CS 553 Cloud Computing

Programming Assignment #2

Sort on Hadoop/Spark

Project Report

Harsh Parikh (A20338453)

Introduction

For all the benchmarks, I experimented on Amazon EC2, c3.large with 2 core CPU, Intel(R)
Xeon(R) CPU E5-2680 v2 @ 2.89GHz.

```
[ec2-user@ip-172-31-13-167 ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    2
Core(s) per socket:    1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 62
Model name:            Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz
Stepping:              4
CPU MHz:               2800.070
BogoMIPS:              5600.14
Hypervisor vendor:     Xen
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              25600K
NUMA node0 CPU(s):     0,1
[ec2-user@ip-172-31-13-167 ~]$
[ec2-user@ip-172-31-13-167 ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    2
Core(s) per socket:    1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 62
Model name:            Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz
Stepping:              4
CPU MHz:               2800.070
BogoMIPS:              5600.14
Hypervisor vendor:     Xen
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              25600K
NUMA node0 CPU(s):     0,1
[ec2-user@ip-172-31-13-167 ~]$
```

This programming assignment covers the Sort application implemented in 3 different ways: Python Shared-Memory, Hadoop, and Spark. I first read a large file and sort it And place the result in .txt format is sorted by its appearance time.

Methodology

For Shared-Memory Sorting Application, I use python and multi-thread(multi-processing) technology to implement.

For Hadoop Sorting Application, I use Java based on a Hadoop framework and Map-reduce.

For Spark Sorting Application, I use Python and Map Reduce and Spark platform.

Runtime Environment Settings

Amazon Web Services EC2

AMI – Amazon Linux (HVM), SSD Volume Type – ami-3d50120d, 2 x 16 GB
Hadoop – 2.7.2
Spark - 1.6.1
Java - 1.7.0_95
Shared-Memory:

EC2 instance type : c3-large

Hadoop part :

EC2 instance type : (master node) c3.4xlarge, (slaves node)c3.large

Spark park:

EC2 instance type : (master node) c3.4xlarge, (slaves node)c3.large ,d2.xlarge

Creating a Virtual Cluster using AWS:

First log in into the Amazon AWS service website (http://aws.amazon.com/) and select EC2 from the network and services section. Then select launch instance here. Then configure different service as per requirements.

Then choose the AMI either Amazon Linux or Ubuntu 14.04. For Hadoop and Spark I worked on Amazon AMI and for Shared-Memory I worked on Ubuntu 14.04. Then choose
the instance type which can be micro, small, medium, large Xlarge, 2Xlarge, 4Xlarge or 8Xlarge depends on the memory requirements.

Then configure the instance i.e I choose the instance to be either spot instance or on
Demand instance. Then add storage to our selected instance and configure the security group. and add TCP, ICMP and all Traffic protocols from the security group.

After doing this. Launch the instance from local machine and
access the instance by connecting to the instance through public DNS address of it.

Implementing Sorting Application in Python(Shared-Memory):

I implemented a Python program to sort the given input file of size 10GB and print the sorted output into a file. I performed sorting on 1,2,4 and 8 threads and measure the Throughput and time of those.

Implementing Sorting Application in HADOOP:

After configuring instances on AWS, first download and install HADOOP and extract its components and adding
the library paths to the bashrc file. The
Sorting application is implemented using HADOOP for 10GB on a single node and 100GB
dataset on 16 nodes.

Implementing Sorting Application in Spark:

After downloading Spark, we need to install Scala and add its library paths to bashrc file. Then we need to implement Spark Sorting Application in either Python, Java or Scala. I implemented Sorting Application in Python. And run spark on 1 and 16 node for 10GB and 100 GB respectively.

Performance:

After generating datasets as per requirements, we need to measure the performance of Sorting Application on Shared-Memory, HADOOP and Spark on 1 node and 16 nodes. These are compared and performance graphs are drawn for them. The performance result of shared memory 1 thread is compared with the performance of Hadoop Sorting on 1 node and Spark's 1 node. For each of the above separate graphs are drawn to show their performance. At the end comparison of Speed-up for all three aspects will be shown.

Creating an instance in Amazon(AWS):

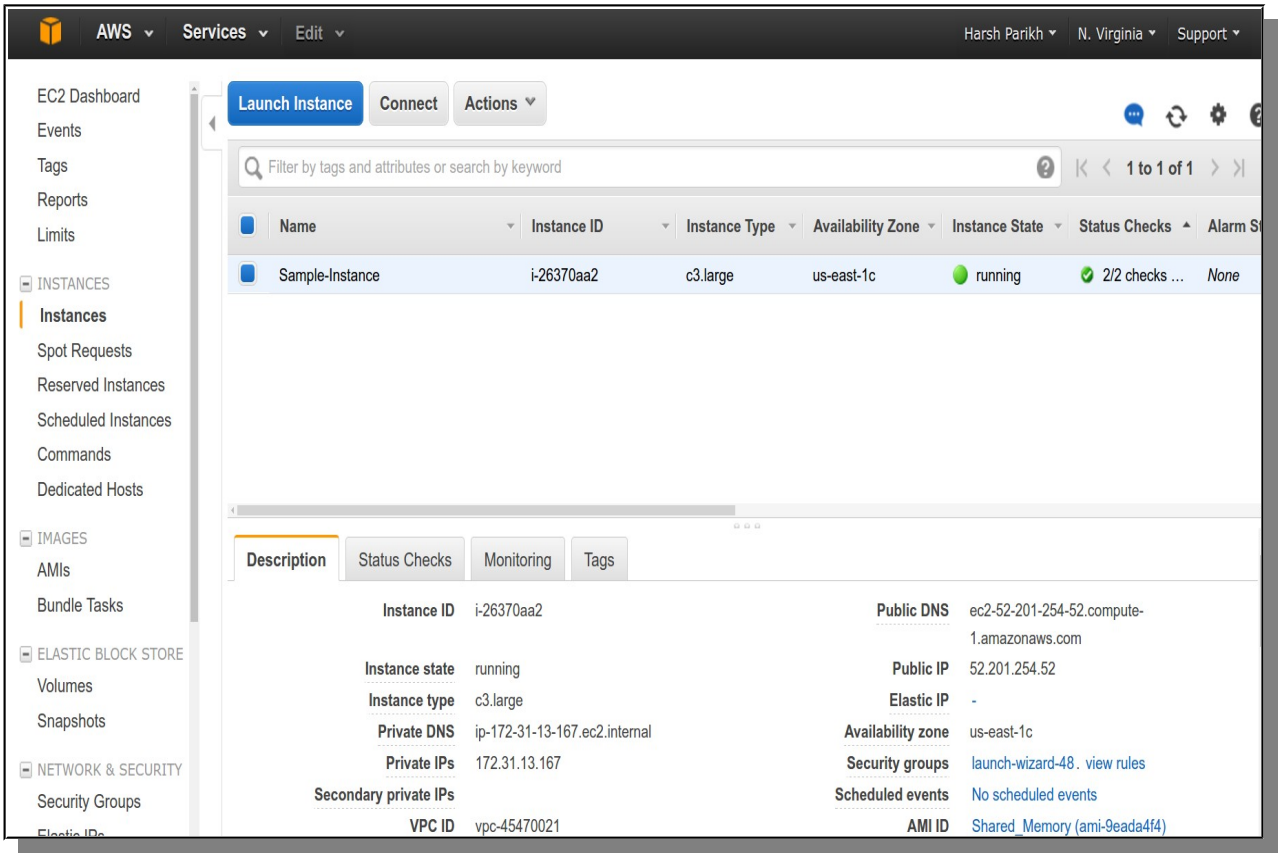


Figure [1] c3.large instance.

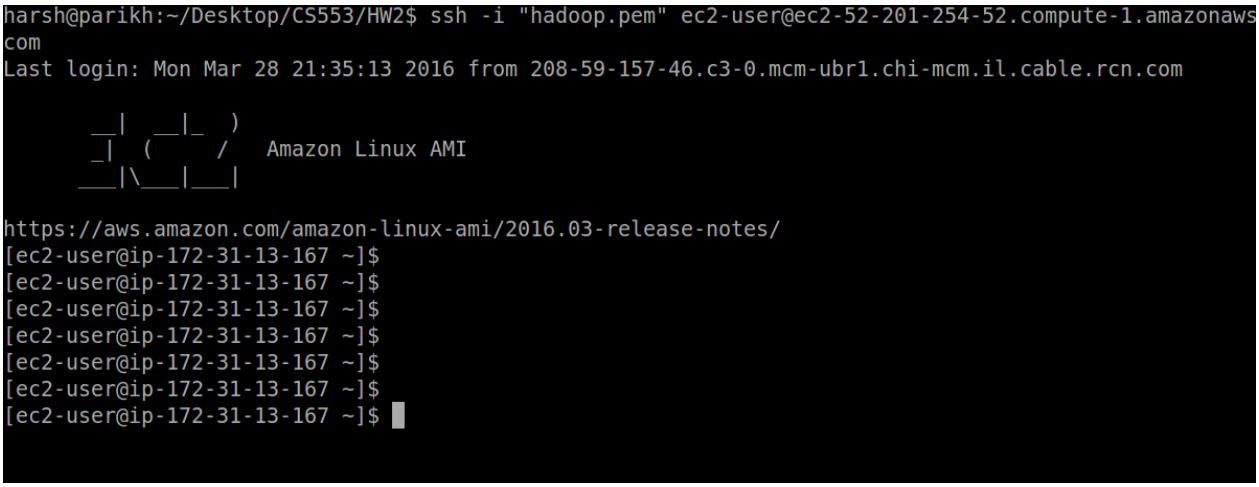


Figure [2] c3.large instance Terminal.

After getting into terminal, we have to follow this process.

chmod 400 key_in_pem (To change the permission, to be read by owner.)

ssh –i key_in_pem IP address

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

First update all applications by **sudo yum update**
Install Java by using **sudo yum install java-devel.**
Now check the java version by using **java --version**.

Sorting Application in Shared-Memory:

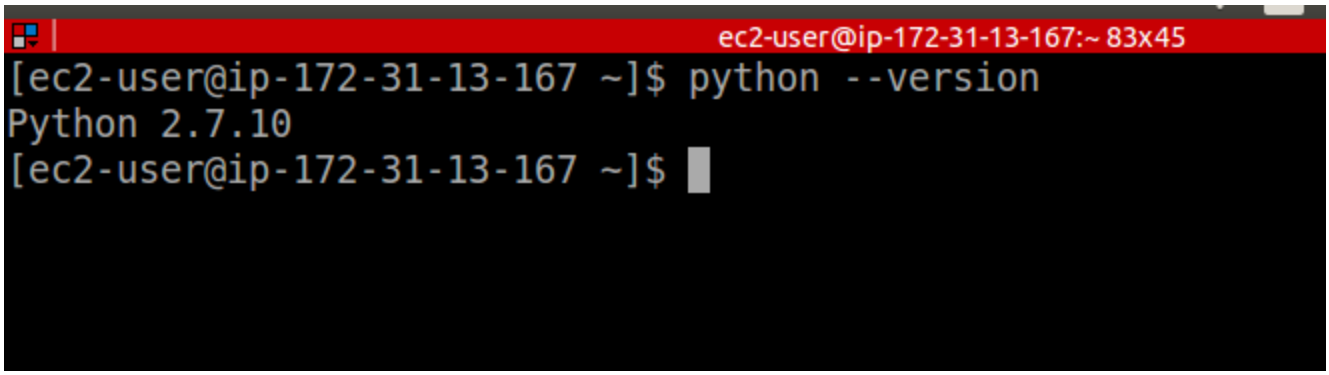


Figure [3] Python Version

Methodology:

From the given datasets in file, first I am counting the file size and reading the data line by line from file.

On the basis of number of lines, I further split the data into equal no. of chunks.

Then perform **Merge Sort** on all chunks' keys and store one by one sorted chunks into File.

Now I have all the files with sorted chunks. Now take the elements from different files one by one and do the merge sort again on them and make a new dictionary containing all this keys.

Now Based on this keys, we need to map it's value and put key-value pair in file with all sorted keys.

The threads are started and execute one file per thread. Numbers of threads are dependent on the number of chunks of a file.

Mutex Lock is to achieve the synchronization and hence avoid deadlocks in threads.

To run the program, *python Shared_Memory_Merge_Sort.py.*

The time of all threads one bye one will be stored in the text file name **output.txt.**

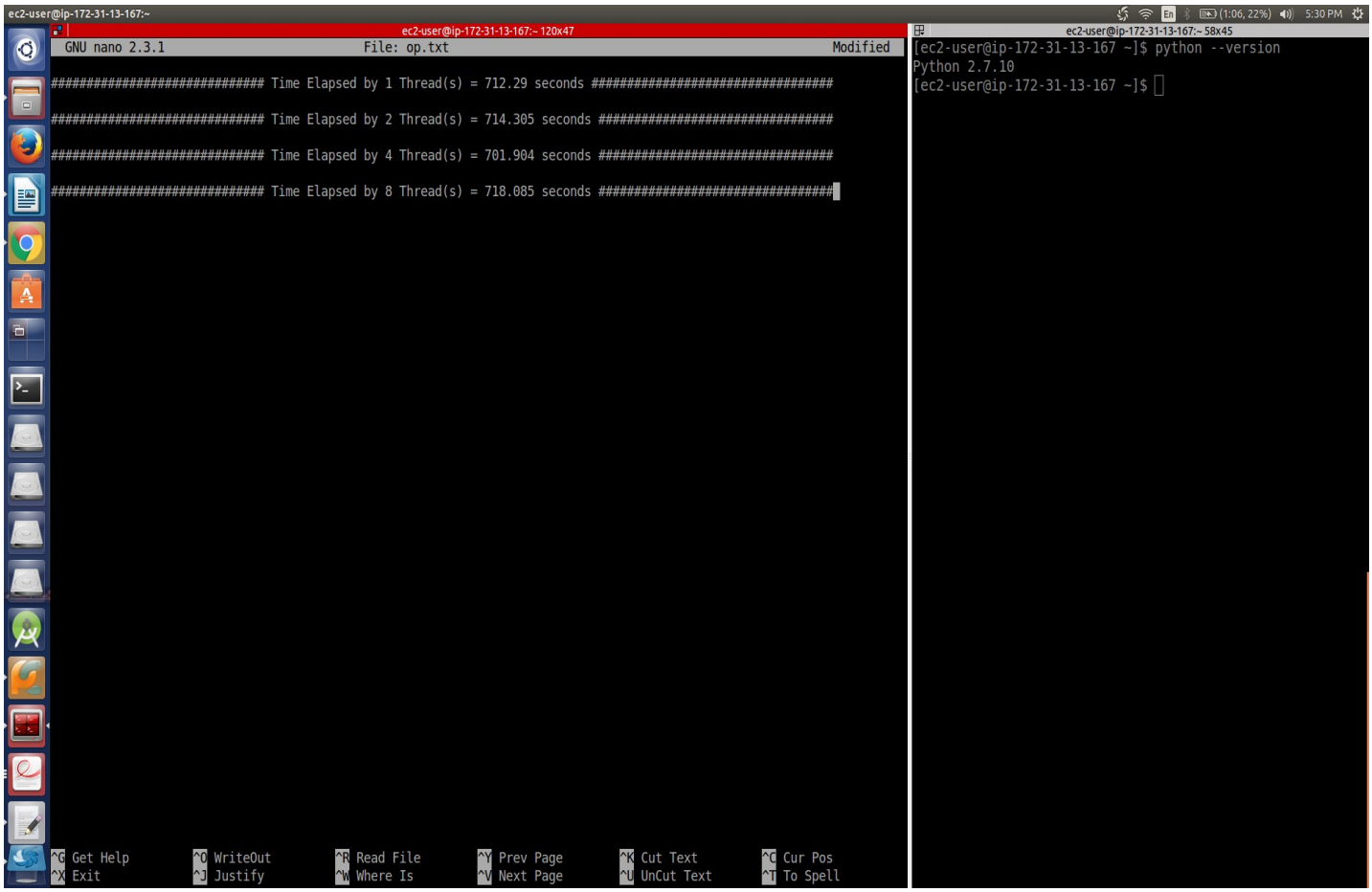
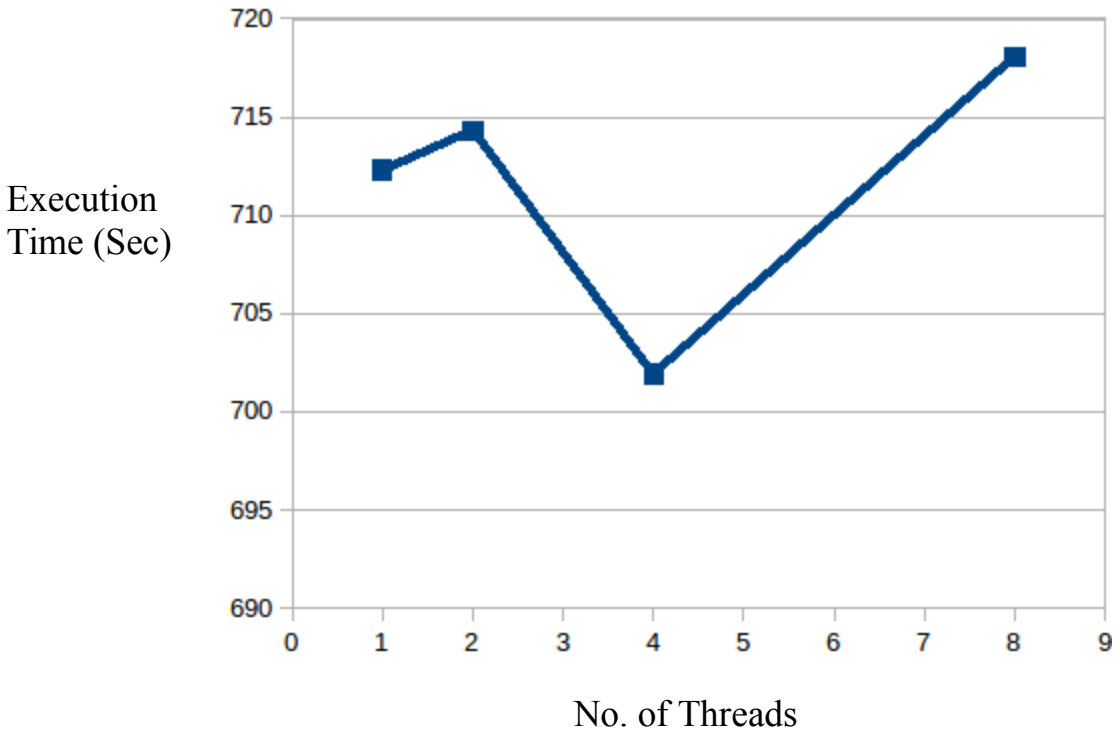


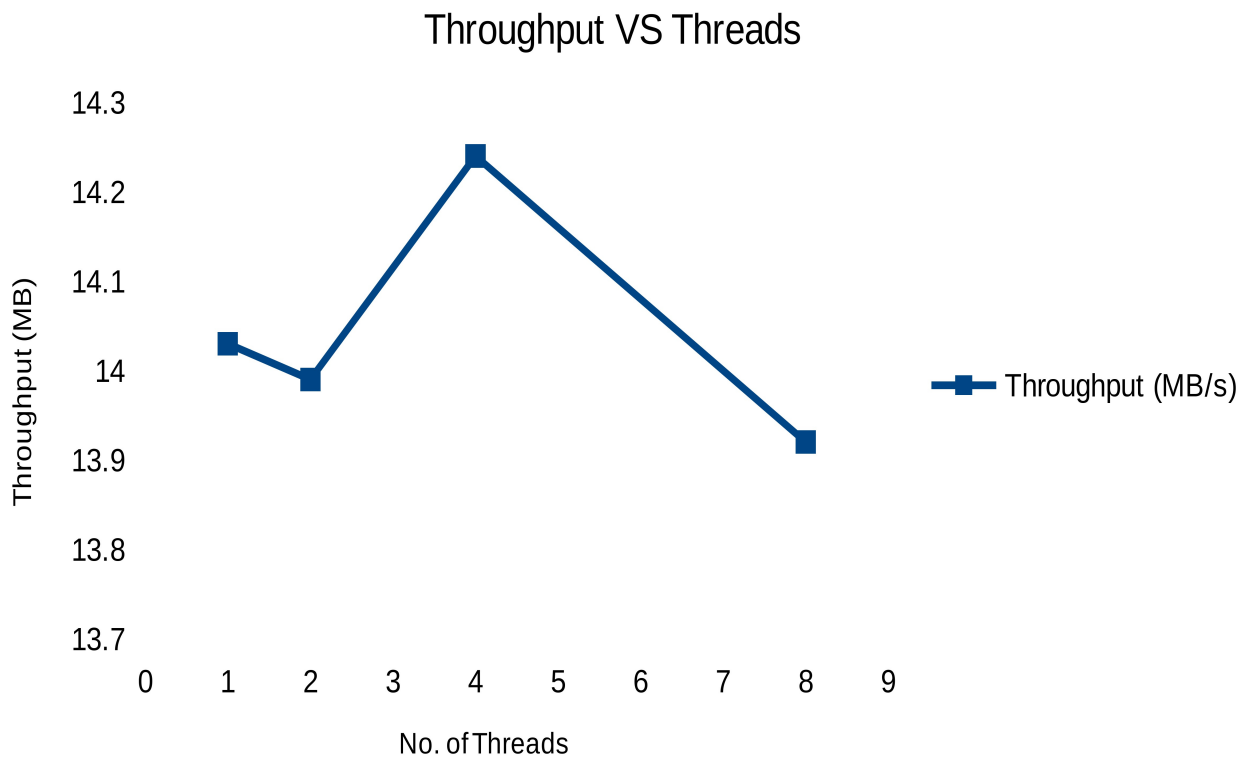
Figure [4] Shared-Memory Output stored in file

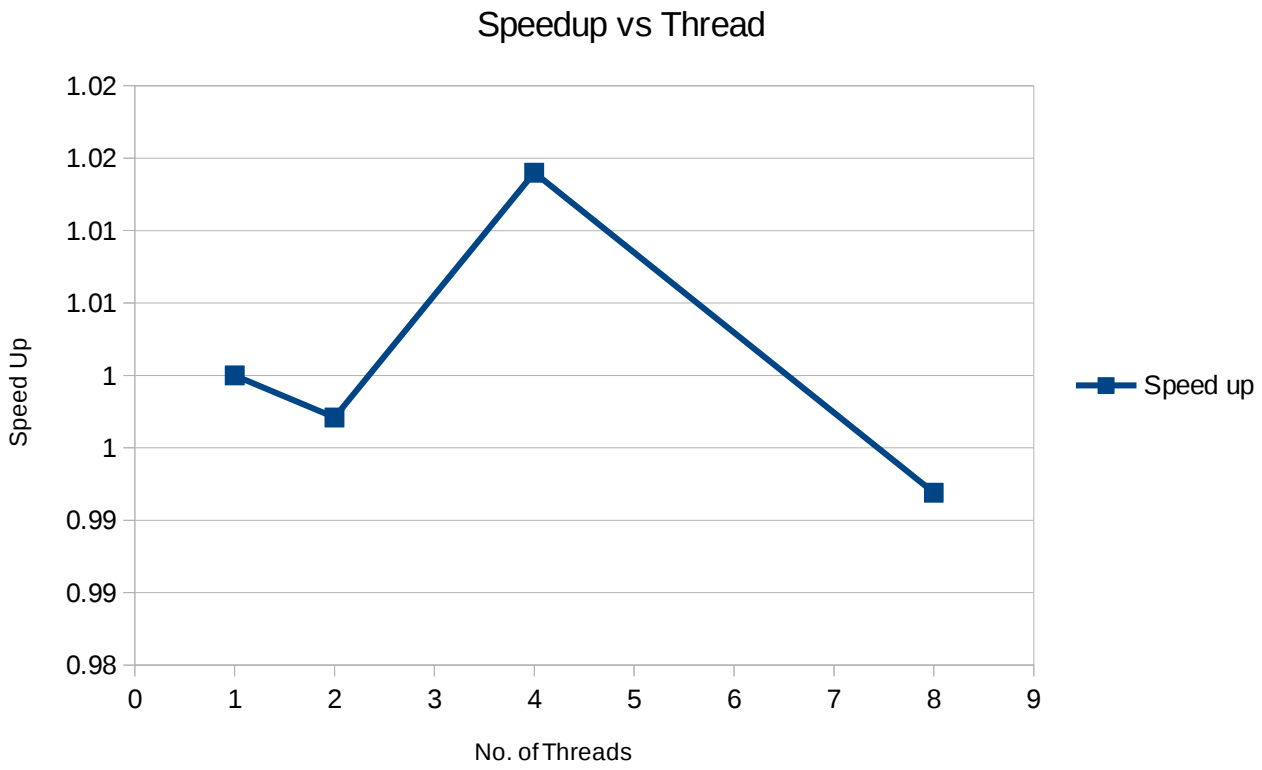
Outputs:

10 GB DATASET on **c3.large instance** (Shared-Memory**)**

Number of Threads	Execution Time	Throughput (MB/s)	Speed up
1	712.29 seconds	14.03	1
2	714.305 seconds	13.99	0.9971
4	701.904 seconds	14.24	1.014
8	718.085 seconds	13.92	0.9919


Execution Time (Sec)
720
715
710
705
700
695
690
0
1
2
3
4
5
6
7
8
9
No. of Threads


Throughput VS Threads
Throughput (MB)
14.3
14.2
14.1
14
13.9
13.8
13.7
0
1
2
3
4
5
6
7
8
9
No. of Threads
Throughput (MB/s)



Valsort on Shared-Memory:

```
[ec2-user@ip-172-31-13-167 64]$ ./valsort /mnt/raid/output
Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
[ec2-user@ip-172-31-13-167 64]$
```

Figure [5] Valsort of 10 GB.

