

# CS 553 Programming Assignment#2 Sort on Hadoop/Spark and Shared-Memory Sort

This document lists the various aspects which needed to be accomplished for successful completion of assignment:

- Setting Virtual Cluster (1 node and 16 nodes)
- Implementing Shared-Memory Sort for 10GB in Java
- Install and Configure Hadoop
- Implementing 10 and 100 GB sort in Hadoop
- Install and Configure Spark
- Implementing 10 and 100 GB sort in Spark
- Measuring the performance of the above task.

## 1 EXPERIMENTAL ENVIRONMENT

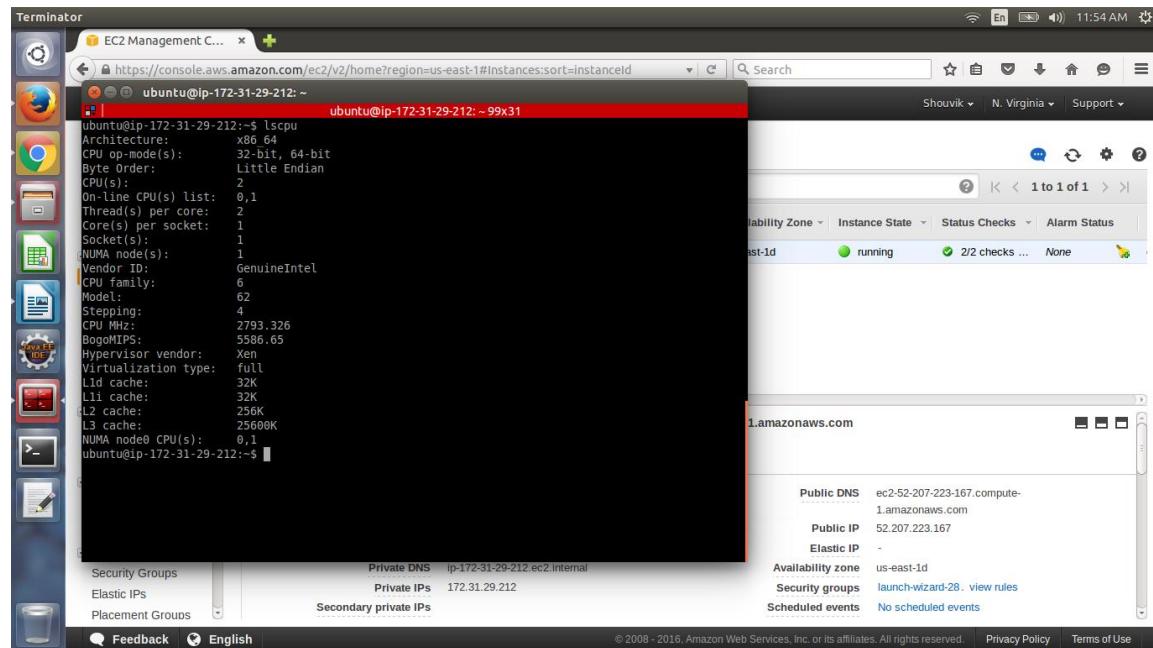
---

I would be running my experiments of on Amazon EC2 C3.Large cloud instance. Below screenshot provides the details of our Experimental Environment (1 node).

Java Version: 1.7.0\_95

Hadoop Version: 2.7.2

Spark Version: 1.6.1



## 2 SETTING VIRTUAL CLUSTER

For setting up virtual cluster follow the below steps:

1. Log into Amazon AWS service Website (<http://aws.amazon.com>)
2. Select EC2 from the compute section.
3. From EC2 dashboard select Instance (You can also go for spot requests).
4. Inside Instance select Launch Instance.
5. In instance type selection screen select “C3.Large” instance. Here if you need a spot instance then you select appropriate checkbox.
6. Add storage to our selected instance.
7. In Security group, we add and configure TCP and ICMP protocol to use public IP address.

The screenshot shows the AWS EC2 Management Console interface. The left sidebar has a tree view with 'AWS' selected, showing 'EC2 Dashboard', 'Events', 'Tags', 'Reports', 'Limits', 'INSTANCES' (which is expanded to show 'Instances', 'Spot Requests', 'Reserved Instances', 'Scheduled Instances', 'Commands', 'Dedicated Hosts'), 'IMAGES', 'ELASTIC BLOCK STORE', and 'NETWORK & SECURITY'. The main content area is titled 'Instances' and shows a table with one row. The table columns are 'Name', 'Instance ID', 'Instance Type', 'Availability Zone', 'Instance State', 'Status Checks', and 'Alarm Status'. The row for 'master\_hadoop' has an 'Actions' button. Below the table, a detailed view for 'Instance: i-8c8b0117 (master\_hadoop)' is shown, including tabs for 'Description', 'Status Checks', 'Monitoring', and 'Tags'. The 'Description' tab displays the instance ID, state, type, and network details like Public DNS, Public IP, and Private IPs. The 'Status Checks' tab shows 2/2 checks passing. The 'Monitoring' and 'Tags' tabs are empty. At the bottom of the page, there are links for 'Feedback', 'English', and copyright information: '© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.' and links to 'Privacy Policy' and 'Terms of Use'.

Above screenshot shows a configured C3.Large in Running Instance Dashboard.

### Steps to mount unmounted disk space and renaming it to /mnt/raid:

1. Type below command to see the size of unmounted partitions:  
>>lsblk
2. Merge and join two unmounted partitions using below commands:  
>>sudo apt-get install mdadm  
>>sudo mdadm --create --verbose /dev/md0 --level=0 --name=hadoop --raid-devices=2 /dev/xvdb /dev/xvdc

3. Label the joined partition with a specific name using below command:

```
>>sudo mkfs.ext4 -L hadoop /dev/md0
```

4. Create a directory which needs to be mapped with the mounted disk:

```
>>sudo mkdir -p /mnt/raid
```

5. Mount the newly created space to the directory that we created in Step4:

```
>>sudo mount LABEL=hadoop /mnt/raid
```

6. To check the mounted path enter the below command:

```
>>df -h
```

7. Grant permissions to the “ubuntu” user using below command:

```
>>sudo chown -R ubuntu:ubuntu mnt
```

### Screenshots for steps for mounting unmounted disk space and renaming it to /mnt/raid:

```
ubuntu@ip-172-31-21-202:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda   202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdb   202:16   0   30G  0 disk
xvdc   202:32   0  15.3G 0 disk
ubuntu@ip-172-31-21-202:~$ sudo mdadm --create --verbose /dev/md0 --level=0 --name=hadoop --raid-devices=2 /dev/xvdb /dev/xvdc
mdadm: chunk size defaults to 512K
mdadm: /dev/xvdc appears to contain an ext2fs file system
      size=15996416K  mtime=Thu Jan  1 00:00:00 1970
Continue creating array? yes
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
ubuntu@ip-172-31-21-202:~$ sudo mkfs.ext4 -L hadoop /dev/md0
mke2fs 1.42.9 (4-Feb-2014)
Filesystem label=hadoop
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=128 blocks, Stripe width=256 blocks
2967888 inodes, 11863168 blocks
593158 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
363 block groups
32768 blocks per group, 32768 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
      4096000, 7962624, 11239424

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

ubuntu@ip-172-31-21-202:~$ sudo mkdir -p /mnt/raid
ubuntu@ip-172-31-21-202:~$ sudo mount LABEL=hadoop /mnt/raid
```

```
ubuntu@ip-172-31-21-202:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G   12K  1.9G  1% /dev
tmpfs           377M  352K  377M  1% /run
/dev/xvda1      7.8G  1.8G  5.6G  25% /
none            4.0K    0  4.0K  0% /sys/fs/cgroup
none            5.0M    0  5.0M  0% /run/lock
none            1.9G    0  1.9G  0% /run/shm
none           100M    0  100M  0% /run/user
/dev/md0         45G   52M  43G  1% /mnt/raid
ubuntu@ip-172-31-21-202:~$ cd ..
ubuntu@ip-172-31-21-202:~/home$ cd ..
ubuntu@ip-172-31-21-202:~/$ sudo su
root@ip-172-31-21-202:/# chown -R ubuntu:ubuntu mnt
root@ip-172-31-21-202:/#
```

### 3 IMPLEMENTING SHARED-MEMORY SORT FOR 10GB

---

In this task I created a multithreaded Java program that sorts a 10GB file. I have printed the output into a file based on the varying number of threads.

#### Program Logic:

- The main computation in my program is done by two functions namely, “`readAndPartition`” and “`mergeTempFiles`”
- The “`readAndPartition`” function reads the raw input file line by line and creates a arraylist in the memory. I allow this arraylist to grow for 20000 lines. Once the arraylist size reaches 20000, I pass the arraylist to “`mySortClass`” which extends from thread. This class sorts the arraylist in parallel. So if I have 2 threads as input then each thread gets 10000 records from arraylist, sorts it and writes it into a temporary file. Thus, after completion of this function, I get small sorted chuck of files. This function returns the total number of intermediate files that gets created.
- The “`mergeTempFiles`” function takes the total number of intermediate files, the temporary directory path and final output file name. This function, based on the total number of intermediate files, creates an array of BufferedReader objects for each file. Also, I create a TreeMap datastructure which reads and stores the first line of all the BufferedReader objects created from the intermediate files. The advantage of putting the first line of all intermediate files into TreeMap is that it internally sorts them while storing. So the first element of the TreeMap would be the smallest line among all intermediate files. I remove this element from the tree and write it into the output file. I read the next line of the particular file that got removed and add it up in the TreeMap object.
- At the end of “`mergeTempFiles`” function I have a new output file that is in sorted order. This function after creating the output file deletes all the intermediate files similar to Map Reduce.

#### Program Flow:

1. Install gensort to generate the input file of 10 GB using:

```
>>wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz  
>>tar -xvf gensort-linux-1.5.tar.gz
```

2. Create a 10GB input file using gensort program.

```
>> cd 64/  
>>./gensort -a 1000000000 [inputfileName]
```

3. I have hard coded the input file, temporary file directory and output file name in my program.

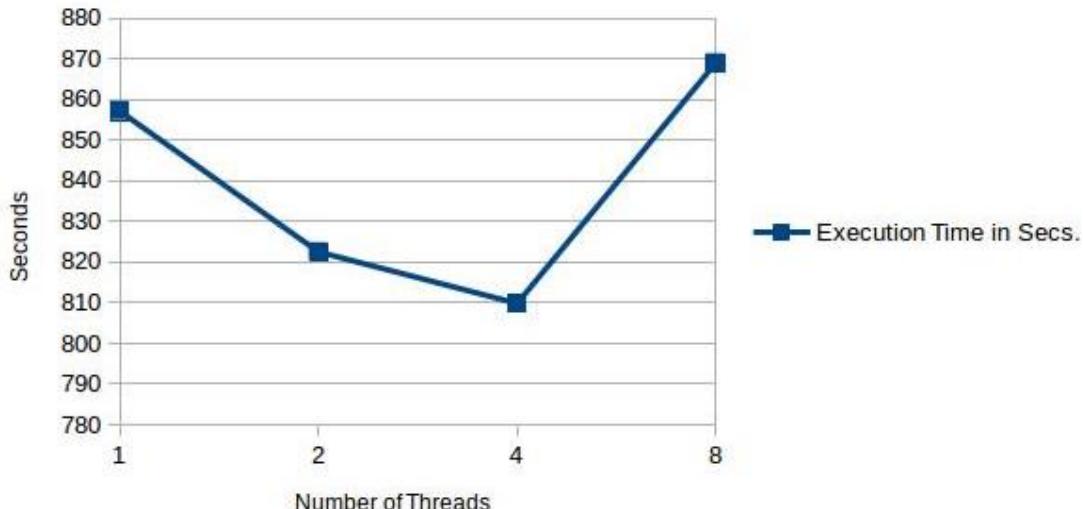
4. I take the number of threads as a command line argument for the program.

5. First I call a user defined function named "`readAndPartition`" which takes source file name, temporary files directory path and number of threads as input.
6. The "`readAndPartition`" function returns the total number of intermediate files that got created by this function.
7. I then call "`mergeTempFiles`" function which takes the total count of intermediate files, temporary files directory path and output file name, merges all the intermediate files present in the temporary file folder and creates a fills in data into the output file.
8. Once the output file is created completely by "`mergeTempFiles`" function, it deletes all the intermediate files.
9. After the program has we get an output file which is in unix format (ASCII text). We need to convert this file into DOS format (ASCII text, with CRLF line terminators) before passing to `valsrt` for final checkup.
10. The command to convert from Unix to Dos format is as below:  
`>>unix2dos <outputFileName>`
11. Later we validate the correctness of our output file by running the `valsrt` program.  
`>>./valsrt <outputFileName>`

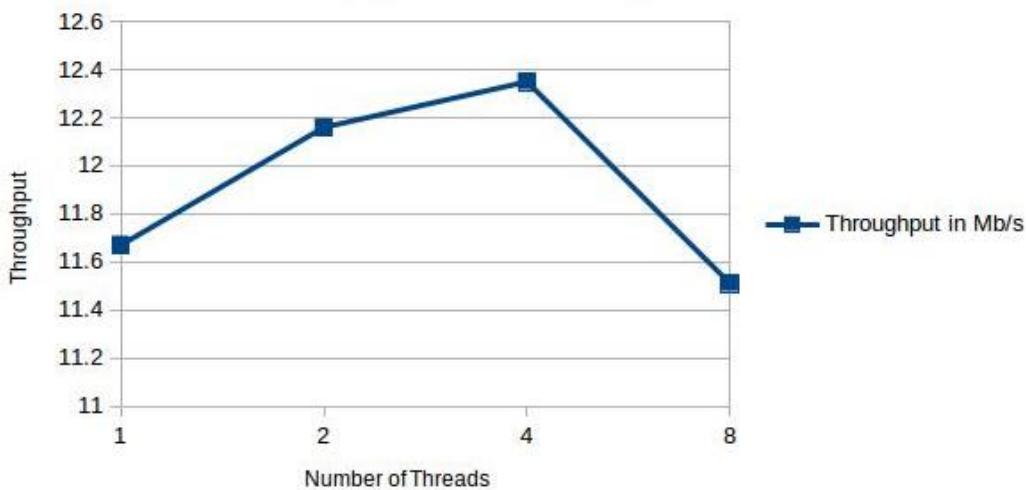
#### Outputs (10 GB Input File):

<i>No. of Threads</i>	<i>Execution Time in Secs.</i>	<i>Throughput in Mb/s</i>	<i>Speed up (w.r.t 1thread throughput)</i>
1	857.141	11.67	1
2	822.525	12.16	1.042
4	809.854	12.35	1.058
8	869.015	11.51	0.986

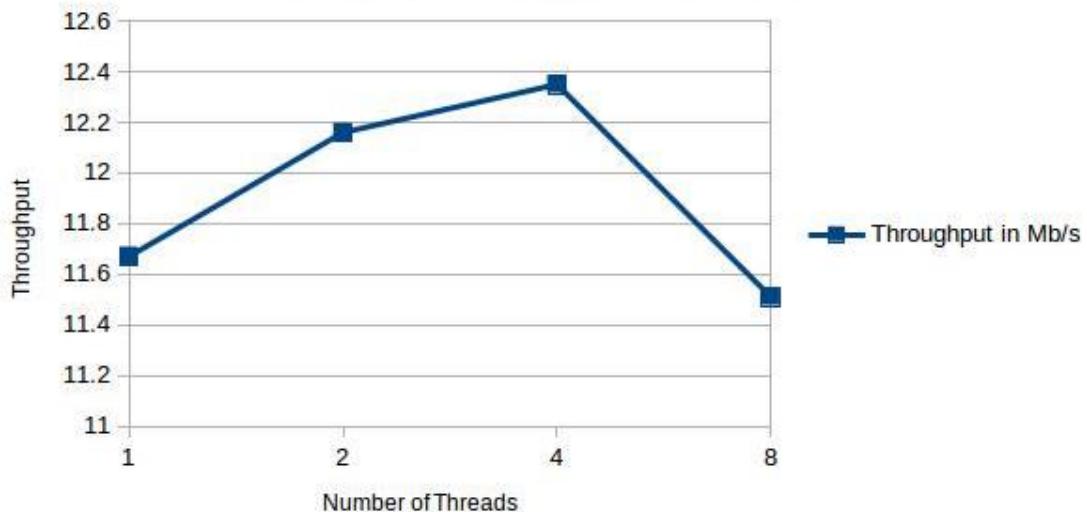
Sort Execution Time vs Number of Threads Graph



Sort Throughput vs Number of Thread Graph



Sort Throughput vs Number of Thread Graph



#### **Result Interpretation:**

The program performs optimally upto 4 threads as visible through the plotted graphs and tables. The programs performance drops drastically with 8 threads. As per the program logic, with the increase in number of threads there is an increase in the number of intermediate file of smaller size. This results in increases the overhead of read and write which adversely affects the programs performance as disk read writes are resource intensive.

