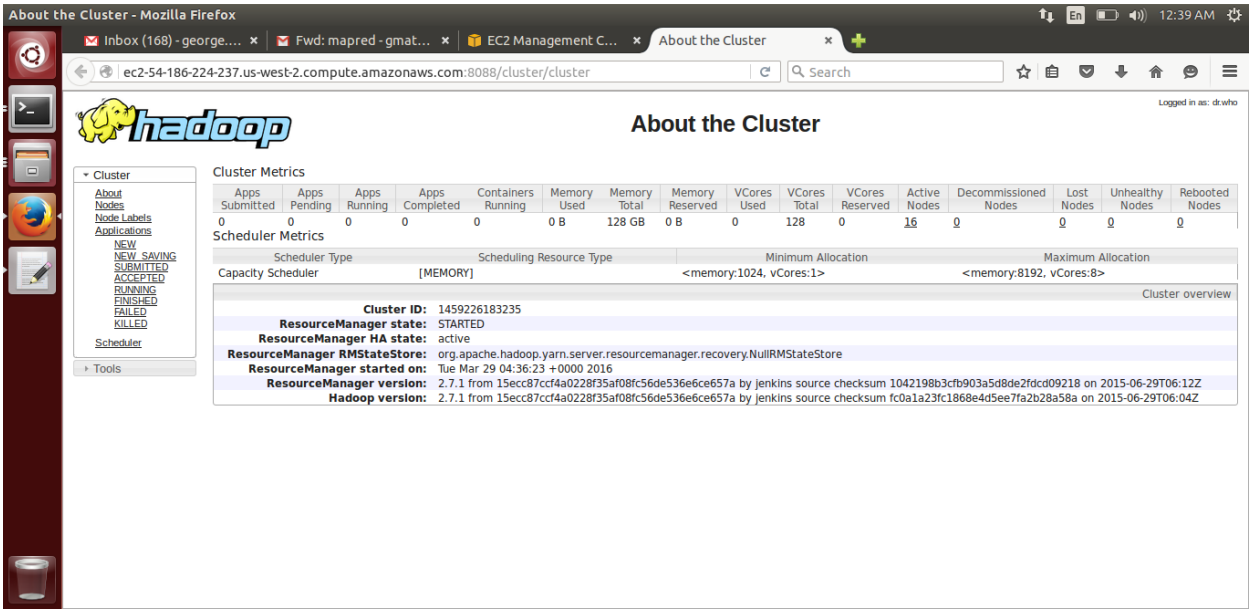
Then dfs is started using the command start-dfs.sh

And then the yarn is started using the command start-yarn.sh

Then using the command jps all the running Hadoop processes can be displayed



The single node was setup on the system with ip 172.31.11.124



On this system the Hadoop program was written srt.java

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class srt {

  public static class Mppr
       extends Mapper<Object, Text, Text, NullWritable>{

    NullWritable n=NullWritable.get();
    Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {

        word.set(value.toString()+" ");
        context.write(word, n);
      }
    }


  public static class Rdcr
       extends Reducer<Text,NullWritable,Text,NullWritable> {


        NullWritable s=NullWritable.get();
public void reduce(Text key, Iterable<NullWritable> values,
                      Context context
                      ) throws IOException, InterruptedException {

      for (NullWritable val : values) {
        s= val;
      }

      context.write(key,s);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(srt.class);
    job.setMapperClass(Mppr.class);
    job.setCombinerClass(Rdcr.class);
    job.setReducerClass(Rdcr.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(NullWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

This was compiled using the command

```
bin/hadoop com.sun.tools.javac.Main srt.java
```

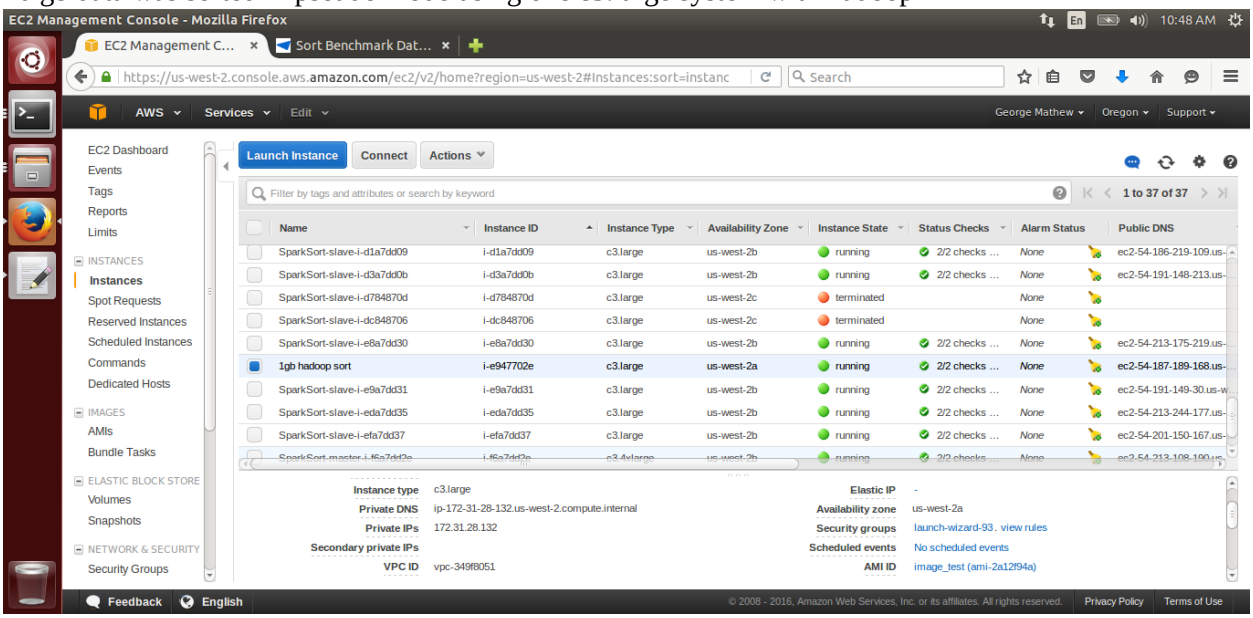
This was made into a jar using the command