Output of Shared-Memory:

```
[ec2-user@ip-172-31-13-167 raid]$ head -10 output
    "O!uve  000000000000000000000000001228D4  77778888000022224444DDDDDDDDEEEE00000000CCCC7777DDDD
    PMd32=  000000000000000000000000003440CC1  FFFFEEEE6666CCCCBBBB999933335555DDDDDDDD777788886666
    ^3CO],  000000000000000000000000000158C5C5  5555AAAA9999EEEE888822229999CCCCDDDD6666555544442222
   !&S3/]]  000000000000000000000000002145D78  8888BBBBDDDD1111CCCC55556666BBBB1111EEEEDDDD22229999
   !,=U#,9  000000000000000000000000019072E3  33332222FFFFBBBB0000FFFFAAAA666655553333DDDD3333CCCC
   !0f[ITd  000000000000000000000000003CAAB4B  9999FFFF555533337777CCCC4444BBBB7777EEEEBBBBDDDD4444
   !f6Suy2  000000000000000000000000003ABFD84  EEEE555555556666AAAA5555BBBBDDDD0000111166660000DDDD
   #%NIpq.  000000000000000000000000003B36FB9  1111000033334444111166666666AAAAAAAA00001111CCCCEEEE
   #'^cl'~  000000000000000000000000002EDC5C8  8888AAAA11114444FFFF77773333EEEE44440000FFFF99999999
   $"-'Q)]  000000000000000000000000005F1265D  CCCC6666EEEE22220000DDDDAAAA88886666BBBB00006666AAAA
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$ tail -10 output
~~~uq2k#=U  000000000000000000000000002C06745  99991111DDDD222211110000FFFFEEEEFFFF33337777CCCC2222
~~~v/0&Qnm  000000000000000000000000004709701  CCCC88883333FFFF0000000000099991111FFFF777744446666
~~~yKOl:gE  000000000000000000000000002048B4F  CCCC1111444488882222666 6BBBB888855557777EEEEBBBB0000
~~~yK^H.il  000000000000000000000000000463D004  44440000FFFF3333999944447777DDDDFFFFAAAA11118888DDDD
~~~yL;C'XE  000000000000000000000000005B0D211  2222EEEE3333000022221111CCCCFFFF555577774444BBBB6666
~~~zbA_ Tt  000000000000000000000000007F9F4F  BBBBCCCC666655559999FFFF8888AAAA11116666AAAABBBB0000
~~~zeO^FEg  000000000000000000000000001E06130  4444CCCCBBBB999922228888555588 88CCCCFFFF000011111111
~~~}GxjWHI  000000000000000000000000000CA1345  777711118888AAAAAAAA22221111BBBB00002222BBBBCCCC2222
~~~}P;]g0g  000000000000000000000000040DA3E4  4444FFFF444466663333EEEE88888888DDDDEEEE44442222DDDD
~~~}kU|K<p  000000000000000000000000005E4A0AA  0000666655551111BBBB88889999AAAA55550000333355557777
[ec2-user@ip-172-31-13-167 raid]$
```

Figure [6] Output Files

Sorting Application on Hadoop:

In Hadoop framework, Map-Reduce function takes the data in <key,value> format. So to get key and value, substring the input data line by line, and assign it to the key and value respectively. Sorting is done by the frame work and after data sorting is done, Reduce function comes in the action. The main work of the reduce function is to combine the same key data, and then to add them in one <key,value> format. After that, write the data in same format as the input, so that it will be easy to validate using the valsort.

Hadoop Configuration:

1 Node Configuration:

Edit **/etc/hosts** file, put following in the file.

172.31.22.75 master
172.31.17.52 slave

edit **~/.ssh/config** file

Host master
HostName ec2-54-85-78-244.compute-1.amazonaws.com
User ec2-user
IdentityFile ~/.ssh/hadoop.pem

Host slave
HostName ec2-54-152-71-220.compute-1.amazonaws.com
User ec2-user
IdentityFile ~/.ssh/hadoop.pem

Edit **.bashrc File** as per following

```
# Set Hadoop-related environment variables

export HADOOP_HOME=$HOME/hadoop

export HADOOP_CONF_DIR=$HOME/hadoop/etc/hadoop

export HADOOP_MAPRED_HOME=$HOME/hadoop

export HADOOP_COMMON_HOME=$HOME/hadoop

export HADOOP_HDFS_HOME=$HOME/hadoop

export YARN_HOME=$HOME/hadoop

# Set JAVA_HOME

export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.95.x86_64/jre

export PATH=$PATH:$JAVA_HOME/bin

# Add Hadoop bin/ directory to PATH

export PATH=$PATH:$HOME/hadoop/bin

export HADOOP_CLASSPATH=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.95.x86_64/lib/tools.jar
~
~
~
~
```

Now, go to **hadoop/etc/hadoop/hdfs-site.xml** file and make the following changes.

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>

<property>
<name>dfs.permission</name>
<value>false</value>
</property>

<property>
<name>dfs.namenode.name.dir</name>
<value>/home/ec2-user/hadoop/hdfs/namenode</value>
</property>

<property>
<name>dfs.datanode.data.dir</name>
<value>/home/ec2-user/hadoop/hdfs/datanode</value>
</property>

</configuration>
```

Now, go to **hadoop/etc/hadoop/hadoop-env.sh** file and make the following changes.

```
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.95.x86_64/
```

Now, go to **hadoop/etc/hadoop/mapred-site.xml** file and make the following changes.

```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>mapreduce.jobtracker.address</name>
<value>hdfs://master:8021</value>
</property>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

And while configuring 16 node, put this along with this properties in Mapred-site.xml.

```
<property>

 <name>mapreduce.job.maps</name>

 <value>2</value>

</property>

<property>

 <name>mapreduce.job.reduces</name>

 <value>2</value>

</property>
```

Now, go to **hadoop/etc/hadoop/core-site.xml** file and make the following changes.

```
<!-- Put site-specific property overrides in this file. -->

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

And while configuring 16 node, put this along with this properties in this file.

<property>

<name>hadoop.tmp.dir</name>

<value>/mnt/raid/tmp/hadoop-ec2-user</value>

</property>

Now, go to **hadoop/etc/hadoop/yarn-site.xml** file and make the following changes.

```
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
</property>
<property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>master:8031</value>
</property>
<property>
        <name>yarn.resourcemanager.address</name>
        <value>master:8032</value>
</property>
<property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>master:8030</value>
</property>
<property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>master:8133</value>
</property>
<property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>master:8088</value>
</property>
</configuration>
```

And at last put **Slave's ip** in **hadoop/etc/hadoop/slaves** file and Master's **ip** in **hadoop/etc/hadoop/masters** file.

Make this all changes for 1 node and then make a replica (Image) for slave. Just delete the **hadoop/etc/hadoop/masters** file.

Moreover, for 1 node we need to run 10gb on c3.large but we don't have enough space on Memory. So first we need to mount the volume. Following are the commands for mounting the disk volume. I have made a script called mount.sh, it includes all commands.

Mount.sh

1. *sudo yum install mdadm*

2 sudo mdadm --create --verbose /dev/md0 --level=0 --name=MY_RAID --raid-devices=2 /dev/xvdb /dev/xvdc

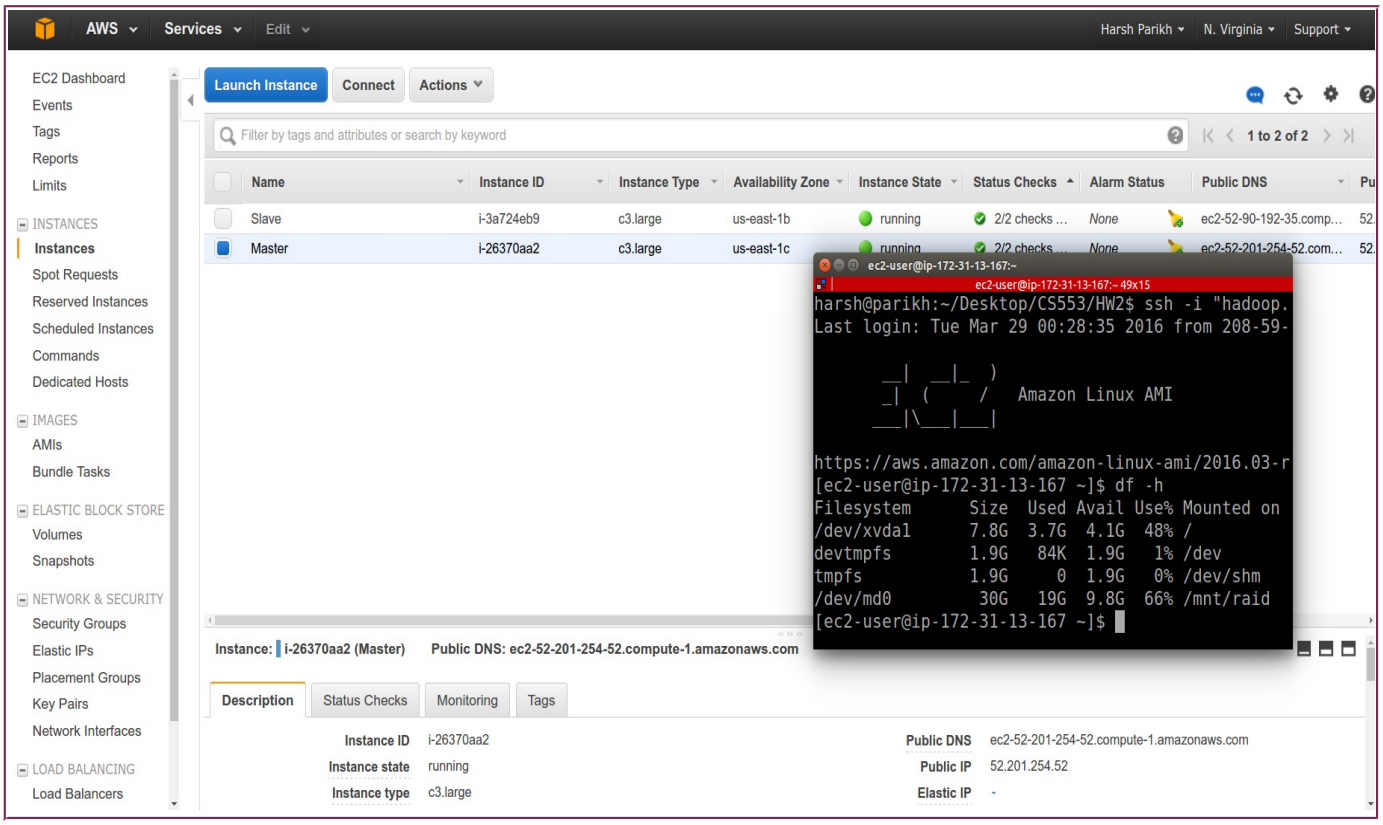
3. *sudo mkfs.ext4 -L MY_RAID /dev/md0*

4. *sudo mkdir -p /mnt*

5. *sudo mount LABEL=MY_RAID /mnt*

6. *sudo chmod 777 /mnt*

After running these commands, 2 disks will be mounted on your instance. It looks similar to this.



Output:-