#### **Formula:**

$$\text{Throughput} = \frac{\text{Sorting File Size (in Mb's)}}{\text{Time Taken to sort (in Secs)}}$$

## Screenshots for Shared-Memory Sort for 1 Thread:

```
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# java sortFile_multithreaded 1 >> outputlog.txt

64 gensorc-linux-1.5.tar.gz mySortClass.class outputlog.txt sortFile_multithreaded.class sortFile_multithreaded.java
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# cat outputlog.txt
1Threads Time taken = 857.141 seconds
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# cd /mnt/raid
root@ip-172-31-21-202:/mnt/raid# ls -l
total 19433784
-rwxr-xr-x 1 root root 10000000000 Mar 26 22:43 input
drwx----- 2 ubuntu ubuntu 16384 Mar 26 22:28 lost+found
-rw-r--r-- 1 root root 9900000000 Mar 26 23:59 output
drwxrwxr-x 2 ubuntu ubuntu 163840 Mar 26 23:59 tmp
root@ip-172-31-21-202:/mnt/raid# file output
output: ASCII text
root@ip-172-31-21-202:/mnt/raid# unix2dos output

unix2dos: converting file output to DOS format ...
root@ip-172-31-21-202:/mnt/raid#
root@ip-172-31-21-202:/mnt/raid#
root@ip-172-31-21-202:/mnt/raid# ls -l
total 19531440
-rwxr-xr-x 1 root root 10000000000 Mar 26 22:43 input
drwx----- 2 ubuntu ubuntu 16384 Mar 26 22:28 lost+found
-rw-r--r-- 1 root root 10000000000 Mar 27 00:09 output
drwxrwxr-x 2 ubuntu ubuntu 163840 Mar 26 23:59 tmp
root@ip-172-31-21-202:/mnt/raid# cd /home/ubuntu/sharedMemory/
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# cd 64
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64# ls
file1 gensorc valsrt
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64# ./valsrt /mnt/raid/output
Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64#
```

## Screenshots for Shared-Memory Sort for 2 Thread:

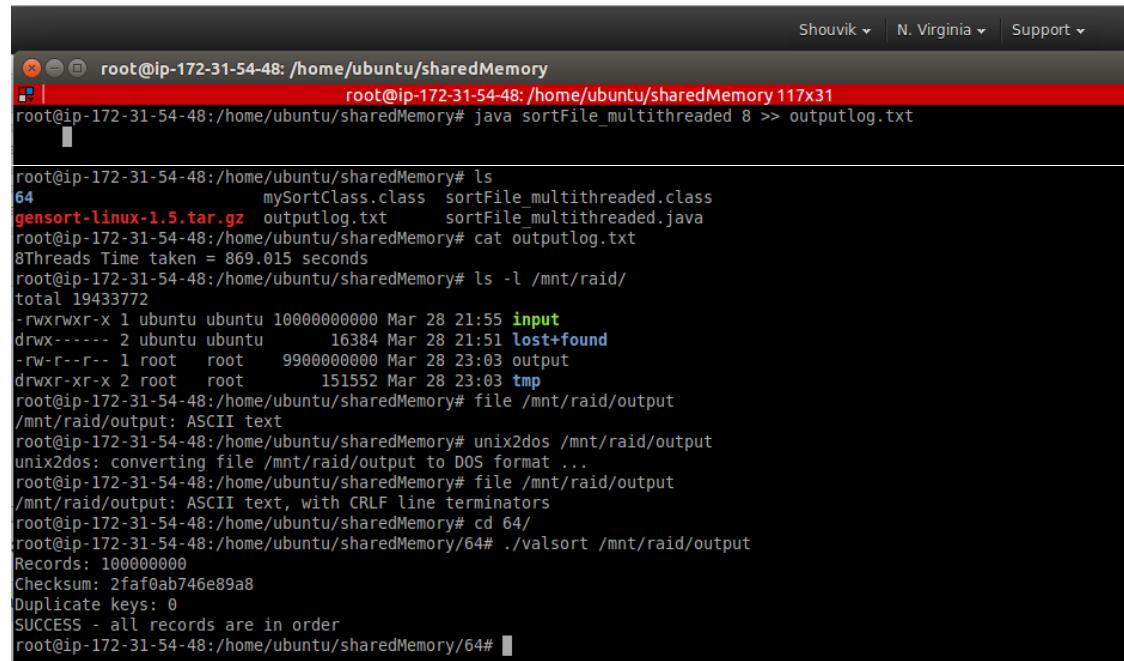
```
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# java sortFile_multithreaded 2 >> outputlog.txt

^[[C
root@ip-172-31-21-202:/home/ubuntu/sharedMemory#
root@ip-172-31-21-202:/home/ubuntu/sharedMemory#
root@ip-172-31-21-202:/home/ubuntu/sharedMemory#
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# ls
64 gensorc-linux-1.5.tar.gz mySortClass.class outputlog.txt sortFile_multithreaded.class sortFile_multithreaded.java
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# cat outputlog.txt
1Threads Time taken = 857.141 seconds
2Threads Time taken = 822.525 seconds
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# cd /mnt/raid/
root@ip-172-31-21-202:/mnt/raid# ls -l
total 19433784
-rwxr-xr-x 1 root root 10000000000 Mar 26 22:43 input
drwx----- 2 ubuntu ubuntu 16384 Mar 26 22:28 lost+found
-rw-r--r-- 1 root root 9900000000 Mar 27 00:40 output
drwxrwxr-x 2 ubuntu ubuntu 163840 Mar 27 00:40 tmp
root@ip-172-31-21-202:/mnt/raid# file output
output: ASCII text
root@ip-172-31-21-202:/mnt/raid# unix2dos output
unix2dos: converting file output to DOS format ...
root@ip-172-31-21-202:/mnt/raid# file output
output: ASCII text, with CRLF line terminators
root@ip-172-31-21-202:/mnt/raid# cd /home/ubuntu/sharedMemory/64/
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64# ./valsrt /mnt/raid/output
^[[CRecords: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64#
```

### Screenshots for Shared-Memory Sort for 4 Thread:

```
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# java sortFile_multithreaded 4 >> outputlog.txt
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# cat outputlog.txt
1Threads Time taken = 857.141 seconds
2Threads Time taken = 822.525 seconds
4Threads Time taken = 809.854 seconds
root@ip-172-31-21-202:/home/ubuntu/sharedMemory# cd /mnt/raid/
root@ip-172-31-21-202:/mnt/raid# ls-l
ls: command not found
root@ip-172-31-21-202:/mnt/raid# ls -l
total 19433784
-rwxr-xr-x 1 root root 10000000000 Mar 26 22:43 input
drwx----- 2 ubuntu ubuntu 16384 Mar 26 22:28 lost+found
-rw-r--r-- 1 root root 9900000000 Mar 27 01:11 output
drwxrwxr-x 2 ubuntu ubuntu 163840 Mar 27 01:11 tmp
root@ip-172-31-21-202:/mnt/raid# file output
output: ASCII text
root@ip-172-31-21-202:/mnt/raid# unix2dos output
unix2dos: converting file output to DOS format ...
root@ip-172-31-21-202:/mnt/raid# file output
output: ASCII text, with CRLF line terminators
root@ip-172-31-21-202:/mnt/raid# ls -l
total 19531440
-rwxr-xr-x 1 root root 10000000000 Mar 26 22:43 input
drwx----- 2 ubuntu ubuntu 16384 Mar 26 22:28 lost+found
-rw-r--r-- 1 root root 10000000000 Mar 27 01:17 output
drwxrwxr-x 2 ubuntu ubuntu 163840 Mar 27 01:11 tmp
root@ip-172-31-21-202:/mnt/raid# cd /home/ubuntu/sharedMemory/64/
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64# ls
file1 gensort valsrt
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64# ./valsrt /mnt/raid/output
Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-21-202:/home/ubuntu/sharedMemory/64#
```

### Screenshots for Shared-Memory Sort for 8 Thread:



The screenshot shows a terminal window with the following session:

```
Shouvik | N. Virginia | Support
root@ip-172-31-54-48:/home/ubuntu/sharedMemory
root@ip-172-31-54-48:/home/ubuntu/sharedMemory 117x31
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# java sortFile_multithreaded 8 >> outputlog.txt
[Output Log]
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# ls
64 mySortClass.class sortFile_multithreaded.class
gensort-linux-1.5.tar.gz outputlog.txt sortFile_multithreaded.java
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# cat outputlog.txt
8Threads Time taken = 869.015 seconds
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# ls -l /mnt/raid/
total 19433772
-rwxrwxr-x 1 ubuntu ubuntu 10000000000 Mar 28 21:55 input
drwx----- 2 ubuntu ubuntu 16384 Mar 28 21:51 lost+found
-rw-r--r-- 1 root root 9900000000 Mar 28 23:03 output
drwxr-xr-x 2 root root 151552 Mar 28 23:03 tmp
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# file /mnt/raid/output
/mnt/raid/output: ASCII text
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# unix2dos /mnt/raid/output
unix2dos: converting file /mnt/raid/output to DOS format ...
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# file /mnt/raid/output
/mnt/raid/output: ASCII text, with CRLF line terminators
root@ip-172-31-54-48:/home/ubuntu/sharedMemory# cd 64/
root@ip-172-31-54-48:/home/ubuntu/sharedMemory/64# ./valsrt /mnt/raid/output
Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-54-48:/home/ubuntu/sharedMemory/64#
```

## Screenshots for top 10 and bottom 10 lines of sorted output file:

```
root@ip-172-31-21-202:/mnt/raid# head -10 output
"0!uve 000000000000000000000000000000001228D4 77778888000022224444DDDDDDDEEEE00000000CCC7777DDDD
PMd32= 000000000000000000000000000000003440CC1 FFFFEEEE6666CCCCBBBBB999933335555DDDDDD77778886666
^3C0], 00000000000000000000000000000000158C5C5 5555AAA9999EEE888822229999CCCCDDDD666655554442222
!&S3/][] 000000000000000000000000000000002145D78 8888BBBBDDDD1111CCCC55556666BBBB1111EEEEDDD22229999
!,=U#,9 0000000000000000000000000000000019072E3 33332222FFFFBBBBB0000FFFFAAAA666655553333DDDD3333CCCC
!0f[ITd 000000000000000000000000000000003CAAB48 9999FFFF555533337777CCC4444BBBB7777EEEEBBBBDDDD4444
!f6Suy2 000000000000000000000000000000003ABFD84 EEEE5555556666AAAA5555BBBBDDDD0000111166660000DDDD
#%NIpq. 000000000000000000000000000000003B36FB9 111100003333444411116666666AAAAAAA00001111CCCCEEEE
#`'cl'~ 000000000000000000000000000000002EDC5C8 8888AAA11114444FFFF77773333EEE44440000FFFF99999999
$"-Q)] 000000000000000000000000000000005F1265D CCCC6666EEEE222200000DDDAAA88886666BBBB00006666AAAA
root@ip-172-31-21-202:/mnt/raid# tail -10 output
~~~uq2k#=U 000000000000000000000000000000002C06745 99991111DDDD222211110000FFFFEEEEFFFF33337777CCCC2222
~~~v/0&Qnm 000000000000000000000000000000004709701 CCCC88883333FFFF00000000009991111FFFF77774446666
~~~yK0l:gE 000000000000000000000000000000002048B4F CCCC11114444888822226666BBBB888855557777EEEEBBBB0000
~~~yK^H.il 00000000000000000000000000000000463D004 44440000FFFF3333999944447777DDDFFFAAAA11118888DDDD
~~~yL;C'XE 00000000000000000000000000000000580D211 2222EEEE3333000022221111CCCCFFF55557774444BBBB6666
~~~zbA_Tt 0000000000000000000000000000000007F9F4F BBBBCCCC666655559999FFFF8888AAAA11116666AAABBBB0000
~~~zeo^FEg 000000000000000000000000000000001E06130 4444CCCCBBBB999922228888555888CCCCFFFF00001111111
~~~}GxjWHI 00000000000000000000000000000000CA1345 777711118888AAAAAAA22221111BBBB00002222BBBBCCCC2222
~~~}P;]g0g 0000000000000000000000000000000040DA3E4 4444FFFF444466663333EEE88888888DDDEEEE44442222DDDD
~~~}kU|K<p 00000000000000000000000000000005E4A0AA 0000666655551111BBBB88889999AAAA55550000333355557777
root@ip-172-31-21-202:/mnt/raid#
```

## 4 INSTALL AND CONFIGURE HADOOP

---

### Steps to install Hadoop:

1. Start EC2 C3.Large instance and log into it.

2. Update all the installed applications using

```
>> sudo apt-get update
```

3. Install Java in the instance using

```
>>sudo apt-get install openjdk-7-jdk
```

4. Download and Extract hadoop packages by using below command

```
>>wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.2/hadoop-2.7.2.tar.gz
```

5. Extract hadoop packages from the tar and rename the folder using

```
>>tar -xvf hadoop-2.7.2.tar.gz
```

```
>>mv hadoop-2.7.2 hadoop
```

6. Copy .pem file from your local filesystem into the instance using

```
>> scp -i "hadoop.pem" hadoop.pem ubuntu@ec2-52-87-176-251.compute-1.amazonaws.com:~/
```

7. Make pem file available to ssh using

```
>> eval `ssh-agent -s`  
>>ssh-add hadoop.pem
```

8. Update .bashrc file to include hadoop variables as below:

```
>>vi .bashrc
```

(Add below variables at the end of .bashrc file)

```
# 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-1.7.0.99.x86_64/jre
```

```
export PATH=$PATH:$JAVA_HOME/bin
```

```
export HADOOP_CLASSPATH=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.99.x86_64/lib/tools.jar
```

```
# Add Hadoop bin/ directory to PATH
```

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

(Save .bashrc file)

```
source .bashrc
```

- 9.** Update hadoop-env.sh file inside /hadoop/etc/hadoop/ as below:

```
>> vi /hadoop/etc/hadoop/hadoop-env.sh  
(update the Java Home path)  
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.99.x86_64/jre  
(Save hadoop-env.sh)
```

- 10.** Add master nodes private ip in etc/hosts file using

```
>> sudo vi /etc/hosts  
(Add private IP of master)  
172.31.8.62 master  
(Save the hosts file)
```

- 11.** Mount and join unmounted disk partition and name it /mnt/raid (See Section2: Setting Virtual Cluster for instructions on how to mount disk)

- 12.** Create a hadoop-ec2-user file in the mounted partition using below command:

```
>> mkdir /mnt/raid/temp/hadoop-ec2-user
```

- 13.** Add configurations in core-site.xml file using below commands:

```
>>vi hadoop/etc/hadoop/core-site.xml  
(Add below entries between the configuration tags)  
<property>  
    <name>fs.default.name</name>  
    <value>hdfs://master:9000</value>  
</property>  
<property>  
    <name>hadoop.tmp.dir</name>  
    <value>/mnt/raid/temp/Hadoop-ec2-user</value>  
</property>  
(Save the core-site.xml file)
```

- 14.** Create a HDFS directory in the mounted disk and then create Namenode and Datanode directories inside HDFS directory using below command:

```
>>mkdir -p /mnt/raid/hdfs  
>>mkdir -p /mnt/raid/hdfs/namenode  
>>mkdir -p /mnt/raid/hdfs/datanode
```

- 15.** Add configurations in hdfs-site.xml file using below commands:

```
>>vi hadoop/etc/hadoop/hdfs-site.xml  
(Add below entries between the configuration tags)  
<property>  
    <name>dfs.replication</name>  
    <value>1</value>  
</property>  
<property>
```

```

<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/mnt/raid/hdfs/namenode</value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
    <value>>/mnt/raid/hdfs/datanode</value>
</property>
(Save the hdfs-site.xml file)

```

- 16.** Create a mapred-site.xml file using below command:

```
>>cp mapred-site.xml.template mapred-site.xml
```

- 17.** Add configurations in mapred-site.xml file using below commands:

```

>>vi hadoop/etc/hadoop/mapred-site.xml
(Add below entries between the configuration tags)
<property>
    <name>mapreduce.jobtracker.address</name>
    <value>hdfs://master:8021</value>
</property>
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
<property>
    <name>mapreduce.job.maps</name>
    <value>2</value>
</property>
<property>
    <name>mapreduce.job.reduces</name>
    <value>2</value>
</property>
(Save the mapred-site.xml file)

```

- 18.** Add configurations in yarn-site.xml file using below commands:

```

>>vi hadoop/etc/hadoop/yarn-site.xml
(Add below entries between the configuration tags)
<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>
(Save the yarn-site.xml file)

```

**19.** Add localhost in hadoop/etc/hadoop/slaves file as below:

```

>> vi hadoop/etc/hadoop/slaves
Add localhost within the slave file and save it (This would be changed when we would
create master node out of this AMI)

```

**20.** For password less login generate Keygen using below commands:

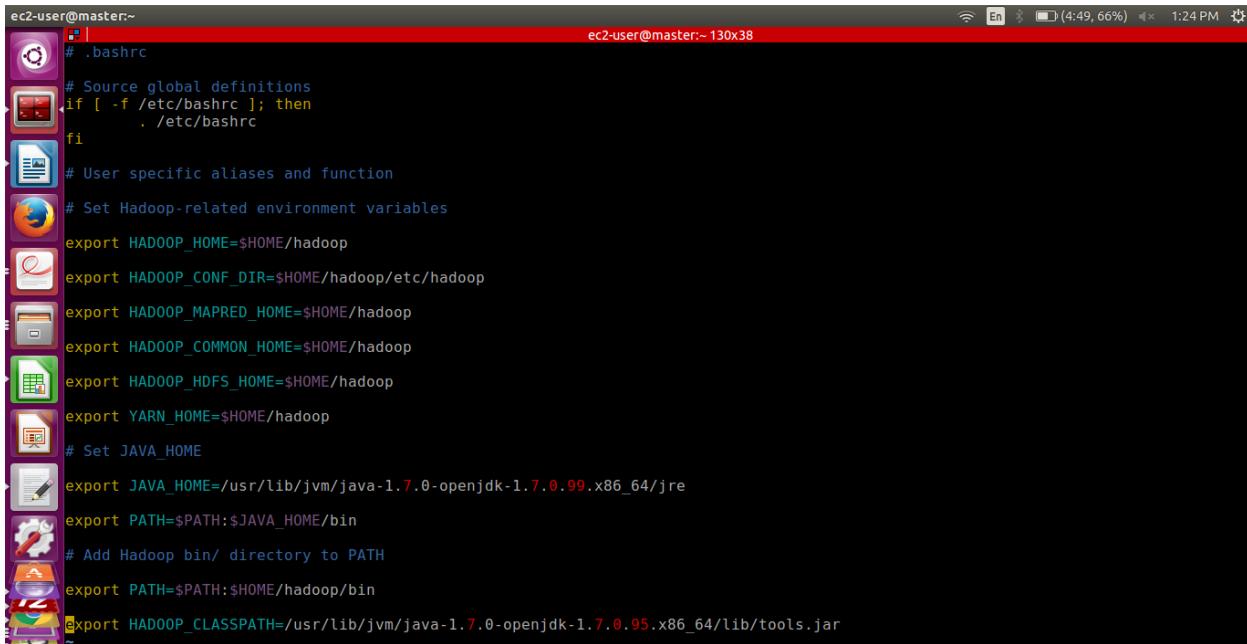
```

>> ssh-keygen -f ~/.ssh/id_rsa -t rsa -P ""
>> cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
>> cat ~/.ssh/id_rsa.pub | ssh slave1 'cat >> ~/.ssh/authorized_keys'