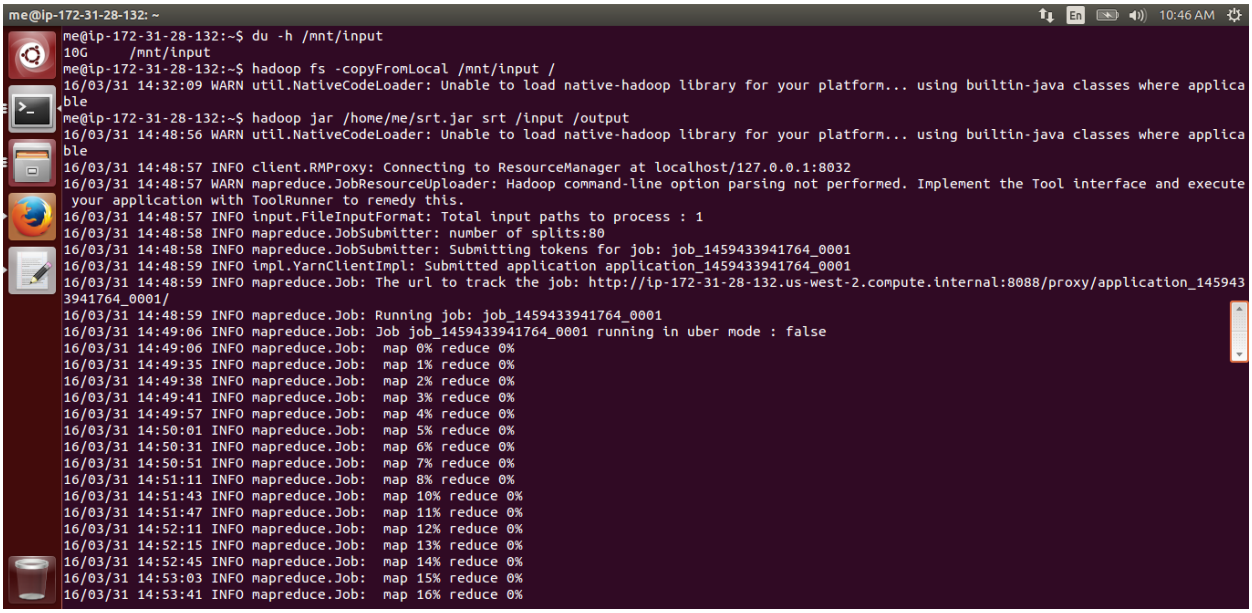
```
jar cf srt.jar srt*.class
```

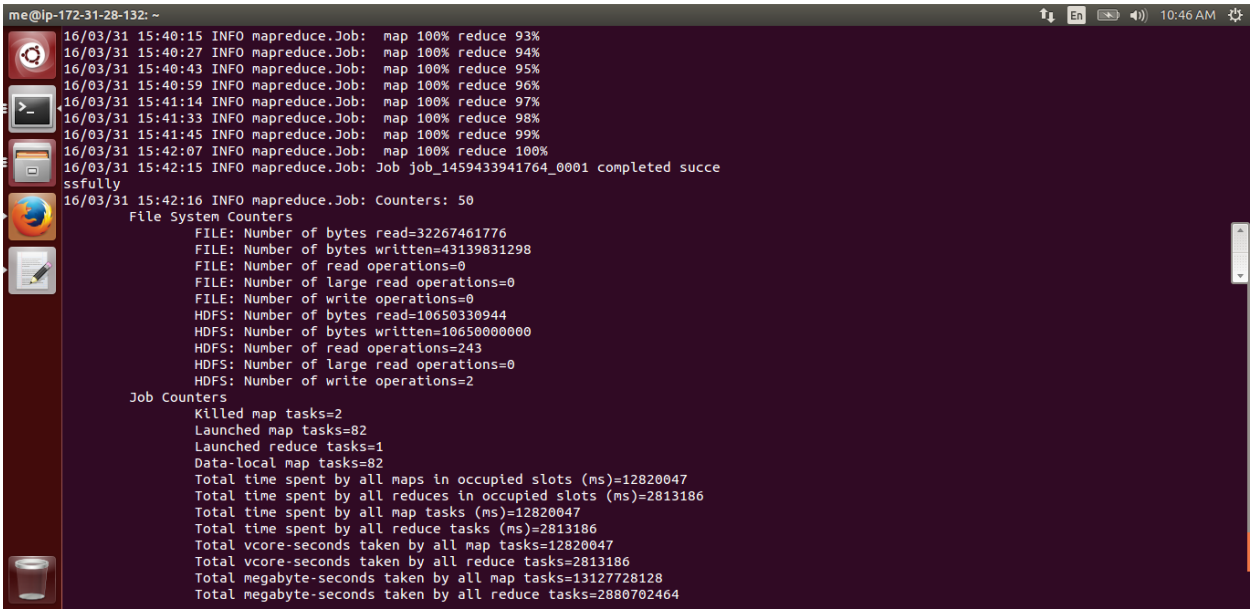
5)Hadoop Sort

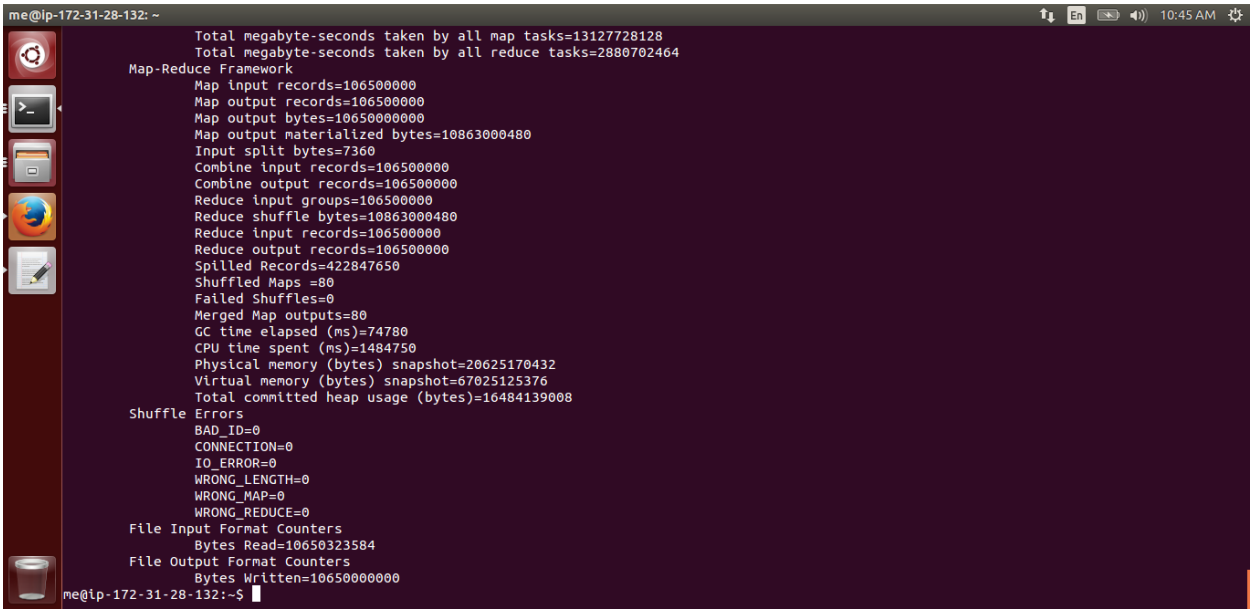
Pseudo mode

10 gb data was sorted in pseudo mode using one c3.large system with hadoop

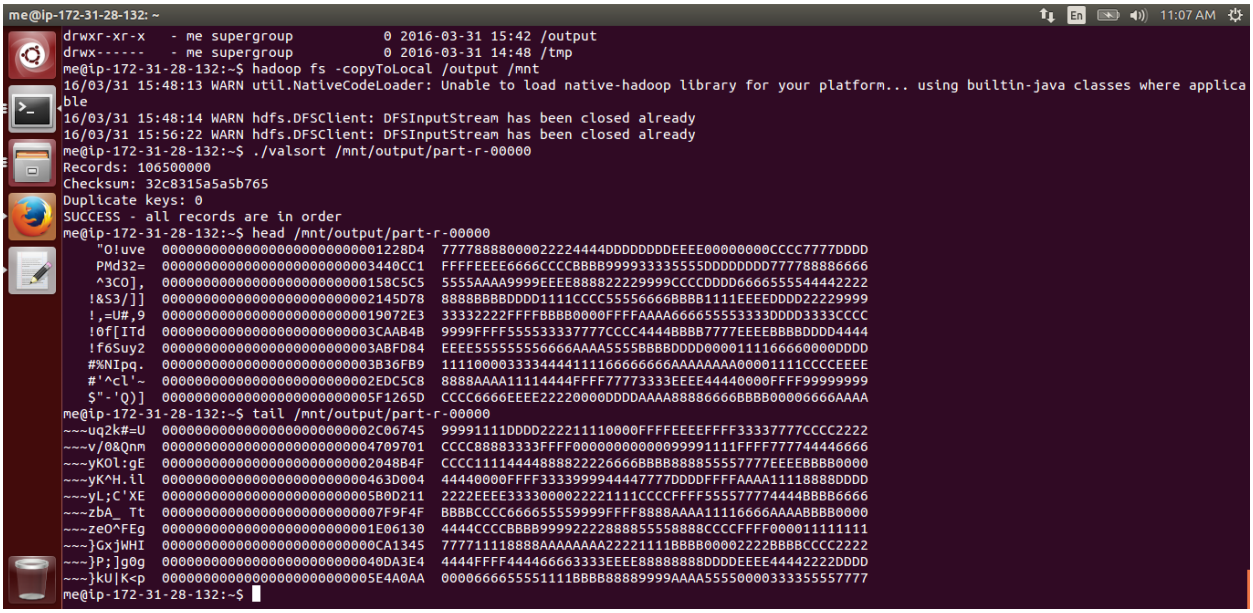








1 GB was sorted in 24.7 minutes



The output was verified using valsort

Hadoop Cluster

Then Hadoop was setup on a 16 node cluster .

The slaves file in master was updated with ips of all the 16 nodes

```
172.31.11.124
172.31.4.234
172.31.13.181
172.31.12.191
172.31.4.190
172.31.14.98
172.31.9.196
172.31.14.174
172.31.0.15
172.31.11.64
172.31.3.97
172.31.6.168
172.31.15.117
172.31.14.57
172.31.7.94
172.31.11.46
```

The core-site of the master was updated with the ip of the master.

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://172.31.14.182:9000</value>
</property>
<property>
```

```
<name>hadoop.tmp.dir</name>
<value>/mnt/raid/tmp/</value>
</property>
</configuration>
```

Hdfs site didn't need any change

Mapred site was updated with the ip of the master

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.jobtracker.address</name>
<value>hdfs://172.31.14.182:9001</value>
</property>
<property>
<name>mapred.tasktracker.map.tasks</name>
<value>2</value>
</property>
<property>
<name>mapred.tasktracker.reduce.tasks</name>
<value>2</value>
</property>
</configuration>
```

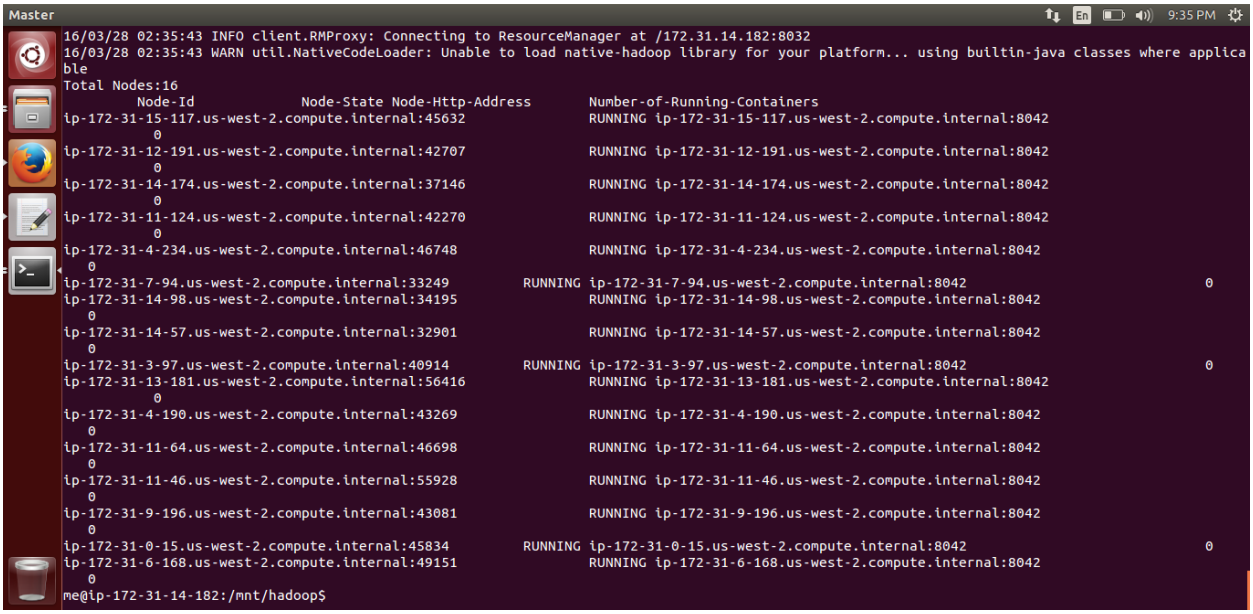
The yarn site was updated with the ip of the master

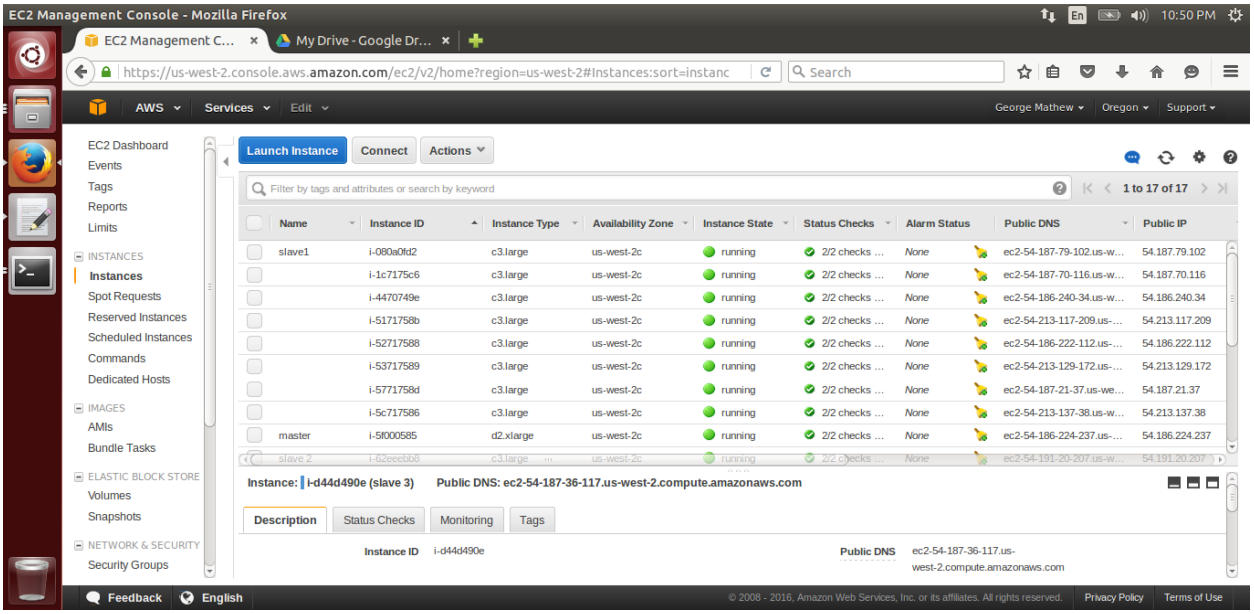
```
<configuration>
<property>
<name>yarn.resourcemanager.hostname></name>
<value>172.31.14.182</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>172.31.14.182:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>172.31.14.182:8032</value>
</property>
```

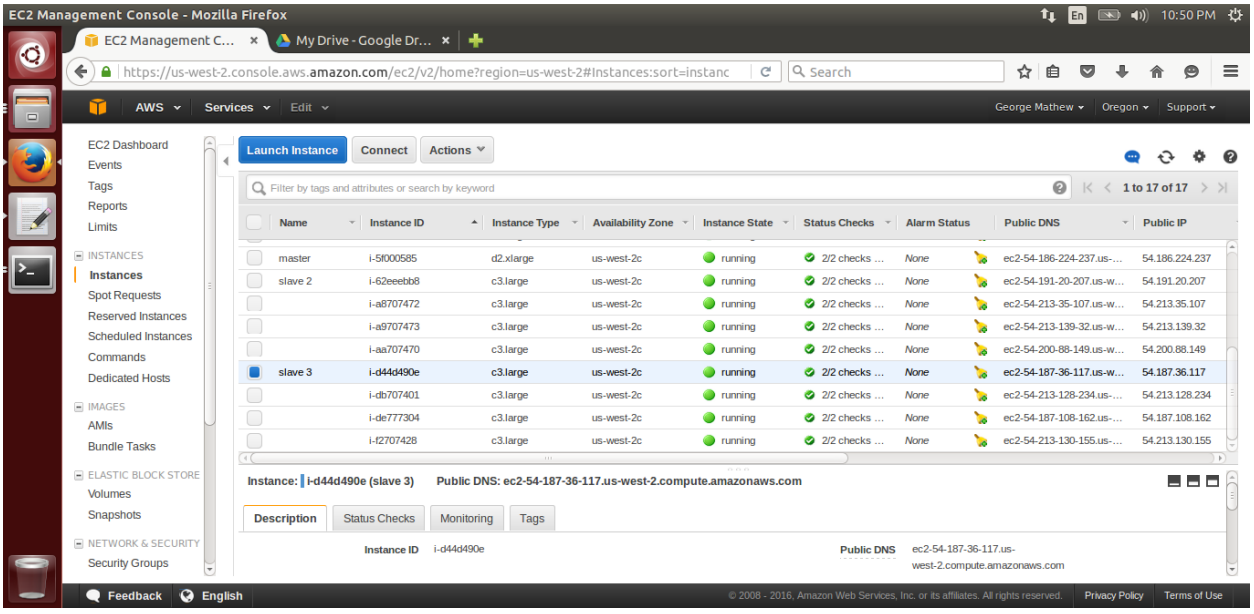
```
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>172.31.14.182:8088</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>172.31.14.182:8031</value>
</property>
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>172.31.14.182:8033</value>
</property>
</configuration>
```