1. The namenode and datanode will run on the master and slave in setup and is started by executing the following command [1]:

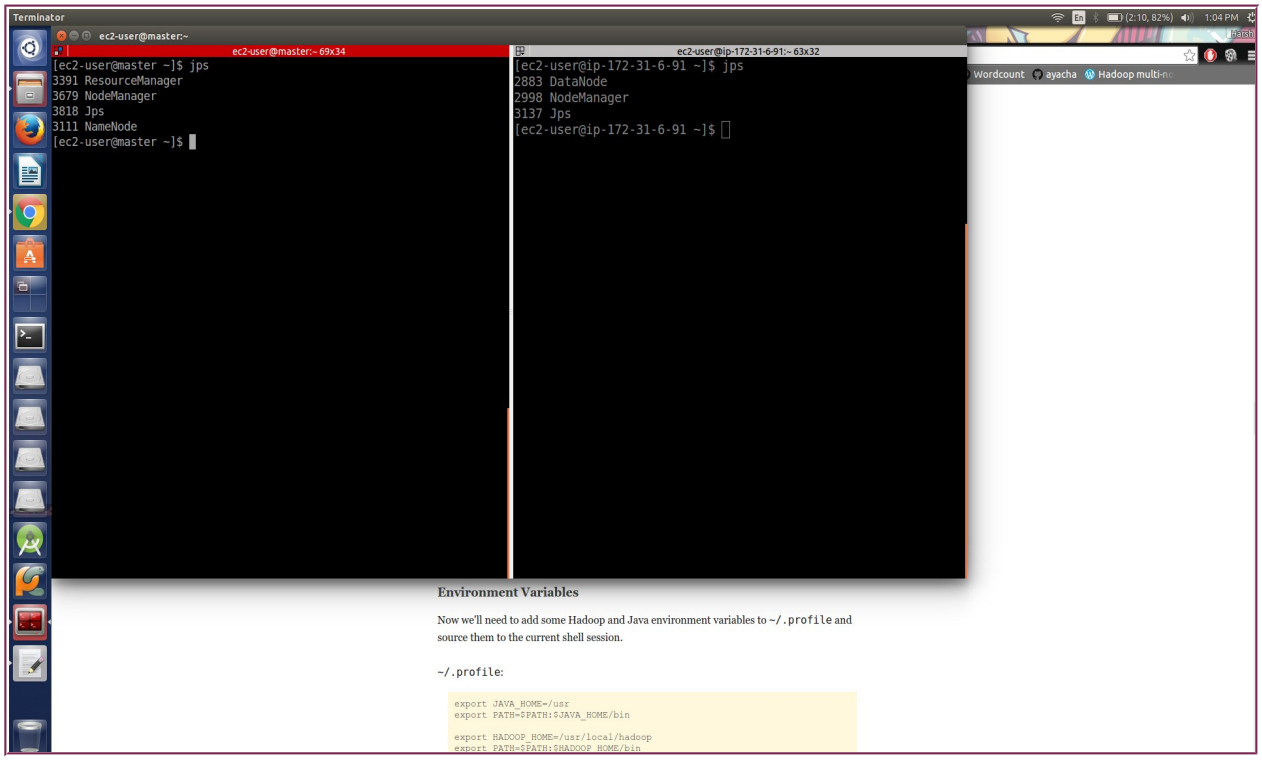
$<HADOOP_HOME>/sbin/start-dfs.sh

2.
Node manager and Resource manager will be started using following command

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

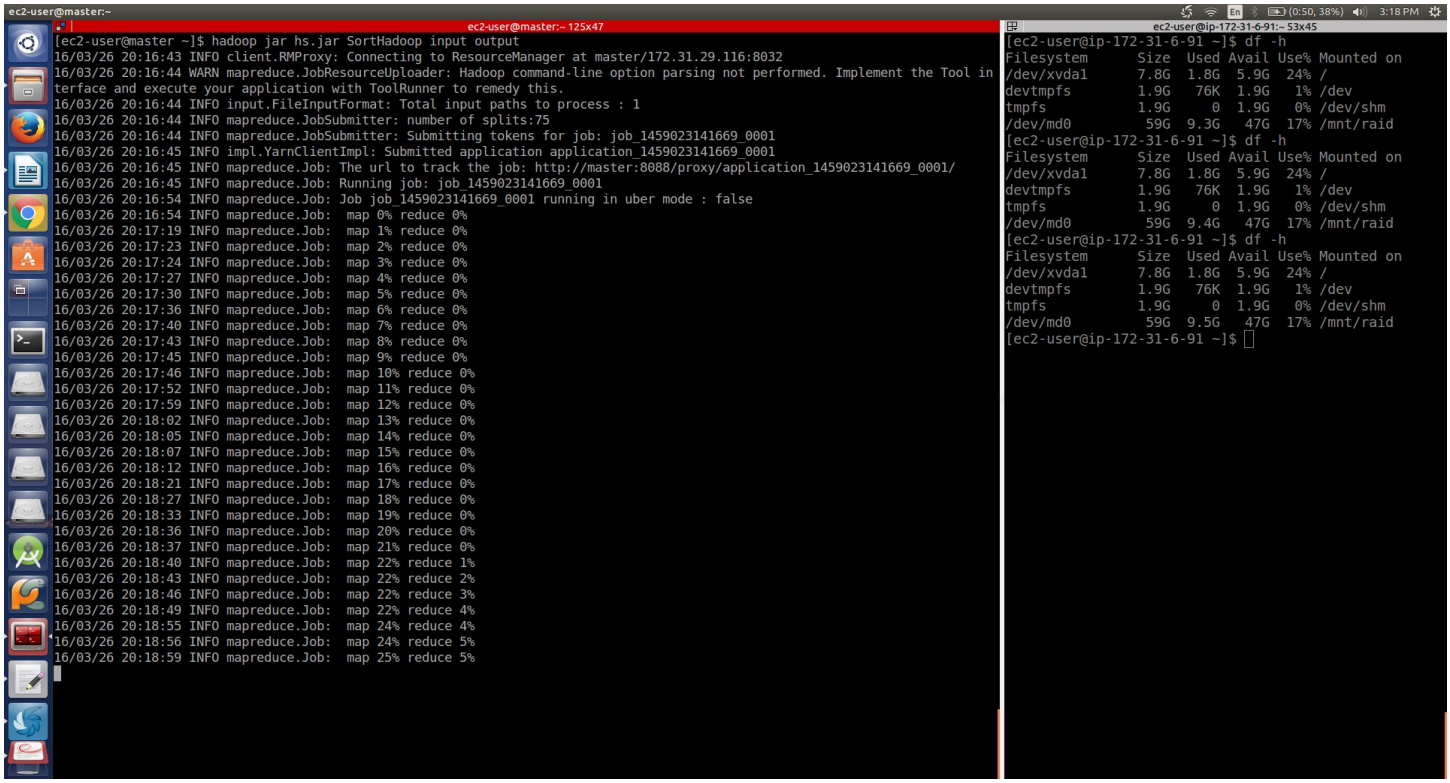
3. To start the services of hadoop file system, run the following command

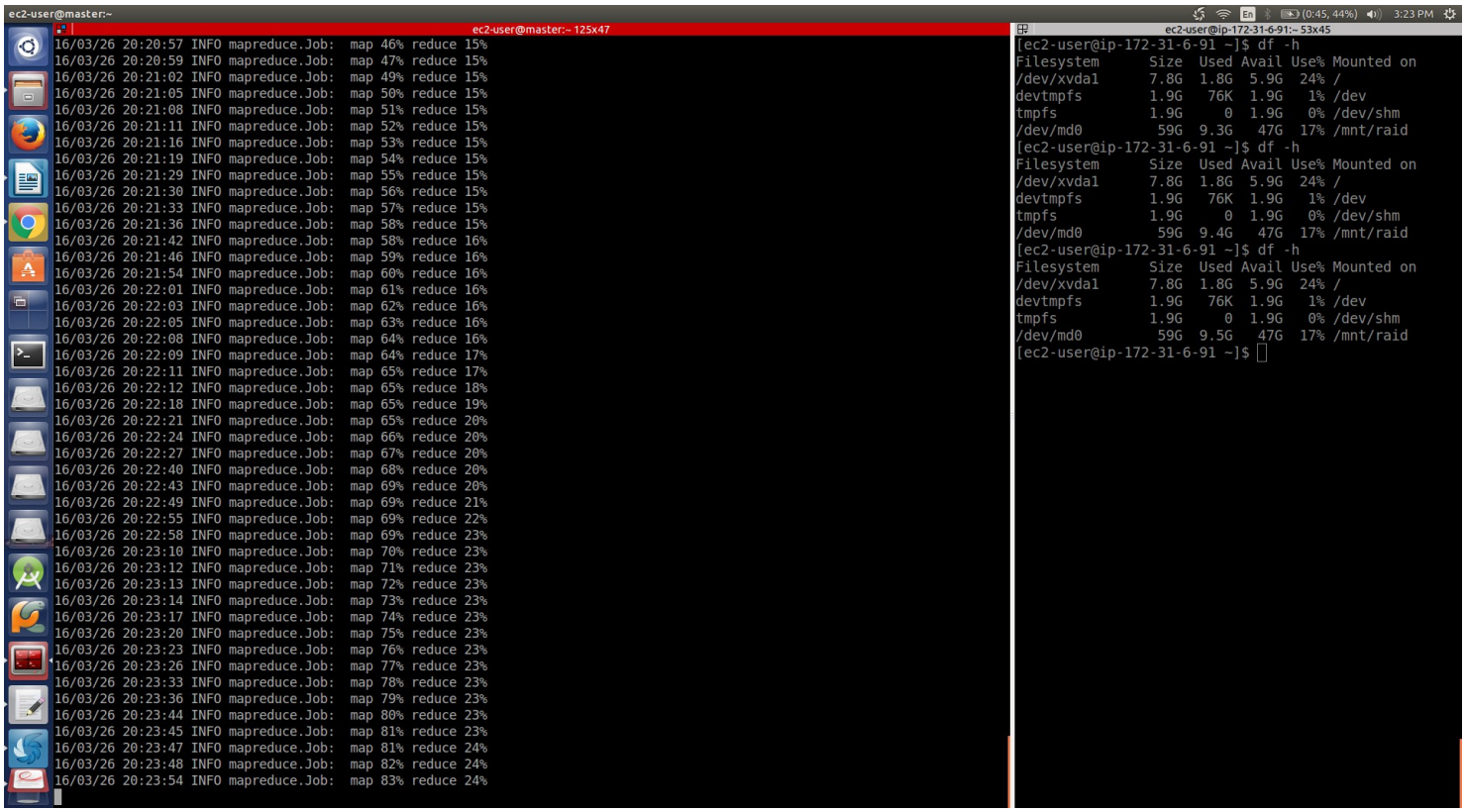
$ jps

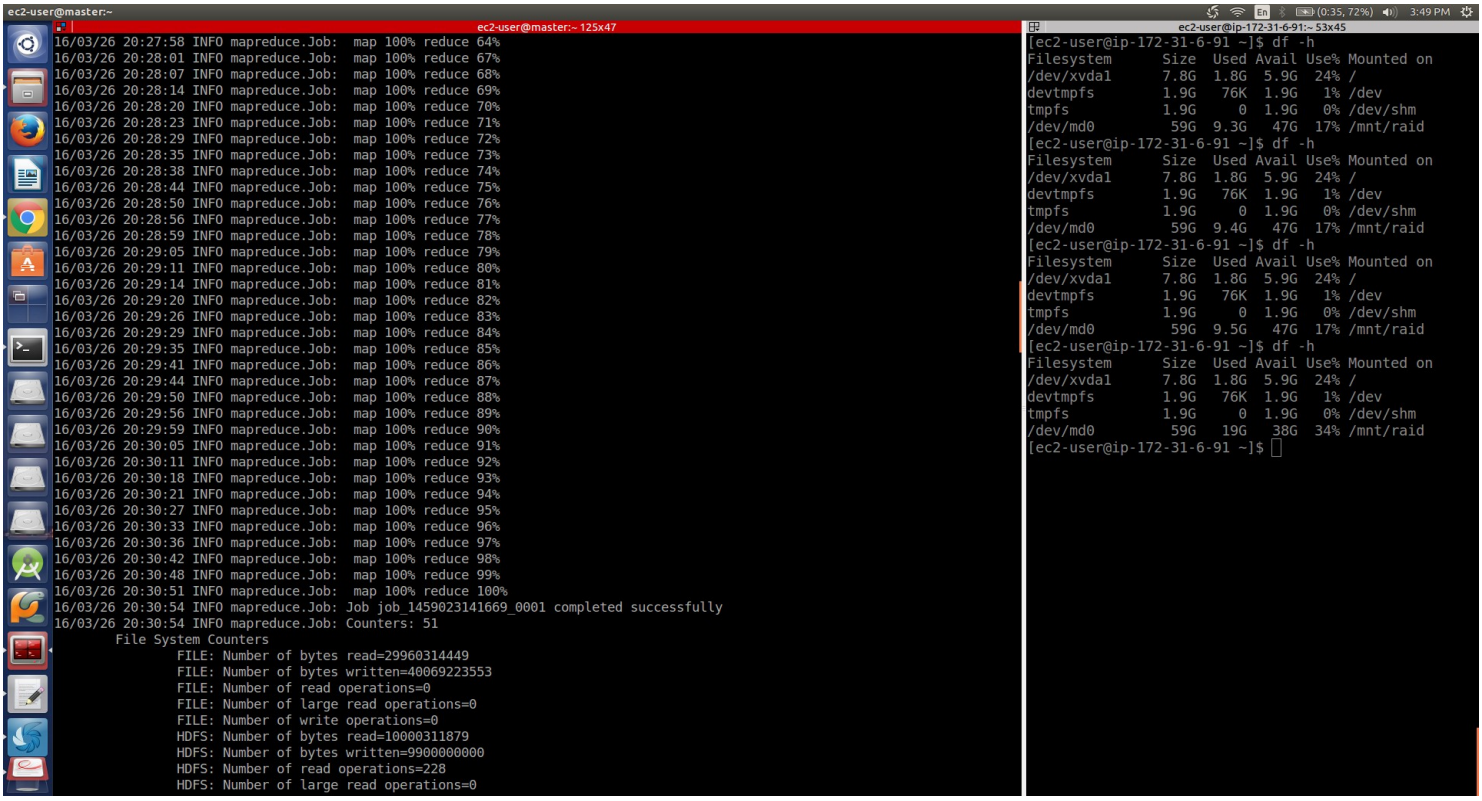


4. To run jar file on hadoop :

$ hadoop jar jar_file_name Class_name input_path output_path




ec2-user@master:~
ec2-user@master:~ 125x47
16/03/26 20:20:57 INFO mapreduce.Job: map 46% reduce 15%
16/03/26 20:20:59 INFO mapreduce.Job: map 47% reduce 15%
16/03/26 20:21:02 INFO mapreduce.Job: map 49% reduce 15%
16/03/26 20:21:05 INFO mapreduce.Job: map 50% reduce 15%
16/03/26 20:21:08 INFO mapreduce.Job: map 51% reduce 15%
16/03/26 20:21:11 INFO mapreduce.Job: map 52% reduce 15%
16/03/26 20:21:16 INFO mapreduce.Job: map 53% reduce 15%
16/03/26 20:21:19 INFO mapreduce.Job: map 54% reduce 15%
16/03/26 20:21:29 INFO mapreduce.Job: map 55% reduce 15%
16/03/26 20:21:30 INFO mapreduce.Job: map 56% reduce 15%
16/03/26 20:21:33 INFO mapreduce.Job: map 57% reduce 15%
16/03/26 20:21:36 INFO mapreduce.Job: map 58% reduce 15%
16/03/26 20:21:42 INFO mapreduce.Job: map 58% reduce 16%
16/03/26 20:21:46 INFO mapreduce.Job: map 59% reduce 16%
16/03/26 20:21:54 INFO mapreduce.Job: map 60% reduce 16%
16/03/26 20:22:01 INFO mapreduce.Job: map 61% reduce 16%
16/03/26 20:22:03 INFO mapreduce.Job: map 62% reduce 16%
16/03/26 20:22:05 INFO mapreduce.Job: map 63% reduce 16%
16/03/26 20:22:08 INFO mapreduce.Job: map 64% reduce 16%
16/03/26 20:22:09 INFO mapreduce.Job: map 64% reduce 17%
16/03/26 20:22:11 INFO mapreduce.Job: map 65% reduce 17%
16/03/26 20:22:12 INFO mapreduce.Job: map 65% reduce 18%
16/03/26 20:22:18 INFO mapreduce.Job: map 65% reduce 19%
16/03/26 20:22:21 INFO mapreduce.Job: map 65% reduce 20%
16/03/26 20:22:24 INFO mapreduce.Job: map 66% reduce 20%
16/03/26 20:22:27 INFO mapreduce.Job: map 67% reduce 20%
16/03/26 20:22:40 INFO mapreduce.Job: map 68% reduce 20%
16/03/26 20:22:43 INFO mapreduce.Job: map 69% reduce 20%
16/03/26 20:22:49 INFO mapreduce.Job: map 69% reduce 21%
16/03/26 20:22:55 INFO mapreduce.Job: map 69% reduce 22%
16/03/26 20:22:58 INFO mapreduce.Job: map 69% reduce 23%
16/03/26 20:23:10 INFO mapreduce.Job: map 70% reduce 23%
16/03/26 20:23:12 INFO mapreduce.Job: map 71% reduce 23%
16/03/26 20:23:13 INFO mapreduce.Job: map 72% reduce 23%
16/03/26 20:23:14 INFO mapreduce.Job: map 73% reduce 23%
16/03/26 20:23:17 INFO mapreduce.Job: map 74% reduce 23%
16/03/26 20:23:20 INFO mapreduce.Job: map 75% reduce 23%
16/03/26 20:23:23 INFO mapreduce.Job: map 76% reduce 23%
16/03/26 20:23:26 INFO mapreduce.Job: map 77% reduce 23%
16/03/26 20:23:33 INFO mapreduce.Job: map 78% reduce 23%
16/03/26 20:23:36 INFO mapreduce.Job: map 79% reduce 23%
16/03/26 20:23:44 INFO mapreduce.Job: map 80% reduce 23%
16/03/26 20:23:45 INFO mapreduce.Job: map 81% reduce 23%
16/03/26 20:23:47 INFO mapreduce.Job: map 81% reduce 24%
16/03/26 20:23:48 INFO mapreduce.Job: map 82% reduce 24%
16/03/26 20:23:54 INFO mapreduce.Job: map 83% reduce 24%
ec2-user@ip-172-31-6-91:~ 53x45
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/xvda1 7.8G 1.8G 5.9G 24% /
devtmpfs 1.9G 76K 1.9G 1% /dev
tmpfs 1.9G 0 1.9G 0% /dev/shm
/dev/md0 59G 9.3G 47G 17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/xvda1 7.8G 1.8G 5.9G 24% /
devtmpfs 1.9G 76K 1.9G 1% /dev
tmpfs 1.9G 0 1.9G 0% /dev/shm
/dev/md0 59G 9.4G 47G 17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/xvda1 7.8G 1.8G 5.9G 24% /
devtmpfs 1.9G 76K 1.9G 1% /dev
tmpfs 1.9G 0 1.9G 0% /dev/shm
/dev/md0 59G 9.5G 47G 17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$


ec2-user@master:~
ec2-user@master:~ 125x47
16/03/26 20:27:58 INFO mapreduce.Job: map 100% reduce 64%
16/03/26 20:28:01 INFO mapreduce.Job: map 100% reduce 67%
16/03/26 20:28:07 INFO mapreduce.Job: map 100% reduce 68%
16/03/26 20:28:14 INFO mapreduce.Job: map 100% reduce 69%
16/03/26 20:28:20 INFO mapreduce.Job: map 100% reduce 70%
16/03/26 20:28:23 INFO mapreduce.Job: map 100% reduce 71%
16/03/26 20:28:29 INFO mapreduce.Job: map 100% reduce 72%
16/03/26 20:28:35 INFO mapreduce.Job: map 100% reduce 73%
16/03/26 20:28:38 INFO mapreduce.Job: map 100% reduce 74%
16/03/26 20:28:44 INFO mapreduce.Job: map 100% reduce 75%
16/03/26 20:28:50 INFO mapreduce.Job: map 100% reduce 76%
16/03/26 20:28:56 INFO mapreduce.Job: map 100% reduce 77%
16/03/26 20:28:59 INFO mapreduce.Job: map 100% reduce 78%
16/03/26 20:29:05 INFO mapreduce.Job: map 100% reduce 79%
16/03/26 20:29:11 INFO mapreduce.Job: map 100% reduce 80%
16/03/26 20:29:14 INFO mapreduce.Job: map 100% reduce 81%
16/03/26 20:29:20 INFO mapreduce.Job: map 100% reduce 82%
16/03/26 20:29:26 INFO mapreduce.Job: map 100% reduce 83%
16/03/26 20:29:29 INFO mapreduce.Job: map 100% reduce 84%
16/03/26 20:29:35 INFO mapreduce.Job: map 100% reduce 85%
16/03/26 20:29:41 INFO mapreduce.Job: map 100% reduce 86%
16/03/26 20:29:44 INFO mapreduce.Job: map 100% reduce 87%
16/03/26 20:29:50 INFO mapreduce.Job: map 100% reduce 88%
16/03/26 20:29:56 INFO mapreduce.Job: map 100% reduce 89%
16/03/26 20:29:59 INFO mapreduce.Job: map 100% reduce 90%
16/03/26 20:30:05 INFO mapreduce.Job: map 100% reduce 91%
16/03/26 20:30:11 INFO mapreduce.Job: map 100% reduce 92%
16/03/26 20:30:18 INFO mapreduce.Job: map 100% reduce 93%
16/03/26 20:30:21 INFO mapreduce.Job: map 100% reduce 94%
16/03/26 20:30:27 INFO mapreduce.Job: map 100% reduce 95%
16/03/26 20:30:33 INFO mapreduce.Job: map 100% reduce 96%
16/03/26 20:30:36 INFO mapreduce.Job: map 100% reduce 97%
16/03/26 20:30:42 INFO mapreduce.Job: map 100% reduce 98%
16/03/26 20:30:48 INFO mapreduce.Job: map 100% reduce 99%
16/03/26 20:30:51 INFO mapreduce.Job: map 100% reduce 100%
16/03/26 20:30:54 INFO mapreduce.Job: Job job_1459023141669_0001 completed successfully
16/03/26 20:30:54 INFO mapreduce.Job: Counters: 51
File System Counters
FILE: Number of bytes read=29960314449
FILE: Number of bytes written=40069223553
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=10000311879
HDFS: Number of bytes written=9900000000
HDFS: Number of read operations=228
HDFS: Number of large read operations=0
ec2-user@ip-172-31-6-91:~ 53x45
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/xvda1 7.8G 1.8G 5.9G 24% /
devtmpfs 1.9G 76K 1.9G 1% /dev
tmpfs 1.9G 0 1.9G 0% /dev/shm
/dev/md0 59G 9.4G 47G 17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/xvda1 7.8G 1.8G 5.9G 24% /
devtmpfs 1.9G 76K 1.9G 1% /dev
tmpfs 1.9G 0 1.9G 0% /dev/shm
/dev/md0 59G 9.5G 47G 17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/xvda1 7.8G 1.8G 5.9G 24% /
devtmpfs 1.9G 76K 1.9G 1% /dev
tmpfs 1.9G 0 1.9G 0% /dev/shm
/dev/md0 59G 19G 38G 34% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$