```

**21.** Create an AMI of the instance so that it can be used to create either a 1 node cluster or 16 node cluster.

## Screenshots to configure Hadoop:



```
ec2-user@master:~$ # .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

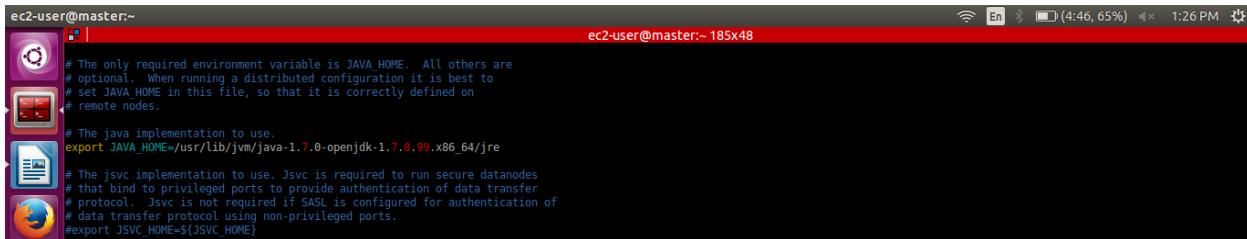
# User specific aliases and functions

# 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-1.7.0.99.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
```

Screen-shot showing variables to be added in .bashrc file for Hadoop

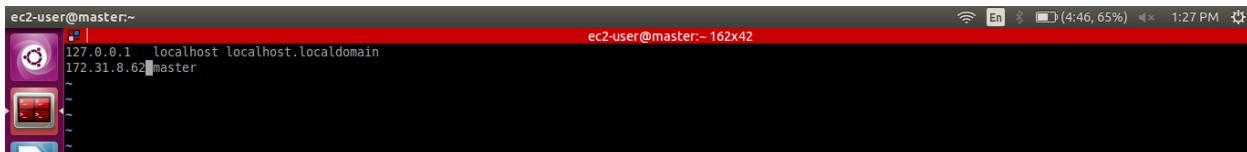


```
ec2-user@master:~$ # The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.99.x86_64/jre

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol. Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}
```

Screen-shot showing Java Home variable set in hadoop-env.sh file for Hadoop



```
ec2-user@master:~$ 127.0.0.1 localhost.localdomain
172.31.8.62 master
```

Screen-shot showing Master alias set in etc/hosts file for Hadoop

```
ec2-user@master:~ ec2-user@master:~ 130x38
[ ] | 
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration SYSTEM "configuration.dtd">
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://master:9000</value>
</property>
<property>
<name>dfs.tmp.dir</name>
<value>/mnt/raid/tmp/hadoop-ec2-user</value>
</property>
</configuration>
~
```

Screen-shot showing changes to be made in core-site.xml file for Hadoop

```
ec2-user@master:~ ec2-user@master:~ 162x42
[ ] | 
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration SYSTEM "configuration.dtd">
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>/mnt/raid/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/mnt/raid/hdfs/datanode</value>
</property>
</configuration>
~ "hadoop/etc/hadoop/hdfs-site.xml" 40L, 1116C
```

Screen-shot showing changes to be made in hdfs-site.xml file for Hadoop

```
ec2-user@master:~          ec2-user@master:~ 130x38

<?xml version="1.0"?>
<xm...-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>mapreduce.jobtracker.address</name>
<value>hdfs://master:8021</value>
</property>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.job.maps</name>
<value>2</value>
</property>
<property>
<name>mapreduce.job.reduces</name>
<value>2</value>
</property>
</configuration>
".hadoop/etc/hadoop/mapred-site.xml" 37L, 1090C
```

Screen-shot showing changes to be made in mapred-site.xml file for Hadoop

```
ec2-user@master:~          ec2-user@master:~ 162x42

<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Site specific YARN configuration properties -->
<configuration>
<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:8033</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>
".hadoop/etc/hadoop/yarn-site.xml" 42L, 1384C
```

Screen-shot showing changes to be made in yarn-site.xml file for Hadoop

## 5 IMPLEMENTING 10 AND 100 GB SORT IN HADOOP

---

### Steps to start 1 node and perform 10GB Sort in Hadoop:

1. Start an instance using the Hadoop AMI which we created in above “Configuring Hadoop” section.
2. Make separate config file in ssh folder using below command

```
>> cd ~/.ssh  
>>sudo vi config  
(Inside config file enter Hostname, Public DNS and identity of .pem file)  
(Save the config file)
```

3. Add private IPs of master and slave in etc/hosts using below command:

```
>> sudo vi /etc/hosts  
(Add private IP of master & slave in master node)  
172.31.51.163 master  
172.31.20.146 slave  
  
(Add private IP of master in slave node)  
172.31.51.163 master  
(Save the hosts file)
```

4. Format the namenode from the master node using below command:

```
>>hadoop namenode -format
```

5. Once the namenode has been formatted we can start Hadoop services using below command:

```
>> cd hadoop/sbin  
>> ./start-all.sh  
>>./yarn-deamon.sh start nodemanager
```

6. Check whether Hadoop services have started using below command:

```
>>jps
```

7. Now we need to run our Hadoop sort program.

8. Install gensort to generate the input file of 10 GB using:

```
>>wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz  
>>tar -xvf gensort-linux-1.5.tar.gz
```

9. Check disk space using below command:

```
>>lsblk
```

10. Generate input file of 10GB size using below command:

```
>> cd 64/
```

```
>>./gensort -a 1000000000 [inputfileName]
```

**11.** We now need to create an input directory within Hadoop file system using below command:

```
>> hadoop fs -mkdir /user  
>>hadoop fs -mkdir /user/shouvik  
>>hadoop fs -mkdir /user/shouvik/input
```

**12.** Now we need to put input file inside hadoop file system using below command:

```
>>hadoop fs -put [inputFileName] /user/shouvik/input/input10gb
```

**13.** You can confirm that input file has been placed within the Hadoop file system using below command:

```
>>hadoop fs -ls /user/shouvik/input/
```

**14.** Compile the source java file using below command:

```
>>hadoop com.sun.tools.javac.Main TeraSort_hadoop.java
```

**15.** After compilation is done we need to create a jar file using below command:

```
>>jar -cf ts.jar TeraSort_hadoop*.class
```

**16.** To run the jar file use below command:

```
>> hadoop jar ts.jar TeraSort_hadoop /user/shouvik/input /user/shouvik/output
```

**17.** Output file get generated in /user/shouvik/output directory. To check use below command:

```
>>hadoop fs -ls /user/shouvik/ts/output/
```

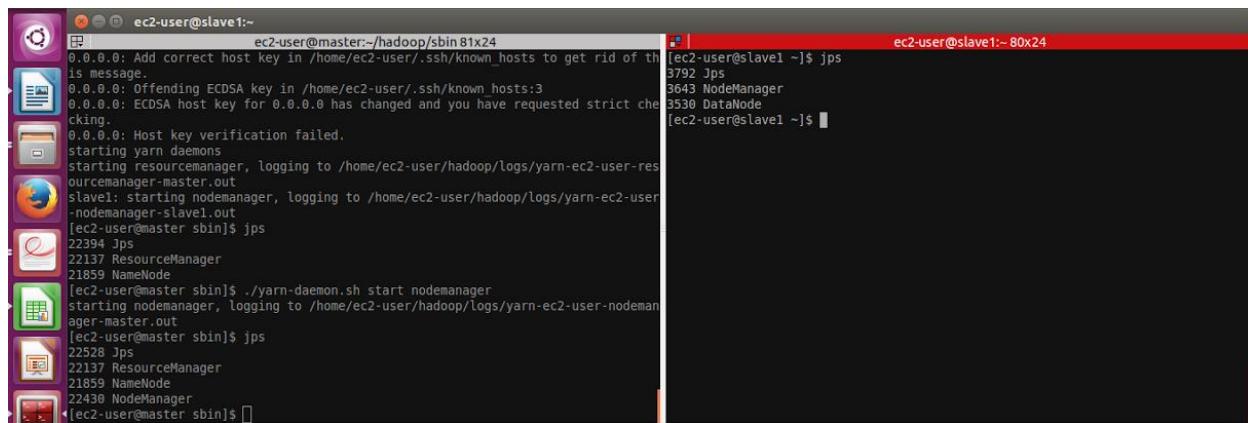
**18.** Copy the generated output file to your local instance disk using below command:

```
>> hadoop fs -get /user/shouvik/output/part-r-00000 /mnt/raid
```

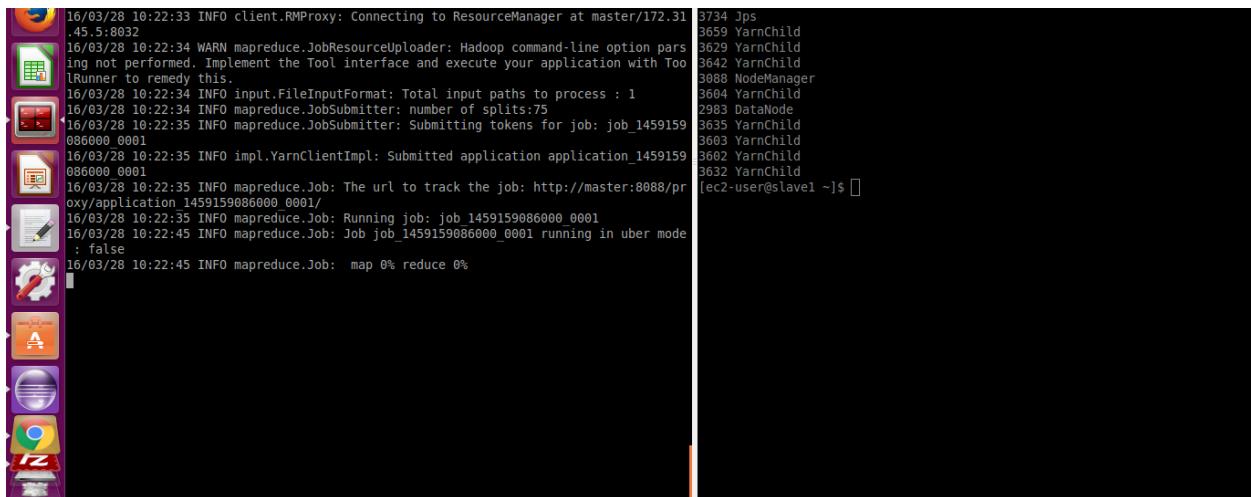
**19.** You can validate the output file using valsor using below command:

```
>> ./valsor /mnt/raid/part-r-00000
```

## Screenshots for 1 node cluster to perform 10GB sort in Hadoop:



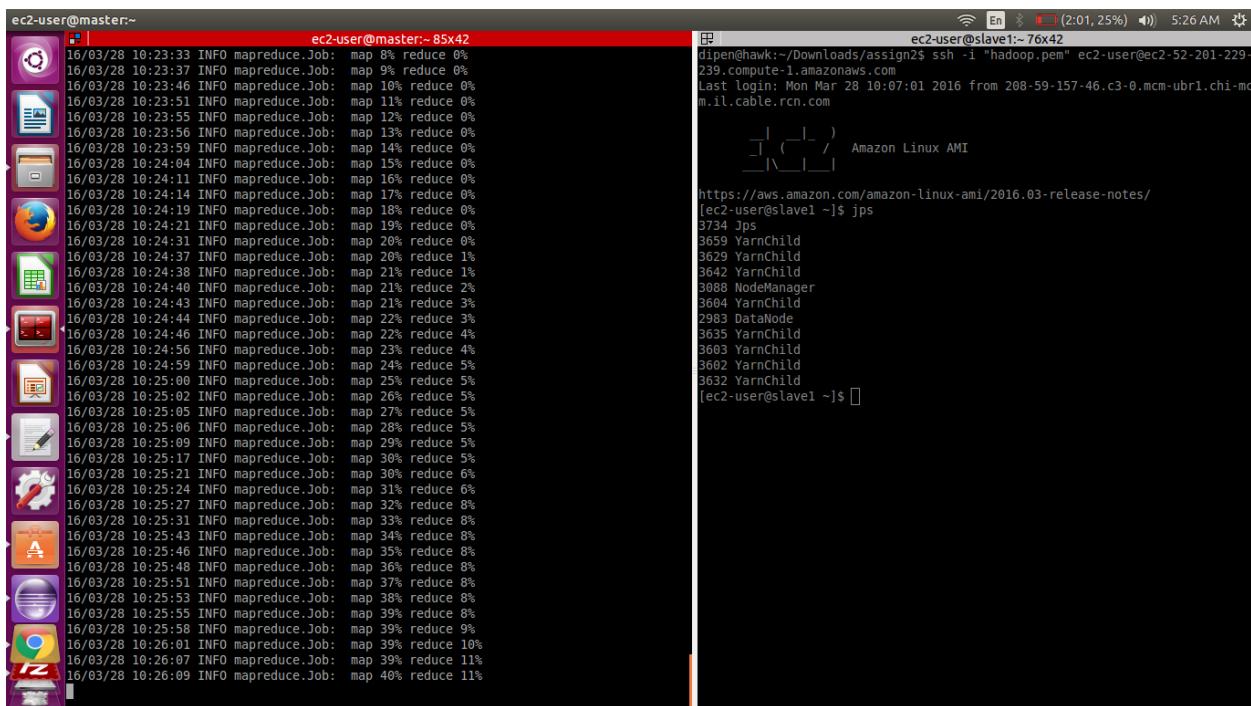
Screen-shot showing JPS command on master and slave for Hadoop



```
16/03/28 10:22:33 INFO client.RMProxy: Connecting to ResourceManager at master/172.31.45.5:8032
16/03/28 10:22:34 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/03/28 10:22:34 INFO input.FileInputFormat: Total input paths to process : 1
16/03/28 10:22:35 INFO mapreduce.JobSubmitter: number of splits:75
16/03/28 10:22:35 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459159086000_0001
16/03/28 10:22:35 INFO impl.YarnClientImpl: Submitted application application_1459159086000_0001
16/03/28 10:22:35 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1459159086000_0001/
16/03/28 10:22:35 INFO mapreduce.Job: Running job: job_1459159086000_0001
16/03/28 10:22:45 INFO mapreduce.Job: Job job_1459159086000_0001 running in uber mode : false
16/03/28 10:22:45 INFO mapreduce.Job: map 0% reduce 0%
```

```
3734 Jps
3659 YarnChild
3629 YarnChild
3642 YarnChild
3088 NodeManager
3604 YarnChild
2983 DataNode
3635 YarnChild
3603 YarnChild
3602 YarnChild
3632 YarnChild
```

Screen-shot showing start of Hadoop program and processes in Slave node



```
ec2-user@master:~ ec2-user@master:~ 85x42
16/03/28 10:23:33 INFO mapreduce.Job: map 8% reduce 0%
16/03/28 10:23:37 INFO mapreduce.Job: map 9% reduce 0%
16/03/28 10:23:46 INFO mapreduce.Job: map 10% reduce 0%
16/03/28 10:23:51 INFO mapreduce.Job: map 11% reduce 0%
16/03/28 10:23:55 INFO mapreduce.Job: map 12% reduce 0%
16/03/28 10:23:56 INFO mapreduce.Job: map 13% reduce 0%
16/03/28 10:23:59 INFO mapreduce.Job: map 14% reduce 0%
16/03/28 10:24:04 INFO mapreduce.Job: map 15% reduce 0%
16/03/28 10:24:11 INFO mapreduce.Job: map 16% reduce 0%
16/03/28 10:24:14 INFO mapreduce.Job: map 17% reduce 0%
16/03/28 10:24:19 INFO mapreduce.Job: map 18% reduce 0%
16/03/28 10:24:22 INFO mapreduce.Job: map 19% reduce 0%
16/03/28 10:24:31 INFO mapreduce.Job: map 20% reduce 0%
16/03/28 10:24:37 INFO mapreduce.Job: map 20% reduce 1%
16/03/28 10:24:38 INFO mapreduce.Job: map 21% reduce 1%
16/03/28 10:24:40 INFO mapreduce.Job: map 21% reduce 2%
16/03/28 10:24:43 INFO mapreduce.Job: map 21% reduce 3%
16/03/28 10:24:44 INFO mapreduce.Job: map 22% reduce 3%
16/03/28 10:24:46 INFO mapreduce.Job: map 22% reduce 4%
16/03/28 10:24:56 INFO mapreduce.Job: map 23% reduce 4%
16/03/28 10:24:59 INFO mapreduce.Job: map 24% reduce 5%
16/03/28 10:25:00 INFO mapreduce.Job: map 25% reduce 5%
16/03/28 10:25:02 INFO mapreduce.Job: map 26% reduce 5%
16/03/28 10:25:05 INFO mapreduce.Job: map 27% reduce 5%
16/03/28 10:25:06 INFO mapreduce.Job: map 28% reduce 5%
16/03/28 10:25:09 INFO mapreduce.Job: map 29% reduce 5%
16/03/28 10:25:17 INFO mapreduce.Job: map 30% reduce 5%
16/03/28 10:25:21 INFO mapreduce.Job: map 30% reduce 6%
16/03/28 10:25:24 INFO mapreduce.Job: map 31% reduce 6%
16/03/28 10:25:27 INFO mapreduce.Job: map 32% reduce 8%
16/03/28 10:25:31 INFO mapreduce.Job: map 33% reduce 8%
16/03/28 10:25:43 INFO mapreduce.Job: map 34% reduce 8%
16/03/28 10:25:46 INFO mapreduce.Job: map 35% reduce 8%
16/03/28 10:25:48 INFO mapreduce.Job: map 36% reduce 8%
16/03/28 10:25:51 INFO mapreduce.Job: map 37% reduce 8%
16/03/28 10:25:53 INFO mapreduce.Job: map 38% reduce 8%
16/03/28 10:25:55 INFO mapreduce.Job: map 39% reduce 8%
16/03/28 10:25:58 INFO mapreduce.Job: map 39% reduce 9%
16/03/28 10:26:01 INFO mapreduce.Job: map 39% reduce 10%
16/03/28 10:26:07 INFO mapreduce.Job: map 39% reduce 11%
16/03/28 10:26:09 INFO mapreduce.Job: map 40% reduce 11%
```

```
ec2-user@slave1:~ 76x42
dipen@hawk:~/Downloads/assign2$ ssh -i "hadoop.pem" ec2-user@ec2-52-201-229-239.compute-1.amazonaws.com
Last login: Mon Mar 28 10:07:01 2016 from 208-59-157-46.c3-0.mcm-ubr1.chi-mc.mil.cable.rcn.com
[ec2-user@slave1 ~]$ jps
3734 Jps
3659 YarnChild
3629 YarnChild
3642 YarnChild
3088 NodeManager
3604 YarnChild
2983 DataNode
3635 YarnChild
3603 YarnChild
3602 YarnChild
3632 YarnChild
```