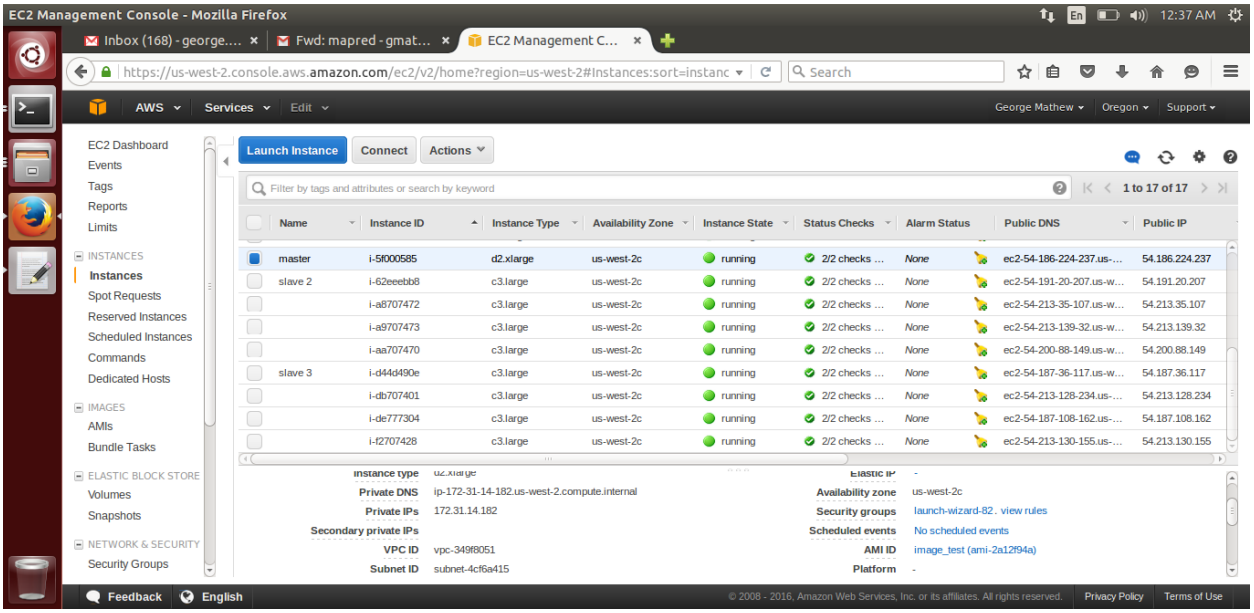
The slaves are specified in the slaves folder and the master node starts the data nodes on these slaves

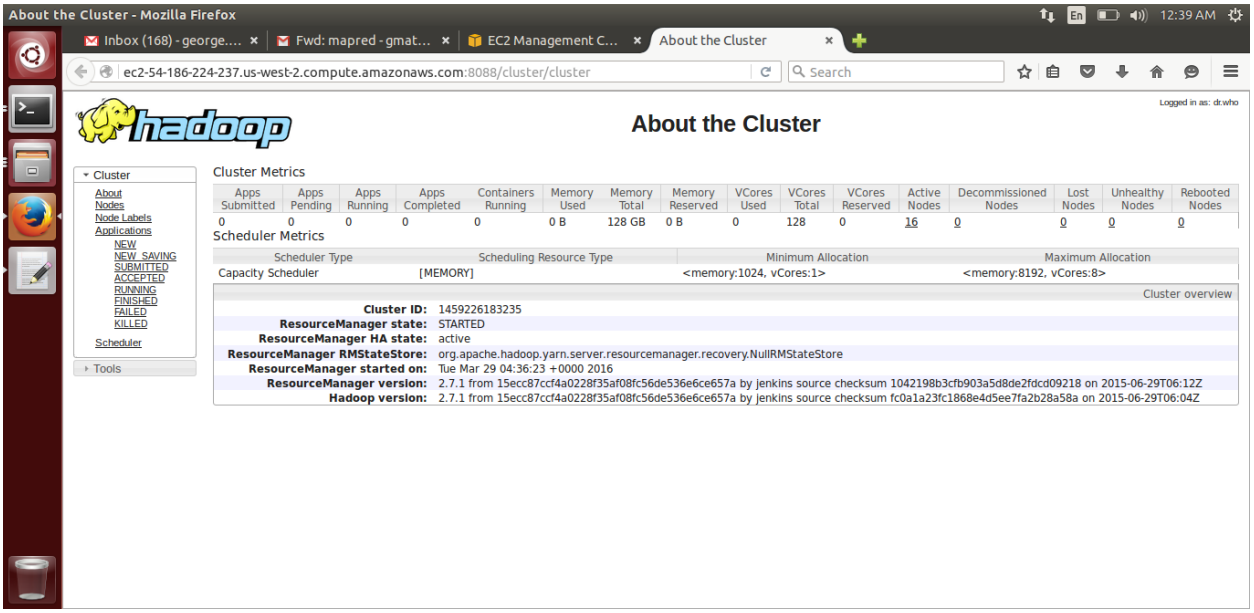
This configured Hadoop is pushed from the master to the slaves .As these files have the ip of the master the node manager,resource manager,namenode for the slaves are working on the master node




EC2 Management Console - Mozilla Firefox
10:50 PM
EC2 Management C...
My Drive - Google Dr...
https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=instanc
Search
AWS
Services
Edit
George Mathew
Oregon
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
Scheduled Instances
Commands
Dedicated Hosts
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 17 of 17
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
Public IP
slave1 i-080a0fd2 c3.large us-west-2c running 2/2 checks ... None ec2-54-187-79-102.us-w... 54.187.79.102
i-1c7175c6 c3.large us-west-2c running 2/2 checks ... None ec2-54-187-70-116.us-w... 54.187.70.116
i-4470749e c3.large us-west-2c running 2/2 checks ... None ec2-54-186-240-34.us-w... 54.186.240.34
i-5171758b c3.large us-west-2c running 2/2 checks ... None ec2-54-213-117-209.us-... 54.213.117.209
i-52717588 c3.large us-west-2c running 2/2 checks ... None ec2-54-186-222-112.us-... 54.186.222.112
i-53717589 c3.large us-west-2c running 2/2 checks ... None ec2-54-213-129-172.us-... 54.213.129.172
i-5771758d c3.large us-west-2c running 2/2 checks ... None ec2-54-187-21-37.us-we... 54.187.21.37
i-5c717586 c3.large us-west-2c running 2/2 checks ... None ec2-54-213-137-38.us-w... 54.213.137.38
master i-5f000585 d2.xlarge us-west-2c running 2/2 checks ... None ec2-54-186-224-237.us-... 54.186.224.237
slave 2 i-62eeebb8 c3.large us-west-2c running 2/2 checks ... None ec2-54-191-20-207.us-w... 54.191.20.207
Instance: i-d44d490e (slave 3) Public DNS: ec2-54-187-36-117.us-west-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID i-d44d490e
Public DNS ec2-54-187-36-117.us-west-2.compute.amazonaws.com
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


EC2 Management Console - Mozilla Firefox
10:50 PM
EC2 Management C...
My Drive - Google Dr...
https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=instanc
Search
AWS
Services
Edit
George Mathew
Oregon
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
Scheduled Instances
Commands
Dedicated Hosts
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 17 of 17
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS
Public IP
master i-5f000585 d2.xlarge us-west-2c running 2/2 checks ... None ec2-54-186-224-237.us-... 54.186.224.237
slave 2 i-62eeebb8 c3.large us-west-2c running 2/2 checks ... None ec2-54-191-20-207.us-w... 54.191.20.207
i-a8707472 c3.large us-west-2c running 2/2 checks ... None ec2-54-213-35-107.us-w... 54.213.35.107
i-a9707473 c3.large us-west-2c running 2/2 checks ... None ec2-54-213-139-32.us-w... 54.213.139.32
i-aa707470 c3.large us-west-2c running 2/2 checks ... None ec2-54-200-88-149.us-w... 54.200.88.149
slave 3 i-d44d490e c3.large us-west-2c running 2/2 checks ... None ec2-54-187-36-117.us-w... 54.187.36.117
i-db707401 c3.large us-west-2c running 2/2 checks ... None ec2-54-213-128-234.us-... 54.213.128.234
i-de777304 c3.large us-west-2c running 2/2 checks ... None ec2-54-187-108-162.us-... 54.187.108.162
i-f2707428 c3.large us-west-2c running 2/2 checks ... None ec2-54-213-130-155.us-... 54.213.130.155
Instance: i-d44d490e (slave 3) Public DNS: ec2-54-187-36-117.us-west-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID i-d44d490e
Public DNS ec2-54-187-36-117.us-west-2.compute.amazonaws.com
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use





What is a master Node and Slave Node?

The resource manager and the Node manager runs on the manager and the keep tracks of which data goes to which node. The nodes that the master use to process the data are called the slave nodes. The master uses the resources of the slave nodes to do the processing

Why do we need to set unique available ports to those configuration files on a shared environment? What errors or side-effects will show if we use same port number for each user?

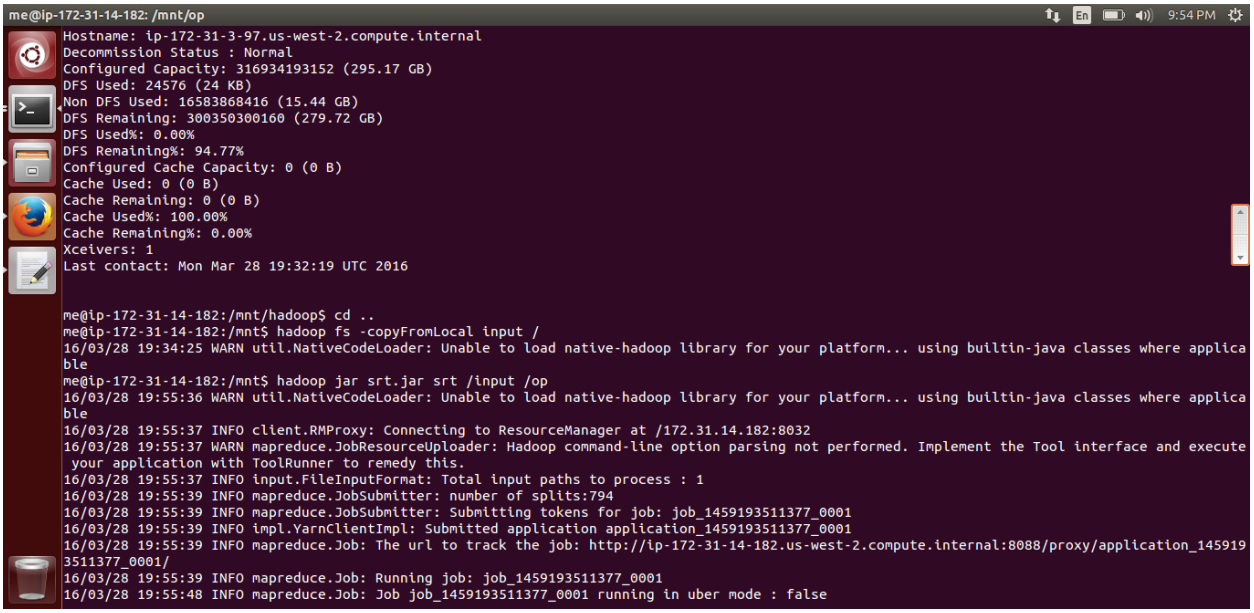
Only one process can run on a single port. If we give the same port number to two different processes, then there will be conflict and the process will fail