```
ec2-user@master:~
ec2-user@master:~ 125x47
16/03/26 20:30:54 INFO mapreduce.Job: Job job_1459023141669_0001 completed successfully
16/03/26 20:30:54 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=29960314449
                FILE: Number of bytes written=40069223553
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=10000311879
                HDFS: Number of bytes written=9900000000
                HDFS: Number of read operations=228
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=4
                Launched map tasks=79
                Launched reduce tasks=1
                Data-local map tasks=44
                Rack-local map tasks=35
                Total time spent by all maps in occupied slots (ms)=6182550
                Total time spent by all reduces in occupied slots (ms)=758087
                Total time spent by all map tasks (ms)=6182550
                Total time spent by all reduce tasks (ms)=758087
                Total vcore-milliseconds taken by all map tasks=6182550
                Total vcore-milliseconds taken by all reduce tasks=758087
                Total megabyte-milliseconds taken by all map tasks=6330931200
                Total megabyte-milliseconds taken by all reduce tasks=776281088
        Map-Reduce Framework
                Map input records=100000000
                Map output records=100000000
                Map output bytes=9900000000
                Map output materialized bytes=10100000450
                Input split bytes=8775
                Combine input records=100000000
                Combine output records=100000000
                Reduce input groups=100000000
                Reduce shuffle bytes=10100000450
                Reduce input records=100000000
                Reduce output records=100000000
                Spilled Records=396636763
                Shuffled Maps =75
                Failed Shuffles=0
                Merged Map outputs=75
                GC time elapsed (ms)=24774
                CPU time spent (ms)=1512940
                Physical memory (bytes) snapshot=19661799424
                Virtual memory (bytes) snapshot=74878418944
```

```
ec2-user@ip-172-31-6-91:~ 53x45
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1      7.8G  1.8G  5.9G  24% /
devtmpfs        1.9G   76K  1.9G   1% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
/dev/md0         59G  9.3G   47G  17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1      7.8G  1.8G  5.9G  24% /
devtmpfs        1.9G   76K  1.9G   1% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
/dev/md0         59G  9.4G   47G  17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1      7.8G  1.8G  5.9G  24% /
devtmpfs        1.9G   76K  1.9G   1% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
/dev/md0         59G  9.5G   47G  17% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1      7.8G  1.8G  5.9G  24% /
devtmpfs        1.9G   76K  1.9G   1% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
/dev/md0         59G   19G   38G  34% /mnt/raid
[ec2-user@ip-172-31-6-91 ~]$
```

Output of first 10 and last 10 lines (10GB) :

```
[ec2-user@ip-172-31-13-167 raid]$ head -10 output
    "O!uve  0000000000000000000000000001228D4  77778888000022224444DDDDDDDDEEEE00000000CCCC7777DDDD
    PMd32=  00000000000000000000000000003440CC1  FFFFEEEE6666CCCCBBBB999933335555DDDDDDDD777788886666
    ^3CO],  0000000000000000000000000000158C5C5  5555AAAA9999EEEE888822229999CCCCDDDD6666555544442222
   !&S3/]]  00000000000000000000000000002145D78  8888BBBBDDDD1111CCCC55556666BBBB1111EEEEDDDD22229999
   !,=U#,9  00000000000000000000000000019072E3  33332222FFFFBBBB0000FFFFAAAA6666555533333DDDD3333CCCC
   !0f[ITd  0000000000000000000000000003CAAB4B  9999FFFF555533337777CCCC4444BBBB7777EEEEBBBBDDDD4444
   !f6Suy2  00000000000000000000000000003ABFD84  EEEE555555556666AAAA5555BBBBDDDD00001111666600000DDDD
   #%NIpq.  00000000000000000000000000003B36FB9  11110000333344441111666666666AAAAAAAA00001111CCCCEEEE
   #'^cl'~  0000000000000000000000000002EDC5C8  8888AAAA11114444FFFF77773333EEEE44440000FFFF99999999
   $"-'Q)]  00000000000000000000000000005F1265D  CCCC6666EEEE22220000DDDDAAAA88886666BBBB00006666AAAA
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$ tail -10 output
~~~uq2k#=U  00000000000000000000000002C06745  99991111DDDD222211110000FFFFEEEEFFFF33337777CCCC2222
~~~v/0&Qnm  00000000000000000000000004709701  CCCC88883333FFFF0000000000099991111FFFF777744446666
~~~yKOl:gE  0000000000000000000000000002048B4F  CCCC111144448888222266666BBBB888855557777EEEEBBBB0000
~~~yK^H.il  00000000000000000000000000463D004  44440000FFFF3333999944447777DDDDFFFFAAAA11118888DDDD
~~~yL;C'XE  00000000000000000000000005B0D211  2222EEEE3333000022221111CCCCFFFF555577774444BBBB6666
~~~zbA_ Tt  0000000000000000000000000007F9F4F  BBBBCCCC666655559999FFFF8888AAAA11116666AAAABBBB0000
~~~zeO^FEg  00000000000000000000000001E06130  4444CCCCBBBB999922228888555588888CCCCFFFF000011111111
~~~}GxjWHI  00000000000000000000000000CA1345  777711118888AAAAAAAA22221111BBBB00002222BBBBCCCC2222
~~~}P;]g0g  0000000000000000000000000040DA3E4  4444FFFF444466663333EEEE88888888DDDDEEEE44442222DDDD
~~~}kU|K<p  00000000000000000000000005E4A0AA  0000666655551111BBBB88889999AAAA55550000333355557777
[ec2-user@ip-172-31-13-167 raid]$
```

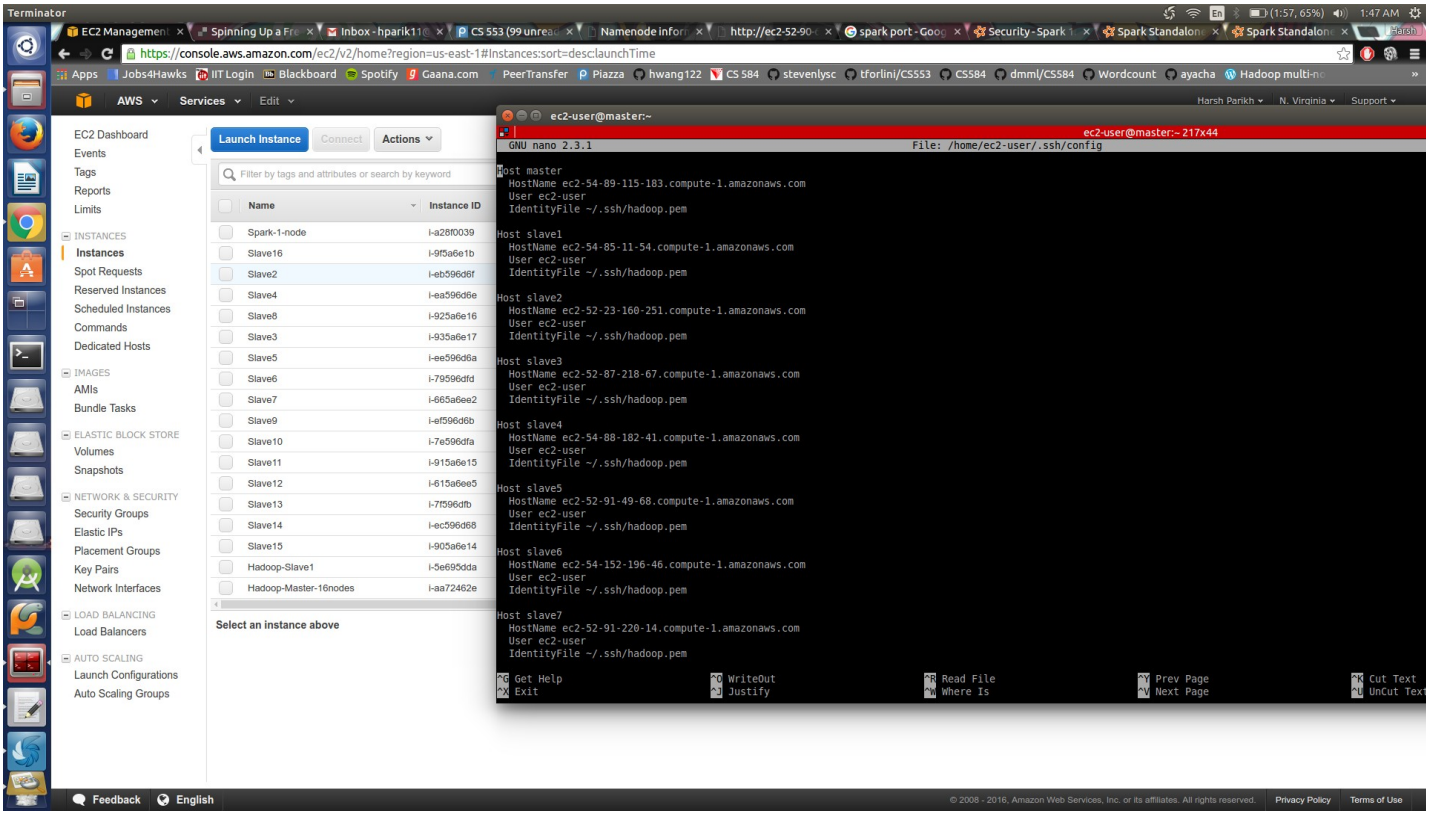
Valsort on 10 GB output:

```
[ec2-user@master 64]$ ./valsort ../output_10gb.txt
Records: 100000000
[ec2-user@master 64]$ ./valsort ../output_10gb.txt
Records: 100000000
Checksum: 2faef2a1dfa8909
Duplicate keys: 0
SUCCESS - all records are in order
[ec2-user@master 64]$
```

Hadoop 16 node Configuration:

```
ec2-user@master:~
ec2-user@master:~ 185x54
  GNU nano 2.3.1                    File: /etc/hosts

127.0.0.1   localhost localhost.localdomain
172.31.8.164 master
172.31.5.221 slave1
172.31.15.113 slave2
172.31.3.117 slave3
172.31.10.9 slave4
172.31.13.36 slave5
172.31.8.230 slave6
172.31.3.180 slave7
172.31.12.138 slave8
172.31.2.11 slave9
172.31.5.43 slave10
172.31.6.60 slave11
172.31.0.110 slave12
172.31.6.192 slave13
172.31.12.199 slave14
172.31.3.75 slave15
172.31.13.204 slave16