Screen-shot showing running Hadoop program and processes in Slave node

```

ec2-user@master:~$ hdfs dfs -ls /user/hawkeye/114x60
16/03/28 10:36:18 INFO mapreduce.Job: Counters: 51
File System Counters
  FILE: Number of bytes read=29960214449
  FILE: Number of bytes written=40069222869
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=10000311654
  HDFS: Number of bytes written=9990000000
  HDFS: Number of read operations=228
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
Job Counters
  Killed map tasks=1
  Launched map tasks=76
  Failed reduce tasks=1
  Data locality tasks=45
  Rack-local map tasks=31
  Total time spent by all maps in occupied slots (ms)=6202082
  Total time spent by all reduces in occupied slots (ms)=726970
  Total time spent by all map tasks=6202082
  Total time spent by all reduce tasks=726970
  Total vcore-milliseconds taken by all map tasks=202082
  Total vcore-milliseconds taken by all reduce tasks=726970
  Total megabyte-milliseconds taken by all map tasks=6359931968
  Total megabyte-milliseconds taken by all reduce tasks=744417280
Map-Reduce Framework
  Map input records=100000000
  Map output records=100000000
  Map input bytes=10000000000
  Map output materialized bytes=10100000458
  Input split bytes=8559
  Combine input records=100000000
  Combine output records=100000000
  Reduce input bytes=1000000000
  Reduce shuffle bytes=10100000450
  Reduce input records=100000000
  Reduce output records=100000000
  Redundant splits=0
  Shuffled Maps =75
  Failed Shuffles=0
  Merged Map outputs=75
  GC time elapsed (ms)=20022
  Total time spent (ms)=1602430
  Physical memory (bytes) snapshot=19410812928
  Virtual memory (bytes) snapshot=4933293956
  Total committed heap usage (bytes)=14615576576
Shuffle
  BAD Ids=0
  CONNECTION=0
  IO_ERROR=0
  MAP_FAILED=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=100000000000
File Output Format Counters
  Bytes Written=99900000000
[ec2-user@master ~]$ 

```

Screen-shot showing completion of Hadoop program and processes in Slave node

```

[ec2-user@master ~]$ cd 64
[ec2-user@master 64]$ ls
gensort input10GB.txt op1.txt op.txt _SUCCESS valsrt
[ec2-user@master 64]$ unix2dos /m/
media/ mnt/
[ec2-user@master 64]$ unix2dos /mnt/raid/
hdfs/   input10GB.txt lost+found/  part-r-00000  temp/
[ec2-user@master 64]$ unix2dos /mnt/raid/part-r-00000
unix2dos: converting file /mnt/raid/part-r-00000 to DOS format ...
[ec2-user@master 64]$ ./valsrt /mnt/raid/part-r-00000
  Records: 100000000
Checksum: 2faef2a1dfa8909
Duplicate Keys: 0
SUCCESS - all records are in order
[ec2-user@master 64]$ 

```

Screen-shot showing valsrt result for 10gb sort of Hadoop program.

## Steps to start 16 nodes and perform 100GB Sort in Hadoop:

1. Start 1 master and 16 slaves instances using the Hadoop AMI which we created in above "Configuring Hadoop" section.
2. Make separate config file in ssh folder of master using below command

```

>> cd ~/.ssh
>>sudo vi config
(Inside config file for each slave enter Hostname, Public DNS and identity of .pem file)
(Save the config file)

```

3. Add private IPs of master and slave in etc/hosts using below command:

```

>> sudo vi /etc/hosts
(Add private IP of master & all slaves in master node)

```

Eg.172.31.51.163 master  
172.31.20.146 slave1  
172.31.45.20 slave2

(Add private IP of master in slave node)  
172.31.51.163 master  
(Save the hosts file)

4. Mount disk in master and all slaves. (For instructions on how to mount disk see “Steps to mount unmounted disk space and renaming it to /mnt/raid” topic in “Setting virtual cluster section”.)
5. Update the hadoop/etc/Hadoop/slaves using below command:

```
>> vi hadoop/etc/hadoop/slaves  
Add all slave aliases as defined in the etc/hosts file and save it.
```

6. Format the namenode from the master node using below command:

```
>>hadoop namenode -format
```

7. Once the namenode has been formatted we can start Hadoop services using below command:

```
>> cd hadoop/sbin  
>> ./start-all.sh  
>>./yarn-deamon.sh start nodemanager
```

8. Check whether Hadoop services have started using below command:

```
>>jps
```

9. Now we need to run our Hadoop sort program.

10. Install gensort to generate the input file of 100 GB using:

```
>>wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz  
>>tar -xvzf gensort-linux-1.5.tar.gz
```

11. Check disk space using below command:

```
>>lsblk
```

12. Generate input file of 100GB size using below command:

```
>> cd 64/  
>>./gensort -a 10000000000 [inputfileName]
```

13. We now need to create an input directory within Hadoop file system using below command:

```
>> hadoop fs -mkdir /user  
>>hadoop fs -mkdir /user/shouvik  
>>hadoop fs -mkdir /user/shouvik/input
```

14. Now we need to put input file inside hadoop file system using below command:

```
>>hadoop fs -put [inputFileName] /user/shouvik/input/input100gb
```

- 15.** You can confirm that input file has been placed within the Hadoop file system using below command:

```
>>hadoop fs -ls /user/shouvik/input/
```

- 16.** Compile the source java file using below command:

```
>>hadoop com.sun.tools.javac.Main TeraSort_hadoop.java
```

- 17.** After compilation is done we need to create a jar file using below command:

```
>>jar -cf ts.jar TeraSort_hadoop*.class
```

- 18.** To run the jar file use below command:

```
>> hadoop jar ts.jar TeraSort_hadoop /user/shouvik/input /user/shouvik/output
```

- 19.** Output file get generated in /user/shouvik/output directory. To check use below command:

```
>>hadoop fs -ls /user/shouvik/ts/output/
```

- 20.** Copy the generated output file to your local instance disk using below command:

```
>>hadoop fs -get /user/shouvik/output/part-r-00000 /mnt/raid
```

- 21.** You can validate the output file using valsor using below command:

```
>>./valsor /mnt/raid/part-r-00000
```

## Screenshots for 16 node cluster to perform 100GB sort in Hadoop:

The screenshot shows the AWS EC2 Management Console interface. On the left, there's a sidebar with various AWS services like EC2 Dashboard, Events, Tags, and Limits. The main area displays a table of 17 running instances. The columns include Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, and Public DNS. The 'Name' column lists entries such as 'Master', 'slave1', 'slave2', etc. The 'Instance ID' column lists unique identifiers for each instance. The 'Instance Type' column shows most are 'c3.large'. The 'Availability Zone' column shows they are all in 'us-east-1e'. The 'Instance State' column shows all are 'running'. The 'Status Checks' column shows '2/2 checks ...' for most, while one instance has '1/2 checks ...'. The 'Alarm Status' column shows 'None' for all. The 'Public DNS' column lists the public IP addresses for each instance. One instance, 'slave9', is highlighted with a blue selection bar. At the bottom of the table, it says 'Instance: i-1c7970e5 (slave9) Public DNS: ec2-52-23-244-222.compute-1.amazonaws.com'.

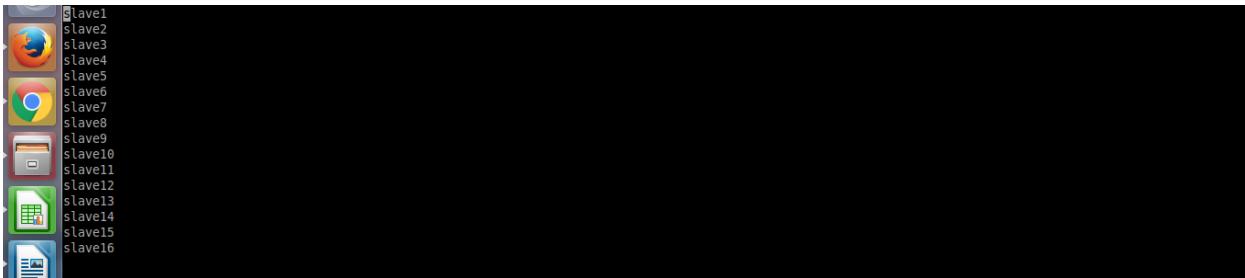
Screen-shot showing 1 master and 16 slaves running in AWS.

```

127.0.0.1 localhost
172.31.33.130 slave1
172.31.45.207 slave2
172.31.43.88 slave3
172.31.33.163 slave4
172.31.34.214 slave5
172.31.43.129 slave6
172.31.37.122 slave7
172.31.38.14 slave8
172.31.47.28 slave9
172.31.42.6 slave10
172.31.34.191 slave11
172.31.39.29 slave12
172.31.40.127 slave13
172.31.46.79 slave14
172.31.35.144 slave15
172.31.43.158 slave16

```

Screen-shot showing 1 master and 16 slaves IP added to /etc/hosts.



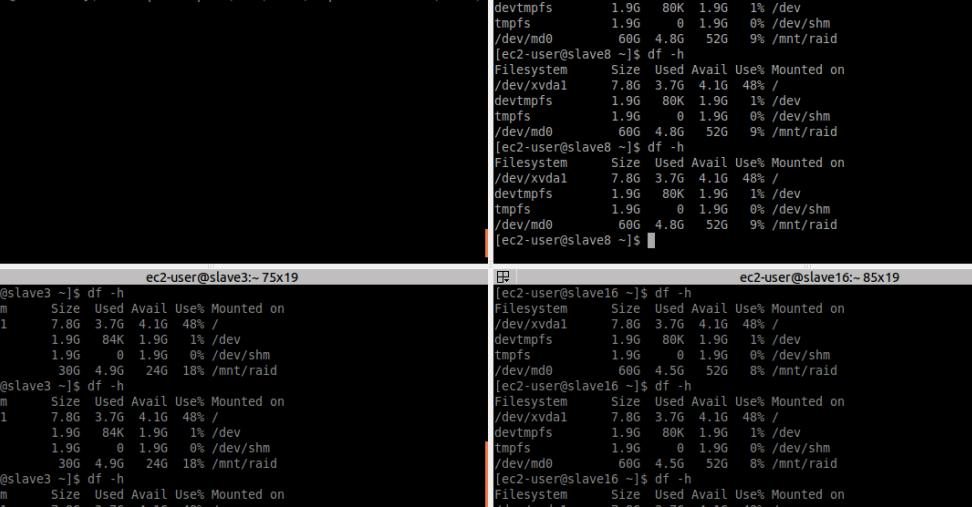
Screen-shot showing 16 slave aliases added in Hadoop/etc/Hadoop/slaves file.

```

ubuntu@ip-172-31-45-44:~/hadoop$ ./hadoop namenode -format
16/03/30 00:31:19 INFO blockmanagement.BlockManager: replicationRecheckInterval = 3000
16/03/30 00:31:19 INFO blockmanagement.BlockManager: encryptDataTransfer = false
16/03/30 00:31:19 INFO blockmanagement.BlockManager: maxNumBlocksToLog = 1600
16/03/30 00:31:19 INFO namenode.FSNamesystem: fsOwner = ubuntu (auth:SIMPLE)
16/03/30 00:31:19 INFO namenode.FSNamesystem: supergroup = supergroup
16/03/30 00:31:19 INFO namenode.FSNamesystem: isPermissionEnabled = false
16/03/30 00:31:19 INFO namenode.FSNamesystem: HA Enabled: false
16/03/30 00:31:19 INFO util.GSet: Computing capacity for map INodeMap
16/03/30 00:31:19 INFO util.GSet: VM type = 64-bit
16/03/30 00:31:19 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB
16/03/30 00:31:19 INFO util.GSet: capacity = 2^20 = 1048576 entries
16/03/30 00:31:19 INFO namenode.FSDirectory: ACLs enabled? false
16/03/30 00:31:19 INFO namenode.FSDirectory: XAttrs enabled? true
16/03/30 00:31:19 INFO namenode.FSDirectory: Maximum size of an xattr: 16384
16/03/30 00:31:19 INFO namenode.NameNode: Caching file names occurring more than 10 times
16/03/30 00:31:19 INFO util.GSet: Computing capacity for map cachedBlocks
16/03/30 00:31:19 INFO util.GSet: VM type = 64-bit
16/03/30 00:31:19 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB
16/03/30 00:31:19 INFO util.GSet: capacity = 2^18 = 262144 entries
16/03/30 00:31:19 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
16/03/30 00:31:19 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
16/03/30 00:31:19 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension = 30000
16/03/30 00:31:19 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.num.buckets = 10
16/03/30 00:31:19 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.num.users = 10
16/03/30 00:31:19 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
16/03/30 00:31:19 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
16/03/30 00:31:19 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
16/03/30 00:31:19 INFO util.GSet: Computing capacity for map NameNodeRetryCache
16/03/30 00:31:19 INFO util.GSet: VM type = 64-bit
16/03/30 00:31:19 INFO util.GSet: 0.02999999329447746% max memory 889 MB = 273.1 KB
16/03/30 00:31:19 INFO util.GSet: capacity = 2^15 = 32768 entries
16/03/30 00:31:19 INFO namenode.FSImage: Allocated new BlockPoolId: BP-1082662551-172.31.45.44-1459297879657
16/03/30 00:31:19 INFO common.Storage: Storage directory /home/ubuntu/hadoop/hdfs/namenode has been successfully formatted.
16/03/30 00:31:19 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
16/03/30 00:31:19 INFO util.ExitUtil: Exiting with status 0
16/03/30 00:31:19 INFO namenode.NameNode: SHUTDOWN_MSG:
*****SHUTDOWN MSG: Shutting down NameNode at ip-172-31-45-44/172.31.45.44*****
ubuntu@ip-172-31-45-44:~/hadoop$ 

```

Screen-shot showing namenode being formatted in Hadoop master node.



```
[ec2-user@master ~]$ hadoop fs -put /mnt/raid/
[ec2-user@master ~]$ hadoop fs -put /mnt/raid/input100GB.txt /user/
[ec2-user@slave8 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.8G  52G  9% /mnt/raid
[ec2-user@slave8 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.8G  52G  9% /mnt/raid
[ec2-user@slave8 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.8G  52G  9% /mnt/raid
[ec2-user@slave8 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.8G  52G  9% /mnt/raid
[ec2-user@slave8 ~]$ 
[ec2-user@slave3 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       300G  4.9G  24G  18% /mnt/raid
[ec2-user@slave3 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   84K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       300G  4.9G  24G  18% /mnt/raid
[ec2-user@slave3 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   84K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       300G  4.9G  24G  18% /mnt/raid
[ec2-user@slave3 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   84K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       300G  4.9G  24G  18% /mnt/raid
[ec2-user@slave3 ~]$ 
[ec2-user@slave16 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.5G  52G  8% /mnt/raid
[ec2-user@slave16 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.5G  52G  8% /mnt/raid
[ec2-user@slave16 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.5G  52G  8% /mnt/raid
[ec2-user@slave16 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.5G  52G  8% /mnt/raid
[ec2-user@slave16 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.8G  3.7G  4.1G  48% /
devtmpfs        1.9G   80K  1.9G  1% /dev
tmpfs          1.9G    0  1.9G  0% /dev/shm
/dev/md0       600G  4.5G  52G  8% /mnt/raid
[ec2-user@slave16 ~]$ 
```

Screen-shot showing 100gb input file being copied into HDFS file system.