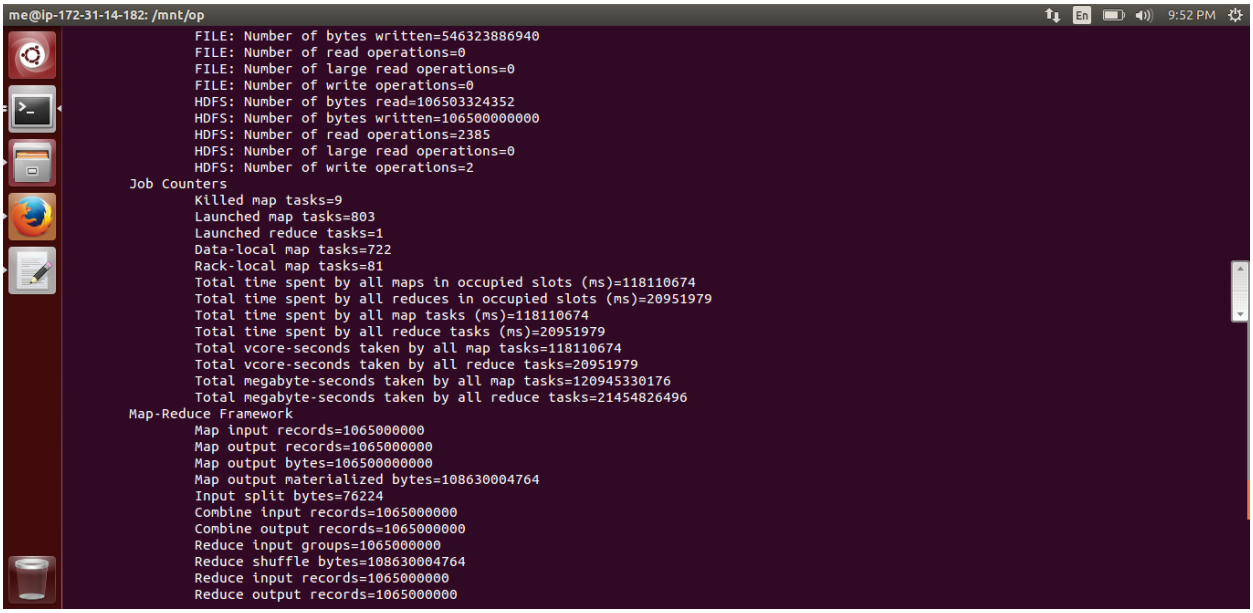
How can we change the number of mappers and reducers from the configuration file?

The number of mappers and reducers can be changed in the map-reduce .xml config file. As mentioned above my cluster was running with two mappers and two reducers as there are two virtual cores in c3.large

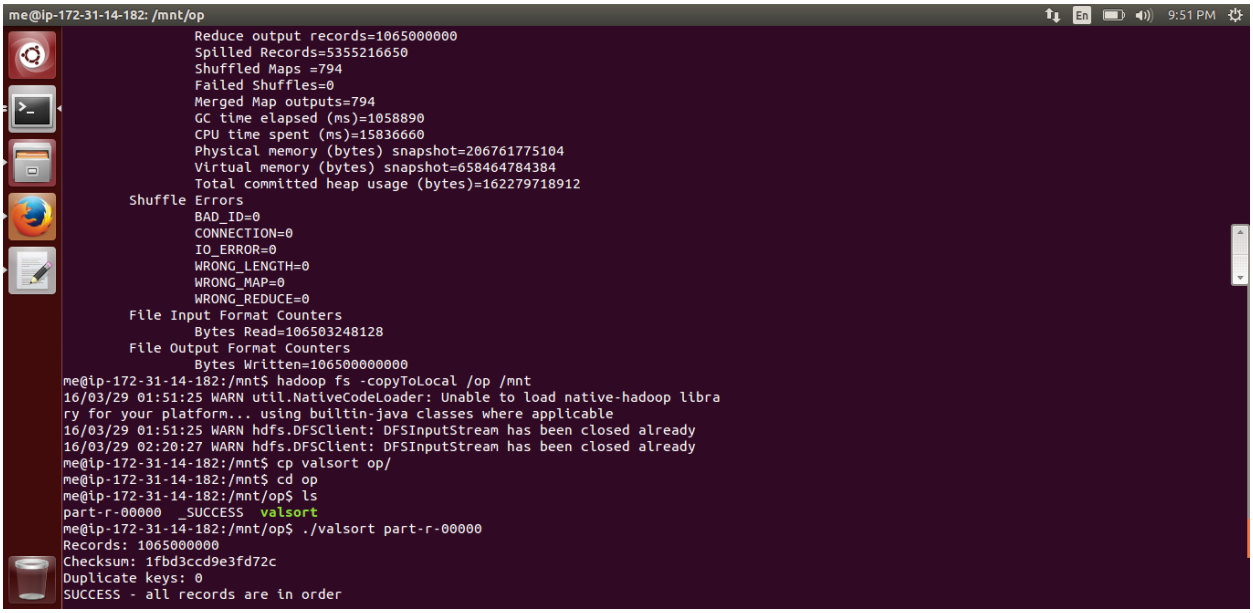
```
<property>
<name>mapred.tasktracker.map.tasks</name>
<value>2</value>
</property>
<property>
<name>mapred.tasktracker.reduce.tasks</name>
<value>2</value>
</property
```

The Hadoop program was run to sort 100gb data





This was completed in 4.39 hours



Valsort was done and the output was verified

6) Spark

Spark was installed on 17 systems.

1 c3.4large system was used as a master and all other systems were c3.large

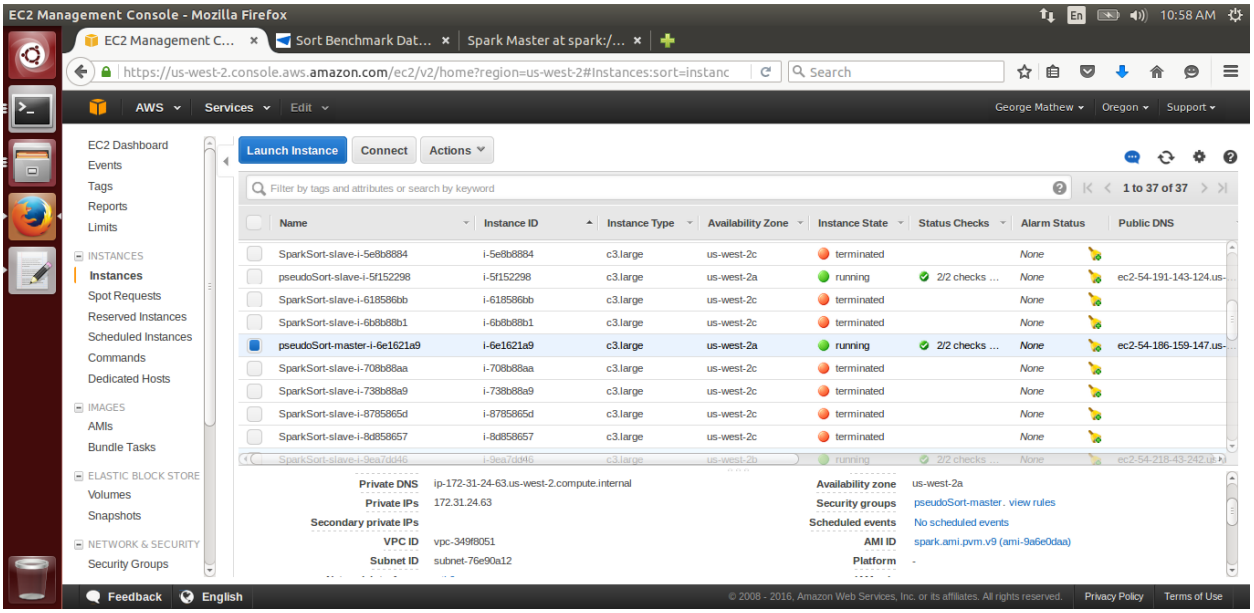
The program to sort the entries was written using python

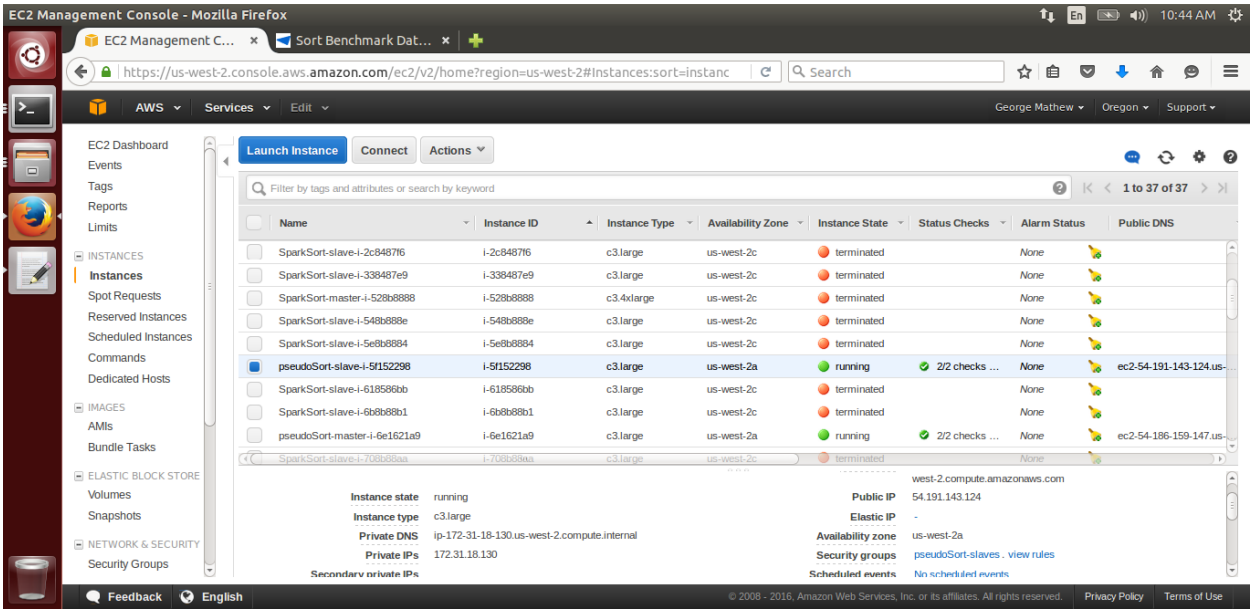
This program was used to sort 10gb in pseudo mode and 100gb in 16 clusters