^G Get Help   ^O WriteOut   ^R Read File   ^Y Prev Page   ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page   ^U UnCut Text ^T To Spell
```

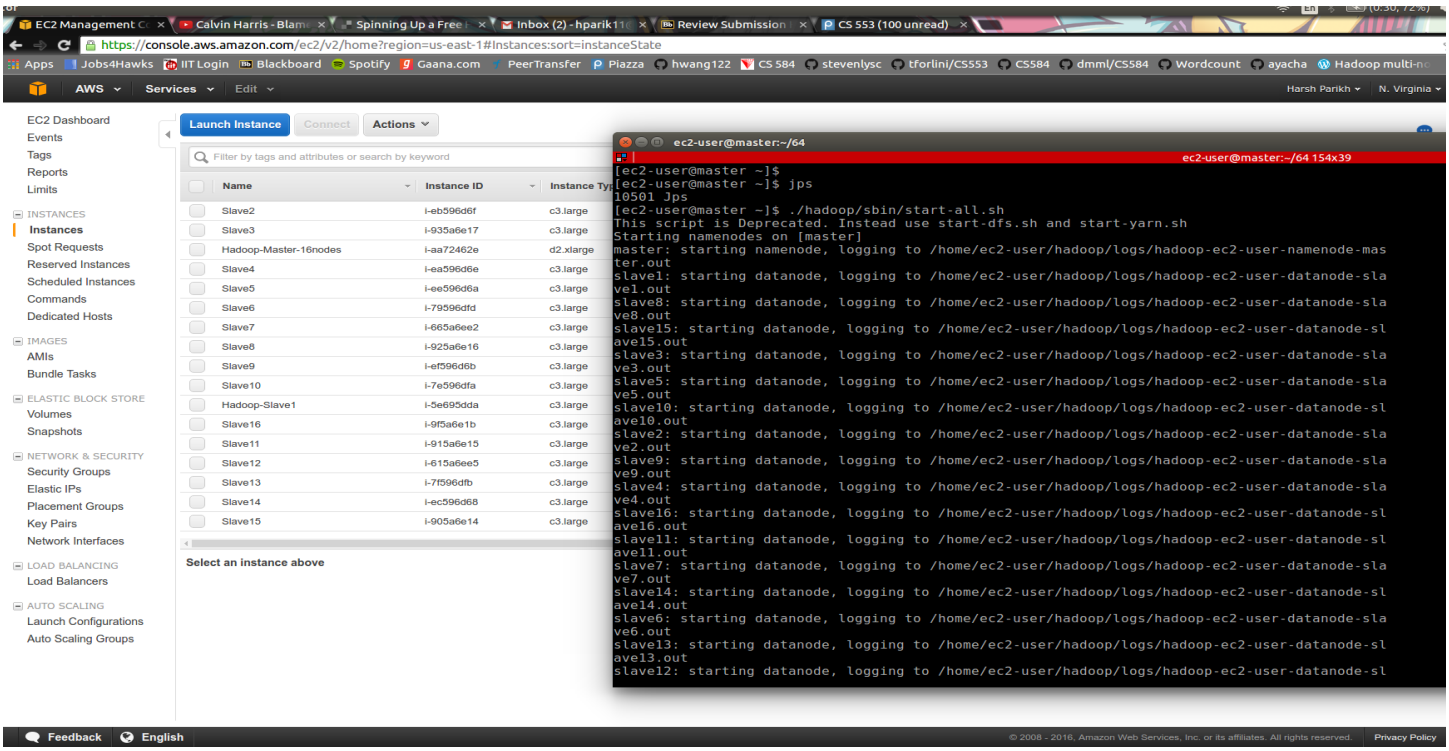
Config file (only on Master):

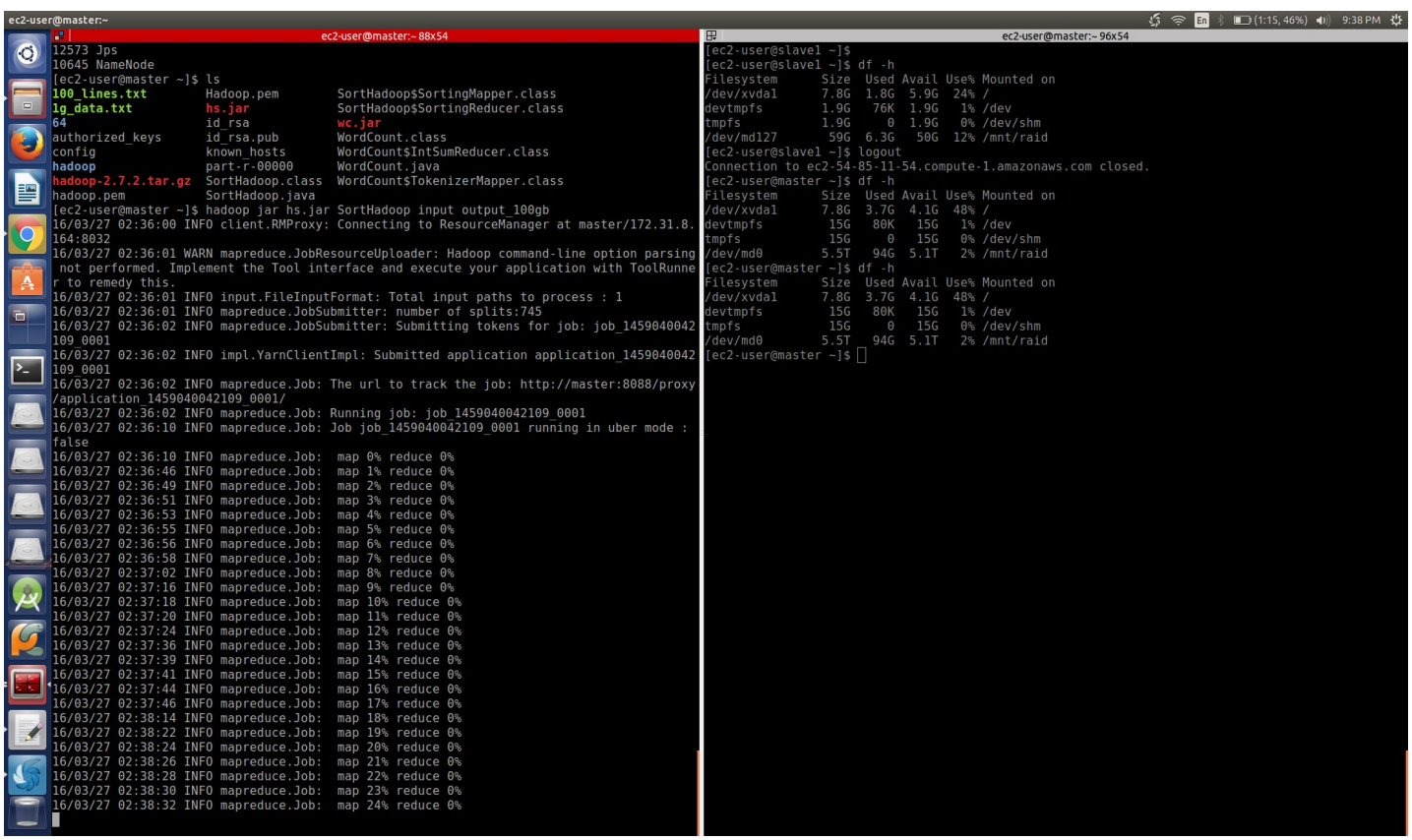


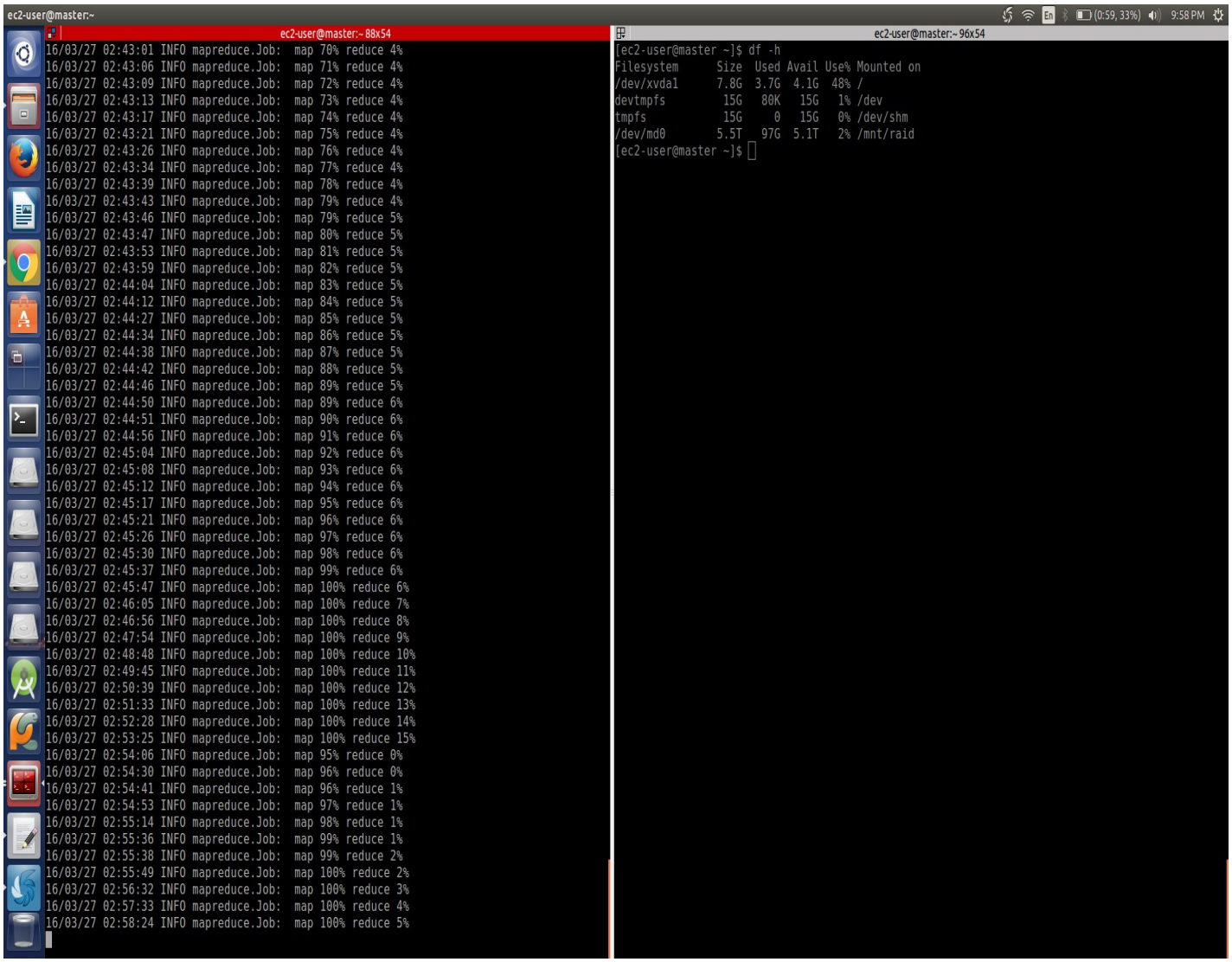
Starting of all nodes:

```
starting yarn daemons
starting resourcemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-resourcemanager
-master.out
slave8: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave8.out
slave1: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave1.out
slave2: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave2.out
slave14: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanage
r-slave14.out
slave6: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave6.out
slave3: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave3.out
slave12: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanage
r-slave12.out
slave13: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanage
r-slave13.out
slave7: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave7.out
slave15: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanage
r-slave15.out
slave16: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanage
r-slave16.out
slave10: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanage
r-slave10.out
slave4: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave4.out
slave5: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave5.out
slave11: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanage
r-slave11.out
slave9: starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager
-slave9.out
[ec2-user@master ~]$ jps
10963 ResourceManager
10645 NameNode
11253 Jps
[ec2-user@master ~]$ ./hadoop/sbin/yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/ec2-user/hadoop/logs/yarn-ec2-user-nodemanager-master.
out
[ec2-user@master ~]$ jps
11425 Jps
11295 NodeManager
10963 ResourceManager
10645 NameNode
```

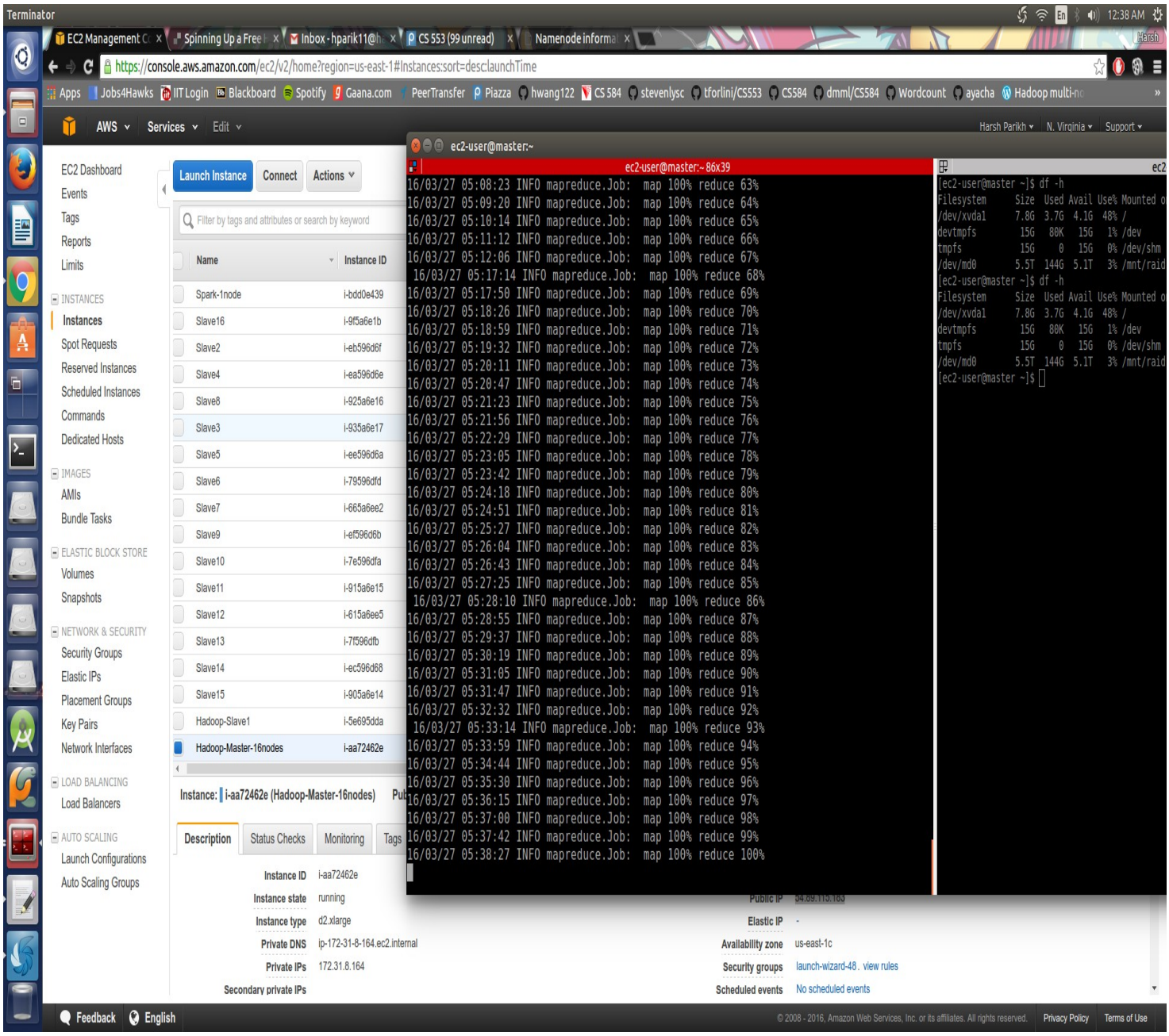
Starting of all 17 Nodes: -





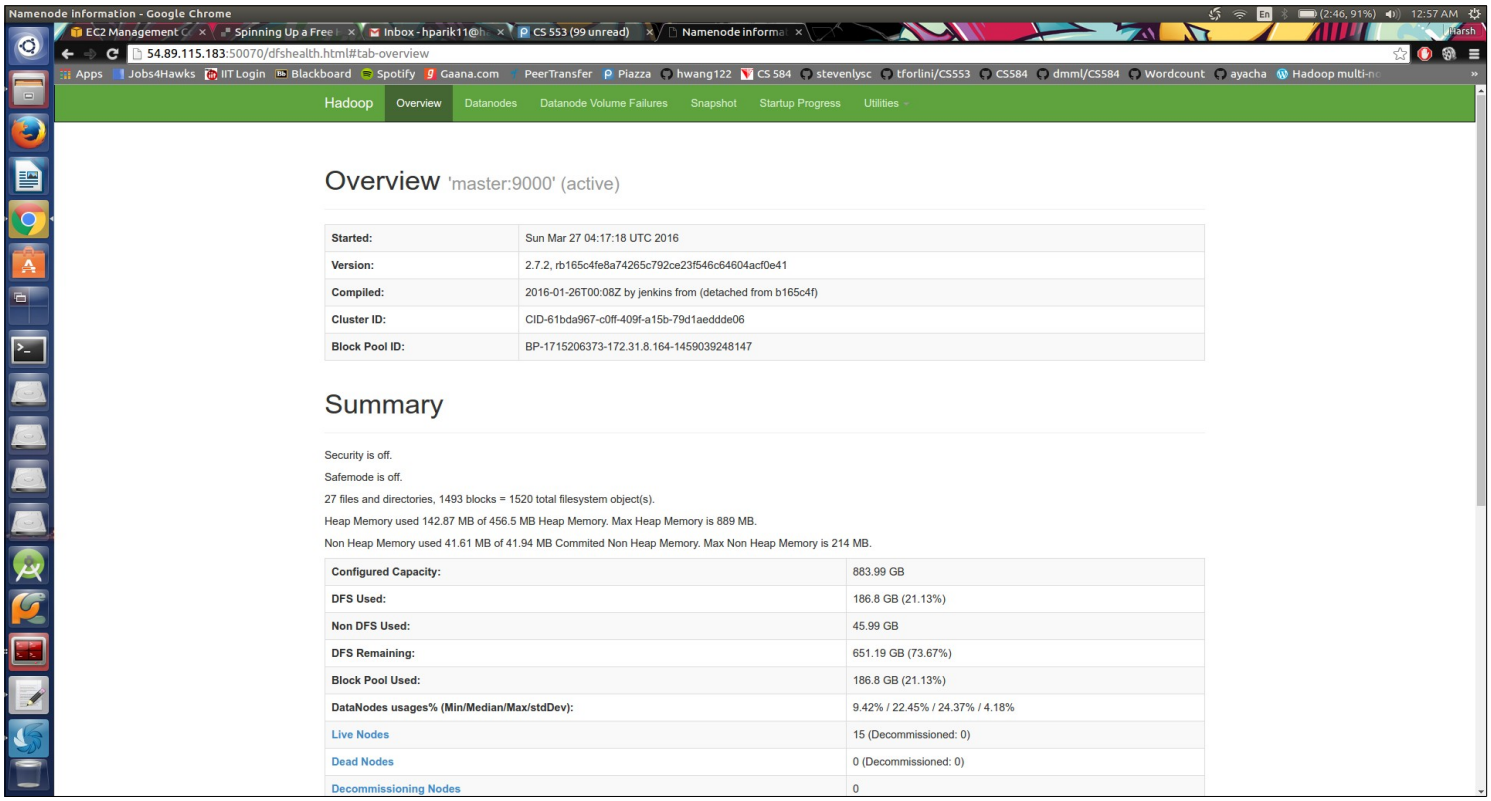


I was getting this error like after Map 100% Reduce 15% , the cycle starts again. It will start Map 100% and Reduce 0%. This is because I haven't put any mapper and reducers in file. So that my scheduler was not working perfectly. After this, I have changed **Mapred-site.xml** file, put number of mappers and reducers in that and later I got the output and it was working smoothly and swiftly compare to previous one. So in next screen shot is after changes.

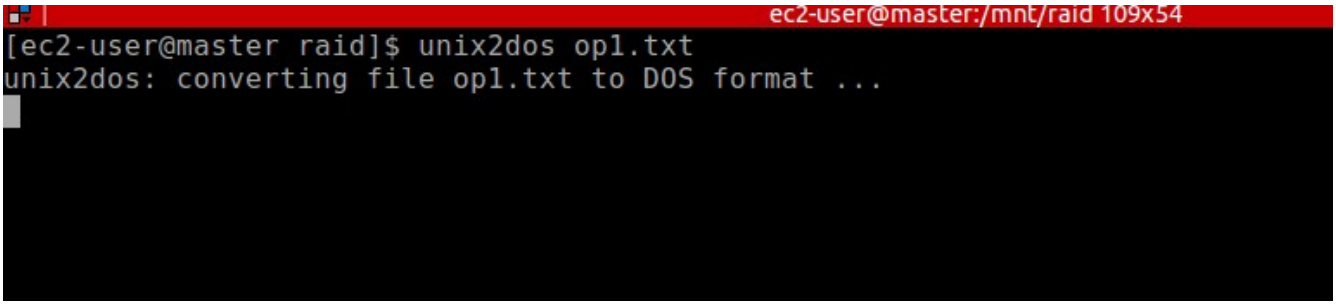


So, Total time elapsed = 70 min 54 sec.

Hadoop Web Interface:



To convert Text file into Dos Format



Valsort of 100GB Output's Screen shot:

```
[ec2-user@master 64]$ ./valsort /mnt/raid/op2.txt
Records: 499987291
Checksum: ee683173c0538cf
Duplicate keys: 0
SUCCESS - all records are in order
[ec2-user@master 64]$
```

After Completion, stopping of All Name and Data Nodes:

```
[ec2-user@master ~]$ ./hadoop/sbin/stop-all.sh
This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh
Stopping namenodes on [master]
master: stopping namenode
slave1: stopping datanode
slave6: stopping datanode
slave15: stopping datanode
slave16: stopping datanode
slave4: stopping datanode
slave2: no datanode to stop
slave9: stopping datanode
slave7: stopping datanode
slave14: stopping datanode
slave3: stopping datanode
slave12: stopping datanode
slave10: stopping datanode
slave13: stopping datanode
slave11: stopping datanode
slave5: stopping datanode
slave8: stopping datanode
Stopping secondary namenodes [0.0.0.0]
```

Valsort of 1 GB 16 node:

```
[ec2-user@master 64]$ ./valsort ../op2.txt
^LRecords: 5001286
Checksum: 262744736bab06
Duplicate keys: 0
SUCCESS - all records are in order
[ec2-user@master 64]$
```

Stopping all Node manager and Resource Manager:

```
stopping yarn daemons
stopping resourcemanager
slave1: stopping nodemanager
slave2: stopping nodemanager
slave3: stopping nodemanager
slave4: stopping nodemanager
slave5: stopping nodemanager
slave6: stopping nodemanager
slave10: stopping nodemanager
slave8: stopping nodemanager
slave16: stopping nodemanager
slave12: stopping nodemanager
slave11: stopping nodemanager
slave14: stopping nodemanager
slave13: stopping nodemanager
slave9: stopping nodemanager
slave15: stopping nodemanager
slave7: stopping nodemanager
slave1: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave2: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave3: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave4: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave5: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave6: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave10: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave8: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave16: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave12: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave11: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave14: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave13: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave9: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave15: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
slave7: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
no proxyserver to stop
```

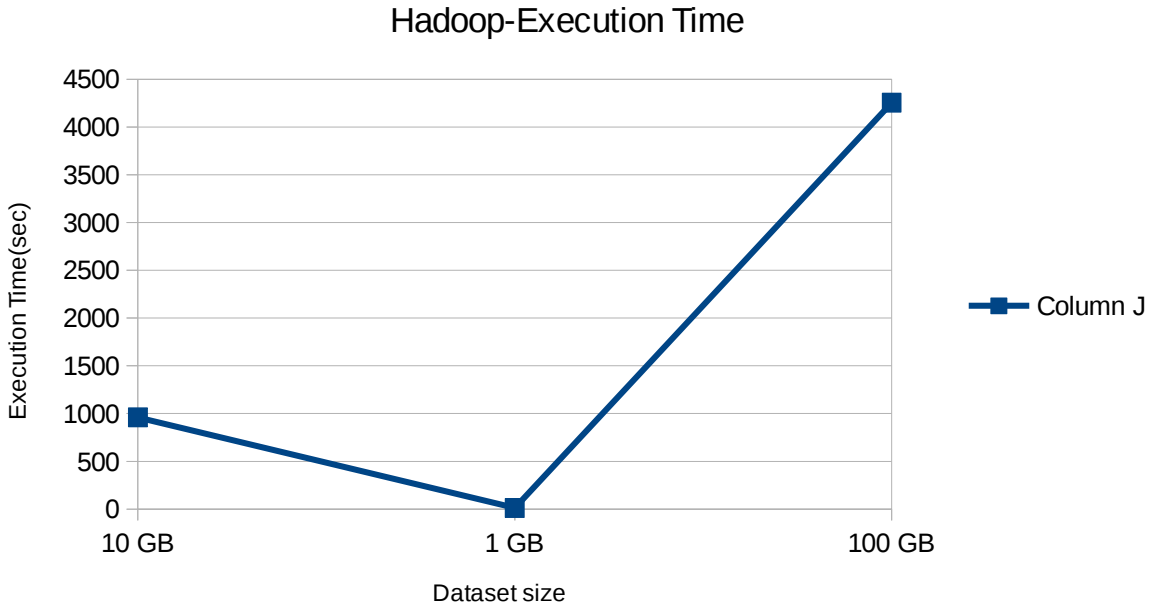
Hadoop Performance Evaluation:

Nodes	Data Set	Execution Time	Throughput
1	10 GB	966 Seconds	10.41 MB/sec
16	1 GB	12 Seconds	83.33 MB/sec
16	100 GB	4254 Seconds	23.50 MB/sec

Output of first 10 and last 10 lines (1 GB) :

```
[ec2-user@ip-172-31-62-211 ~]$ nano 1GB_Sort_AWS.py
[ec2-user@ip-172-31-62-211 ~]$ python 1GB_Sort_AWS.py
10000000
************************************************** First 10 Lines **************************************************


    "O!uve  00000000000000000000000001228D4  77778888000022224444DDDDDDDDEEEE00000000CCCC7777DDDD

   ,K4a-:v  00000000000000000000000001B8132  5555EEEE888899994444FFFF1111CCCCEEEE1111EEEE6666FFFF

   .FuD\}u  00000000000000000000000000797631  5555DDDDBBBB00007777222211112222444 4DDDDDDDD99996666

   ;5YThct  00000000000000000000000007D3DF5  2222AAAACCCCFFFFAAAA44445555EEEE44442222DDDD99992222

   =2G^9{-  00000000000000000000000000809EE  5555DDDD1111CCCC9999BBBB0000BBBBCCCCFFFFCCCC44443333

   N}M9?sP  00000000000000000000000000429597  5555FFFF00007777555599991111CCCC66669999AAAAEEEE8888

   P0X?Rs&  00000000000000000000000000041162E  888833339999FFFF1111CCCC8888CCCC9999EEEEDDDD00003333

   [Xq\\$%  00000000000000000000000000097A5F0  66666666EEEEDDDD7777FFFF00005555FFFFFFFF888855551111

   rAmQg4v  00000000000000000000000000008180D3  BBBB111111119999FFFFFFFFFFFFFFFF4444BBBB88884444CCCC

  !&))Jf3;  00000000000000000000000000004602E1  4444FFFFCCCC88888888CCCCFFFFCCCCEEEE5555666666666666

************************************************** Last 10 Lines **************************************************


~~~%A NB_t  00000000000000000000000000003DE5EC  FFFF7777EEEEBBBB4444EEEEEEEE33339999DDDD999900005555

~~~-c-CQ(>  00000000000000000000000000832611  9999FFFF111177773333777700001111444444440000BBBB6666

~~~8Y}Fql*  00000000000000000000000000742A4C  BBBB1111CCCCEEEE888800000000777733333333DDDD22225555

~~~>Dd=QT]  00000000000000000000000000674C0F  9999DDDD000055556666CCCC22220000FFFFEEEEDDDDFFFF0000

~~~]JA(}j$  00000000000000000000000000060BCBE  111122223333444455559999AAAABBBB9999FFFFDDDDBBBB3333

~~~]Zp.#/+  00000000000000000000000000003B9A5A  CCCC8888EEEEAAAAEEEE33333333377770000FFFFCCCC66667777

~~~_jQepix  00000000000000000000000000011E5D4  1111999911115555BBBB111100002222EEEE6666BBBB7777DDDD

~~~nt=ZH[N  00000000000000000000000000332A13  44441111BBBBBBBB33337777FFFF4444555555553333 0000CCCC

~~~s/Pq,-E  00000000000000000000000000006BE930  2222DDDDDDDD77771111EEEECCCC7777BBBB4444888811111111

~~~zbA_ Tt  00000000000000000000000000007F9F4F  BBBBCCCC666655559999FFFF8888AAAA11116666AAAABBBB0000

Start Time: 1458955468.97
End Time: 1458955519.51
Total Time Elapsed:  50.5440180302
[ec2-user@ip-172-31-62-211 ~]$
```

Total elapsed time = 12 sec.



Question-Answers:

1(a). What is Master - node ?

• The Master-Node is the center of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. It does scheduling and divide the data into all slaves equally.

1(b) What is Data-node:

• A Data Node stores data in the Hadoop File System. A functional file system has more than one DataNode, with data replicated across them. In most of cases, it replicates data 3 times.

2. Why we need to set unique port in shared environment service? What Errors or side-effects of using same port?

Each service uses different port to uniquely identified. So it won't throw errors but it will not able to perform either of the service that has the same port numbers. So for those ports, it may conflict with them and those services would not work unless change the port numbers.

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

In **hadoop/etc/hadoop/** there is a file called mapred-site.xml . We can add the value as number of mappers and reducers for **mapreduce.job.maps** and **mapreduce.job.reduces**.

4. Changes made while scaling from 1 node to 16 Node:

We need to mention all the 16 nodes private ip in /etc/hosts file and public dns in ~/.ssh/config file. Add slaves public dns in known_host file resides in same location as config file. Change as mentioned in configuration file in etc/hadoop/mapred-site.xml. And we need to change hadoop/etc/hadoop/slaves file.

Sorting Application on Spark

Spark Configuration:

To install, one has to write the following commands

#To Download spark
wget http://mirrors.sonic.net/apache/spark/spark-1.6.1/spark-1.6.1-bin-hadoop2.6.tgz

#To extract spark

tar -xvzf spark-1.6.1-bin-hadoop2.6.tgz

#To update the All application
sudo apt-get update

#To Install java
sudo apt-get install openjdk-7-jdk

Export Java_Home
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64/jre

To Install Scala
wget http://www.scala-lang.org/files/archive/scala-2.10.4.tgz

sudo mkdir /usr/local/src/scala
sudo tar xvf scala-2.10.4.tgz -C /usr/local/src/scala/

#To Export Scala Path

export SCALA_HOME=/usr/local/src/scala/scala-2.10.4
export PATH=$SCALA_HOME/bin:$PATH

After Installation of Spark,

To invoke 1 node cluster run this following command:-

./spark-ec2 -k spark -i spark.pem -s 1 --instance-type=t2.micro launch Spark-Cluster

```
ubuntu@ip-172-31-45-14:~/spark-1.6.1-bin-hadoop2.6/ec2$ ./spark-ec2 -k spark -i
spark.pem -s 1 --instance-type=d2.xlarge launch Spark-Cluster
Downloading external libraries that spark-ec2 needs from PyPI to /home/ubuntu/sp
ark-1.6.1-bin-hadoop2.6/ec2/lib...
This should be a one-time operation.
 - Downloading boto...
 - Finished downloading boto.
Setting up security groups...
Searching for existing cluster Spark-Cluster in region us-east-1...
Spark AMI: ami-35b1885c
Launching instances...
Launched 1 slave in us-east-1c, regid = r-3cb5c5ee
Launched master in us-east-1c, regid = r-52b6c680
Waiting for AWS to propagate instance metadata...
Waiting for cluster to enter 'ssh-ready' state.............