The image shows a Ubuntu desktop environment with four terminal windows open, each displaying the output of the `jps` command on different Hadoop nodes:

- Master Node (Terminal 1):** ec2-user@master:~\$ jps  
21391 NodeManager  
22853 ResourceManager  
22499 NameNode  
23205 Jps
- Slave5 Node (Terminal 2):** ec2-user@slave5:~\$ jps  
6487 Jps  
6222 DataNode  
6338 NodeManager
- Slave10 Node (Terminal 3):** ec2-user@slave10:~\$ jps  
6406 Jps  
6257 NodeManager  
6141 DataNode
- Slave16 Node (Terminal 4):** ec2-user@slave16:~\$ jps  
6270 Jps  
6073 DataManager  
5957 DataNode

Screen-shot showing processes running in master and slaves after staring Hadoop using JPS command.

```
16/03/30 08:31:52 INFO client.RMProxy: Connecting to ResourceManager at master/172.31.8.62:8032
16/03/30 08:31:52 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/03/30 08:31:53 INFO input.FileInputFormat: Total input paths to process : 1
16/03/30 08:31:53 INFO mapreduce.JobSubmitter: number of splits:745
16/03/30 08:31:54 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459325552741_0001
16/03/30 08:31:54 INFO impl.YarnClientImpl: Submitted application application_1459325552741_0001
16/03/30 08:31:54 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1459325552741_0001/
16/03/30 08:31:54 INFO mapreduce.Job: Running job: job_1459325552741_0001
16/03/30 08:32:02 INFO mapreduce.Job: Job job_1459325552741_0001 running in uber mode : false
16/03/30 08:32:02 INFO mapreduce.Job: map 0% reduce 0%
16/03/30 08:32:55 INFO mapreduce.Job: map 8% reduce 0%
16/03/30 08:33:10 INFO mapreduce.Job: map 9% reduce 0%
16/03/30 08:33:12 INFO mapreduce.Job: map 10% reduce 0%
16/03/30 08:33:13 INFO mapreduce.Job: map 11% reduce 0%
16/03/30 08:33:25 INFO mapreduce.Job: map 12% reduce 0%
16/03/30 08:33:29 INFO mapreduce.Job: map 13% reduce 0%
16/03/30 08:33:33 INFO mapreduce.Job: map 14% reduce 0%
16/03/30 08:33:36 INFO mapreduce.Job: map 15% reduce 0%
16/03/30 08:33:38 INFO mapreduce.Job: map 16% reduce 0%
16/03/30 08:33:44 INFO mapreduce.Job: map 17% reduce 0%
```

Screen-shot showing start of Hadoop program

```

16/03/30 08:41:18 INFO mapreduce.Job: map 85% reduce 11%
16/03/30 08:41:23 INFO mapreduce.Job: map 86% reduce 11%
16/03/30 08:41:27 INFO mapreduce.Job: map 87% reduce 11%
16/03/30 08:41:39 INFO mapreduce.Job: map 88% reduce 12%
16/03/30 08:41:53 INFO mapreduce.Job: map 89% reduce 12%
16/03/30 08:42:01 INFO mapreduce.Job: map 90% reduce 13%
16/03/30 08:42:05 INFO mapreduce.Job: map 91% reduce 13%
16/03/30 08:42:10 INFO mapreduce.Job: map 92% reduce 13%
16/03/30 08:42:18 INFO mapreduce.Job: map 93% reduce 13%
16/03/30 08:42:26 INFO mapreduce.Job: map 94% reduce 13%
16/03/30 08:42:29 INFO mapreduce.Job: map 94% reduce 14%
16/03/30 08:42:34 INFO mapreduce.Job: map 95% reduce 14%
16/03/30 08:42:42 INFO mapreduce.Job: map 96% reduce 14%
16/03/30 08:42:50 INFO mapreduce.Job: map 97% reduce 14%
16/03/30 08:42:56 INFO mapreduce.Job: map 98% reduce 14%
16/03/30 08:43:02 INFO mapreduce.Job: map 98% reduce 15%
16/03/30 08:43:04 INFO mapreduce.Job: map 99% reduce 15%
16/03/30 08:43:22 INFO mapreduce.Job: map 100% reduce 15%
16/03/30 08:43:27 INFO mapreduce.Job: map 100% reduce 16%
16/03/30 08:43:54 INFO mapreduce.Job: map 100% reduce 17%
16/03/30 08:44:21 INFO mapreduce.Job: map 100% reduce 18%

```

Screen-shot showing running hadoop program and processes in Slave node

Started:	Wed Mar 30 08:12:21 UTC 2016
Version:	2.7.2, rb165c4fe8a74265c792ca23f546c54d04acf0e41
Compiled:	2016-01-26T00:08Z by jenkins from (detached from b165c4f)
Cluster ID:	CID-843331b1-0df4-b889-b3dd-d7623c6d97a7
Block Pool ID:	BP-2107450885-172.31.8.62-1459323779568

### Summary

Security is off.

Safenode is off.

26 files and directories, 754 blocks = 780 total filesystem objects(s).

Heap Memory used 110.27 MB of 448.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 33.22 MB of 33.38 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity:	802.91 GB
DFS Used:	93.88 GB (11.69%)
Non DFS Used:	370.2 GB
DFS Remaining:	338.83 GB (42.2%)
Block Pool Used:	93.88 GB (11.69%)
DataNodes usages% (Min/Median/Max/stdDev):	9.11% / 11.45% / 26.96% / 4.63%
Live Nodes	15 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	3/30/2016, 3:12:21 AM

Screen-shot showing summary page for Hadoop in browser.

```

0.0.0.0: IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
0.0.0.0: Someone could be eavesdropping on you right now (man-in-the-middle attack)!
0.0.0.0: It is also possible that a host key has just been changed.
0.0.0.0: The fingerprint for the ECDSA key sent by the remote host is
0.0.0.0: 90:38:53:a7:d4:f3:4b:2e:db:20:e1:6c:78:37:bc:9e.
0.0.0.0: Please contact your system administrator.
0.0.0.0: Add correct host key in /home/ec2-user/.ssh/known_hosts to get rid of this message.
0.0.0.0: Offending ECDSA key in /home/ec2-user/.ssh/known_hosts: 3
0.0.0.0: ECDSA host key for 0.0.0.0 has changed and you have requested strict checking.
0.0.0.0: Host key verification failed.
stopping yarn daemons
stopping resourcemanager
slave2: stopping nodemanager
slave7: stopping nodemanager
slave13: stopping nodemanager
slave6: stopping nodemanager
slave1: stopping nodemanager
slave3: stopping nodemanager
slave5: stopping nodemanager
slave16: stopping nodemanager
slave11: stopping nodemanager
slave15: stopping nodemanager
slave9: stopping nodemanager
slave4: no nodemanager to stop
slave10: stopping nodemanager
slave8: stopping nodemanager
slave14: stopping nodemanager
slave12: stopping nodemanager
no proxyserver to stop
[ec2-user@master ~]$ 
[ec2-user@slave5 ~]$ jps
6511 Jps
6222 DataNode
6338 NodeManager
[ec2-user@slave5 ~]$ jps
6594 Jps
[ec2-user@slave5 ~]$ 

```

Screen-shot showing Hadoop processes stopping in all slave instance.

## **Questions and Answers:**

1. What is a Master node? What is a Slaves node?

**Ans:** Master node manages the whole process like opening, closing input file, partitioning the input file, assigning the partitioned files to workers, and so on.

Slave node are the workers which are assigned to execute branch to finish assigned input part on which processing needs to be done. Basically slaves work on the actual data whereas master's work is to schedule and synchronize this work.

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

**Ans:** Each service uses different port to uniquely identify itself. If two services are given the same port either of the service would work at a single time. Both services won't be able to work simultaneously. Thus there would be port collision between different function modules if we use same port number.

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

**Ans:** We can change the number of mappers and reducers by adding two properties in "mapred-site.xml" file as below:

```
<property>
    <name> mapreduce.job.maps</name>
    <value>2</value>
</property>
<property>
    <name> mapreduce.job.reduces</name>
    <value>2</value>
</property>
```

## 6 INSTALL AND CONFIGURE SPARK

---

### Steps to get an Identity and Access Management Key:

1. Click on your Account Name on top right corner and then select “Security Credentials”.
2. Expand the Access Keys tab.
3. Click on “Create New Access Key”
4. A popup will show up. Click on the “Download key file” and store the key(.csv file) in a secure location.

### Steps to install Spark:

1. Start EC2 C3.Large instance and log into it.
2. Update all the installed applications using

```
>> sudo apt-get update
```
3. Install Java in the instance using

```
>>sudo apt-get install openjdk-7-jdk
```
4. Download Spark packages by using below command

```
>>wget http://apache.arvixe.com/spark/spark-1.6.1/spark-1.6.1-bin-hadoop2.6.tgz
```
5. Extract Spark packages from the tar and rename the folder using

```
>>tar -xvf spark-1.6.1-bin-hadoop2.6.tgz  
>>mv spark-1.6.1-bin-hadoop2.6 spark
```
6. Download Scala packages by using below command

```
>>wget http://www.scala-lang.org/files/archive/scala-2.10.4.tgz
```
7. Create a folder in usr/local for Scala using below command

```
>> sudo mkdir /usr/local/src/scala
```
8. Update .bashrc file with Scala variables

```
>> vi .bashrc  
(Inside .bashrc file enter below 2 entries)  
export SCALA_HOME=/usr/local/src/scala/scala-2.10.4  
export PATH=$SCALA_HOME/bin:$PATH  
(Save the .bashrc file)  
>> source .bashrc
```
9. Copy .pem file from your local filesystem into the instance using

```
>> scp -i "spark.pem" spark.pem ubuntu@ec2-54-164-89-172.compute-1.amazonaws.com:~/
```

**10.** Export your IAM AccessKeyID and SecurityAccess Key as below

```
>>export AWS_ACCESS_KEY_ID=[Your Access Key]  
>>export AWS_SECRET_ACCESS_KEY=[Your Secret Access Key]
```

**11.** Create an AMI of the instance so that it can be used to create either a 1 node cluster or 16 node cluster

## Screenshots to configure Spark:

Screenshot showing Spark Installation

## Screenshot showing Scala Installation

## 7 IMPLEMENTING 10 AND 100 GB SORT IN SPARK

---

### Steps to start 1 node and perform 10GB Sort in Spark:

1. Start an instance using the spark AMI which we creating in above section.
2. To start a cluster of 1 node write below command (Below command will create a 1 master and 1 slave cluster):

```
>> cd spark/ec2/  
>>./spark-ec2 -k spark -i spark.pem -s 1 -t c3.large --spot-price=0.025 --ebs-vol-size=30 --  
ebs-vol-num=1 launch spark_tsort
```

3. Login into the cluster using the below command (By default we login into Master node):

```
>>./spark-ec2 -k spark -i spark.pem login spark_tsort
```

4. Below all commands are now written in the master node of Spark

5. Install gensort to generate the input file of 10 GB using:

```
>>wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz  
>>tar -xvf gensort-linux-1.5.tar.gz
```

6. Check disk space using below command:

```
>>lblk
```

7. Generate input file of 10GB size using below command:

```
>> cd 64/  
>>./gensort -a 1000000000 [inputfileName]
```

8. Modify hdfs-site.xml to using below command:

```
>> cd ephemeral-hdfs/conf/  
>> vi hdfs-site.xml  
(Inside hdfs-site.xml modify the dfs.replication to make its value to 1 as shown below)  
<property>  
    <name>dfs.replication</name>  
    <value>1</value>  
</property>  
(Save the hdfs-site.xml file)
```

9. Now we need to copy this configuration into the slave node. This is done by below command:

```
>> cd spark/ec2/  
>>./spark-ec2/copy-dir ephemeral-hdfs/conf/
```

10. Now we need to stop SPARK and HDFS service and start again as configuration file was modified using

```
>>./stop-all.sh (inside spark/sbin/ directory)
```

```

>>./stop-dfs.sh (inside ephemeral-hdfs/bin/ directory)
>>./hadoop namenode -format
>> ./start-all.sh (inside spark/sbin/ directory)
>>./start-dfs.sh (inside ephemeral-hdfs/bin/ directory)

```

11. Now we need to put input file inside HDFS file system using below command:

```
>>ephemeral-hdfs/bin/hadoop fs -put [inputFileName] /user/shouvik/input/input10gb
```

12. Now Run Program of Sort written in Scala into spark shell.

```

>>cd spark/bin
>>./spark-shell

```

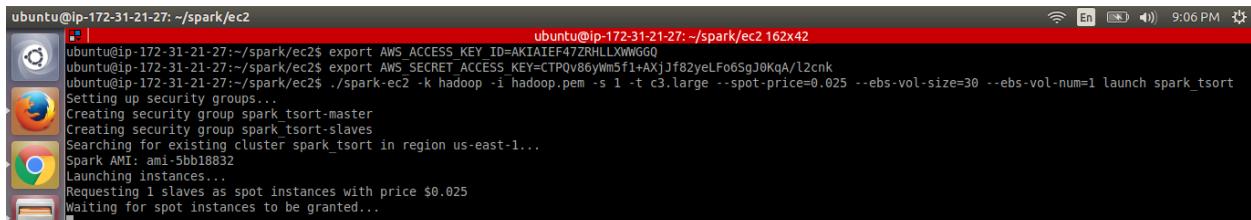
13. Copy and paste following command into shell. (Note: Run one by one)

```

>>val sortFile = sc.textFile("hdfs://ec2-52-207-231-47.compute-1.amazonaws.com:9000/user/shouvik/input/input10gb")
>>val sortedobj =
sortFile.flatMap(line=>line.split("\n")).map(word=>(word.substring(0,10),word.substring(10))).sortByKey()
>>sortedobj.saveAsTextFile("hdfs://ec2-52-207-231-47.compute-1.amazonaws.com:9000/user/shouvik/output10gb")

```

### Screenshots for 1 node cluster to perform 10GB sort in Spark:

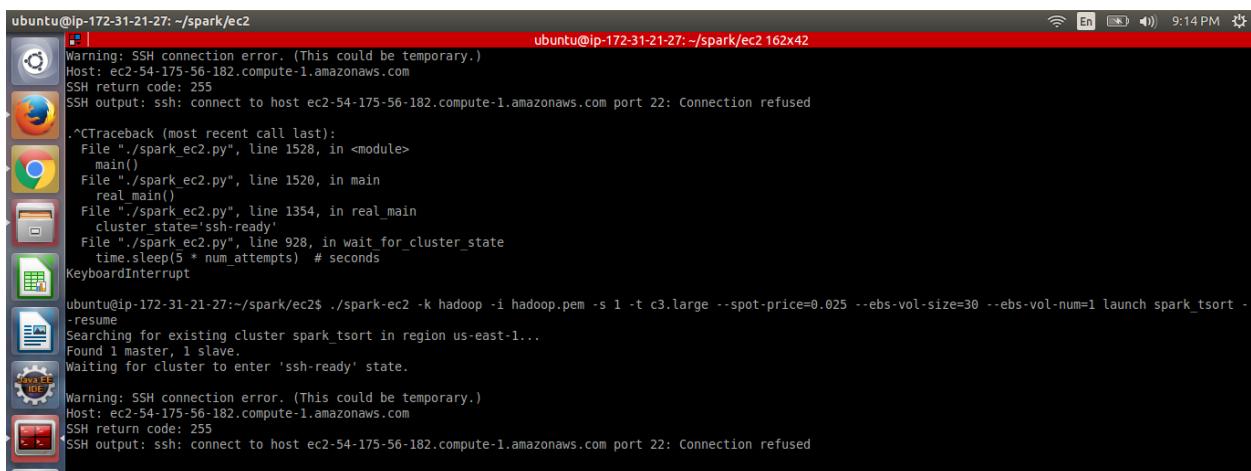


```

ubuntu@ip-172-31-21-27: ~/spark/ec2
[Ubuntu] [ ] ubuntu@ip-172-31-21-27: ~/spark/ec2 162x42
ubuntu@ip-172-31-21-27:~/spark/ec2$ export AWS_ACCESS_KEY_ID=AKIAIEF47ZKHLXWGG0
ubuntu@ip-172-31-21-27:~/spark/ec2$ export AWS_SECRET_ACCESS_KEY=CTPQv86yM5fI+AXjJf82yeLFo65gJ0KqA/l2cnk
ubuntu@ip-172-31-21-27:~/spark/ec2$ ./spark-ec2 -k hadoop -i hadoop.pem -s 1 -t c3.large --spot-price=0.025 --ebs-vol-size=30 --ebs-vol-num=1 launch spark_tsort
Setting up security groups...
Creating security group spark_tsort-master
Creating security group spark_tsort-slaves
Searching for existing cluster spark_tsort in region us-east-1...
Spark AMI: ami-5bb18832
Launching instances...
Requesting 1 slaves as spot instances with price $0.025
Waiting for spot instances to be granted...