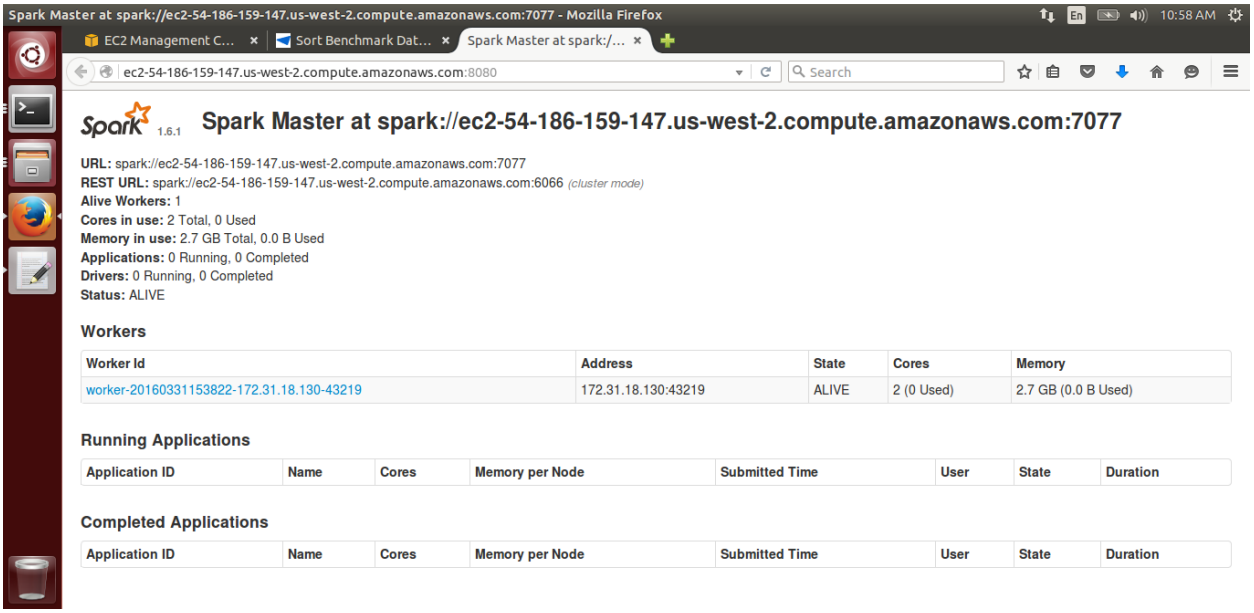
7)Spark Sort

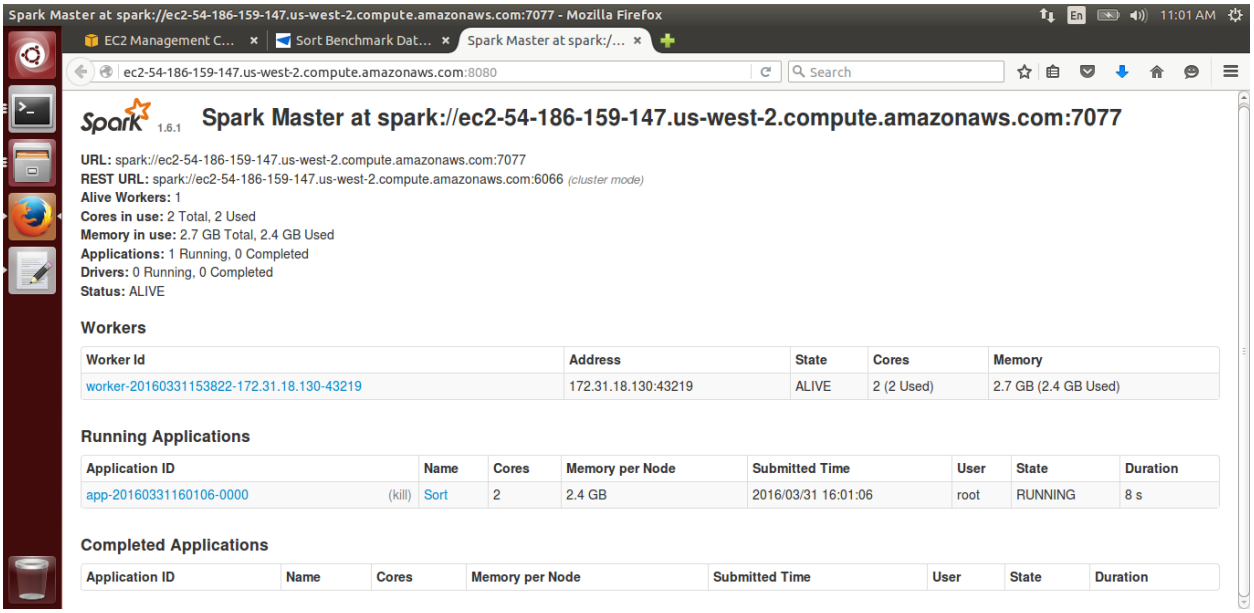
Pseudo Mode

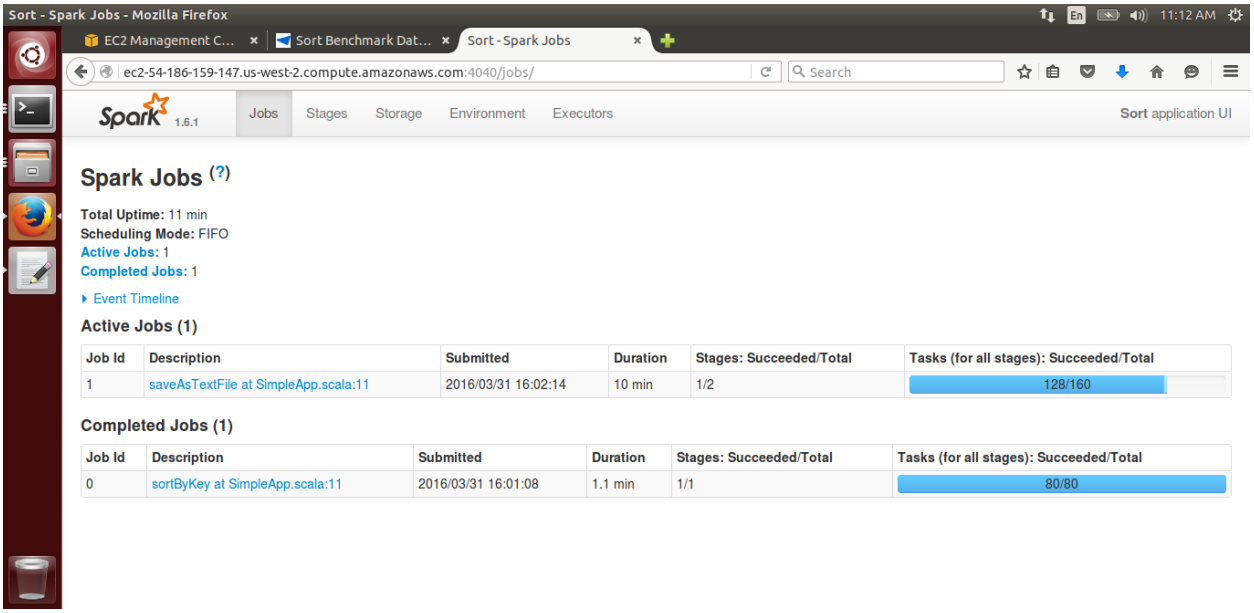
10 gb of data was sorted using the program on a single node spark system

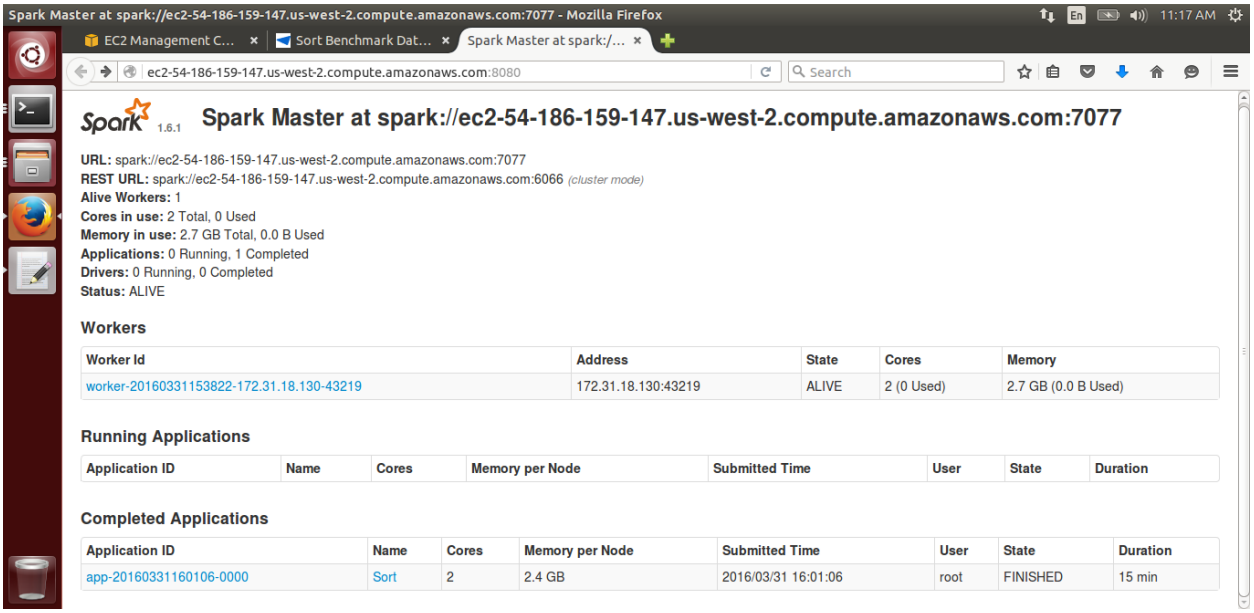




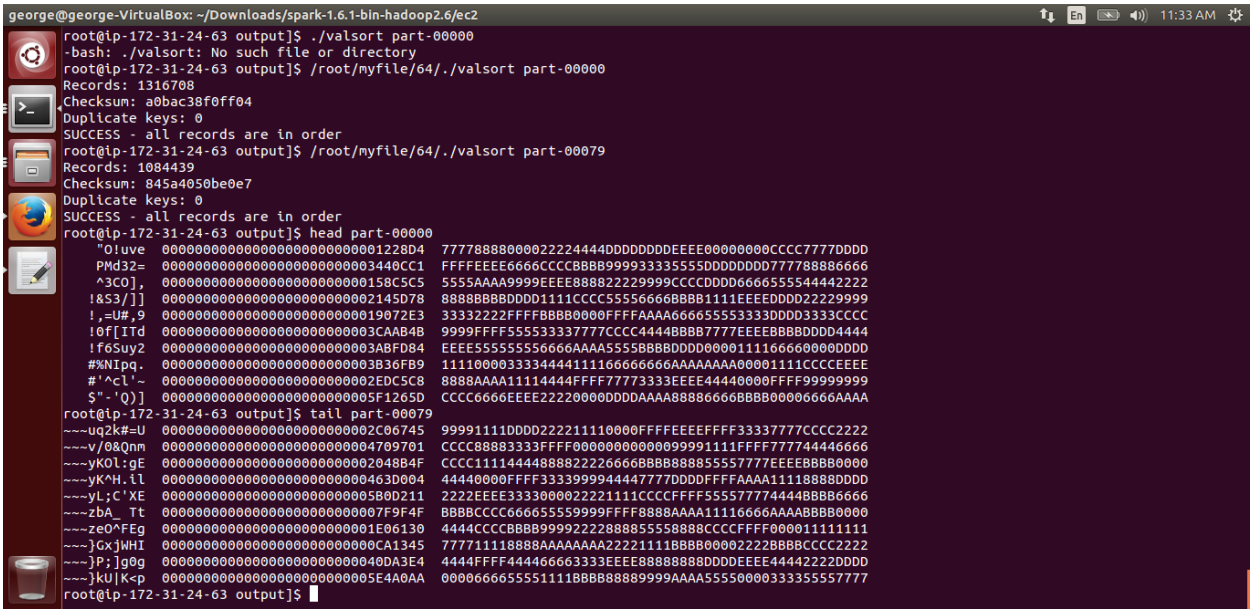

Spark Master at spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:7077 - Mozilla Firefox
10:58 AM
EC2 Management C...
Sort Benchmark Dat...
Spark Master at spark:/...
ec2-54-186-159-147.us-west-2.compute.amazonaws.com:8080
Search
Spark 1.6.1
Spark Master at spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:7077
URL: spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:7077
REST URL: spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:6066 (cluster mode)
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 2.7 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE
Workers
Worker Id
Address
State
Cores
Memory
worker-20160331153822-172.31.18.130-43219
172.31.18.130:43219
ALIVE
2 (0 Used)
2.7 GB (0.0 B Used)
Running Applications
Application ID
Name
Cores
Memory per Node
Submitted Time
User
State
Duration
Completed Applications
Application ID
Name
Cores
Memory per Node
Submitted Time
User
State
Duration


Spark Master at spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:7077 - Mozilla Firefox
11:01 AM
EC2 Management C...
Sort Benchmark Dat...
Spark Master at spark:/...
ec2-54-186-159-147.us-west-2.compute.amazonaws.com:8080
Search
Spark 1.6.1
Spark Master at spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:7077
URL: spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:7077
REST URL: spark://ec2-54-186-159-147.us-west-2.compute.amazonaws.com:6066 (cluster mode)
Alive Workers: 1
Cores in use: 2 Total, 2 Used
Memory in use: 2.7 GB Total, 2.4 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE
Workers
Worker Id
Address
State
Cores
Memory
worker-20160331153822-172.31.18.130-43219
172.31.18.130:43219
ALIVE
2 (2 Used)
2.7 GB (2.4 GB Used)
Running Applications
Application ID
Name
Cores
Memory per Node
Submitted Time
User
State
Duration
app-20160331160106-0000
(kill)
Sort
2
2.4 GB
2016/03/31 16:01:06
root
RUNNING
8 s
Completed Applications
Application ID
Name
Cores
Memory per Node
Submitted Time
User
State
Duration