Warning: SSH connection error. (This could be temporary.)
Host: ec2-52-23-161-128.compute-1.amazonaws.com
SSH return code: 255
SSH output: ssh: connect to host ec2-52-23-161-128.compute-1.amazonaws.com port
22: Connection refused

.^[[C^Z
[1]+  Stopped                 ./spark-ec2 -k spark -i spark.pem -s 1 --instance-
type=d2.xlarge launch Spark-Cluster
ubuntu@ip-172-31-45-14:~/spark-1.6.1-bin-hadoop2.6/ec2$ ./spark-ec2 -k spark -i
spark.pem -s 1 --instance-type=d2.xlarge launch Spark-Cluster --resume
Searching for existing cluster Spark-Cluster in region us-east-1...
Found 1 master, 1 slave.
Waiting for cluster to enter 'ssh-ready' state.

Warning: SSH connection error. (This could be temporary.)
Host: ec2-52-23-161-128.compute-1.amazonaws.com
SSH return code: 255
SSH output: ssh: connect to host ec2-52-23-161-128.compute-1.amazonaws.com port
22: Connection refused
```

If you get Connection refused error you have to run this command:

./spark-ec2 -k spark -i spark.pem -s 1 --instance-type=t2.micro launch Spark-Cluster --resume

Installation Process(1 node 10 GB):-

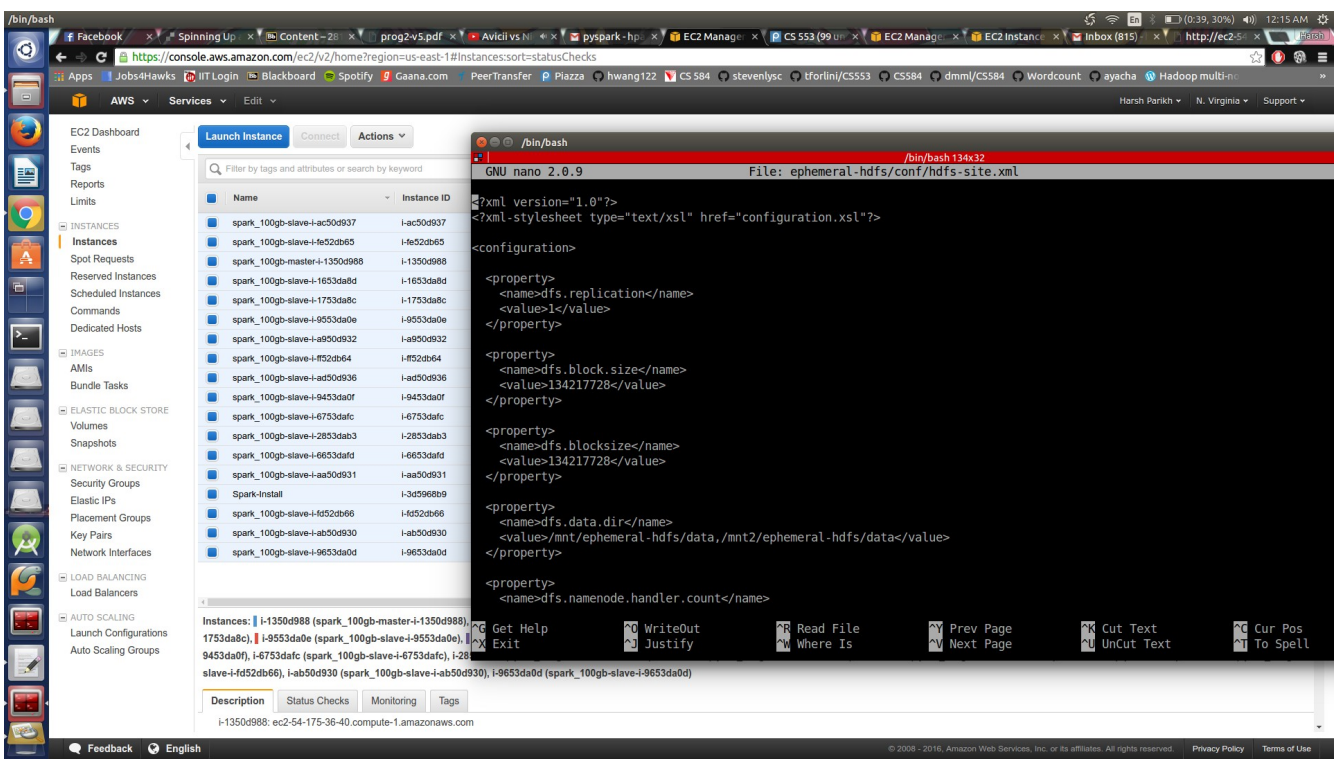
```
Cluster is now in 'ssh-ready' state. Waited 111 seconds.
Generating cluster's SSH key on master...
Warning: Permanently added 'ec2-52-23-161-128.compute-1.amazonaws.com,172.31.3.9
7' (ECDSA) to the list of known hosts.
Connection to ec2-52-23-161-128.compute-1.amazonaws.com closed.
Warning: Permanently added 'ec2-52-23-161-128.compute-1.amazonaws.com,172.31.3.9
7' (ECDSA) to the list of known hosts.
Transferring cluster's SSH key to slaves...
ec2-52-87-185-138.compute-1.amazonaws.com
Warning: Permanently added 'ec2-52-87-185-138.compute-1.amazonaws.com,172.31.9.2
21' (ECDSA) to the list of known hosts.
Cloning spark-ec2 scripts from https://github.com/amplab/spark-ec2/tree/branch-1
.5 on master...
Warning: Permanently added 'ec2-52-23-161-128.compute-1.amazonaws.com,172.31.3.9
7' (ECDSA) to the list of known hosts.
Cloning into 'spark-ec2'...
remote: Counting objects: 2069, done.
remote: Total 2069 (delta 0), reused 0 (delta 0), pack-reused 2069
Receiving objects: 100% (2069/2069), 355.93 KiB, done.
Resolving deltas: 100% (787/787), done.
Connection to ec2-52-23-161-128.compute-1.amazonaws.com closed.
Deploying files to master...
Warning: Permanently added 'ec2-52-23-161-128.compute-1.amazonaws.com,172.31.3.9
7' (ECDSA) to the list of known hosts.
sending incremental file list
root/spark-ec2/ec2-variables.sh

sent 1,727 bytes  received 40 bytes  3,534.00 bytes/sec
total size is 1,576  speedup is 0.89
Running setup on master...
Warning: Permanently added 'ec2-52-23-161-128.compute-1.amazonaws.com,172.31.3.9
7' (ECDSA) to the list of known hosts.
Connection to ec2-52-23-161-128.compute-1.amazonaws.com closed.
Warning: Permanently added 'ec2-52-23-161-128.compute-1.amazonaws.com,172.31.3.9
7' (ECDSA) to the list of known hosts.
Setting up Spark on ip-172-31-3-97.ec2.internal...
Setting executable permissions on scripts...
RSYNC'ing /root/spark-ec2 to other cluster nodes...
ec2-52-87-185-138.compute-1.amazonaws.com
Warning: Permanently added 'ec2-52-87-185-138.compute-1.amazonaws.com,172.31.9.2
21' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-52-87-185-138.compute-1.amazonaws.com,172.31.9.2
21' (ECDSA) to the list of known hosts.
id_rsa                                    100% 1679     1.6KB/s   00:00
[timing] rsync /root/spark-ec2:  00h 00m 00s
Running setup-slave on all cluster nodes to mount filesystems, etc...
[1] 08:39:01 [SUCCESS] ec2-52-87-185-138.compute-1.amazonaws.com
checking/fixing resolution of hostname
Setting up slave on ip-172-31-9-221.ec2.internal... of type d2.xlarge
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 0.581249 s, 1.8 GB/s
mkswap: /mnt/swap: warning: don't erase bootbits sectors
        on whole disk. Use -f to force.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 30531249 (29M) [application/x-compressed]
Saving to: 'scala-2.10.3.tgz'

100%[===================================>] 30,531,249  2.24MB/s   in 16s

2016-03-20 08:39:34 (1.80 MB/s) - 'scala-2.10.3.tgz' saved [30531249/30531249]

[timing] scala init:  00h 00m 17s
Initializing spark
--2016-03-20 08:39:35--  http://s3.amazonaws.com/spark-related-packages/spark-1.
6.1-bin-hadoop1.tgz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.80.163
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.80.163|:80... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 243673720 (232M) [application/x-compressed]
Saving to: 'spark-1.6.1-bin-hadoop1.tgz'

100%[===================================>] 243,673,720 58.3MB/s   in 3.9s

2016-03-20 08:39:39 (59.9 MB/s) - 'spark-1.6.1-bin-hadoop1.tgz' saved [243673720
/243673720]

Unpacking Spark
[timing] spark init:  00h 00m 06s
Initializing ephemeral-hdfs
--2016-03-20 08:39:41--  http://s3.amazonaws.com/spark-related-packages/hadoop-1
.0.4.tar.gz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.97.235
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.97.235|:80... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 62793050 (60M) [application/x-gzip]
Saving to: 'hadoop-1.0.4.tar.gz'

100%[===================================>] 62,793,050  65.8MB/s   in 0.9s

2016-03-20 08:39:42 (65.8 MB/s) - 'hadoop-1.0.4.tar.gz' saved [62793050/62793050
]

Unpacking Hadoop
RSYNC'ing /root/ephemeral-hdfs to slaves...
ec2-52-87-185-138.compute-1.amazonaws.com
Warning: Permanently added 'ec2-52-87-185-138.compute-1.amazonaws.com,172.31.9.2
21' (ECDSA) to the list of known hosts.
[timing] ephemeral-hdfs init:  00h 00m 09s
Initializing persistent-hdfs
--2016-03-20 08:39:50--  http://s3.amazonaws.com/spark-related-packages/hadoop-1
.0.4.tar.gz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.9.72
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.9.72|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 62793050 (60M) [application/x-gzip]
Saving to: 'hadoop-1.0.4.tar.gz'

100%[===================================>] 62,793,050  47.4MB/s   in 1.3s
```

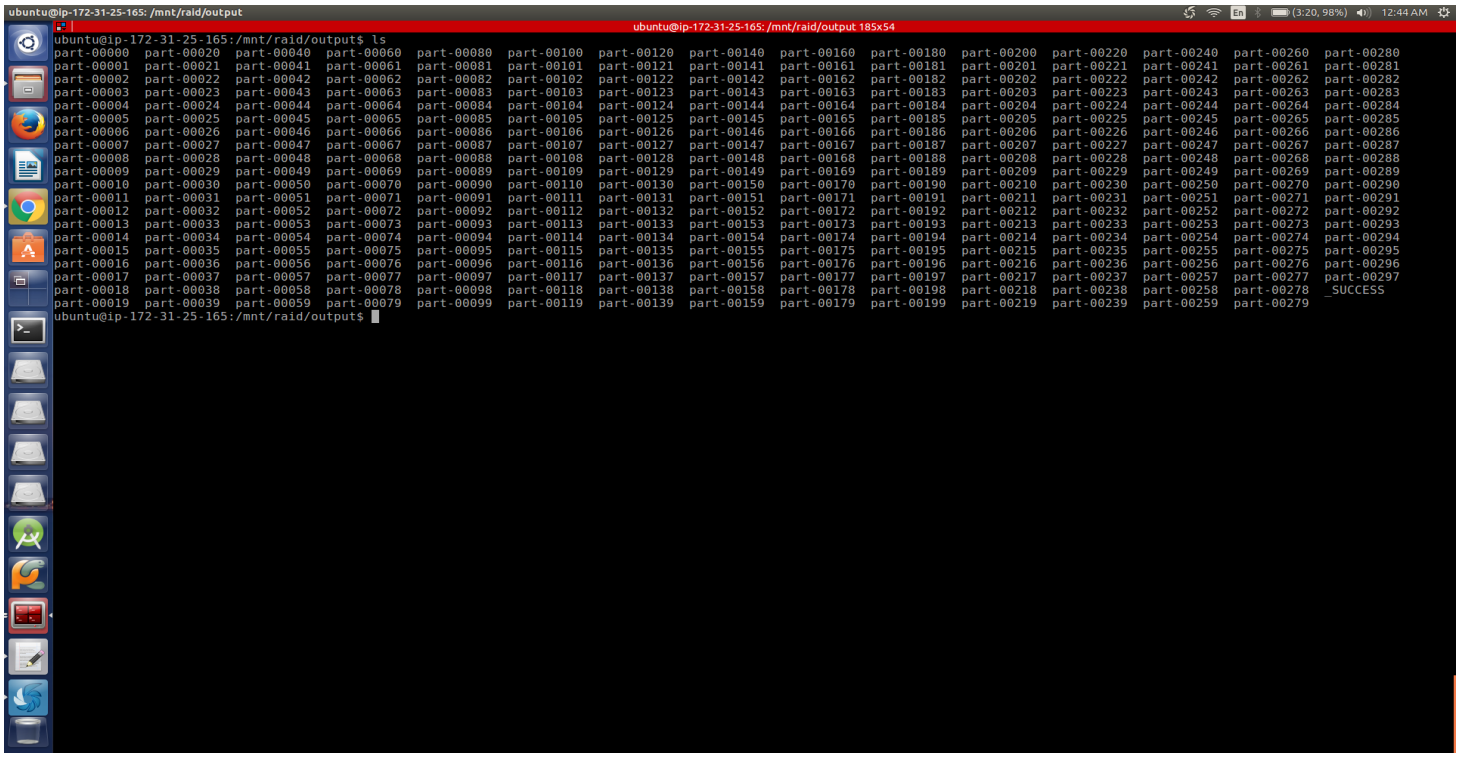
```
ec2-52-87-185-138.compute-1.amazonaws.com
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-52-87-185-138.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmetad:                              [FAILED]
Starting GANGLIA gmetad:                                   [  OK  ]
Stopping httpd:                                            [FAILED]
Starting httpd:                                            [  OK  ]
[timing] ganglia setup:  00h 00m 02s
Connection to ec2-52-23-161-128.compute-1.amazonaws.com closed.
Spark standalone cluster started at http://ec2-52-23-161-128.compute-1.amazonaws
.com:8080
Ganglia started at http://ec2-52-23-161-128.compute-1.amazonaws.com:5080/ganglia
Done!
ubuntu@ip-172-31-45-14:~/spark-1.6.1-bin-hadoop2.6/ec2$ ls
deploy.generic  lib  README  spark-ec2  spark_ec2.py  spark.pem
ubuntu@ip-172-31-45-14:~/spark-1.6.1-bin-hadoop2.6/ec2$ cd
ubuntu@ip-172-31-45-14:~$ ls
sbt-0.13.5.deb    spark-1.6.1-bin-hadoop2.6      spark.pem
scala-2.10.4.tgz  spark-1.6.1-bin-hadoop2.6.tgz
ubuntu@ip-172-31-45-14:~$
```

hdfs-site.xml File path to change:

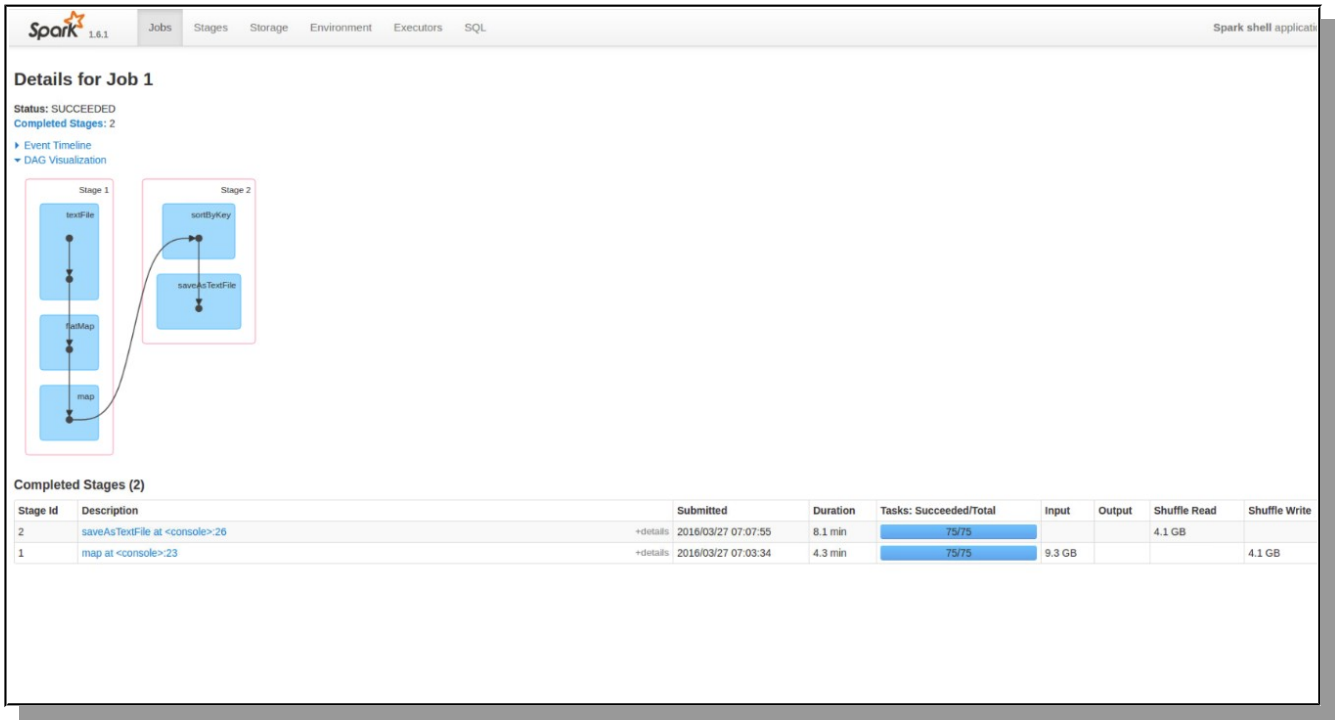
Make replication of hdfs as 1 instead of 3.



Output of 10 GB (297 files):



Flow of Program:



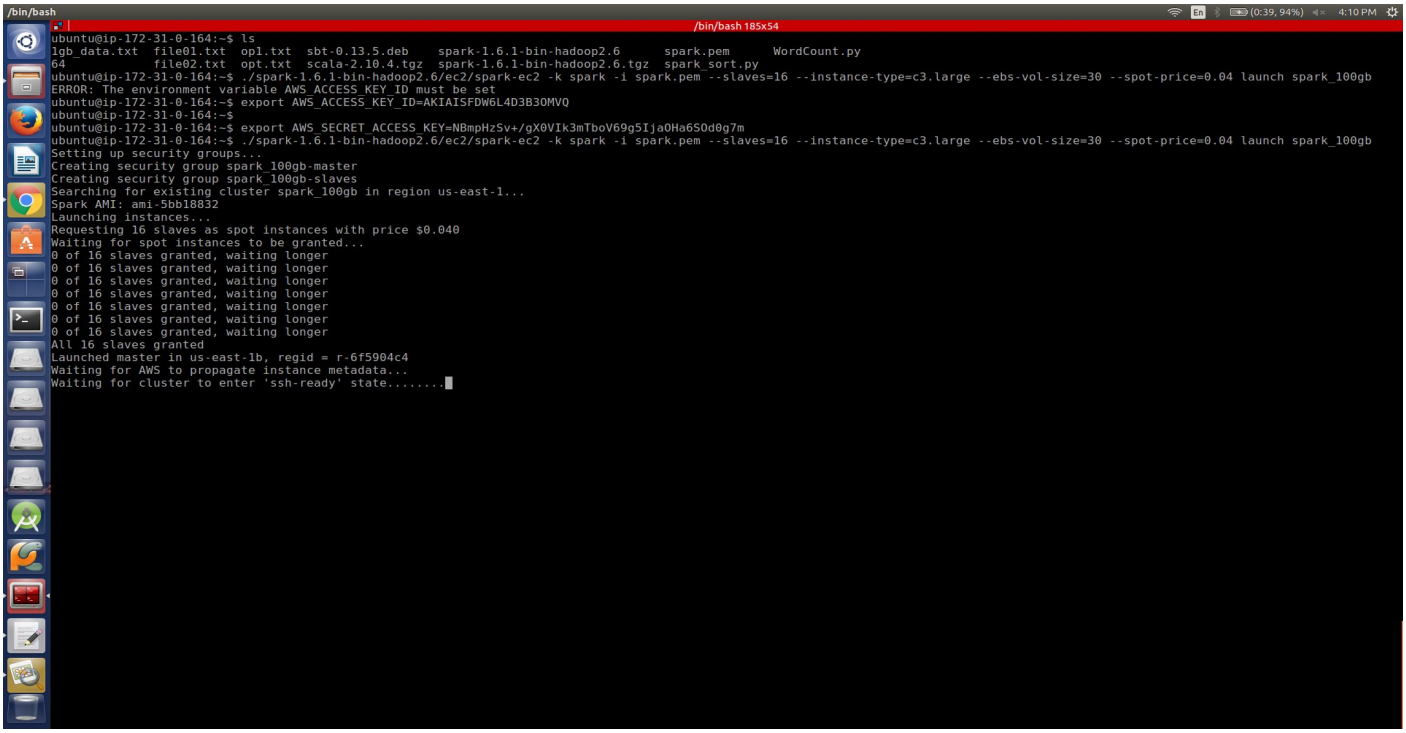
Output of 10 GB (1 Nodes)(First and Last file out of 297 files):

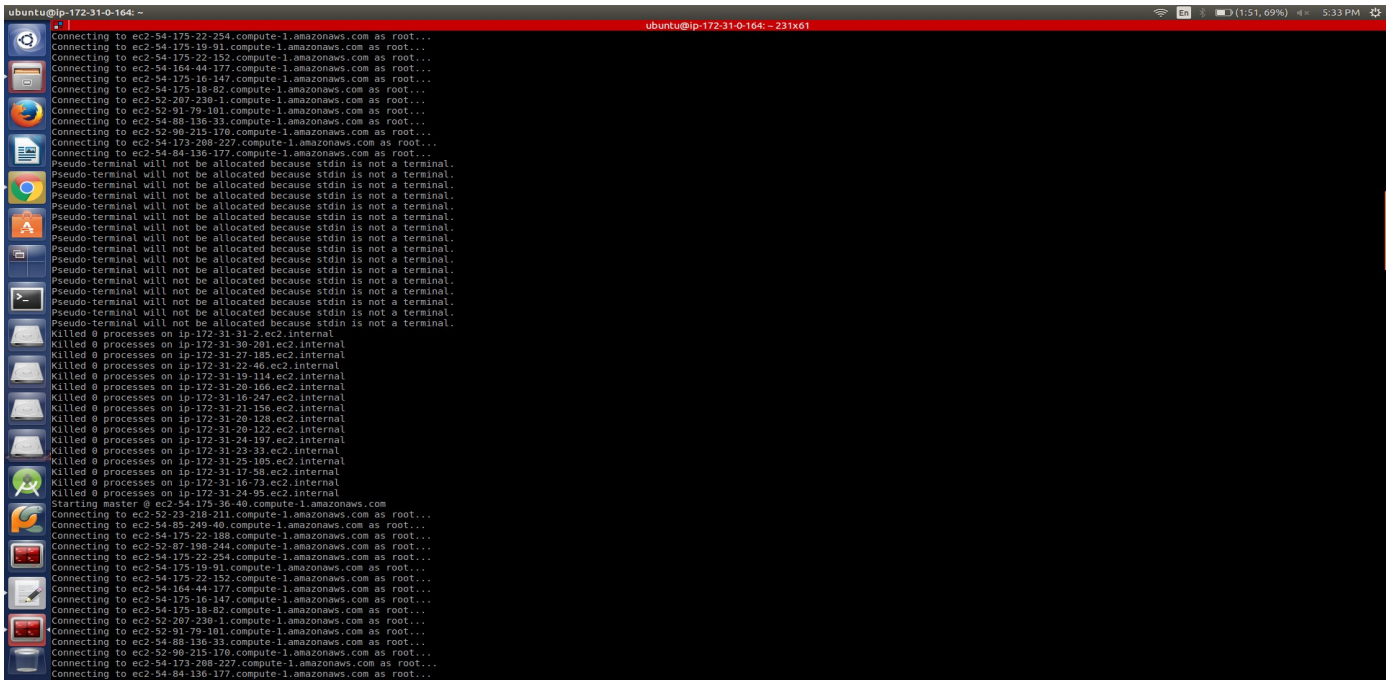
```
[ec2-user@ip-172-31-13-167 raid]$ head -10 output
    "O!uve  00000000000000000000000001228D4  7777888800022224444DDDDDDDDEEEE00000000CCCC7777DDDD
    PMd32=  00000000000000000000000003440CC1  FFFFEEEE6666CCCCBBBB999933335555DDDDDDDD777788886666
    ^3CO],  00000000000000000000000000158C5C5  5555AAAA9999EEEE888822229999CCCCDDDD6666555544442222
   !&S3/]]  00000000000000000000000002145D78  8888BBBBDDDD1111CCCC55556666BBBB1111EEEEDDDD22229999
   !,=U#,9  00000000000000000000000019072E3  33332222FFFFBBBB0000FFFFAAAA666655553333DDDD3333CCCC
   !0f[ITd  00000000000000000000000003CAAB4B  9999FFFF555533337777CCCC4444BBBB7777EEEEBBBBDDDD4444
   !f6Suy2  00000000000000000000000003ABFD84  EEEE555555556666AAAA5555BBBBDDDD0000111166660000DDDD
   #%NIpq.  00000000000000000000000003B36FB9  1111000033334444111166666666AAAAAAAA00001111CCCCEEEE
   #'^cl'~  00000000000000000000000002EDC5C8  8888AAAA11114444FFFF77773333EEEE44440000FFFF99999999
   $"-'Q)]  00000000000000000000000005F1265D  CCCC6666EEEE22220000DDDDAAAA88886666BBBB00006666AAAA
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$
[ec2-user@ip-172-31-13-167 raid]$ tail -10 output
~~~uq2k#=U  00000000000000000000000002C06745  99991111DDDD222211110000FFFFEEEEFFFF33337777CCCC2222
~~~v/0&Qnm  00000000000000000000000004709701  CCCC88883333FFFF0000000000000999991111FFFF777744446666
~~~yKOl:gE  00000000000000000000000002048B4F  CCCC1111444488882222666 6BBBB888855557777EEEEBBBB0000
~~~yK^H.il  00000000000000000000000000463D004  44440000FFFF3333999944447777DDDDFFFFAAAA11118888DDDD
~~~yL;C'XE  00000000000000000000000005B0D211  2222EEEE3333000022221111CCCCFFFF555577774444BBBB6666
~~~zbA_ Tt  00000000000000000000000007F9F4F  BBBBCCCC666655559999FFFF8888AAAA11116666AAAABBBB0000
~~~zeO^FEg  00000000000000000000000001E06130  4444CCCCBBBB999922228888555588888CCCCFFFF000011111111
~~~}GxjWHI  00000000000000000000000000CA1345  777711118888AAAAAAAA22221111BBBB00002222BBBBCCCC2222
~~~}P;]g0g  00000000000000000000000040DA3E4  4444FFFF444466663333EEEE88888888DDDDEEEE44442222DDDD
~~~}kU|K<p  00000000000000000000000005E4A0AA  0000666655551111BBBB88889999AAAA55550000333355557777
[ec2-user@ip-172-31-13-167 raid]$
```

Total Execution Time for 10 GB = 8.1+4.3 = 12.4 min = 744 sec

Now to invoke 16 nodes, use the following script:

./spark-ec2 -k spark -i spark.pem -s 16 --instance-type=c3.large -m c3.4xlarge --spot-price=0.14 launch spark_100gb

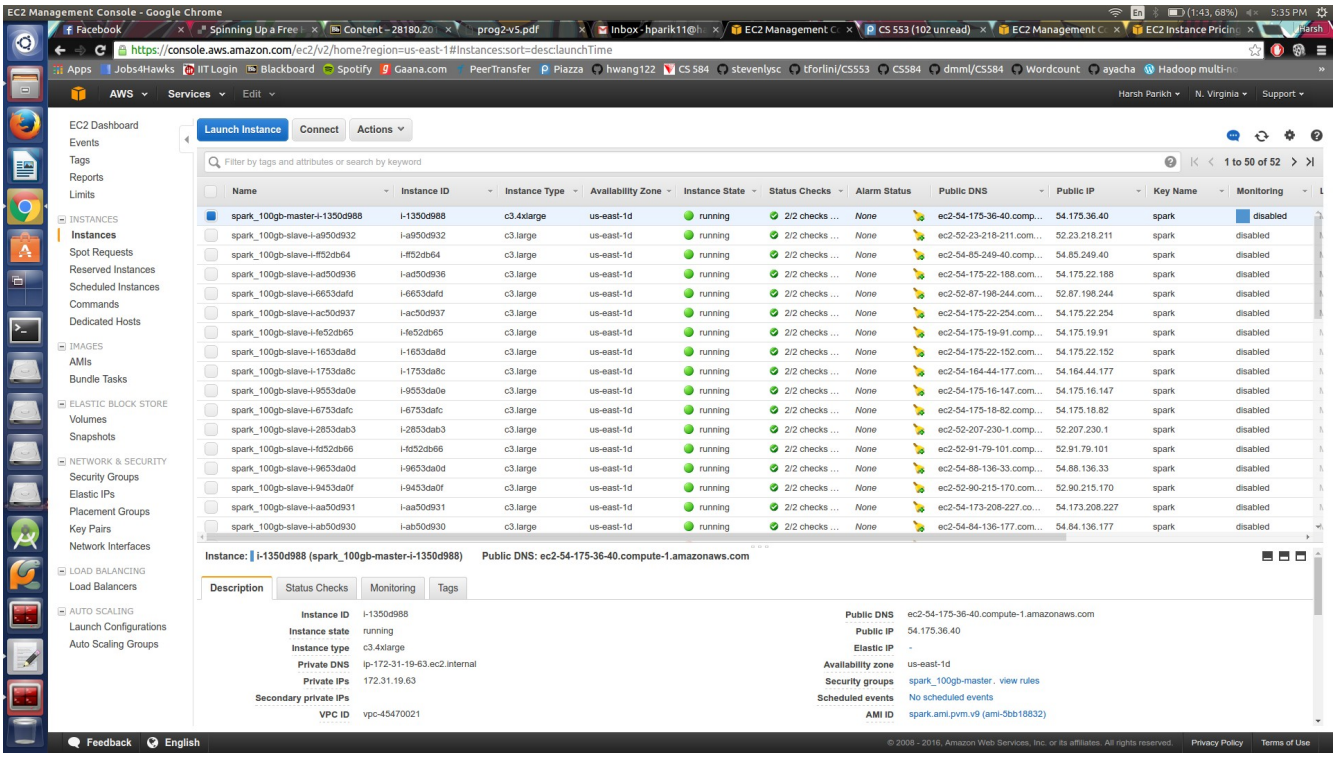




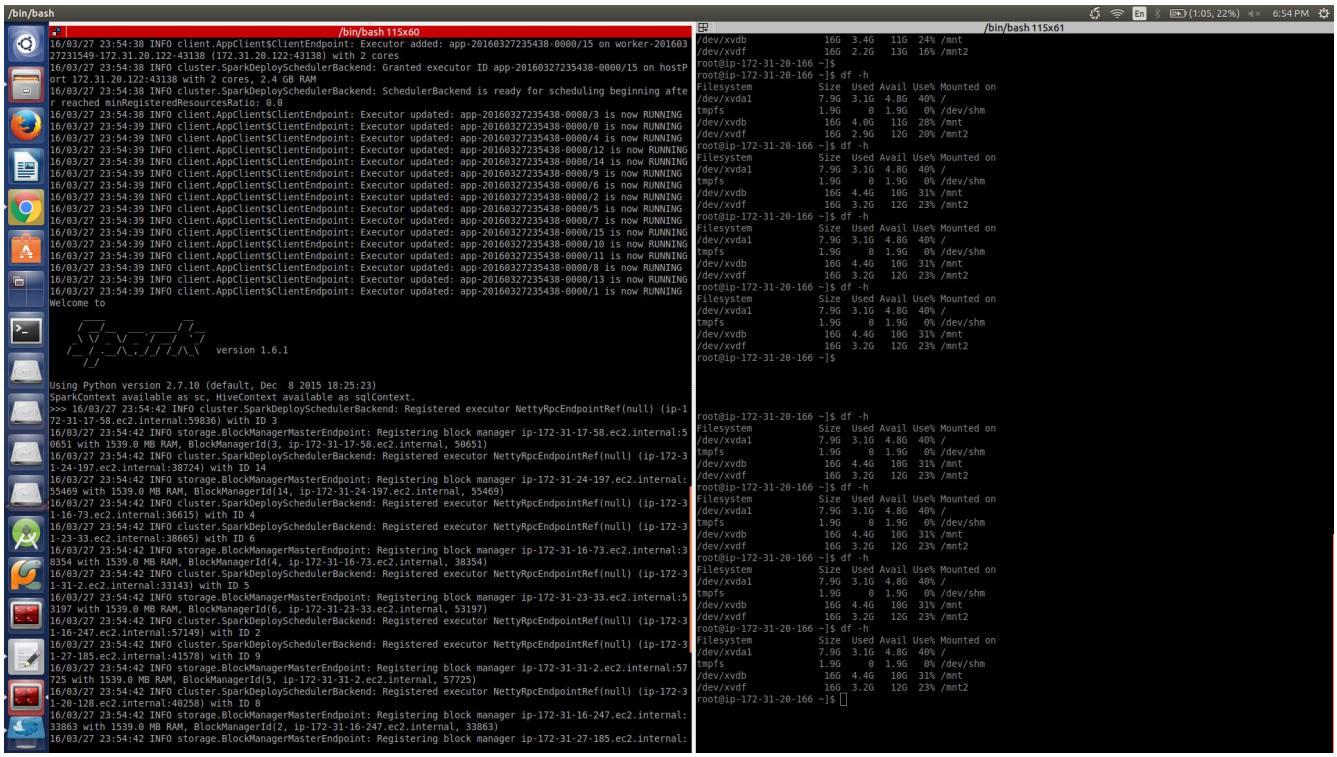
Running this script will launch 17 instances