```

Screenshot showing start of 1 node cluster in Spark



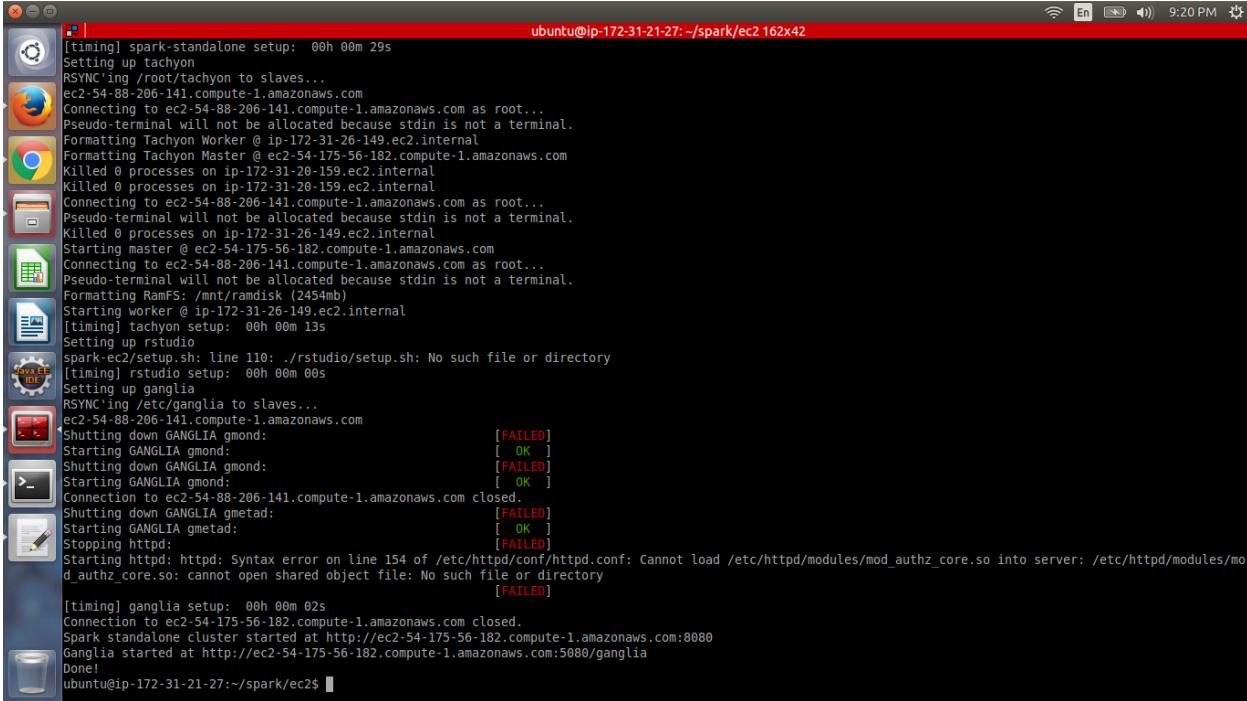
```

ubuntu@ip-172-31-21-27: ~/spark/ec2
[Ubuntu] [ ] ubuntu@ip-172-31-21-27: ~/spark/ec2 162x42
Warning: SSH connection error. (This could be temporary.)
Host: ec2-54-175-56-182.compute-1.amazonaws.com
SSH return code: 255
SSH output: ssh: connect to host ec2-54-175-56-182.compute-1.amazonaws.com port 22: Connection refused
.^CTraceback (most recent call last):
  File "./spark_ec2.py", line 1528, in <module>
    main()
  File "./spark_ec2.py", line 1520, in main
    real_main()
  File "./spark_ec2.py", line 1354, in real_main
    cluster_state='ssh-ready'
  File "./spark_ec2.py", line 928, in wait_for_cluster_state
    time.sleep(5 * num_attempts) # seconds
KeyboardInterrupt

ubuntu@ip-172-31-21-27:~/spark/ec2$ ./spark-ec2 -k hadoop -i hadoop.pem -s 1 -t c3.large --spot-price=0.025 --ebs-vol-size=30 --ebs-vol-num=1 launch spark_tsort -resume
Searching for existing cluster spark_tsort 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: ec2-54-175-56-182.compute-1.amazonaws.com
SSH return code: 255
SSH output: ssh: connect to host ec2-54-175-56-182.compute-1.amazonaws.com port 22: Connection refused

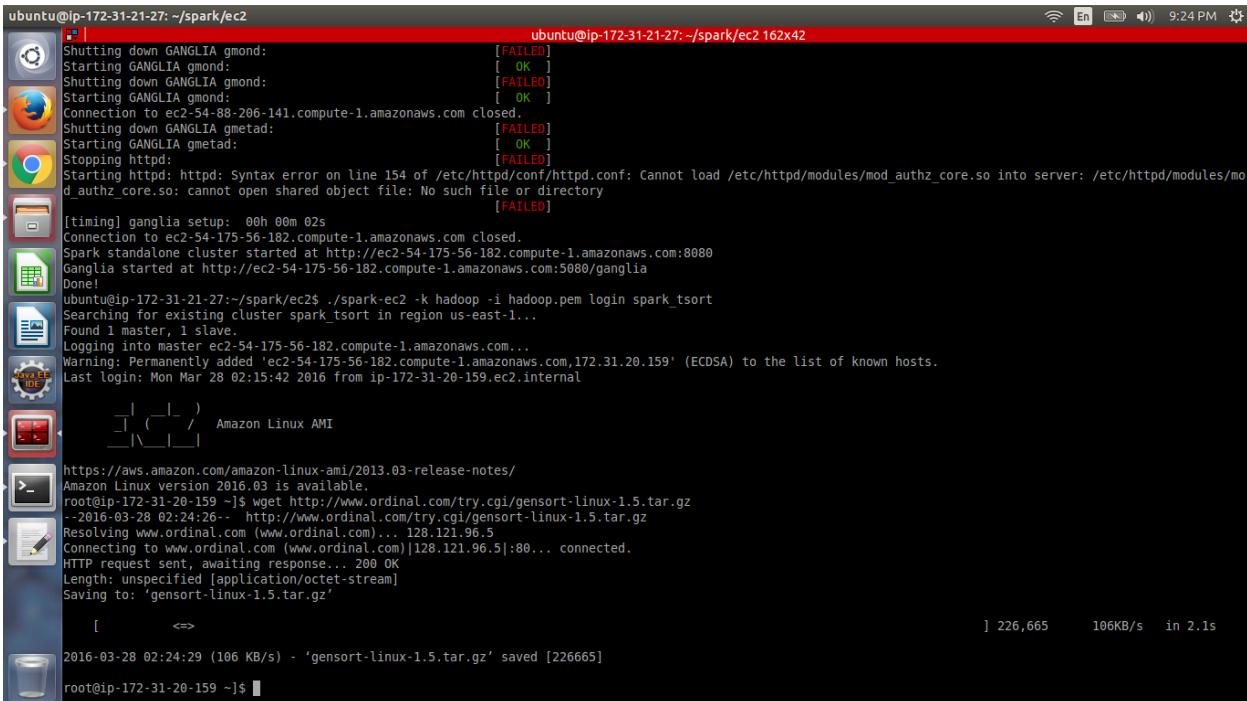
```

Screenshot showing resume of 1 node cluster in Spark



```
[timing] spark-standalone setup: 00h 00m 29s
Setting up tachyon
RSYNC'ing /root/tachyon to slaves...
ec2-54-88-206-141.compute-1.amazonaws.com
Connecting to ec2-54-88-206-141.compute-1.amazonaws.com as root...
Pseudo-terminal will not be allocated because stdin is not a terminal.
Formatting Tachyon Master @ ec2-54-175-56-182.compute-1.amazonaws.com
Killed 0 processes on ip-172-31-20-159.ec2.internal
Connecting to ec2-54-88-206-141.compute-1.amazonaws.com as root...
Pseudo-terminal will not be allocated because stdin is not a terminal.
Killed 0 processes on ip-172-31-20-159.ec2.internal
Starting master @ ec2-54-175-56-182.compute-1.amazonaws.com
Connecting to ec2-54-88-206-141.compute-1.amazonaws.com as root...
Pseudo-terminal will not be allocated because stdin is not a terminal.
Formatting RamFS: /mnt/ramdisk (2454mb)
Starting worker @ ip-172-31-26-149.ec2.internal
[timing] tachyon setup: 00h 00m 13s
Setting up rstudio
spark-ec2/setup.sh: line 110: ./rstudio/setup.sh: No such file or directory
[timing] rstudio setup: 00h 00m 00s
Setting up ganglia
RSYNC'ing /etc/ganglia to slaves...
ec2-54-88-206-141.compute-1.amazonaws.com
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-54-88-206-141.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 directory [FAILED]
[timing] ganglia setup: 00h 00m 02s
Connection to ec2-54-175-56-182.compute-1.amazonaws.com closed.
Spark standalone cluster started at http://ec2-54-175-56-182.compute-1.amazonaws.com:8080
Ganglia started at http://ec2-54-175-56-182.compute-1.amazonaws.com:5080/ganglia
Done!
ubuntu@ip-172-31-21-27:~/spark/ec2$
```

Screenshot showing successful completion of creation of 1 node cluster in Spark



```
ubuntu@ip-172-31-21-27:~/spark/ec2
[timing] ganglia setup: 00h 00m 02s
Connection to ec2-54-175-56-182.compute-1.amazonaws.com closed.
Spark standalone cluster started at http://ec2-54-175-56-182.compute-1.amazonaws.com:8080
Ganglia started at http://ec2-54-175-56-182.compute-1.amazonaws.com:5080/ganglia
Done!
ubuntu@ip-172-31-21-27:~/spark/ec2$ ./spark-ec2 -k hadoop.pem login spark_tsort
Searching for existing cluster spark_tsort in region us-east-1...
Found 1 master, 1 slave.
Logging into master ec2-54-175-56-182.compute-1.amazonaws.com...
Warning: Permanently added 'ec2-54-175-56-182.compute-1.amazonaws.com,172.31.20.159' (ECDSA) to the list of known hosts.
Last login: Mon Mar 28 02:15:42 2016 from ip-172-31-20-159.ec2.internal
[   | ( --_ ) _ /   Amazon Linux AMI
   | \_\_|__|_
https://aws.amazon.com/amazon-linux-ami/2013.03-release-notes/
Amazon Linux version 2016.03 is available.
root@ip-172-31-20-159 ~\$ wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz
--2016-03-28 02:24:26-- http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz
Resolving www.ordinal.com (www.ordinal.com)... 128.121.96.5
Connecting to www.ordinal.com (www.ordinal.com)|128.121.96.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/octet-stream]
Saving to: 'gensort-linux-1.5.tar.gz'

[          =>                                ] 226,665      106KB/s  in 2.1s
2016-03-28 02:24:29 (106 KB/s) - 'gensort-linux-1.5.tar.gz' saved [226665]
root@ip-172-31-20-159 ~\$
```

Screenshot showing login to master node in Spark

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.block.size</name>
    <value>134217728</value>
  </property>
  <property>
    <name>dfs.blocksize</name>
    <value>134217728</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/mnt/ephemeral-hdfs/data,/mnt2/ephemeral-hdfs/data</value>
  </property>
  <property>
    <name>dfs.namenode.handler.count</name>
    <value>25</value>
  </property>
  <property>
    <name>dfs.datanode.handler.count</name>
    <value>8</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
  <property>
    <name>dfs.</name>
    <value>hdfs-site.xml" 61L, 1223C</value>
  </property>
</configuration>
```

Screenshot showing modification to be made in hdfs-site.xml file in Spark

```
root@ip-172-31-20-159 conf]$ cd /root/ephemeral-hdfs/conf to slaves...
root@ip-172-31-20-159 ~$ ./spark-ec2/copy-dir ephemeral-hdfs/conf
RSYNC'ing /root/ephemeral-hdfs/conf to slaves...
ec2-54-88-206-141.compute-1.amazonaws.com
root@ip-172-31-20-159 ~$
```

Screenshot showing config file being copied to all slaves in Spark

```
root@ip-172-31-20-159 bin]$ jps
6230 Jps
5756 TachyonMaster
root@ip-172-31-20-159 bin]$ ls
hadoop     .hadoop-daemons.sh  start-all.sh      start-jobhistoryserver.sh  stop-balancer.sh      stop-mapred.sh
hadoop-config.sh  rcc          start-balancer.sh  start-mapred.sh     stop-dfs.sh        task-controller
hadoop-daemon.sh  slaves.sh   start-dfs.sh      stop-all.sh       stop-jobhistoryserver.sh
root@ip-172-31-20-159 bin]$ ./hadoop namenode -format
Warning: $HADOOP_HOME is deprecated.

16/03/28 03:15:08 INFO namenode.NameNode: STARTUP_MSG:
*****STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = ip-172-31-20-159.ec2.internal/172.31.20.159
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.0.4
STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.0 -r 1393290; compiled by 'hortonfo' on Wed Oct  3 05:13:58 UTC 2012
*****
Re-format filesystem in /mnt/ephemeral-hdfs/dfs/name ? (Y or N) y
Format aborted in /mnt/ephemeral-hdfs/dfs/name
16/03/28 03:15:10 INFO namenode.NameNode: SHUTDOWN_MSG:
*****SHUTDOWN_MSG: Shutting down NameNode at ip-172-31-20-159.ec2.internal/172.31.20.159
*****
```

Screenshot showing namenode formatted in Spark

```

shouvik@shouvik-dutta: ~/CloudComputing/Assignment2
shouvik@shouvik-dutta: ~/CloudComputing/Assignment2 81x42
root@ip-172-31-20-159 bin]$ cd ..
root@ip-172-31-20-159 ephemeral-hdfs]$ cd
root@ip-172-31-20-159 ~]$ cd spark/conf/
root@ip-172-31-20-159 conf]$ cd ..
root@ip-172-31-20-159 spark]$ cd sbin/
root@ip-172-31-20-159 sbin]$ ./start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /root/spark/logs/spark
ec2-54-88-206-141.compute-1.amazonaws.com: starting org.apache.spark.deploy.worke
worker-1-ip-172-31-26-149.ec2.internal.out
root@ip-172-31-20-159 sbin]$ jps
6383 Jps
5750 TachyonMaster
6311 Master
root@ip-172-31-20-159 sbin]$ cd
root@ip-172-31-20-159 ~]$ cd ephemeral-hdfs/bin/
root@ip-172-31-20-159 bin]$ ./start-dfs.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /mnt/ephemeral-hdfs/logs/hadoop-root-namenode-ip-17
ec2-54-88-206-141.compute-1.amazonaws.com: Warning: $HADOOP_HOME is deprecated.
ec2-54-88-206-141.compute-1.amazonaws.com: starting datanode, logging to /mnt/eph
ec2-54-175-56-182.compute-1.amazonaws.com: Warning: $HADOOP_HOME is deprecated.
ec2-54-175-56-182.compute-1.amazonaws.com: starting secondarynamenode, logging to
rnal.out
root@ip-172-31-20-159 bin]$ jps
6435 NameNode
6615 SecondaryNameNode
5750 TachyonMaster
6311 Master
6664 Jps
root@ip-172-31-20-159 bin]$ 

```

Screenshot showing master and slave running in Spark

```

shouvik@shouvik-dutta: ~/CloudComputing/Assignment2
shouvik@shouvik-dutta: ~/CloudComputing/Assignment2 81x20
root@ip-172-31-20-159 bin]$ cd
root@ip-172-31-20-159 ~]$ ephemeral-hdfs/bin/hadoop fs -put /vol0/file_10.txt /us
er/root/input/file_10.txt
Warning: $HADOOP_HOME is deprecated.

root@ip-172-31-26-149 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.9G  3.1G  4.8G  40% /
tmpfs          1.9G    0  1.9G   0% /dev/shm
/dev/xvdb      16G   3.1G  12G  22% /mnt
/dev/xvdf      16G   2.0G  13G  14% /mnt2
/dev/xvds      30G   33M  30G   1% /vol0
root@ip-172-31-26-149 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.9G  3.1G  4.8G  40% /
tmpfs          1.9G    0  1.9G   0% /dev/shm
/dev/xvdb      16G   3.3G  12G  23% /mnt
/dev/xvdf      16G   2.2G  13G  16% /mnt2
/dev/xvds      30G   33M  30G   1% /vol0
root@ip-172-31-20-159 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1     7.9G  3.4G  4.5G  43% /
tmpfs          1.9G    0  1.9G   0% /dev/shm
/dev/xvdb      16G   1.3G  14G   9% /mnt
/dev/xvdf      16G  167M  15G   2% /mnt2
/dev/xvds      30G   9.4G  21G  32% /vol0
root@ip-172-31-20-159 ~]$ 

```

Screenshot showing input data being put in HDFS in Spark

```
shouvik@shouvik-dutta: ~/CloudComputing/Assignment2
```

shouvik@shouvik-dutta: ~/CloudComputing/Assignment2 162x42  
16/03/28 03:25:12 INFO util.Utils: Successfully started service 'HTTP class server' on port 39159.  
Welcome to  
 version 1.6.1  
Using Scala version 2.10.5 (OpenJDK 64-Bit Server VM, Java 1.7.0\_95)  
Type in expressions to have them evaluated.  
Type :help for more information.  
16/03/28 03:25:17 INFO spark.SparkContext: Running Spark version 1.6.1  
16/03/28 03:25:17 WARN spark.SparkConf:  
SPARK\_WORKER\_INSTANCES was detected (set to '1').  
This is deprecated in Spark 1.0+.  
Please instead use:  
- ./spark-submit with --num-executors to specify the number of executors  
- Or set SPARK\_EXECUTOR\_INSTANCES  
- spark.executor.instances to configure the number of instances in the spark config.  
  
16/03/28 03:25:17 INFO spark.SecurityManager: Changing view acls to: root  
16/03/28 03:25:17 INFO spark.SecurityManager: Changing modify acls to: root  
16/03/28 03:25:17 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(root); users with modify permissions: Set(root)  
16/03/28 03:25:18 INFO util.Utils: Successfully started service 'sparkDriver' on port 39730.  
16/03/28 03:25:18 INFO slf4j.Slf4jLogger: Slf4jLogger started  
16/03/28 03:25:18 INFO Remoting: Starting remoting  
16/03/28 03:25:18 INFO Remoting: Remoting started: listening on addresses :[akka.tcp://sparkDriverActorSystem@172.31.20.159:60319]  
16/03/28 03:25:18 INFO util.Utils: Successfully started service 'sparkDriverActorSystem' on port 60319.  
16/03/28 03:25:18 INFO spark.SparkEnv: Registering MapOutputTracker  
16/03/28 03:25:18 INFO spark.SparkEnv: Registering BlockManagerMaster  
16/03/28 03:25:18 INFO storage.DiskBlockManager: Created local directory at /mnt/spark/blockmgr-3c580120-2b75-4889-8f4d-d73b4d03b670  
16/03/28 03:25:18 INFO storage.DiskBlockManager: Created local directory at /mnt2/spark/blockmgr-c24e/c8f-0f90-4a04-b110-4e/0817/9ede  
16/03/28 03:25:18 INFO storage.MemoryStore: MemoryStore started with capacity 511.5 MB  
16/03/28 03:25:18 INFO spark.SparkEnv: Registering OutputCommitCoordinator  
16/03/28 03:25:19 INFO server.Server: jetty-8.y.z-SNAPSHOT  
16/03/28 03:25:19 INFO server.AbstractConnector: Started SelectChannelConnector@0.0.0.0:4040  
16/03/28 03:25:19 INFO util.Utils: Successfully started service 'SparkUI' on port 4040.  
16/03/28 03:25:19 INFO ui.SparkUI: Started SparkUI at http://ec2-54-175-56-182.compute-1.amazonaws.com:4040  
16/03/28 03:25:19 INFO client.AppClients\$ClientEndpoint: Connecting to master spark://ec2-54-175-56-182.compute-1.amazonaws.com:7077...

Screenshot showing shell running in Spark

```
shouvik@shouvik-dutta: ~/CloudComputing/Assignment2          shouvik@shouvik-dutta: ~/CloudComputing/Assignment2 162x42
[INFO] +-----+
| at java.lang.reflect.Method.invoke(Method.java:606)
| at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$$runMain(SparkSubmit.scala:731)
| at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:181)
| at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:206)
| at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:121)
| at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
Caused by: java.io.IOException: Filesystem closed
| at org.apache.hadoop.hdfs.DFSClient.checkOpen(DFSClient.java:323)
| at org.apache.hadoop.hdfs.DFSClient.getFileInfo(DFSClient.java:1057)
| at org.apache.hadoop.hdfs.DistributedFileSystem.getFileStatus(DistributedFileSystem.java:554)
| at org.apache.hadoop.hive.ql.session.SessionState.createRootDFSDir(SessionState.java:599)
| at org.apache.hadoop.hive.ql.session.SessionState.createSessionDirs(SessionState.java:554)
| at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:508)
... 62 more

<console>:16: error: not found: value sqlContext
      import sqlContext.implicits._