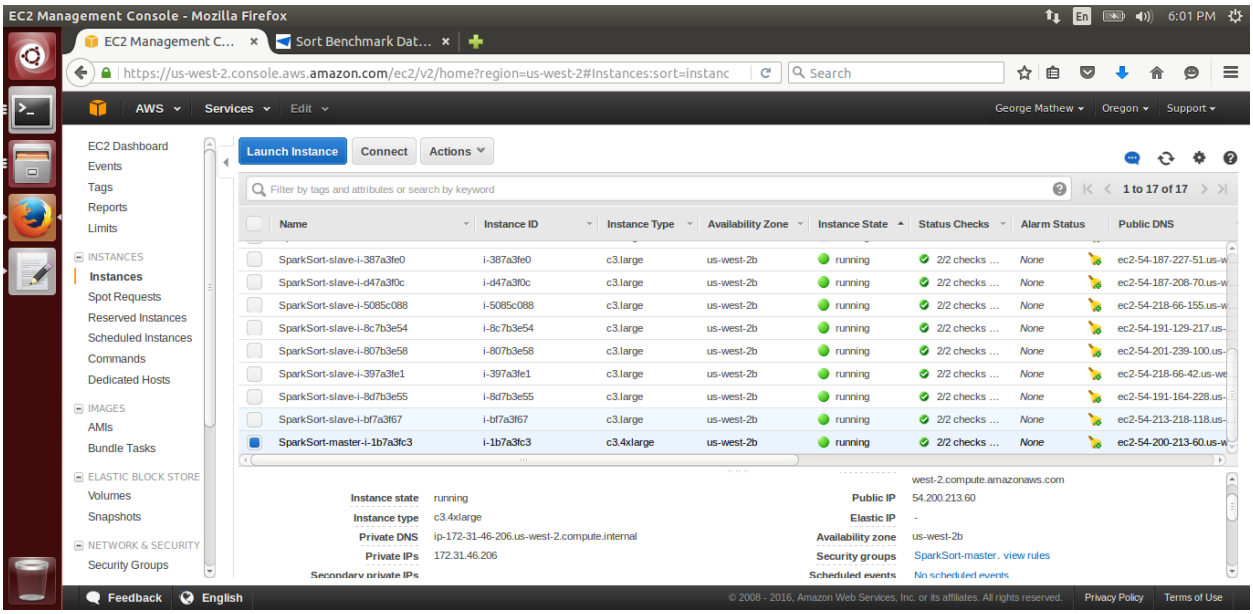
The sorting was completed in 15 mins

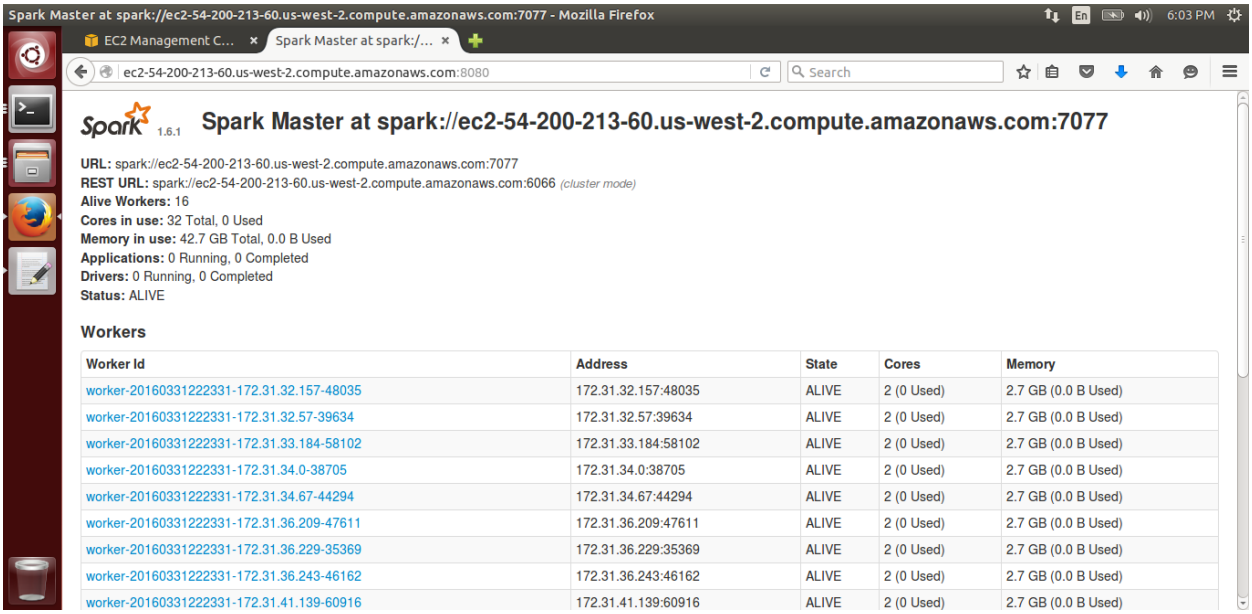


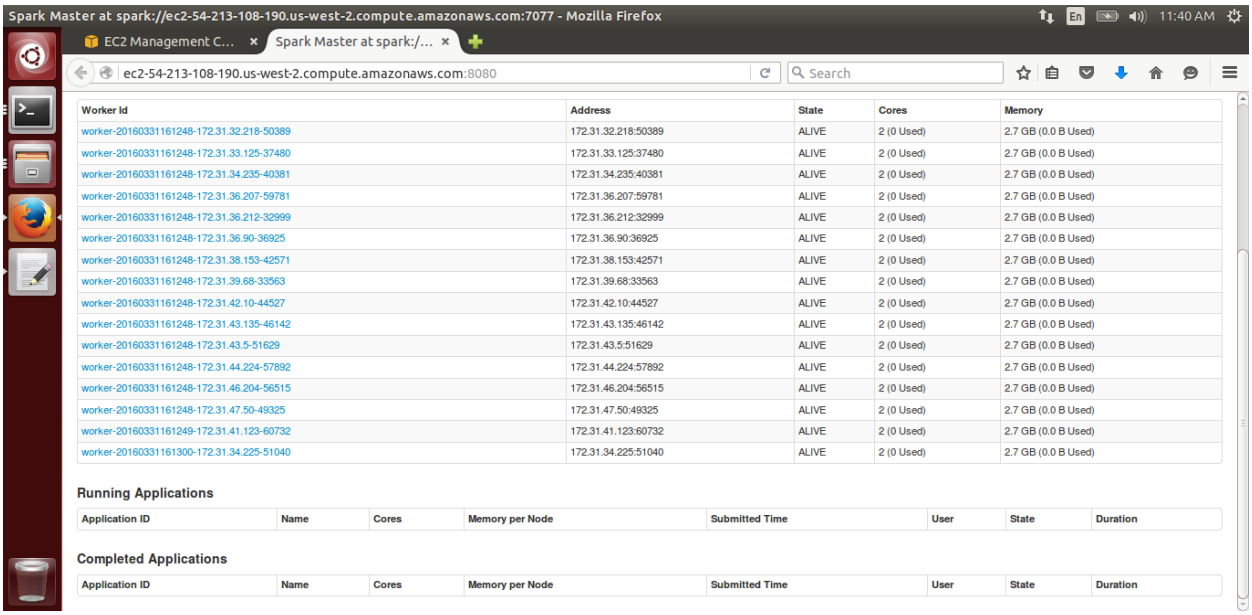
The output was verified using valsort

Cluster Sort

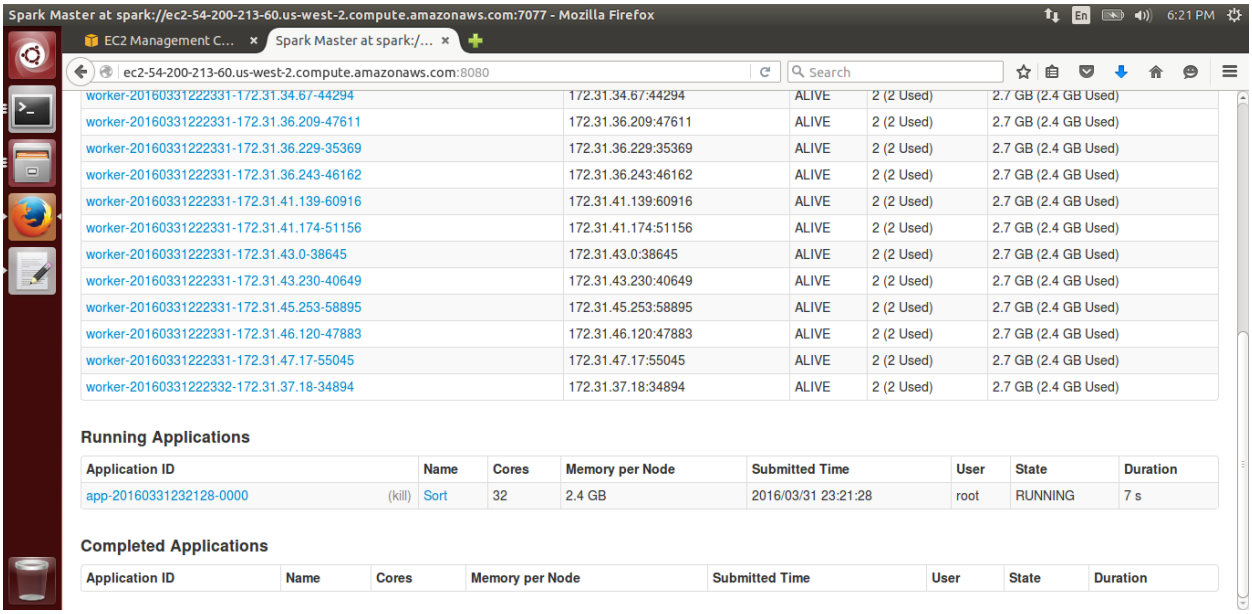
The cluster was setup with 17 instances 1 c4.large as master and 16 c3.large instances as slaves were initiated