```
ubuntu@ip-172-31-0-164: ~
ubuntu@ip-172-31-0-164: ~ 231x61
ec2-52-91-79-101.compute-1.amazonaws.com
ec2-54-88-136-33.compute-1.amazonaws.com
ec2-52-90-215-170.compute-1.amazonaws.com
ec2-54-173-208-227.compute-1.amazonaws.com
ec2-54-84-136-177.compute-1.amazonaws.com
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-52-23-218-211.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-85-249-40.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-175-22-188.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-52-87-198-244.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-175-22-254.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-175-19-91.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-175-22-152.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-164-44-177.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-175-16-147.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-175-18-82.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-52-207-230-1.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-52-91-79-101.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-88-136-33.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-52-90-215-170.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-173-208-227.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond:                               [FAILED]
Starting GANGLIA gmond:                                    [  OK  ]
Connection to ec2-54-84-136-177.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmetad:                              [FAILED]
Starting GANGLIA gmetad:                                   [  OK  ]
Stopping httpd:                                            [FAILED]
Starting httpd: httpd: Syntax error on line 154 of /etc/httpd/conf/httpd.conf: Cannot load /etc/httpd/modules/mod_authz_core.so into server: /etc/httpd/modules/mod_authz_core.so: cannot open shared object file: No such file or dire
ctory
                                                           [FAILED]
```

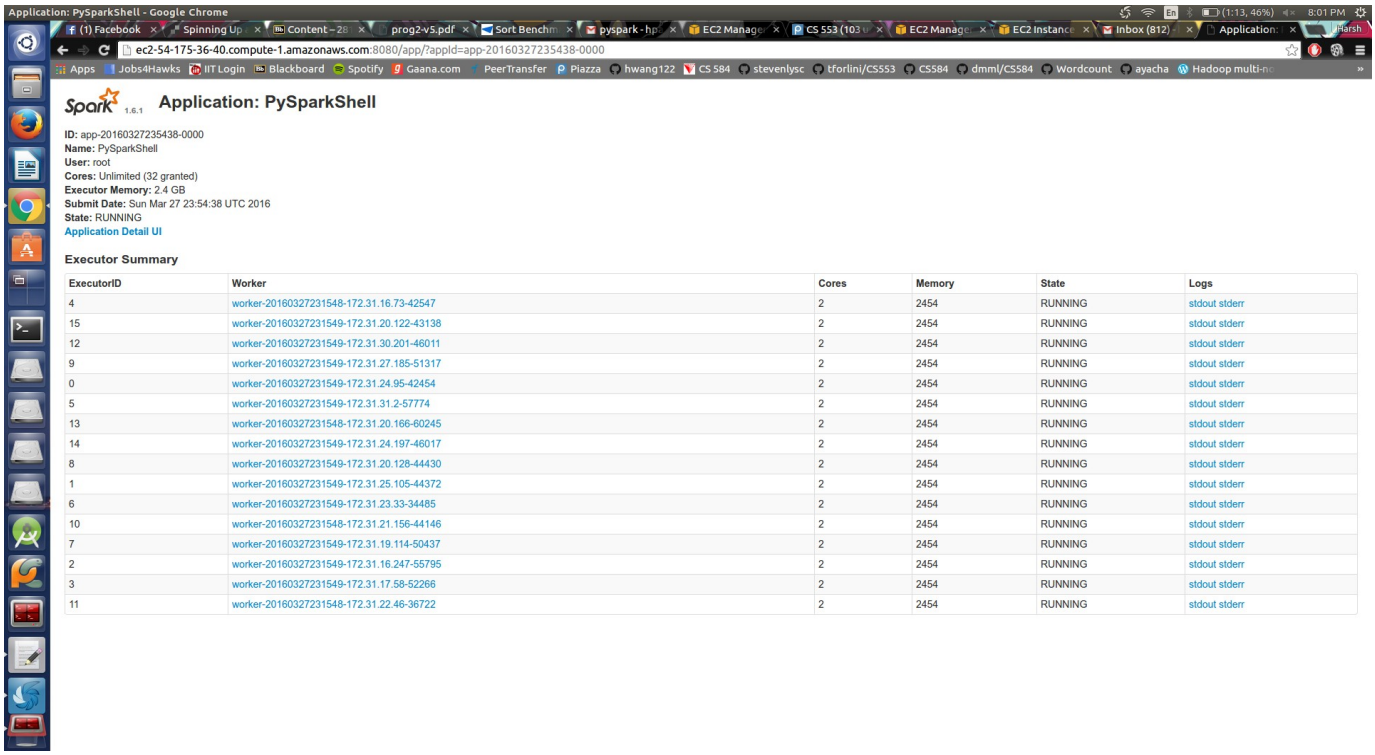
17 Instances in Running state:



To open Pyspark to run all commands:

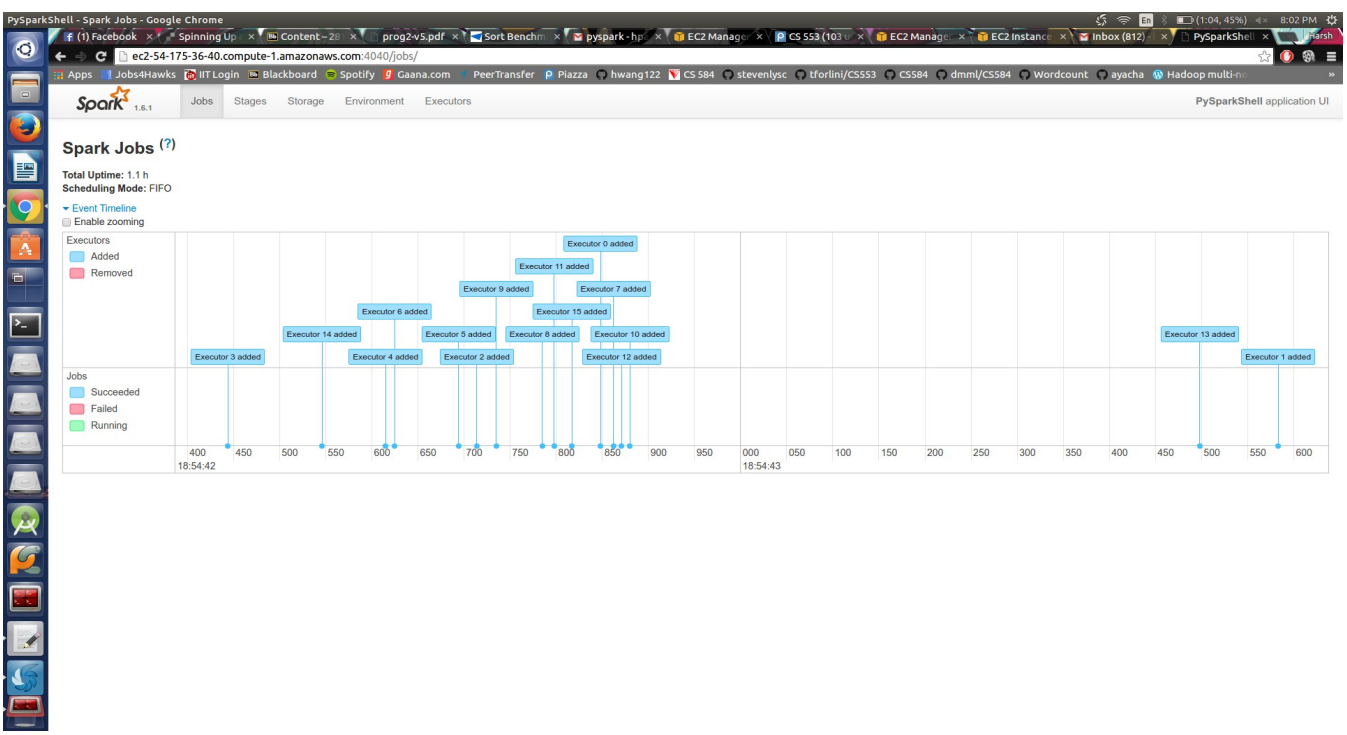


All Workers(Slaves) in running state while Sort Application running

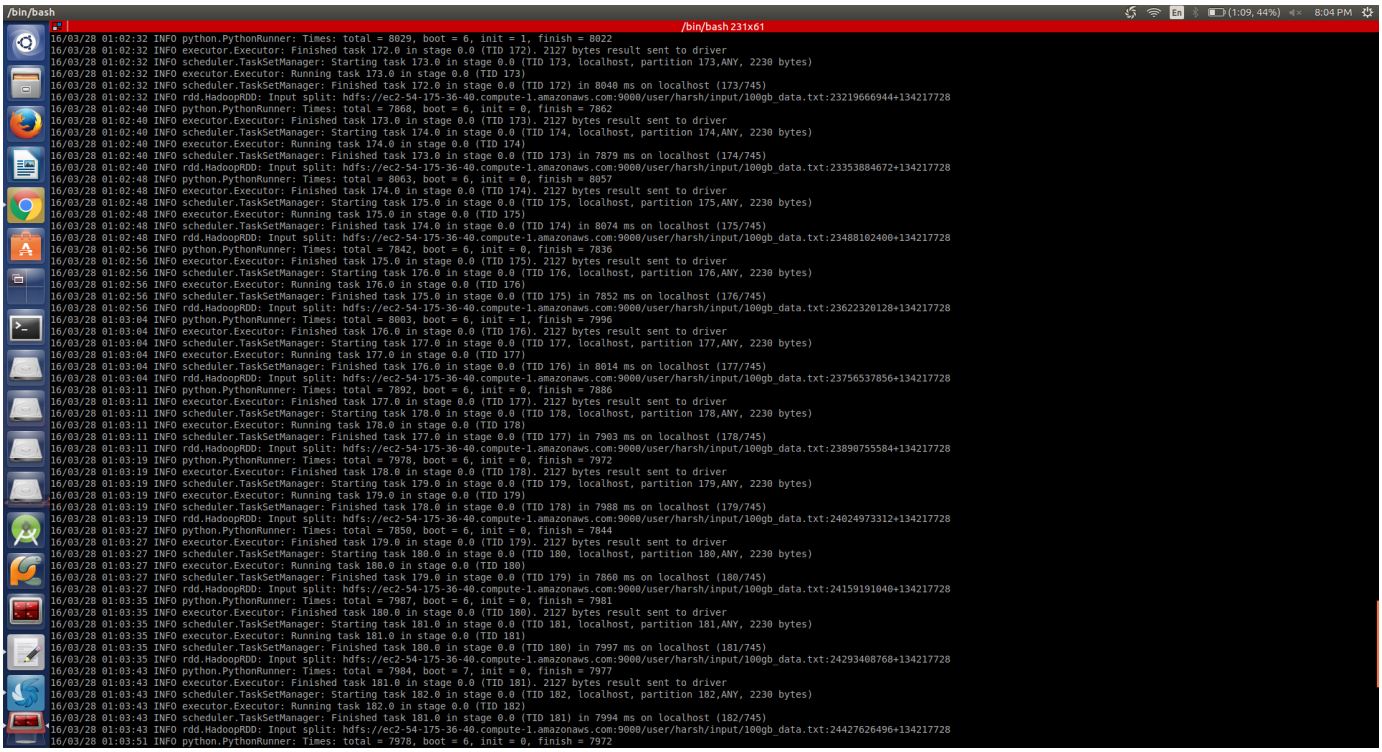


ExecutorID	Worker	Cores	Memory	State	Logs
4	worker-20160327231548-172.31.16.73-42547	2	2454	RUNNING	stdout stderr
15	worker-20160327231549-172.31.20.122-43138	2	2454	RUNNING	stdout stderr
12	worker-20160327231549-172.31.30.201-46011	2	2454	RUNNING	stdout stderr
9	worker-20160327231549-172.31.27.185-51317	2	2454	RUNNING	stdout stderr
0	worker-20160327231549-172.31.24.95-42454	2	2454	RUNNING	stdout stderr
5	worker-20160327231549-172.31.31.2-57774	2	2454	RUNNING	stdout stderr
13	worker-20160327231548-172.31.20.166-60245	2	2454	RUNNING	stdout stderr
14	worker-20160327231549-172.31.24.197-46017	2	2454	RUNNING	stdout stderr
8	worker-20160327231549-172.31.20.128-44430	2	2454	RUNNING	stdout stderr
1	worker-20160327231549-172.31.25.105-44372	2	2454	RUNNING	stdout stderr
6	worker-20160327231549-172.31.23.33-34485	2	2454	RUNNING	stdout stderr
10	worker-20160327231548-172.31.21.156-44146	2	2454	RUNNING	stdout stderr
7	worker-20160327231549-172.31.19.114-50437	2	2454	RUNNING	stdout stderr
2	worker-20160327231549-172.31.16.247-55795	2	2454	RUNNING	stdout stderr
3	worker-20160327231549-172.31.17.58-52266	2	2454	RUNNING	stdout stderr
11	worker-20160327231548-172.31.22.46-36722	2	2454	RUNNING	stdout stderr

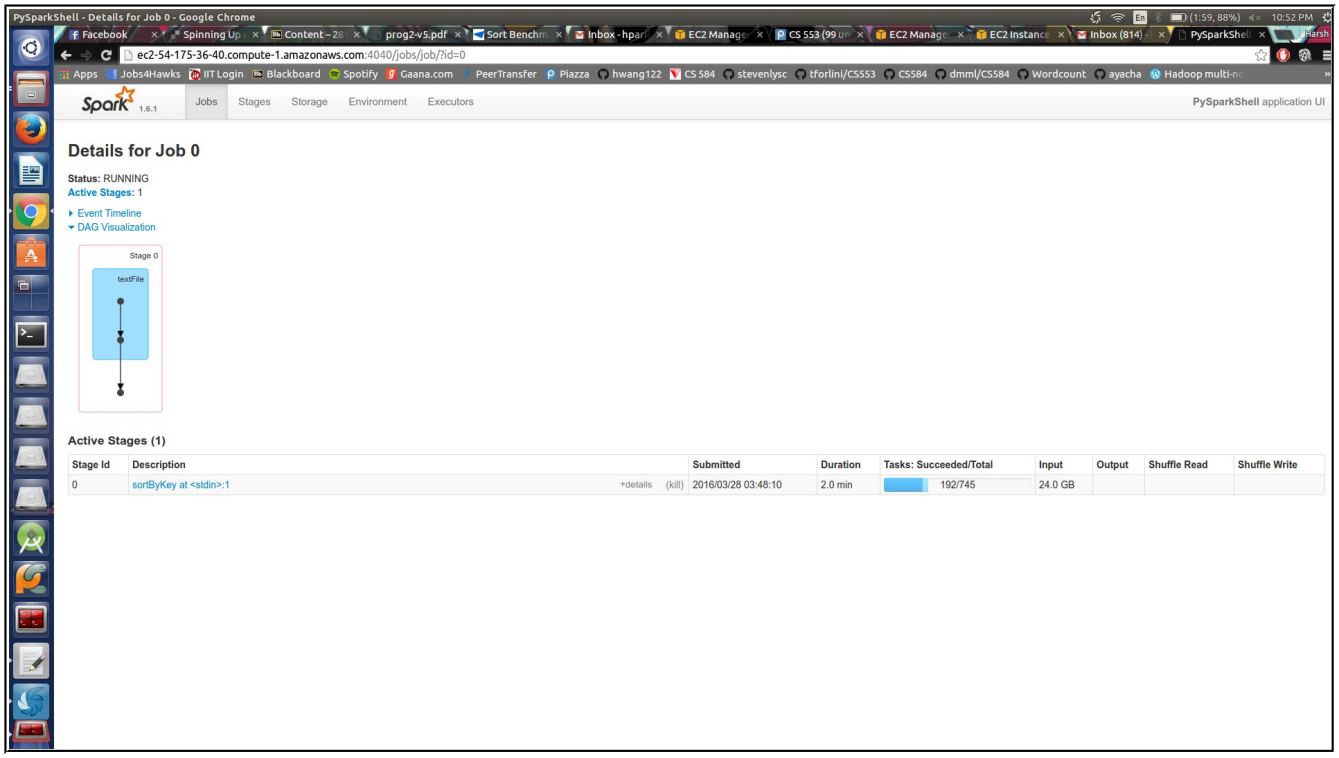
Event Time line of Each worker:



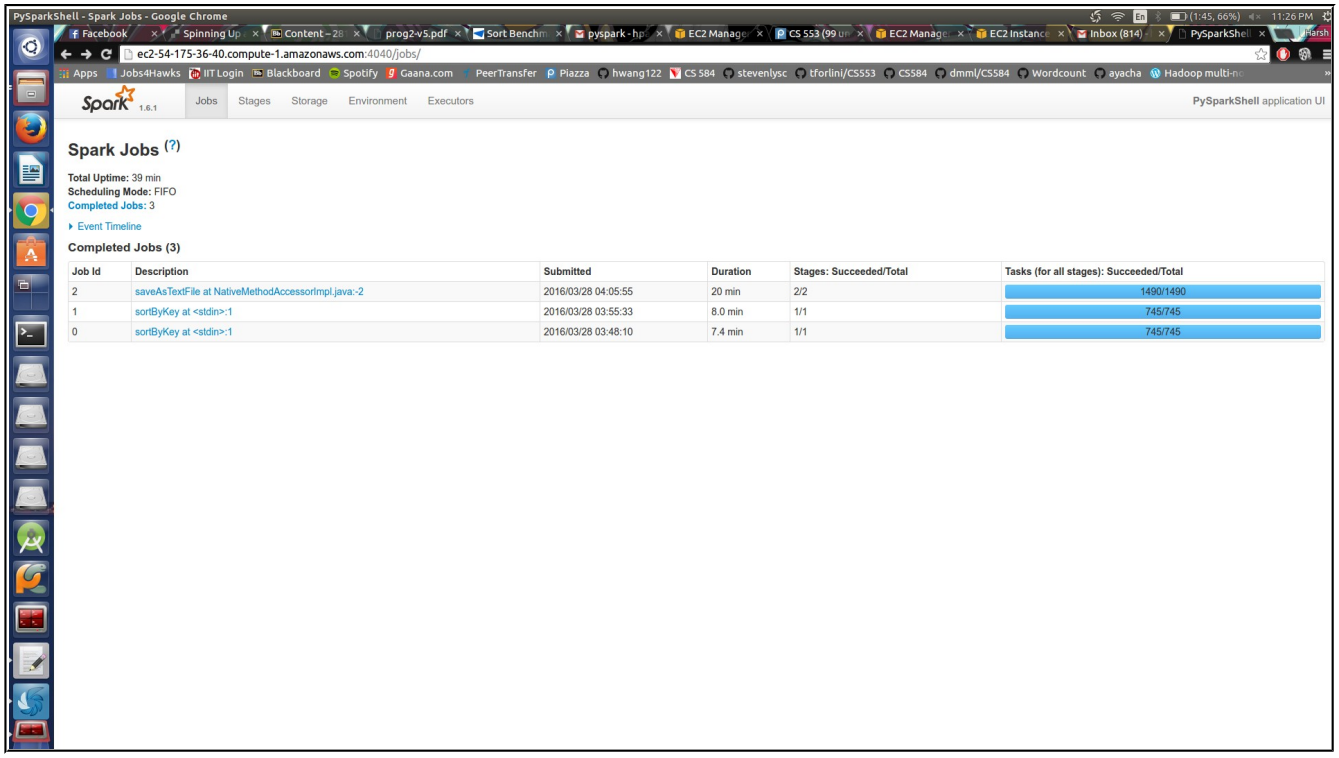
In running state of 100 GB (16 node):



Sort by Key operation (100 GB):



Result after completion: (100 GB)



Output of 100 GB (16 Nodes)(total 745 files):

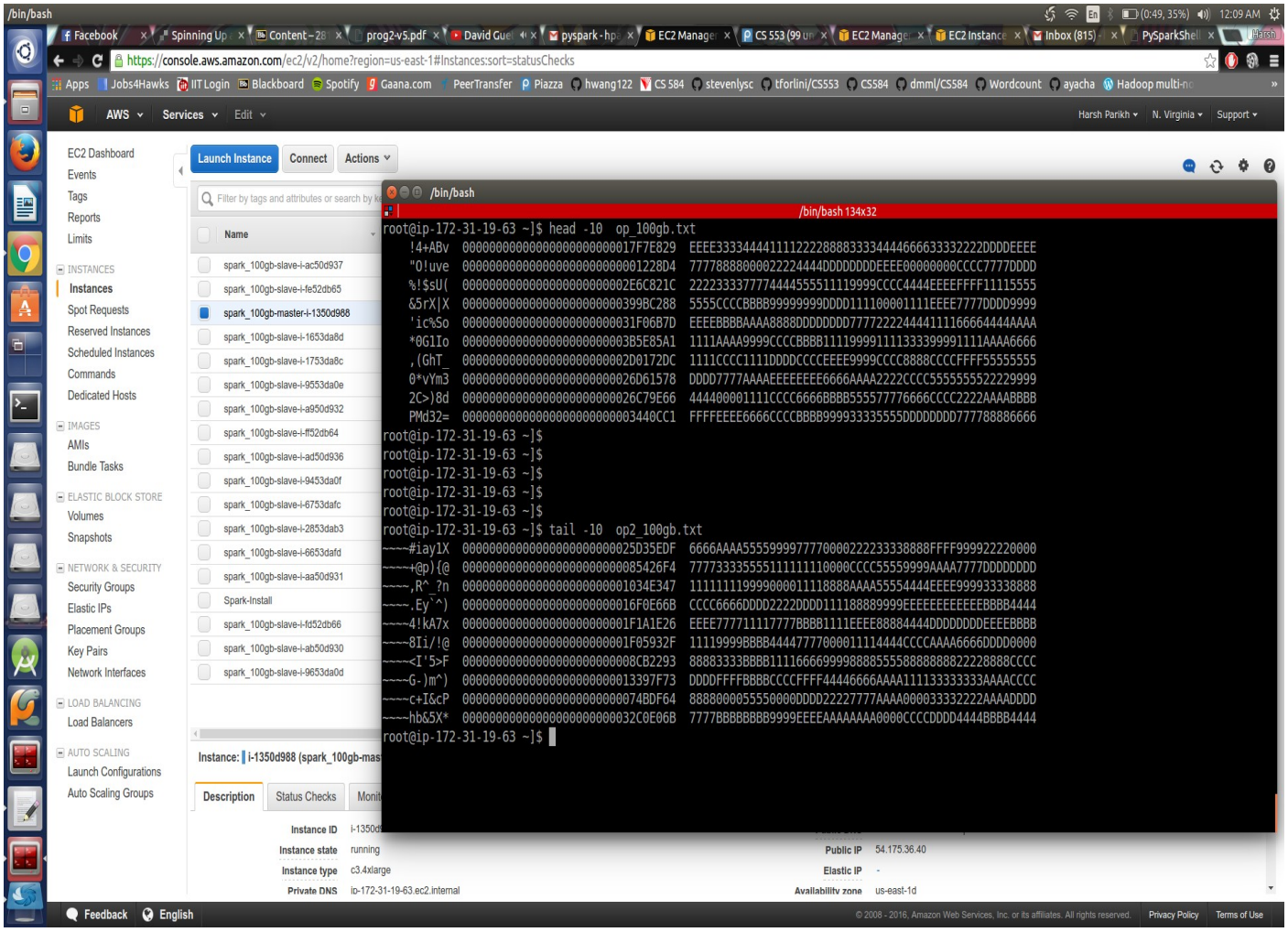
```
/bin/bash                                                                                                                    (1:29, 56%)  11:41 PM
/bin/bash 205x57
-rw-r--r--   1 root supergroup  106979796 2016-03-28 04:24 /user/harsh/output_100gb/part-00689
-rw-r--r--   1 root supergroup  129908988 2016-03-28 04:24 /user/harsh/output_100gb/part-00690
-rw-r--r--   1 root supergroup  151701561 2016-03-28 04:24 /user/harsh/output_100gb/part-00691
-rw-r--r--   1 root supergroup  116333712 2016-03-28 04:24 /user/harsh/output_100gb/part-00692
-rw-r--r--   1 root supergroup  102920697 2016-03-28 04:24 /user/harsh/output_100gb/part-00693
-rw-r--r--   1 root supergroup  112191948 2016-03-28 04:24 /user/harsh/output_100gb/part-00694
-rw-r--r--   1 root supergroup   70450677 2016-03-28 04:24 /user/harsh/output_100gb/part-00695
-rw-r--r--   1 root supergroup  133001451 2016-03-28 04:24 /user/harsh/output_100gb/part-00696
-rw-r--r--   1 root supergroup  142779483 2016-03-28 04:24 /user/harsh/output_100gb/part-00697
-rw-r--r--   1 root supergroup  117434196 2016-03-28 04:24 /user/harsh/output_100gb/part-00698
-rw-r--r--   1 root supergroup  153311103 2016-03-28 04:24 /user/harsh/output_100gb/part-00699
-rw-r--r--   1 root supergroup   74294253 2016-03-28 04:24 /user/harsh/output_100gb/part-00700
-rw-r--r--   1 root supergroup  105435693 2016-03-28 04:24 /user/harsh/output_100gb/part-00701
-rw-r--r--   1 root supergroup  127316772 2016-03-28 04:24 /user/harsh/output_100gb/part-00702
-rw-r--r--   1 root supergroup  123708222 2016-03-28 04:24 /user/harsh/output_100gb/part-00703
-rw-r--r--   1 root supergroup  147575835 2016-03-28 04:25 /user/harsh/output_100gb/part-00704
-rw-r--r--   1 root supergroup  121415580 2016-03-28 04:25 /user/harsh/output_100gb/part-00705
-rw-r--r--   1 root supergroup  102222252 2016-03-28 04:24 /user/harsh/output_100gb/part-00706
-rw-r--r--   1 root supergroup  101472327 2016-03-28 04:24 /user/harsh/output_100gb/part-00707
-rw-r--r--   1 root supergroup  141059259 2016-03-28 04:25 /user/harsh/output_100gb/part-00708
-rw-r--r--   1 root supergroup  110276892 2016-03-28 04:25 /user/harsh/output_100gb/part-00709
-rw-r--r--   1 root supergroup   94730328 2016-03-28 04:25 /user/harsh/output_100gb/part-00710
-rw-r--r--   1 root supergroup  190837449 2016-03-28 04:25 /user/harsh/output_100gb/part-00711
-rw-r--r--   1 root supergroup  172395333 2016-03-28 04:25 /user/harsh/output_100gb/part-00712
-rw-r--r--   1 root supergroup  124293708 2016-03-28 04:25 /user/harsh/output_100gb/part-00713
-rw-r--r--   1 root supergroup  120305097 2016-03-28 04:25 /user/harsh/output_100gb/part-00714
-rw-r--r--   1 root supergroup  107904258 2016-03-28 04:25 /user/harsh/output_100gb/part-00715
-rw-r--r--   1 root supergroup   79425027 2016-03-28 04:25 /user/harsh/output_100gb/part-00716
-rw-r--r--   1 root supergroup  127654461 2016-03-28 04:25 /user/harsh/output_100gb/part-00717
-rw-r--r--   1 root supergroup   78449877 2016-03-28 04:25 /user/harsh/output_100gb/part-00718
-rw-r--r--   1 root supergroup  128487051 2016-03-28 04:25 /user/harsh/output_100gb/part-00719
-rw-r--r--   1 root supergroup  132851565 2016-03-28 04:25 /user/harsh/output_100gb/part-00720
-rw-r--r--   1 root supergroup  116209764 2016-03-28 04:25 /user/harsh/output_100gb/part-00721
-rw-r--r--   1 root supergroup  135995607 2016-03-28 04:25 /user/harsh/output_100gb/part-00722
-rw-r--r--   1 root supergroup  150924114 2016-03-28 04:25 /user/harsh/output_100gb/part-00723
-rw-r--r--   1 root supergroup  147515445 2016-03-28 04:25 /user/harsh/output_100gb/part-00724
-rw-r--r--   1 root supergroup  143295075 2016-03-28 04:25 /user/harsh/output_100gb/part-00725
-rw-r--r--   1 root supergroup   77077440 2016-03-28 04:25 /user/harsh/output_100gb/part-00726
-rw-r--r--   1 root supergroup  124562493 2016-03-28 04:25 /user/harsh/output_100gb/part-00727
-rw-r--r--   1 root supergroup  163677096 2016-03-28 04:25 /user/harsh/output_100gb/part-00728
-rw-r--r--   1 root supergroup  147236958 2016-03-28 04:25 /user/harsh/output_100gb/part-00729
-rw-r--r--   1 root supergroup   92219985 2016-03-28 04:25 /user/harsh/output_100gb/part-00730
-rw-r--r--   1 root supergroup  143139942 2016-03-28 04:25 /user/harsh/output_100gb/part-00731
-rw-r--r--   1 root supergroup  122086503 2016-03-28 04:25 /user/harsh/output_100gb/part-00732
-rw-r--r--   1 root supergroup  117310545 2016-03-28 04:25 /user/harsh/output_100gb/part-00733
-rw-r--r--   1 root supergroup  181297809 2016-03-28 04:25 /user/harsh/output_100gb/part-00734
-rw-r--r--   1 root supergroup  111680316 2016-03-28 04:25 /user/harsh/output_100gb/part-00735
-rw-r--r--   1 root supergroup  148061331 2016-03-28 04:25 /user/harsh/output_100gb/part-00736
-rw-r--r--   1 root supergroup  166205358 2016-03-28 04:25 /user/harsh/output_100gb/part-00737
-rw-r--r--   1 root supergroup  113725557 2016-03-28 04:25 /user/harsh/output_100gb/part-00738
-rw-r--r--   1 root supergroup  176472549 2016-03-28 04:25 /user/harsh/output_100gb/part-00739
-rw-r--r--   1 root supergroup  117385587 2016-03-28 04:25 /user/harsh/output_100gb/part-00740
-rw-r--r--   1 root supergroup  108249669 2016-03-28 04:25 /user/harsh/output_100gb/part-00741
-rw-r--r--   1 root supergroup  185911506 2016-03-28 04:25 /user/harsh/output_100gb/part-00742
-rw-r--r--   1 root supergroup  113241645 2016-03-28 04:25 /user/harsh/output_100gb/part-00743
-rw-r--r--   1 root supergroup  114428952 2016-03-28 04:25 /user/harsh/output_100gb/part-00744
root@ip-172-31-19-63 ~]$
```

Valsort of 100 GB(first and last file out of 745 files)