<console>:16: error: not found: value sqlContext
      import sqlContext.sql

scala>
scala>
scala>

scala> val sortFile = sc.textFile("hdfs://ec2-54-175-56-182.compute-1.amazonaws.com:9000/user/root/file_10.txt")
<console>:21: error: value textFile is not a member of org.apache.spark.SparkContext
      val sortFile = sc.textFile("hdfs://ec2-54-175-56-182.compute-1.amazonaws.com:9000/user/root/file_10.txt")

scala> val sortFile = sc.textFile("hdfs://ec2-54-175-56-182.compute-1.amazonaws.com:9000/user/root/file_10.txt")
16/03/28 03:29:05 INFO storage.MemoryStore: Block broadcast_0 stored as values in memory (estimated size 26.5 KB, free 26.5 KB)
16/03/28 03:29:05 INFO storage.MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 5.6 KB, free 32.1 KB)
16/03/28 03:29:05 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on 172.31.20.159:42964 (size: 5.6 KB, free: 511.5 MB)
16/03/28 03:29:05 INFO spark.SparkContext: Created broadcast 0 from textfile at <console>:21
sortfile: org.apache.spark.rdd.RDD[String] = hdfs://ec2-54-175-56-182.compute-1.amazonaws.com:9000/user/root/file_10.txt MapPartitionsRDD[1] at textFile at <console>:21
scala>
```

Screenshot showing scala sort program written in Spark Shell

```

shouvik@shouvik-dutta: ~/CloudComputing/Assignment2
shouvik@shouvik-dutta: ~/CloudComputing/Assignment2 162x42
16/03/28 03:34:24 INFO scheduler.TaskSetManager: Finished task 15.0 in stage 0.0 (TID 15) in 1692 ms on ip-172-31-26-149.ec2.internal (16/75)
16/03/28 03:34:25 INFO scheduler.TaskSetManager: Starting task 18.0 in stage 0.0 (TID 18, ip-172-31-26-149.ec2.internal, partition 18, NODE LOCAL, 2184 bytes)
16/03/28 03:34:25 INFO scheduler.TaskSetManager: Finished task 17.0 in stage 0.0 (TID 17) in 1682 ms on ip-172-31-26-149.ec2.internal (17/75)
16/03/28 03:34:25 INFO scheduler.TaskSetManager: Starting task 19.0 in stage 0.0 (TID 19, ip-172-31-26-149.ec2.internal, partition 19, NODE LOCAL, 2184 bytes)
16/03/28 03:34:25 INFO scheduler.TaskSetManager: Finished task 16.0 in stage 0.0 (TID 16) in 1723 ms on ip-172-31-26-149.ec2.internal (18/75)
16/03/28 03:34:27 INFO scheduler.TaskSetManager: Starting task 20.0 in stage 0.0 (TID 20, ip-172-31-26-149.ec2.internal, partition 20, NODE LOCAL, 2184 bytes)
16/03/28 03:34:27 INFO scheduler.TaskSetManager: Finished task 18.0 in stage 0.0 (TID 18) in 1680 ms on ip-172-31-26-149.ec2.internal (19/75)
16/03/28 03:34:27 INFO scheduler.TaskSetManager: Starting task 21.0 in stage 0.0 (TID 21, ip-172-31-26-149.ec2.internal, partition 21, NODE LOCAL, 2184 bytes)
16/03/28 03:34:27 INFO scheduler.TaskSetManager: Finished task 19.0 in stage 0.0 (TID 19) in 1745 ms on ip-172-31-26-149.ec2.internal (20/75)
16/03/28 03:34:29 INFO scheduler.TaskSetManager: Starting task 20.0 in stage 0.0 (TID 20) in 1639 ms on ip-172-31-26-149.ec2.internal (21/75)
16/03/28 03:34:29 INFO scheduler.TaskSetManager: Finished task 23.0 in stage 0.0 (TID 23, ip-172-31-26-149.ec2.internal, partition 23, NODE LOCAL, 2184 bytes)
16/03/28 03:34:29 INFO scheduler.TaskSetManager: Finished task 21.0 in stage 0.0 (TID 21) in 1778 ms on ip-172-31-26-149.ec2.internal (22/75)
16/03/28 03:34:30 INFO scheduler.TaskSetManager: Starting task 24.0 in stage 0.0 (TID 24, ip-172-31-26-149.ec2.internal, partition 24, NODE LOCAL, 2184 bytes)
16/03/28 03:34:30 INFO scheduler.TaskSetManager: Finished task 22.0 in stage 0.0 (TID 22) in 1702 ms on ip-172-31-26-149.ec2.internal (23/75)
16/03/28 03:34:31 INFO scheduler.TaskSetManager: Starting task 25.0 in stage 0.0 (TID 25, ip-172-31-26-149.ec2.internal, partition 25, NODE LOCAL, 2184 bytes)
16/03/28 03:34:31 INFO scheduler.TaskSetManager: Finished task 23.0 in stage 0.0 (TID 23) in 1898 ms on ip-172-31-26-149.ec2.internal (24/75)
16/03/28 03:34:32 INFO scheduler.TaskSetManager: Starting task 26.0 in stage 0.0 (TID 26, ip-172-31-26-149.ec2.internal, partition 26, NODE LOCAL, 2184 bytes)
16/03/28 03:34:32 INFO scheduler.TaskSetManager: Finished task 24.0 in stage 0.0 (TID 24) in 1666 ms on ip-172-31-26-149.ec2.internal (25/75)
16/03/28 03:34:32 INFO scheduler.TaskSetManager: Starting task 27.0 in stage 0.0 (TID 27, ip-172-31-26-149.ec2.internal, partition 27, NODE LOCAL, 2184 bytes)
16/03/28 03:34:32 INFO scheduler.TaskSetManager: Finished task 25.0 in stage 0.0 (TID 25) in 1757 ms on ip-172-31-26-149.ec2.internal (26/75)
16/03/28 03:34:34 INFO scheduler.TaskSetManager: Starting task 28.0 in stage 0.0 (TID 28, ip-172-31-26-149.ec2.internal, partition 28, NODE LOCAL, 2184 bytes)
16/03/28 03:34:34 INFO scheduler.TaskSetManager: Finished task 26.0 in stage 0.0 (TID 26) in 1671 ms on ip-172-31-26-149.ec2.internal (27/75)
16/03/28 03:34:34 INFO scheduler.TaskSetManager: Starting task 29.0 in stage 0.0 (TID 29, ip-172-31-26-149.ec2.internal, partition 29, NODE LOCAL, 2184 bytes)
16/03/28 03:34:34 INFO scheduler.TaskSetManager: Finished task 27.0 in stage 0.0 (TID 27) in 1731 ms on ip-172-31-26-149.ec2.internal (28/75)
16/03/28 03:34:35 INFO scheduler.TaskSetManager: Starting task 30.0 in stage 0.0 (TID 30, ip-172-31-26-149.ec2.internal, partition 30, NODE LOCAL, 2184 bytes)
16/03/28 03:34:35 INFO scheduler.TaskSetManager: Finished task 28.0 in stage 0.0 (TID 28) in 1733 ms on ip-172-31-26-149.ec2.internal (29/75)
16/03/28 03:34:36 INFO scheduler.TaskSetManager: Starting task 31.0 in stage 0.0 (TID 31, ip-172-31-26-149.ec2.internal, partition 31, NODE LOCAL, 2184 bytes)
16/03/28 03:34:36 INFO scheduler.TaskSetManager: Finished task 29.0 in stage 0.0 (TID 29) in 1661 ms on ip-172-31-26-149.ec2.internal (30/75)
16/03/28 03:34:37 INFO scheduler.TaskSetManager: Starting task 32.0 in stage 0.0 (TID 32, ip-172-31-26-149.ec2.internal, partition 32, NODE LOCAL, 2184 bytes)
16/03/28 03:34:37 INFO scheduler.TaskSetManager: Finished task 30.0 in stage 0.0 (TID 30) in 1762 ms on ip-172-31-26-149.ec2.internal (31/75)
16/03/28 03:34:37 INFO scheduler.TaskSetManager: Starting task 33.0 in stage 0.0 (TID 33, ip-172-31-26-149.ec2.internal, partition 33, NODE LOCAL, 2184 bytes)
16/03/28 03:34:37 INFO scheduler.TaskSetManager: Finished task 31.0 in stage 0.0 (TID 31) in 1681 ms on ip-172-31-26-149.ec2.internal (32/75)
16/03/28 03:34:39 INFO scheduler.TaskSetManager: Starting task 34.0 in stage 0.0 (TID 34, ip-172-31-26-149.ec2.internal, partition 34, NODE LOCAL, 2184 bytes)
16/03/28 03:34:39 INFO scheduler.TaskSetManager: Finished task 32.0 in stage 0.0 (TID 32) in 1746 ms on ip-172-31-26-149.ec2.internal (33/75)
16/03/28 03:34:39 INFO scheduler.TaskSetManager: Starting task 35.0 in stage 0.0 (TID 35, ip-172-31-26-149.ec2.internal, partition 35, NODE LOCAL, 2184 bytes)
16/03/28 03:34:39 INFO scheduler.TaskSetManager: Finished task 33.0 in stage 0.0 (TID 33) in 1689 ms on ip-172-31-26-149.ec2.internal (34/75)
16/03/28 03:34:40 INFO scheduler.TaskSetManager: Starting task 36.0 in stage 0.0 (TID 36, ip-172-31-26-149.ec2.internal, partition 36, NODE LOCAL, 2184 bytes)
16/03/28 03:34:40 INFO scheduler.TaskSetManager: Finished task 34.0 in stage 0.0 (TID 34) in 1648 ms on ip-172-31-26-149.ec2.internal (35/75)
16/03/28 03:34:41 INFO scheduler.TaskSetManager: Starting task 37.0 in stage 0.0 (TID 37, ip-172-31-26-149.ec2.internal, partition 37, NODE LOCAL, 2184 bytes)
16/03/28 03:34:41 INFO scheduler.TaskSetManager: Finished task 35.0 in stage 0.0 (TID 35) in 1765 ms on ip-172-31-26-149.ec2.internal (36/75)

```

Screenshot showing scala sort program running in Spark Shell

## Spark Jobs (?)

Total Uptime: 12 min  
Scheduling Mode: FIFO  
Active Jobs: 1  
Completed Jobs: 1

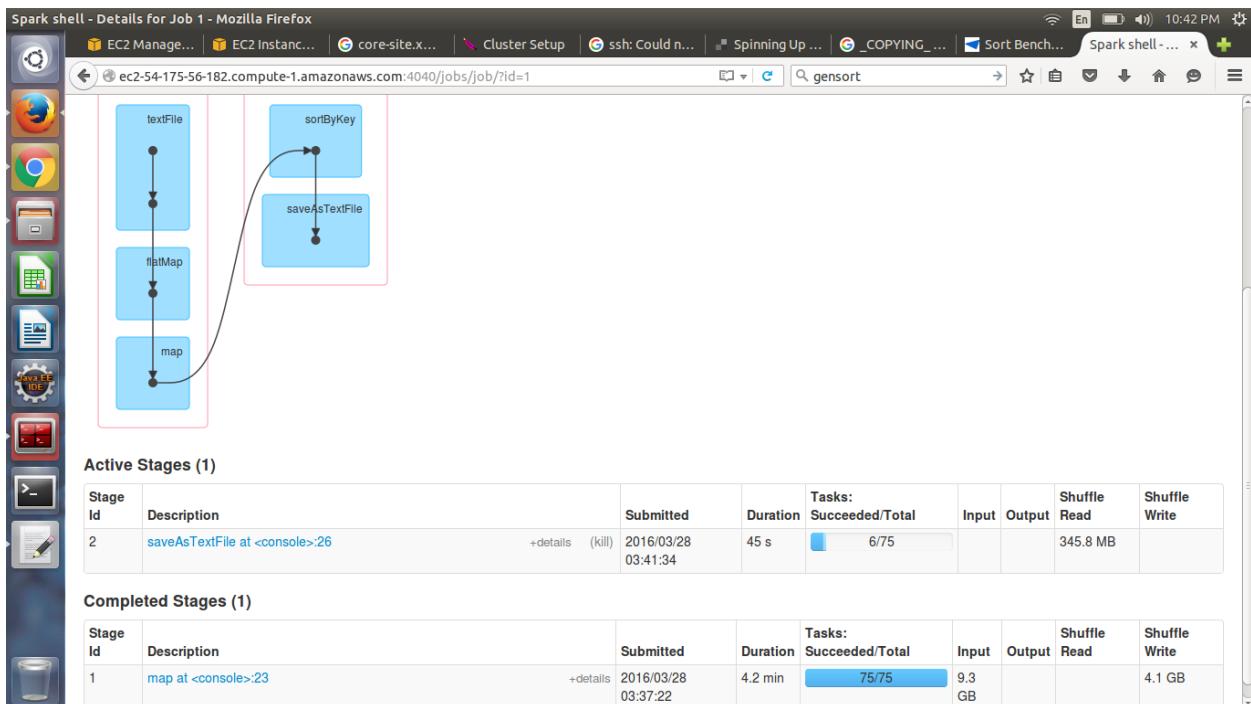
[Event Timeline](#)

Active Jobs (1)					
Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	saveAsTextFile at <console>:26	2016/03/28 03:37:22	17 s	0/2	4/150

[Completed Jobs \(1\)](#)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	sortByKey at <console>:23	2016/03/28 03:34:06	1.1 min	1/1	75/75

Screenshot showing spark jobs in browser



Screenshot showing visualization for scala sort program in browser

```
shouvik@shouvik-dutta: ~/CloudComputing/Assignment2
root@ip-172-31-20-159 ~]$ ls
64 ephemeral-hdfs gensort-linux-1.5.tar.gz hadoop-native mapreduce part-00000 part-00074 persistent-hdfs scala spark spark-spark-ec2 tachyon
root@ip-172-31-20-159 ~$ head -10 part-00000
( "0!uve, 000000000000000000000000000000001228D4 777788800002224444DDDDDDDEEE00000000CCC7777DDDD)
( PMd32=, 000000000000000000000000000000003440CC1 FFFFFFFEE666CCCCBBB9993335550DDDDDD7778886666)
( ^3C0|., 00000000000000000000000000000000158C5C5 5555AAA9999EEF8882229999CCCCDD666655554442222)
( &S3/1], 000000000000000000000000000000002145078 8888888888DD1111CCC55566668888111EEFEDDD2229999)
( _!=#_9, 0000000000000000000000000000000019072E3 33332222FFBBB80000FFFAAA66665553330DD3333CCCC)
( !f6su2, 000000000000000000000000000000003ABF084 9999FFFF555533337777CCC4444BBBB7777EEE88888888)
( #&Npd., 000000000000000000000000000000003B30FB9 EEEE55555556666AAA5555BBBBDD00000111666600000DDD)
( #`^Q`~, 000000000000000000000000000000002EDC58 1110000333444111166666666AAAAAA0000111CCCCEEE)
( $"-'Q), 0000000000000000000000000000000051265D 8888AAA11114444FFF7773333EEE44440000FFF9999999)
( ~`-Q), 0000000000000000000000000000000051265D CCCCC6666EEE22220000DDDAAA88886666BBB80006666AAA)
root@ip-172-31-20-159 ~$ tail -10 part-00074
(~uq2k#=U, 00000000000000000000000000000002C06745 99991111DDDD22221110000FFFEEEEEFFF33337777CCCC2222)
(~v0$Qnm, 000000000000000000000000000000479701 CCCCR8888333FFF000000000009999111FF7774446666)
(~yK0L:GE, 000000000000000000000000000000204884F CCCC1111444488882226666888888555777EEE8888888000)
(~yK'H.jl, 0000000000000000000000000000004630064 44440000FFF33339999444477770DDDFFFFAAA1118888000)
(~yL;cXE, 0000000000000000000000000000005B0D211 2222EEE333300002221111CCCFFFF55557774444BBBB6666)
(~zBa_Tt, 00000000000000000000000000000007F9F4F BBBBCCCC66665559999FFF8888AAA1116666AAA8888888000)
(~ze0`Feg, 0000000000000000000000000000001E06130 4444CCCCBBB99992228885558888CCCF000011111111)
(~g)xjWHI, 000000000000000000000000000000CA1345 777711118888AAAAAA2222111BBB8000222888888888)
(~P;jg0g, 00000000000000000000000000000040DA3E4 4444FFF444466663333EEE888888888DDDEEE4442222000)
(~KU)Kep, 0000000000000000000000000000005E4A0AA 000066665555111BBB888889999AAA555500003335557777)
root@ip-172-31-20-159 ~$
```

Screenshots for top 10 and bottom 10 lines of sorted output file

## **Steps to start 16 nodes and perform 100GB Sort in Spark:**

1. Start an instance using the spark AMI which we creating in above section.
2. To start a cluster of 1 node write below command (Below command creates a C3.4xlarge master instance and 16 C3.large slave instances)

```
>> cd spark/ec2/  
>>./spark-ec2 -k spark -i spark.pem -s 16 --instance-type=c3.large -m c3.4xlarge --spot-price=0.14 launch spark_tsort_100gb
```

3. Login into the cluster using the below command (By default we login into Master node):

```
>>./spark-ec2 -k spark -i spark.pem login spark_tsort_100gb
```

4. Below all commands are now written in the master node of Spark

5. Install gensort to generate the input file of 100 GB using:

```
>>wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz  
>>tar -xvf gensort-linux-1.5.tar.gz
```

6. Check disk space using below command:

```
>>lsblk
```

7. Generate input file of 100GB size using below command:

```
>> cd 64/  
>>./gensort -a 10000000000 [inputfileName]
```

8. Modify hdfs-site.xml to using below command:

```
>> cd ephemeral-hdfs/conf/  
>> vi hdfs-site.xml  
(Inside hdfs-site.xml modify the dfs.replication to make its value to 1 as shown below)  
<property>  
    <name>dfs.replication</name>  
    <value>1</value>  
</property>  
(Save the hdfs-site.xml file)
```

9. Now we need to copy this configuration into all 16 slave nodes. This is done by below command:

```
>> cd spark/ec2/  
>>./spark-ec2/copy-dir ephemeral-hdfs/conf/
```

10. Now we need to stop SPARK and HDFS service and start again as configuration file was modified using

```
>>./stop-all.sh (inside spark/sbin/ directory)  
>>./stop-dfs.sh (inside ephemeral-hdfs/bin/ directory)  
>>./hadoop namenode -format
```

```
>> ./start-all.sh (inside spark/sbin/ directory)
>>./start-dfs.sh (inside ephemeral-hdfs/bin/ directory)
```

- Now we need to put input file inside HDFS file system using below command:

```
>>ephemeral-hdfs/bin/hadoop fs -put [inputFileName] /user/shouvik/input/input100gb
```

- Now Run Program of Sort written in Scala into spark shell.