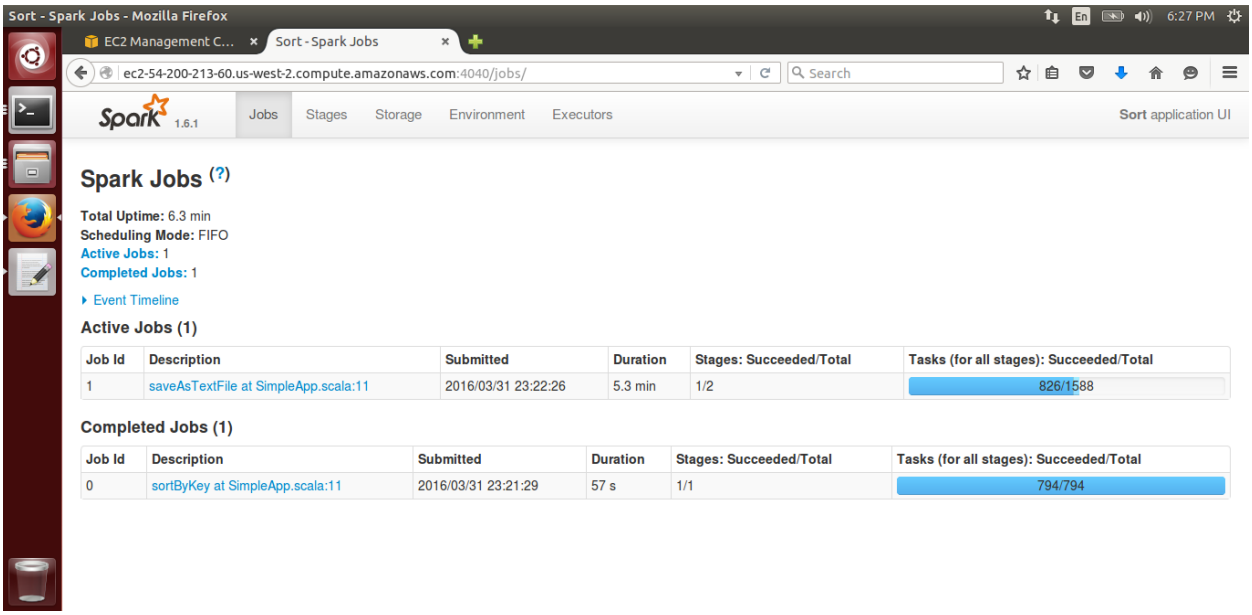


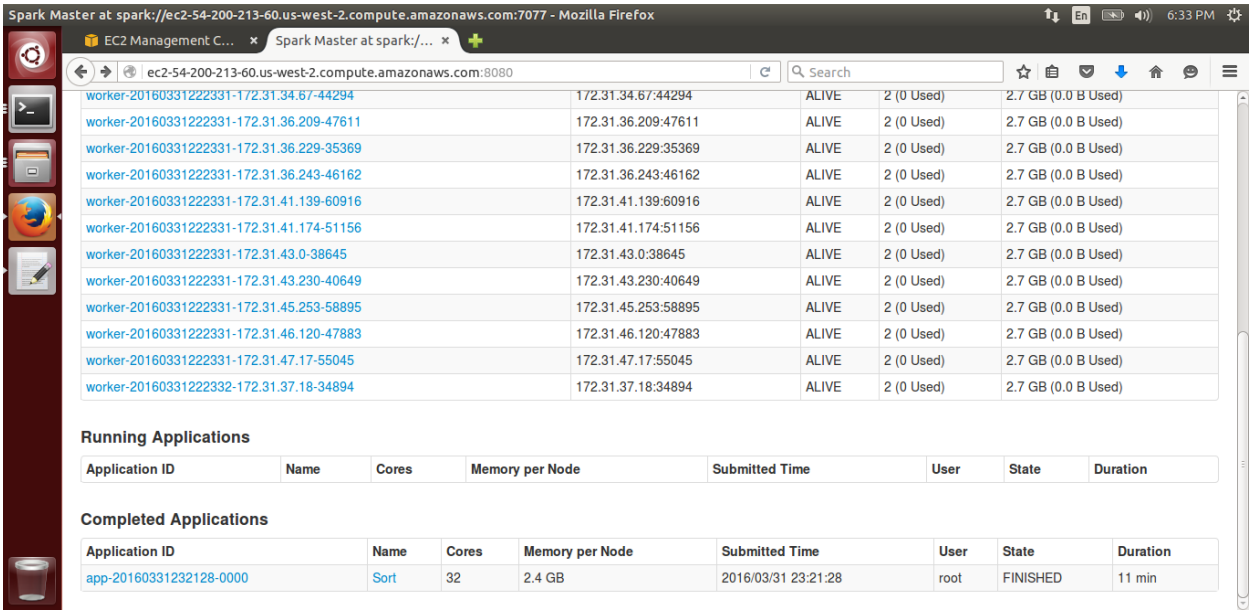




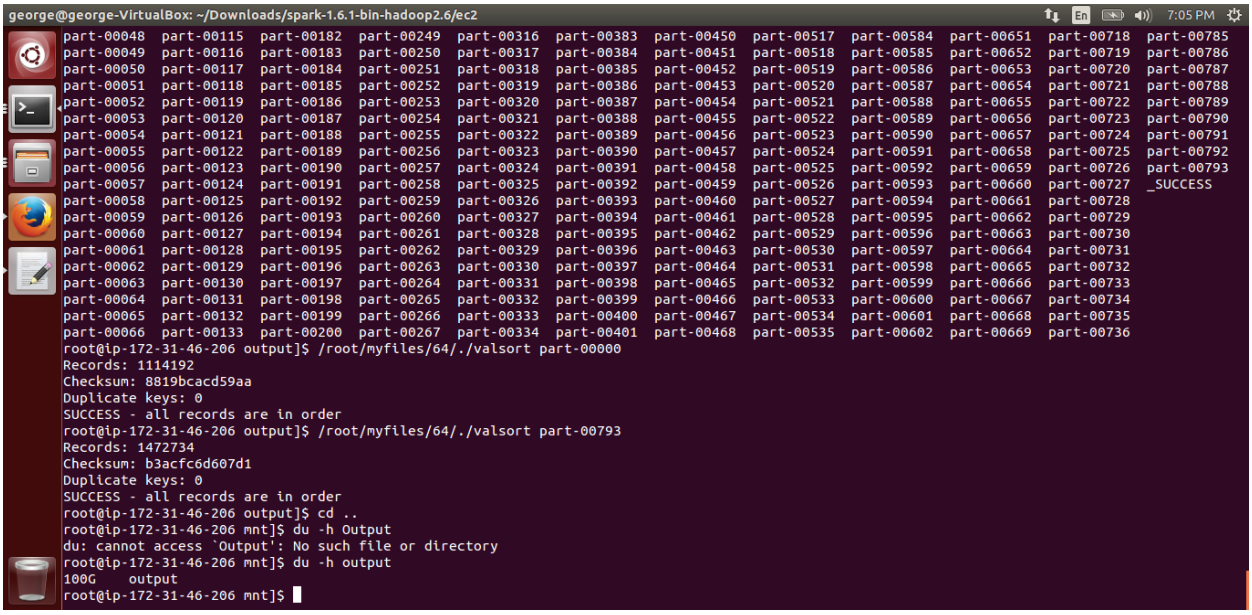
The program was run on this cluster to sort 100gb data


Spark Master at spark://ec2-54-200-213-60.us-west-2.compute.amazonaws.com:7077 - Mozilla Firefox
6:21 PM
EC2 Management C...
Spark Master at spark:/...
ec2-54-200-213-60.us-west-2.compute.amazonaws.com:8080
Search
worker-20160331222331-172.31.34.67-44294 172.31.34.67:44294 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.36.209-47611 172.31.36.209:47611 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.36.229-35369 172.31.36.229:35369 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.36.243-46162 172.31.36.243:46162 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.41.139-60916 172.31.41.139:60916 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.41.174-51156 172.31.41.174:51156 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.43.0-38645 172.31.43.0:38645 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.43.230-40649 172.31.43.230:40649 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.45.253-58895 172.31.45.253:58895 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.46.120-47883 172.31.46.120:47883 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222331-172.31.47.17-55045 172.31.47.17:55045 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
worker-20160331222332-172.31.37.18-34894 172.31.37.18:34894 ALIVE 2 (2 Used) 2.7 GB (2.4 GB Used)
Running Applications
Application ID Name Cores Memory per Node Submitted Time User State Duration
app-20160331232128-0000 (kill) Sort 32 2.4 GB 2016/03/31 23:21:28 root RUNNING 7 s
Completed Applications
Application ID Name Cores Memory per Node Submitted Time User State Duration


Sort - Spark Jobs - Mozilla Firefox
6:27 PM
EC2 Management C...
Sort - Spark Jobs
ec2-54-200-213-60.us-west-2.compute.amazonaws.com:4040/jobs/
Search
Spark 1.6.1
Jobs Stages Storage Environment Executors
Sort application UI
Spark Jobs (?)
Total Uptime: 6.3 min
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 1
Event Timeline
Active Jobs (1)
Job Id Description Submitted Duration Stages: Succeeded/Total Tasks (for all stages): Succeeded/Total
1 saveAsTextFile at SimpleApp.scala:11 2016/03/31 23:22:26 5.3 min 1/2 826/1588
Completed Jobs (1)
Job Id Description Submitted Duration Stages: Succeeded/Total Tasks (for all stages): Succeeded/Total
0 sortByKey at SimpleApp.scala:11 2016/03/31 23:21:29 57 s 1/1 794/794



The sorting was completed in 11 minutes



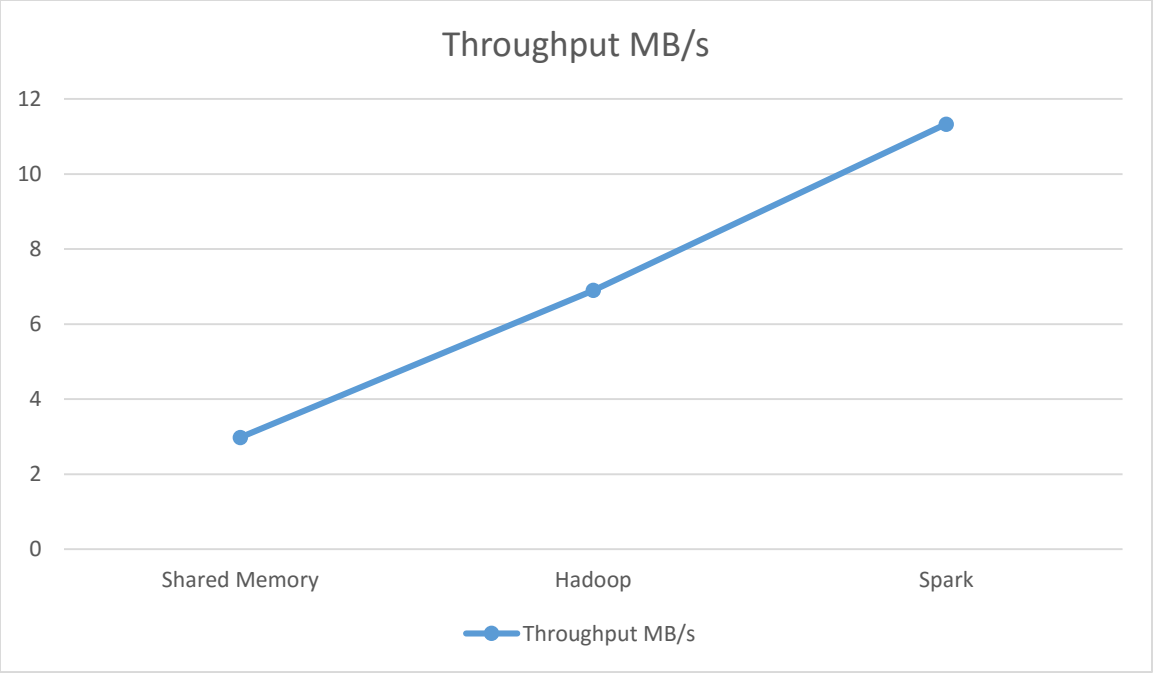
The output was verified using valsort

8) Performance

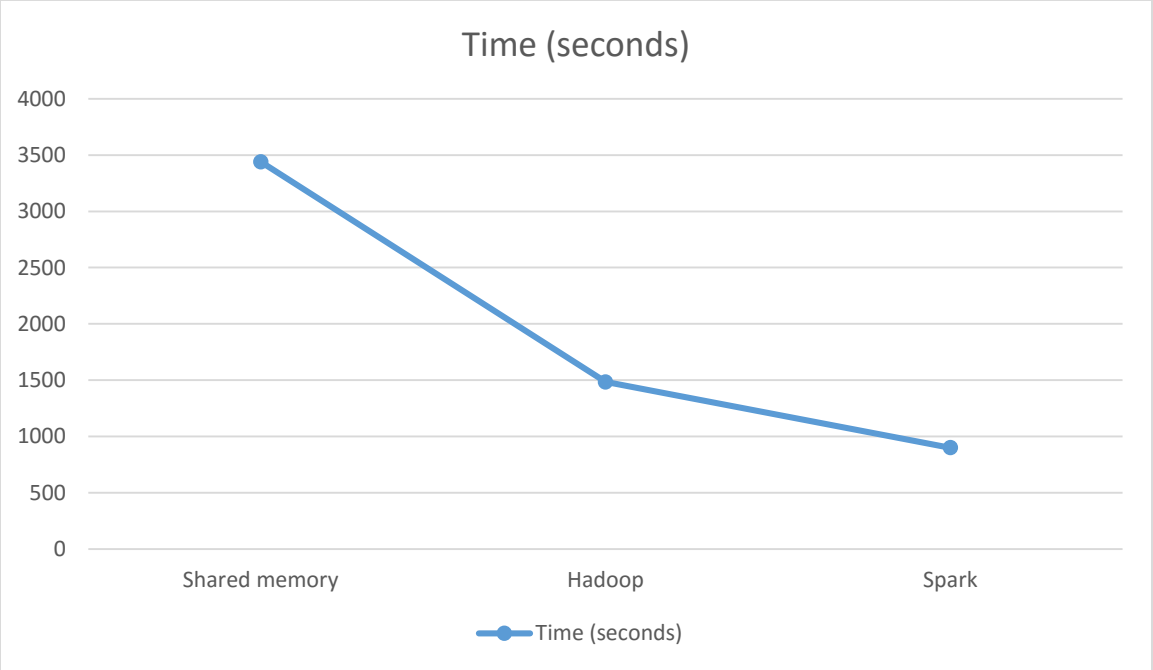
<u>1 node performance</u>

10GB sorts

Name	Time	Throughput
Shared Memory	3440 s	2.98 MB/s
Hadoop	1484 s	6.90 MB/s
Spark	900 s	11.33 MB/s