```
root@ip-172-31-19-63 64]$ ./valsort ../op_100gb.txt
Records: 1703516
Checksum: d00172f4a600f
Duplicate keys: 0
SUCCESS - all records are in order
```

```
root@ip-172-31-19-63 64]$ ./valsort ../op2_100gb.txt
Records: 1155848
Checksum: 8d049ac370508
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-19-63 64]$
```

Output of First and Last file (10 lines):

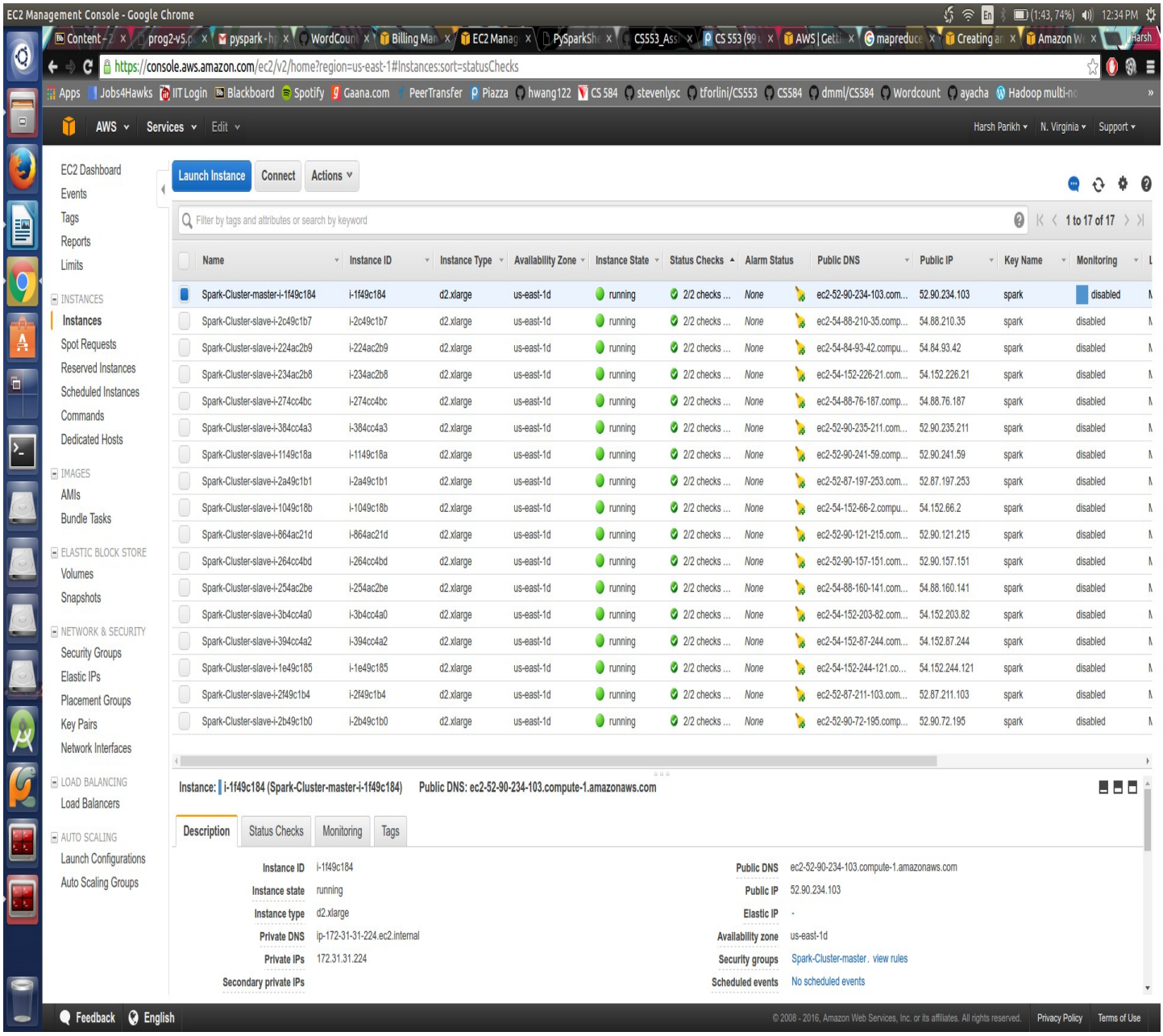


Total Time taken for 100 GB = 20 + 7.4 = 27.4 min = 1644 sec

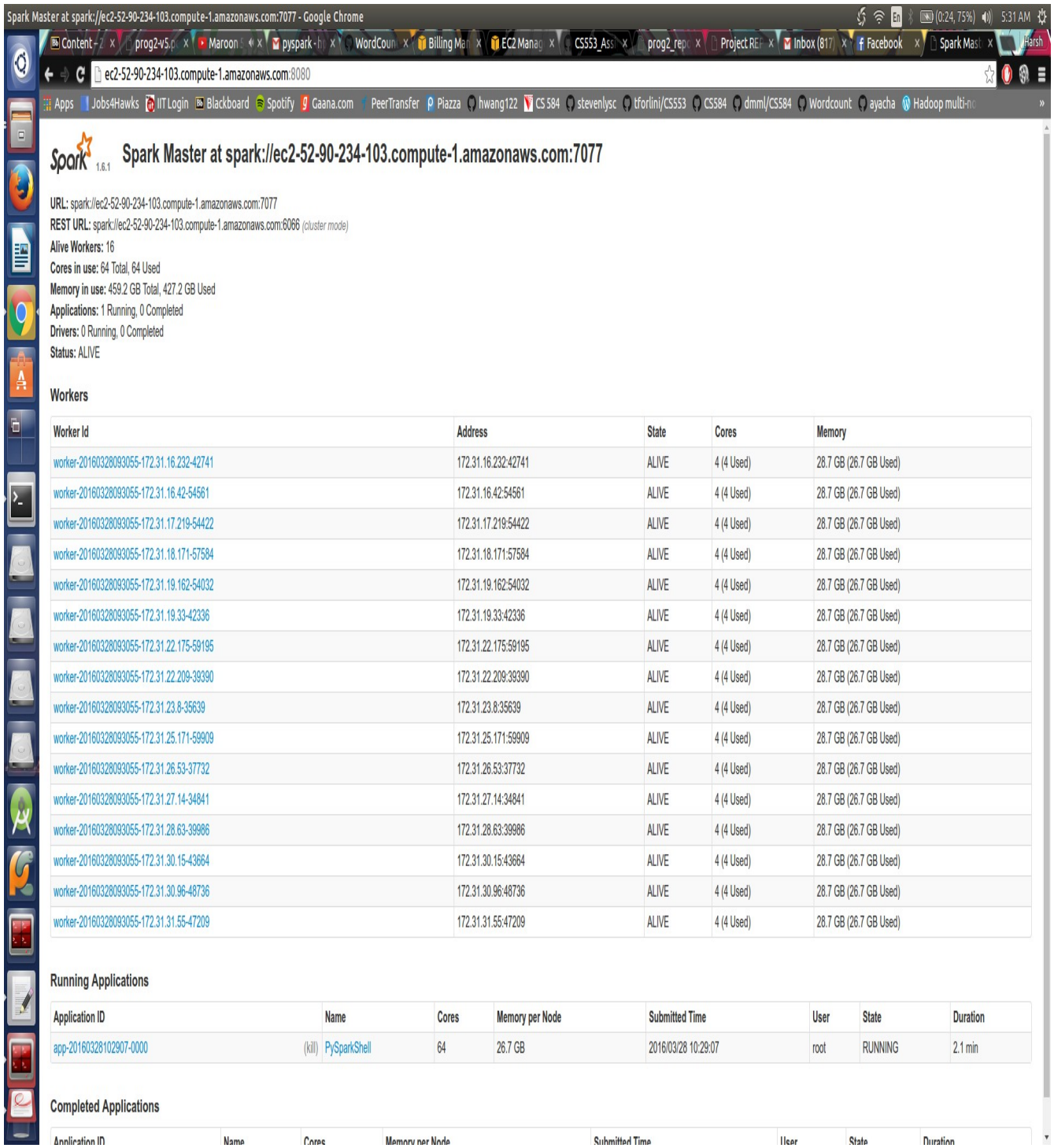
Output of 1 TB (16 Nodes):

After Launching d2.xlarge 17 instances by running following script:

./spark-ec2 -k spark -i spark.pem -s 16 --instance-type=d2.xlarge --spot-price=0.20 launch Spark-Cluster



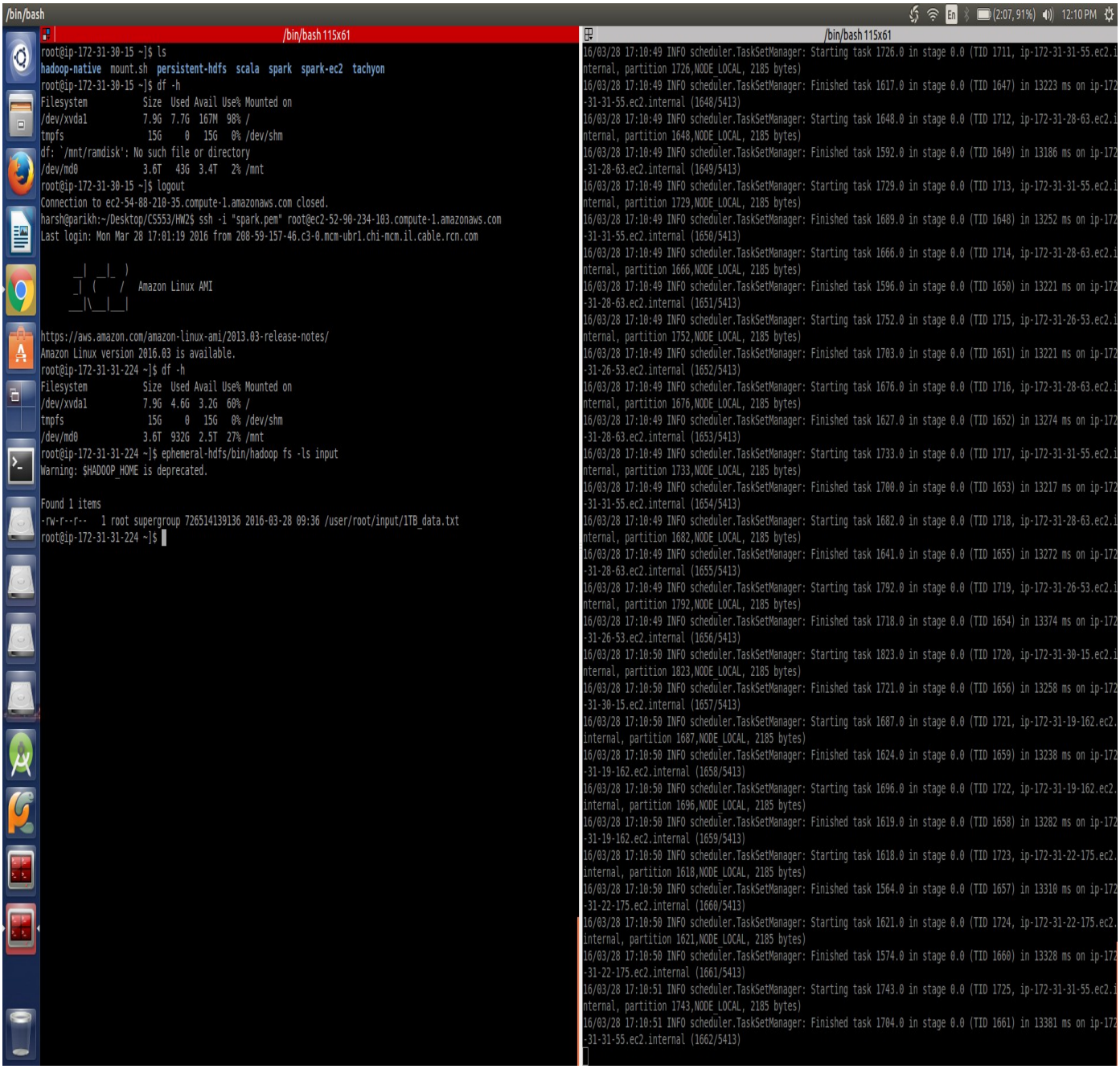
All working Slaves on web interface:



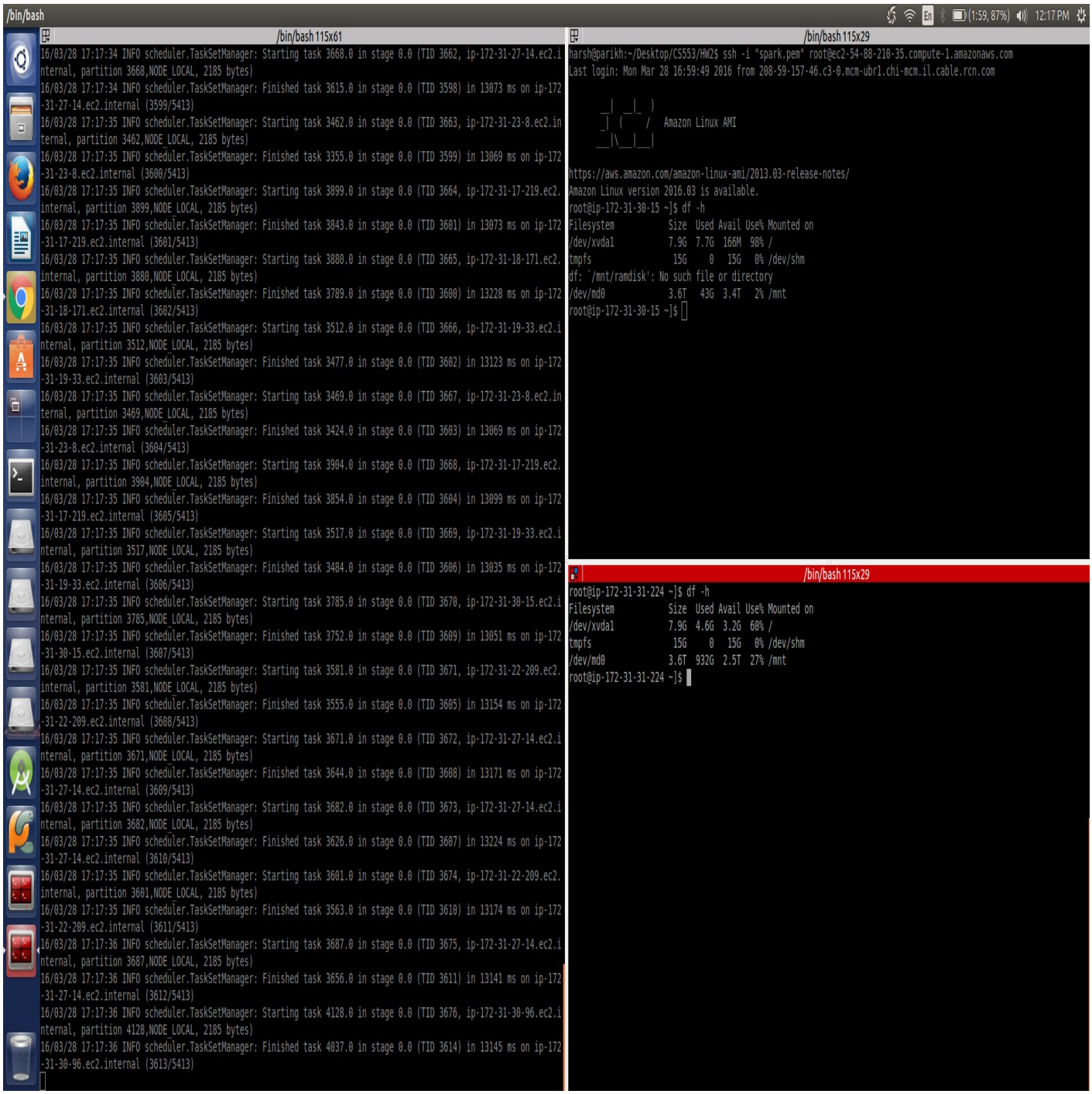
Worker Id	Address	State	Cores	Memory
worker-20160328093055-172.31.16.232-42741	172.31.16.232:42741	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.16.42-54561	172.31.16.42:54561	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.17.219-54422	172.31.17.219:54422	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.18.171-57584	172.31.18.171:57584	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.19.162-54032	172.31.19.162:54032	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.19.33-42336	172.31.19.33:42336	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.22.175-59195	172.31.22.175:59195	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.22.209-39390	172.31.22.209:39390	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.23.8-35639	172.31.23.8:35639	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.25.171-59909	172.31.25.171:59909	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.26.53-37732	172.31.26.53:37732	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.27.14-34841	172.31.27.14:34841	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.28.63-39986	172.31.28.63:39986	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.30.15-43664	172.31.30.15:43664	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.30.96-48736	172.31.30.96:48736	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)
worker-20160328093055-172.31.31.55-47209	172.31.31.55:47209	ALIVE	4 (4 Used)	28.7 GB (26.7 GB Used)

Application ID		Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160328102907-0000	(kill)	PySparkShell	64	26.7 GB	2016/03/28 10:29:07	root	RUNNING	2.1 min

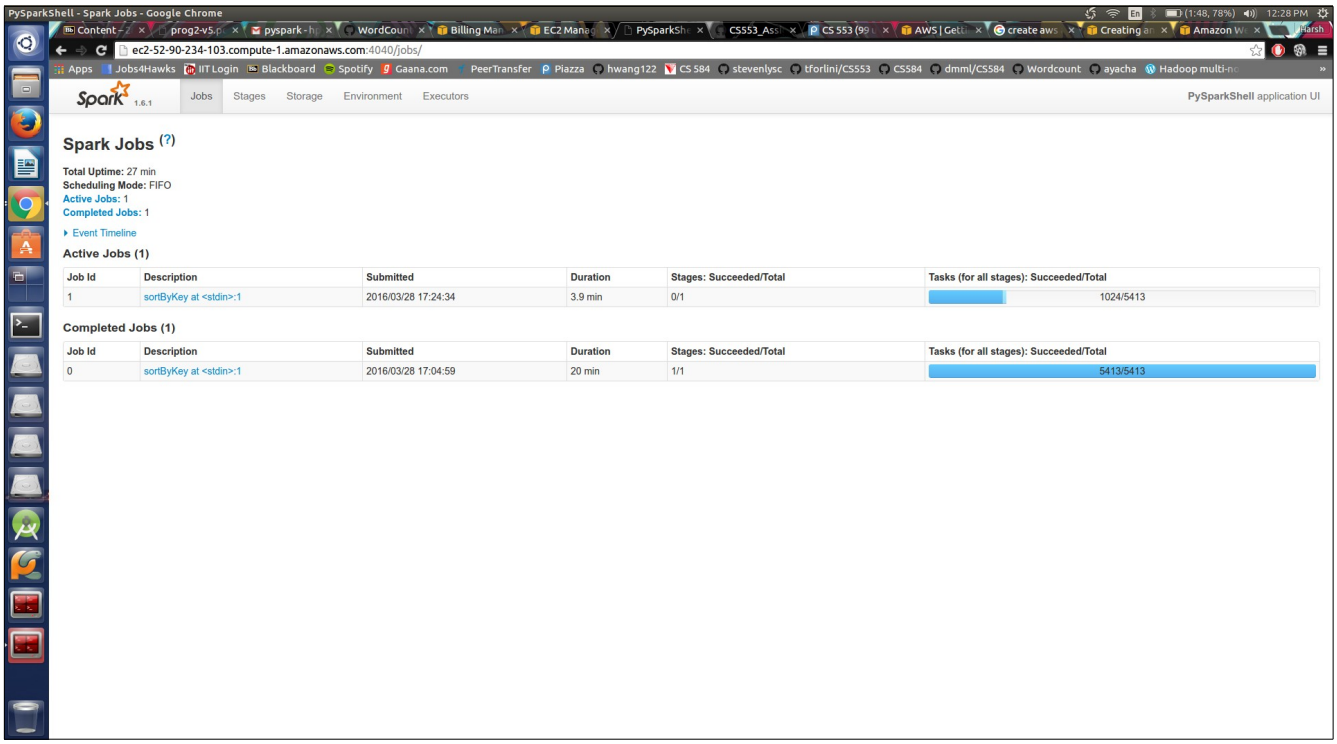
After generating 1TB on mounted disk:



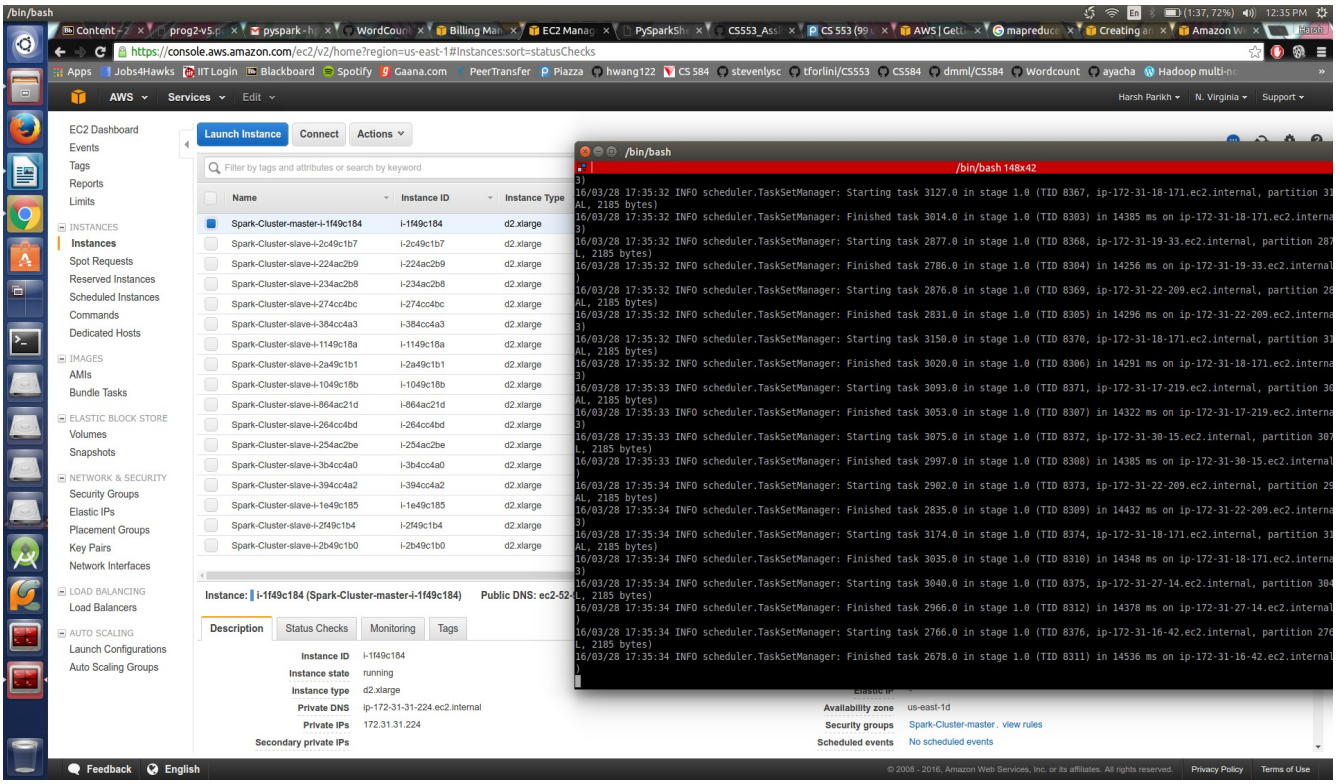
While process running, datanode memory:



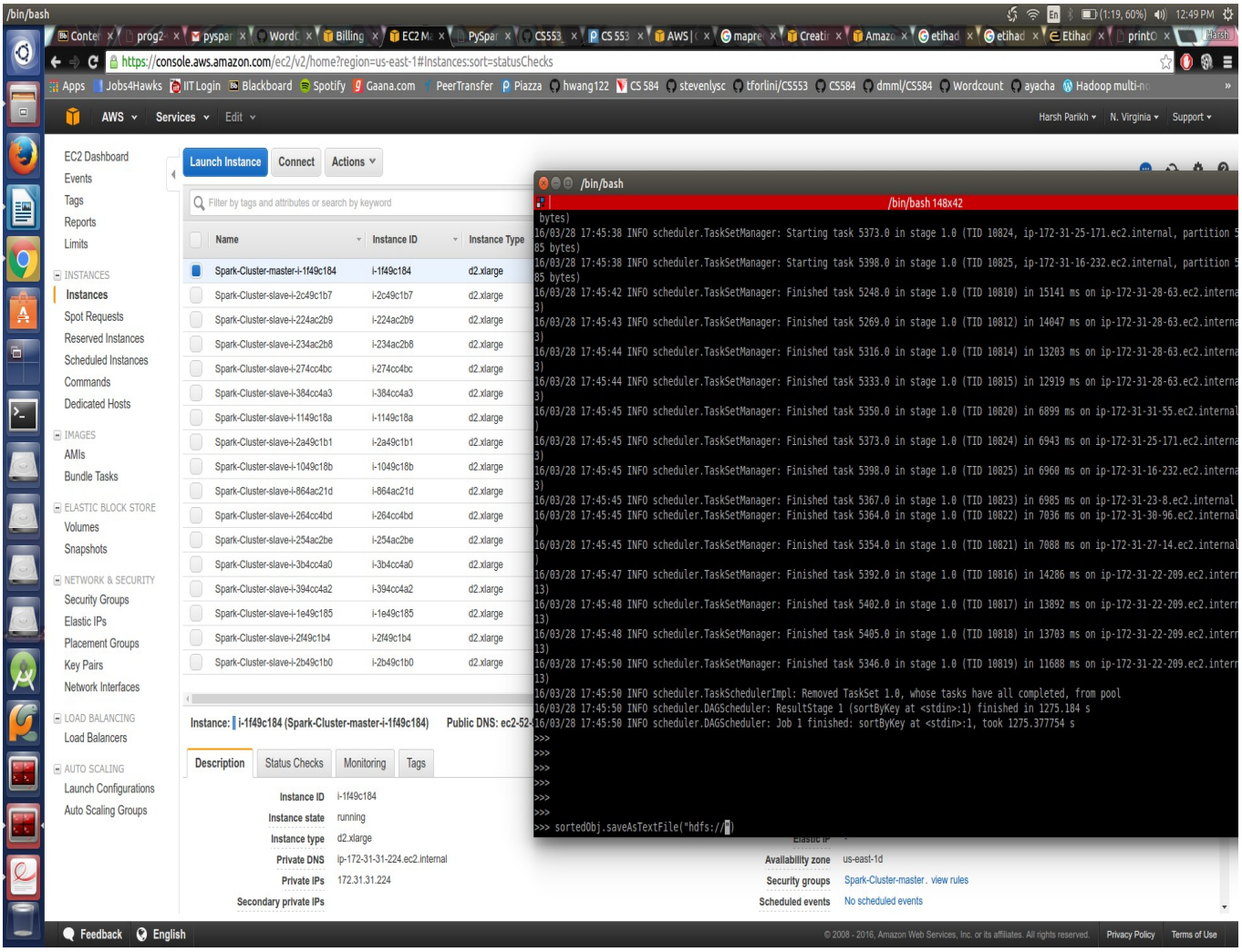
After completion of sortbykey operation:



Scheduler running:



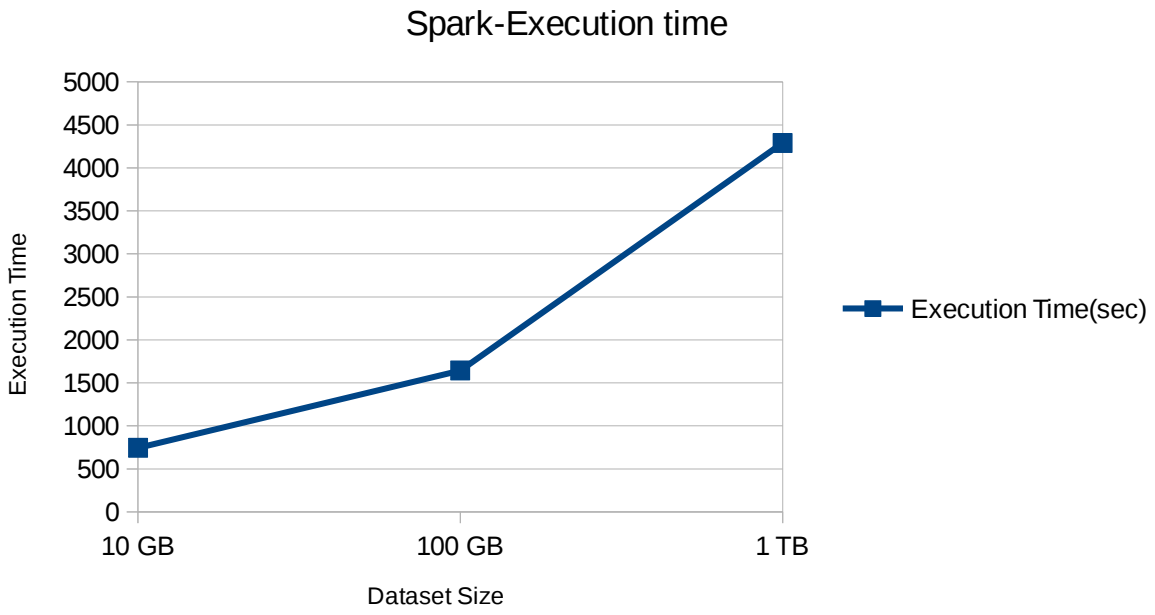
After completion of map-reduce:



Total Time Elapsed = 20 + 3.9 + 47.6 = 71.5 min = 4290 sec

Sort time for Spark:

Nodes	Datasets	Execution Time(sec)	Throughput(MB/s)
1	10 GB	744	13.44
16	100 GB	1644	60.82
16	1 TB	4290	233.100

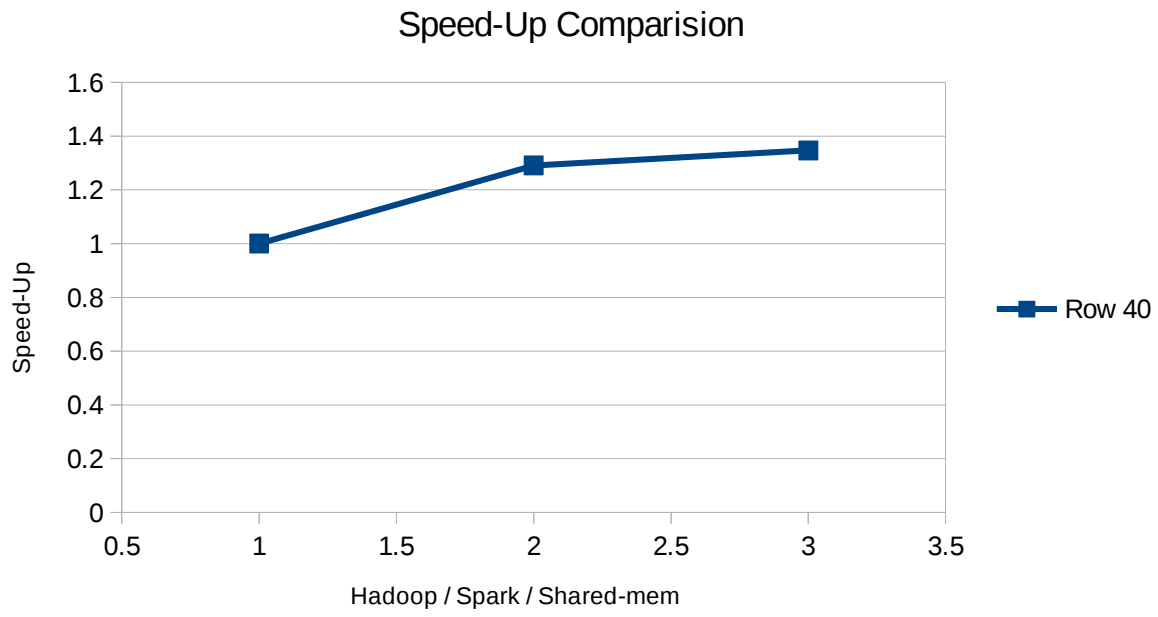


Speed-up (Hadoop-Spark)

Nodes	Datasets	Spark-Speed-up	Hadoop-Speed-up
1	10 GB	1.291	1
16	100 GB	2.58	1

Speed-up Comparison:

Node	Hadoop	Spark	Shared-Memory
1 Node	1	1.291	1.347



Conclusion:

For 1 thread, shared-memory takes less time than Hadoop and Spark. As per the speed-up graph, shared-memory has the highest throuh-put. For 16 threads, Spark has a better performance than Hadoop.

After several weeks' hard-work, I finally get used to use Hadoop and Spark. After taking long time configuration of Hadoop, we can implement MapReduce on AWS EC2 cluster. It takes a lot of time to configure on 16 nodes because I haven't made any script before but later it became very easy to configure on each worker. But only constrain is we need to go to each slave and mount drive for 100 GB and 1 TB. Through the comparison of performance, we can find a balance between time and cost on build virtual cluster, which makes calculations more effective.

Problems :

While configuring Hadoop on cloud, I dealt with problem listed as below: I had a problem while configuring ssh. But by using the link [1] , I configured ssh.

While I run program on multiple node, it gets stuck at map 0% , reduce 0%.

Besides that few errors occurred which were being solved by "try and error".

Question-Answers:

1) Which seems to be best at 1 node scale?

Shared-Memory performs best at 1 node compare to Hadoop and Spark compare to Speed-up Comparison graph.

2) How about 16 nodes?

On 16 nodes, Spark performs better compare to Hadoop.

3) Can you predict which would be best at 100 node scale?

I think Spark would be best for 100 node scale because it stores the intermediate data so that we can directly use those data when we want at any time but the only constrain is that It consumes a lot of memory.

4) How about 1000 node scales?

It depends on the system what we are using and how many resources we have. Because Spark takes large system resources. Though Spark is 100 times faster than Hadoop, it's not very friendly to intensive jobs. So still for intensive and large scale jobs, Hadoop would be the best choice for 1000 nodes.

Reference:

[1] http://insightdataengineering.com/blog/hadoopdevops/

[2]https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

[3] https://spark.apache.org/docs/0.9.1/python-programming-guide.html

[4] http://www.ordinal.com/gensort.html

[5] http://codexpi.com/merge-sort-python-iterative-recursive-implementations/