```
>>cd spark/bin
>>./spark-shell
```

- Copy and paste following command into shell. (Note: Run one by one)

```
>>val sortFile = sc.textFile("hdfs://ec2-52-207-231-47.compute-1.amazonaws.com:9000/user/shouvik/input/input100gb")
>>val sortedobj =
sortFile.flatMap(line=>line.split("\n")).map(word=>(word.substring(0,10),word.substring(10))).sortByKey()
>>sortedobj.saveAsTextFile("hdfs://ec2-52-207-231-47.compute-1.amazonaws.com:9000/user/shouvik/output10gb")
```

### Screenshots for 16 node cluster to perform 100GB sort in Spark:

```
ubuntu@ip-172-31-21-27: ~/spark/ec2
[...]
ubuntu@ip-172-31-21-27:~/spark/ec2$ ls
deploy_generic hadoop.pem lib README spark-ec2 spark ec2.py
ubuntu@ip-172-31-21-27:~/spark/ec2$ export AWS_ACCESS_KEY_ID=AKIAIEF47ZRHLXW6G6
ubuntu@ip-172-31-21-27:~/spark/ec2$ export AWS_SECRET_ACCESS_KEY=CTPQv86yMm5f1+Axjf82yeLFo65gJ0KqA/l2cnk
ubuntu@ip-172-31-21-27:~/spark/ec2$ ./spark-ec2 -k hadoop -i hadoop.pem -s 16 --instance-type=c3.large -m c3.4xlarge --spot-price=0.14 launch spark_tsort_100gb
Setting up security groups...
Creating security group spark_tsort_100gb-master
Creating security group spark_tsort_100gb-slaves
Searching for existing cluster spark_tsort_100gb in region us-east-1...
Spark AMI: ami-5bb18832
Launching instances...
Requesting 16 slaves as spot instances with price $0.140
Waiting for spot instances to be granted...
```

Screenshot showing start of 16 nodes cluster in Spark

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
spark_tsort_100gb-slave-i24f4c4a0	i-24f4c4a0	c3.large	us-east-1c	running	Initializing	None	ec2-52-90-53-211.com...	52.90.53.211
spark_tsort_100gb-slave-i26f4c4a2	i-26f4c4a2	c3.large	us-east-1c	running	Initializing	None	ec2-52-23-160-235.co...	52.23.160.235
spark_tsort_100gb-slave-i27f4c4a3	i-27f4c4a3	c3.large	us-east-1c	running	Initializing	None	ec2-52-23-188-41.com...	52.23.188.41
spark_tsort-master-i34dd55fa	i-34dd55fa	c3.large	us-east-1d	terminated	None	None	ec2-52-241-215.co...	52.241.215
spark_tsort_100gb-slave-i4af4c4ce	i-4af4c4ce	c3.large	us-east-1c	running	Initializing	None	ec2-52-207-241-215.co...	52.207.241.215
spark_tsort_100gb-slave-i4ff4c4cb	i-4ff4c4cb	c3.large	us-east-1c	running	Initializing	None	ec2-52-207-252-109.co...	52.207.252.109
spark_tsort_100gb-slave-i52f5c5d6	i-52f5c5d6	c3.large	us-east-1c	running	Initializing	None	ec2-52-23-192-82.com...	52.23.192.82
spark_tsort_100gb-slave-i54f5c5d0	i-54f5c5d0	c3.large	us-east-1c	running	Initializing	None	ec2-52-91-53-128.com...	52.91.53.128
spark_tsort_100gb-slave-i55f5c5d1	i-55f5c5d1	c3.large	us-east-1c	running	Initializing	None	ec2-52-87-167-96.com...	52.87.167.96
spark_tsort_100gb-slave-i56f5c5d2	i-56f5c5d2	c3.large	us-east-1c	running	Initializing	None	ec2-52-91-250-41.com...	52.91.250.41
spark_tsort_100gb-slave-i57f5c5d3	i-57f5c5d3	c3.large	us-east-1c	running	Initializing	None	ec2-52-91-59-6.comput...	52.91.59.6
spark_tsort_100gb-master-i5ef5c5da	i-5ef5c5da	c3.4xlarge	us-east-1c	running	Initializing	None	ec2-52-207-231-47.co...	52.207.231.47
spark_tsort_100gb-slave-i70f4c4f4	i-70f4c4f4	c3.large	us-east-1c	running	Initializing	None	ec2-52-91-173-198.co...	52.91.173.198
spark_tsort_100gb-slave-i71f4c4f5	i-71f4c4f5	c3.large	us-east-1c	running	Initializing	None	ec2-52-91-225-139.co...	52.91.225.139
spark_tsort_100gb-slave-i73f4c4f7	i-73f4c4f7	c3.large	us-east-1c	running	Initializing	None	ec2-52-201-244-80.co...	52.201.244.80
spark_tsort-slave-i84dd55f1	i-84dd55f1	c3.large	us-east-1d	terminated	None	None	ec2-52-91-59-6.comput...	52.91.59.6
spark_tsort_100gb-slave-i8f5c55c	i-8f5c55c	c3.large	us-east-1c	running	Initializing	None	ec2-52-90-108-240.co...	52.90.108.240
spark_tsort_100gb-slave-iad5c55e	i-ad5c55e	c3.large	us-east-1c	running	Initializing	None	ec2-52-207-244-243.co...	52.207.244.243
spark_tsort_100gb-slave-i-db5c55f1	i-db5c55f1	c3.large	us-east-1c	running	Initializing	None	ec2-52-91-92-6.comput...	52.91.92.6

Screenshot showing 16 instance running in AWS dashboard

```

ubuntu@ip-172-31-21-27: ~/spark/ec2$ 
[...]
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-91-173-198.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-201-244-80.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-91-59-6.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-23-160-235.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-207-241-215.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-91-250-41.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-207-252-109.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-23-188-41.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-91-92-6.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [OK]
Connection to ec2-52-91-225-139.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 directory [FAILED]
[timing] ganglia setup: 00h 00m 11s
Connection to ec2-52-207-231-47.compute-1.amazonaws.com closed.
Spark standalone cluster started at http://ec2-52-207-231-47.compute-1.amazonaws.com:8080
Ganglia started at http://ec2-52-207-231-47.compute-1.amazonaws.com:5080/ganglia
Done!
ubuntu@ip-172-31-21-27:~/spark/ec2$ 

```

Screenshot showing 16 node cluster installation done

```

shouvik@shouvik-dutta: ~/CloudComputing/Assignment2$ 
root@ip-172-31-7-6 mnt]$ ls -l
total 98801372
drwxr-xr-x 4 root root 4096 Mar 28 06:05 ephemeral-hdfs
drwxr-xr-x 3 root root 4096 Mar 28 06:04 ganglia
drwxr-xr-x 5 root root 4096 Mar 28 06:05 hadoop
drwxr-xr-x 2 root root 4096 Mar 28 06:05 hadoop-logs
-rw-r--r-- 1 root root 10000000000 Mar 28 06:57 inputlogb
drwxr-w--w- 2 root root 16384 Dec 24 15:23 lost+found
drwxr-xr-x 3 root root 4096 Mar 28 06:05 persistent-hdfs
drwxrwxrwx 2 root root 4096 Mar 28 06:04 swap
drwxr-xr-x 2 root root 4096 Mar 28 06:05 yarn-local
drwxr-xr-x 2 root root 4096 Mar 28 06:05 yarn-logs
root@ip-172-31-7-6 mnt]$ cd ~
root@ip-172-31-7-6 ~]$ ls
64 ephemeral-hdfs gensorc-linux-1.5.tar.gz hadoop-native mapreduce persistent-hdfs scala spark spark-ec2 tachyon
root@ip-172-31-7-6 ~]$ cd ephemeral-hdfs/conf/
root@ip-172-31-7-6 conf]$ vi hdfs-site.xml
root@ip-172-31-7-6 conf]$ vi hdfs-site.xml
root@ip-172-31-7-6 conf]$ cd ..
root@ip-172-31-7-6 ephemeral-hdfs]$ cd ..
root@ip-172-31-7-6 ~]$ ls
64 ephemeral-hdfs gensorc-linux-1.5.tar.gz hadoop-native mapreduce persistent-hdfs scala spark spark-ec2 tachyon
root@ip-172-31-7-6 ~]$ ./spark-ec2/copy-dir ephemeral-hdfs/conf/
RSYNC'ing /root/ephemeral-hdfs/conf to slaves...
ec2-52-91-53-128.compute-1.amazonaws.com
ec2-52-90-108-240.compute-1.amazonaws.com
ec2-52-87-167-96.compute-1.amazonaws.com
ec2-52-23-192-82.compute-1.amazonaws.com
ec2-52-90-53-211.compute-1.amazonaws.com
ec2-52-207-244-243.compute-1.amazonaws.com
ec2-52-91-173-198.compute-1.amazonaws.com
ec2-52-201-244-80.compute-1.amazonaws.com
ec2-52-91-59-6.compute-1.amazonaws.com
ec2-52-23-160-235.compute-1.amazonaws.com
ec2-52-207-241-215.compute-1.amazonaws.com
ec2-52-91-250-41.compute-1.amazonaws.com
ec2-52-207-252-109.compute-1.amazonaws.com
ec2-52-23-188-41.compute-1.amazonaws.com
ec2-52-91-92-6.compute-1.amazonaws.com
ec2-52-91-225-139.compute-1.amazonaws.com
root@ip-172-31-7-6 ~]$ 

```

Screenshot showing config file being copied to all slaves in Spark

The screenshot shows the Spark shell application running on a Firefox browser. The main page displays the following information:

- Total Uptime:** 17 min
- Scheduling Mode:** FIFO
- Active Jobs:** 1
- Completed Jobs:** 1

Below this, there are two sections: "Event Timeline" and "Active Jobs (1)". The "Active Jobs (1)" section shows a table with one row:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	saveAsTextFile at <console>:26	2016/03/28 08:03:28	18 s	0/2	32/1490

At the bottom, there is a "Completed Jobs (1)" section showing another table with one row:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	sortByKey at <console>:23	2016/03/28 08:00:10	53 s	1/1	745/745

## Screenshot showing spark jobs in browser

The screenshot shows the Spark shell application UI running on a Linux desktop. The title bar reads "Spark shell - Spark Jobs - Mozilla Firefox". The main content area displays the "Completed Jobs" section, which lists two completed tasks:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	saveAsTextFile at <console>:26	2016/03/28 08:03:28	9.7 min	2/2	1490/1490
0	sortByKey at <console>:23	2016/03/28 08:00:10	53 s	1/1	745/745

Screenshot showing completion of 100GB sort in browser

Screenshots for top 10 and bottom 10 lines of sorted output file

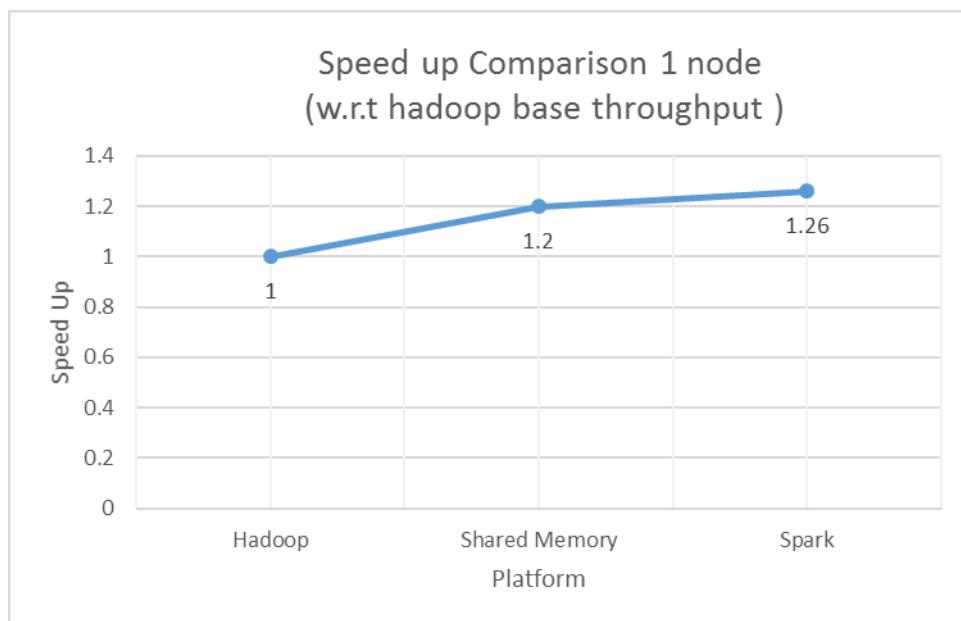
## 8 PERFORMANCE EVALUATION

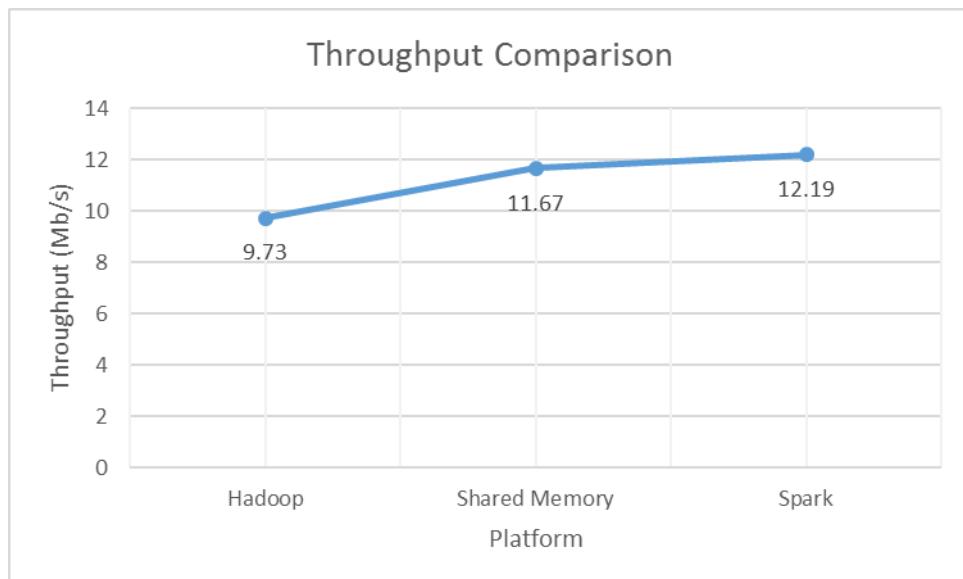
---

Data Size = 10GB

Number of Nodes/ Threads = 1

Platform	Execution Time in Secs.	Throughput in Mb/s	Speed up (w.r.t hadoop base throughput )
Hadoop	1027.882	9.73	1
Shared Memory	857.141	11.67	1.2
Spark	820.437	12.19	1.26

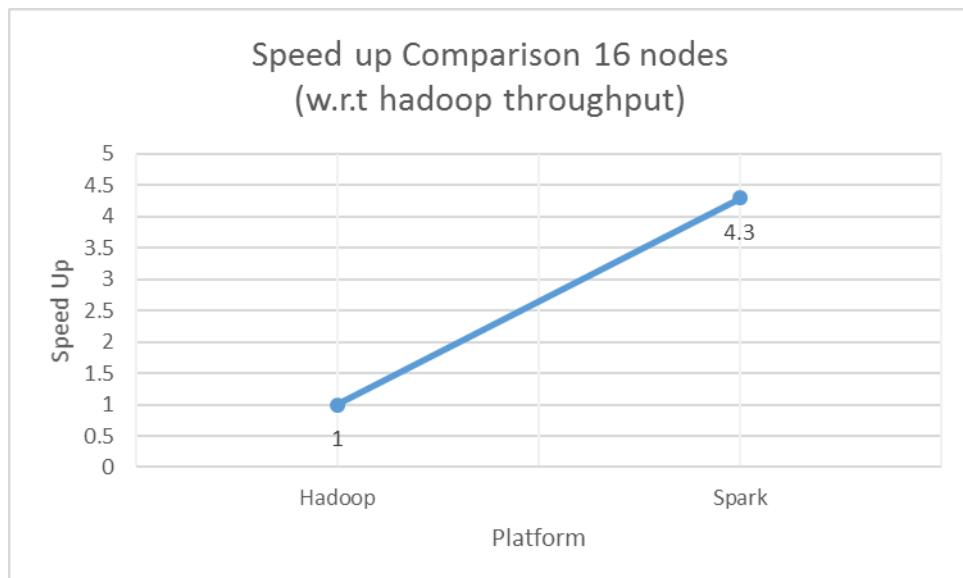




**Data Size = 100GB**

**Number of Nodes/ Threads = 16**

<i>Platform</i>	<i>Execution Time in Secs.</i>	<i>Throughput in Mb/s</i>	<i>Speed up (w.r.t hadoop throughput)</i>
Hadoop	2735	36.57	1
Spark	635	157.48	4.3



### **Result Interpretation:**

For 1 node/1 thread, spark does a fair job and is the fastest to sort 10GB of data. My shared memory program does better than Hadoop. As Hadoop has a lot of overheads it is only suitable for doing data intensive task when data size is quite large. For 100GB data, spark was significantly faster than Hadoop. One reason might be as Spark keeps its final output in several files instead of merging into a single output file like Hadoop. This saves Spark some of the computation cycle which Hadoop lacks.

For 100 node scale, my bets would be on Spark based on the results that we got by performing 16 nodes Hadoop and Spark experiment.

For 1000 nodes, I guess Hadoop would fare better as it doesn't store the intermediate data. Spark might run out of resources in terms of disk space while performing a 1000 node based sort.

### **References:**

- <http://javarevisited.blogspot.com/2014/08/quicksort-sorting-algorithm-in-java-in-place-example.html>
- <http://insightdataengineering.com/blog/hadoopdevops/>
- <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>
- <http://spark.apache.org/docs/latest/ec2-scripts.html>