Graph showing increased throughput for Hadoop and spark as compared to shared memory sort

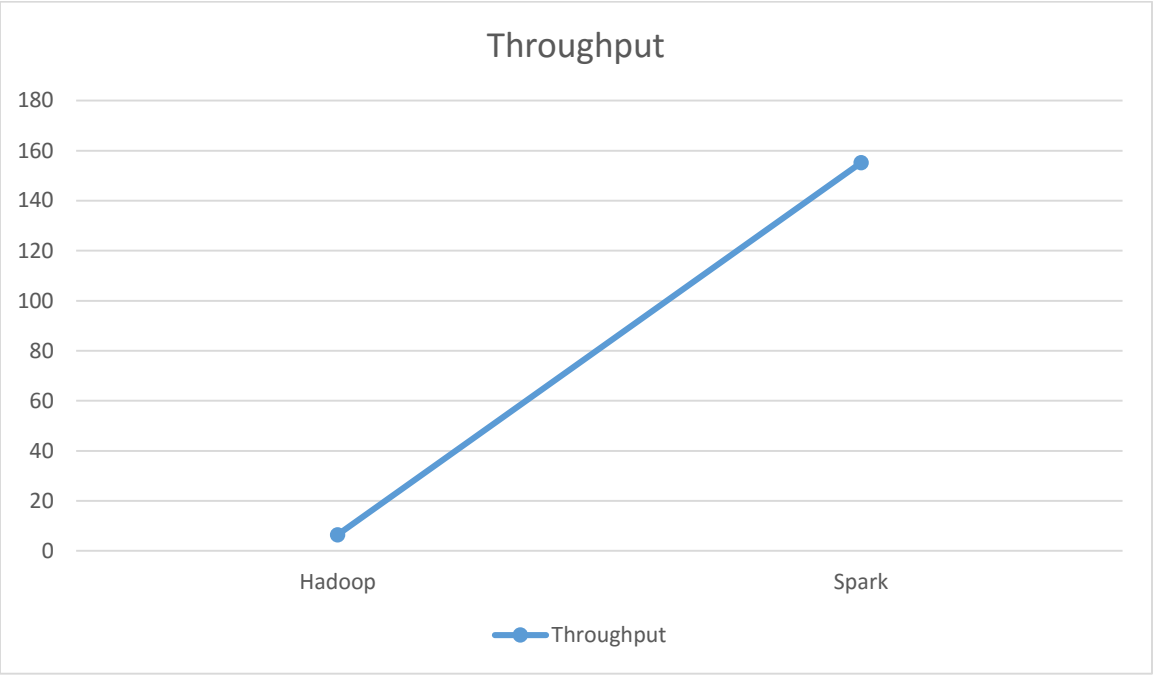


Graph showing Decreased Time of execution as compared to shared memory

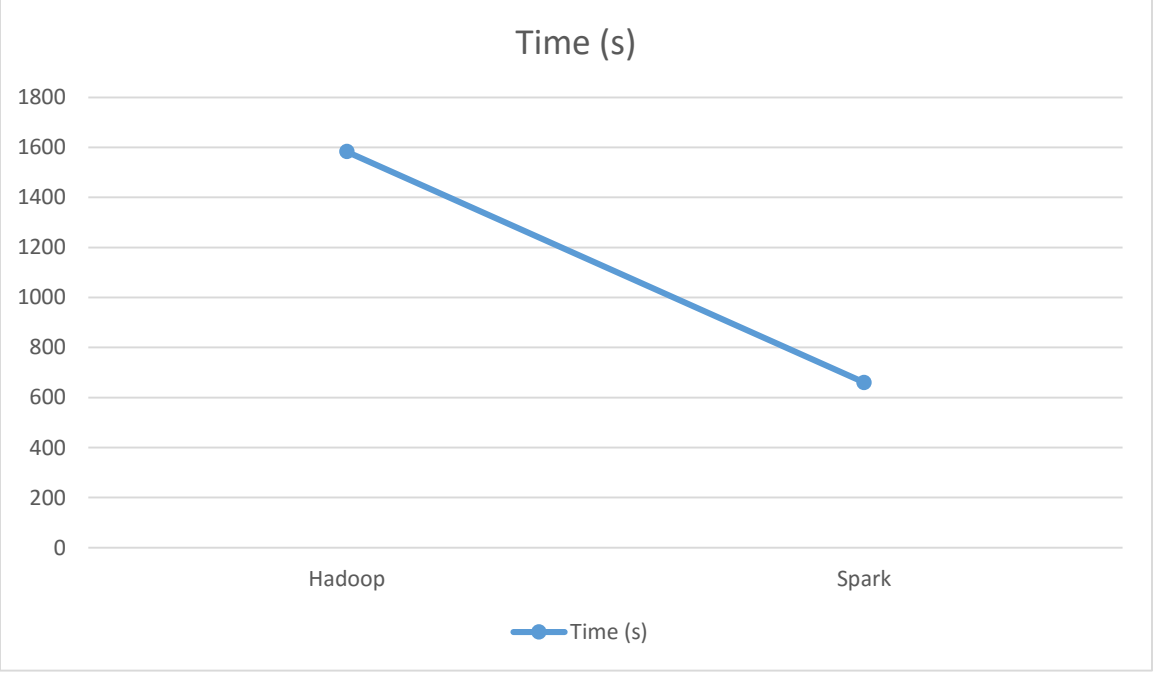
16 node performance

100GB sorts

Name	Time	Throughput
Hadoop	15836s	6.46 MB/s
Spark	660 s	155.15 MB/s



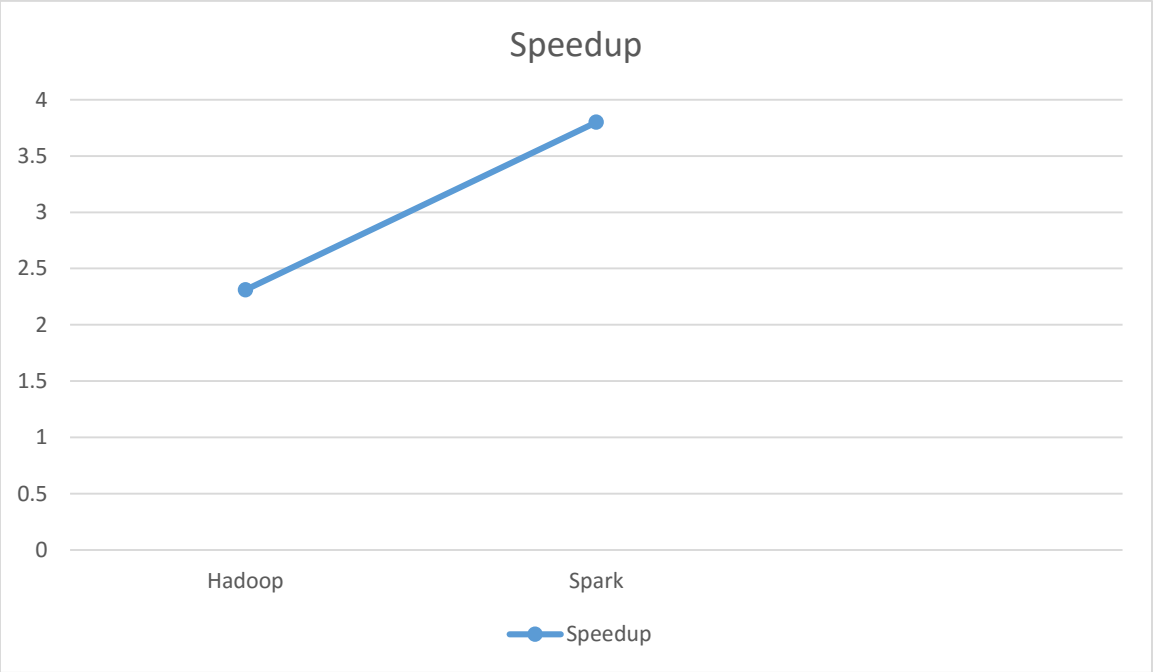
Graph showing the increased throughput of spark compared to Hadoop



Graph showing the decrease in time of execution of Spark as compared to Hadoop

Speed up Shared Memory

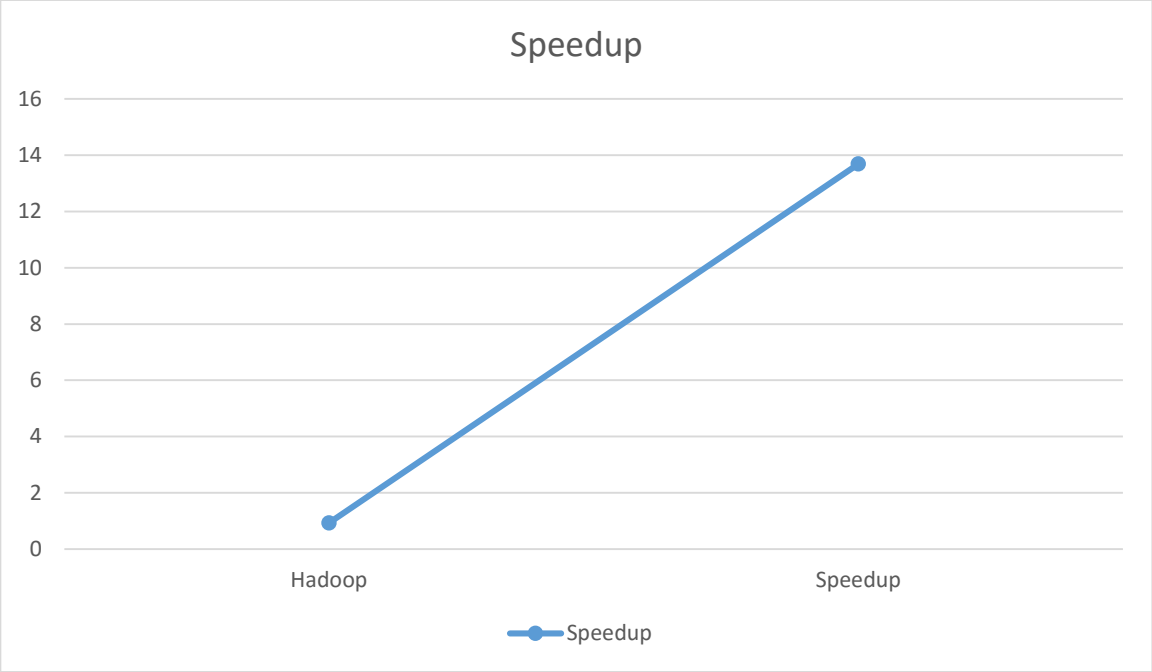
Name	Throughput (1 node)	Speed Up
Shared Memory	2.98 MB/s	
Hadoop	6.90 MB/s	2.31
Spark	11.33 MB/s	3.80



Graph showing the high increase in speedup of spark as compared to Hadoop and shared memory

Speedup Spark and Speedup Hadoop

Name	Throughput (1 node)	Throughput (16 node)	Speed Up
Hadoop	6.90 MB/s	6.46 MB/s	0.93
Spark	11.33 MB/s	155.15 MB/s	13.69



Graph showing the increased speed up of spark as compared to Hadoop

It can be concluded that Hadoop has more throughput than the shared memory sort and is better than normal programming practices

Spark is definitely an improvement over Hadoop in throughput and the speed up attained when more nodes are added

9)TeraSort

Terasort was attempted for shared memory. It was taking too much time to execute so it was not completed to save money The completed code has been submitted with the assignment

10)Conclusions

Environment Details

OS: **Ubuntu Server 14.04 LTS (HVM), SSD Volume Type** - ami-9abea4fb 64 bit

Java: 1.7.0_95

Hadoop: 2.7.2

Spark: Spark 1.6.1

Difficulties Faced:

-Shared Memory:

The external sort algorithm was tricky to implement

-Hadoop

Pseudo Mode was easy but setting up a cluster was hard especially because lot of repetitive tasks were there

-Spark

Spark environment was easy to setup but the program was hard to write mainly because I decided to do it in python and I was new to python

It was definitely an experience to do this assignment. The challenges involved were huge as the learning curve was very high. But in the end it felt nice to have done a sorting over data bigger than memory, setting up clusters and writing codes on that