

**CS553 Cloud Computing**

**Programming Assignment #2**

**Report**

Pranitha Nagavelli(A20345406)

**The following are the important aspects in this assignment:**

- Creating a Virtual Cluster using AMAZON WEB SERVICES(AWS)
- Implementation of Sorting using JAVA, HADOOP and SPARK
- Measuring the performance of the above

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

First we log in into the Amazon AWS service website (<http://aws.amazon.com>) then select EC2 from the network and services section. Select launch instance here. Then we configure the service to meet the requirements. choose the AMI we needed, instance type which can be micro, small, medium, large Xlarge, 2Xlarge, 4Xlarge or 8Xlarge. Then configure the instance i.e choose the instance to be either spot instance or on Demand instance and also can add storage to our selected instance and configure the security group add TCP, UDP and ICMP protocols from the security group. At last then launch the instance

#### **Sort in JAVA:**

Implemented a java program that would be sort large files of 10GB. This is done using External merge sort and merge sort. The files are first divided into equal parts between threads and then send that file into each thread it is further divided into sub files which are sent into merge sort method where that file is sorted and then all these files are merged using external sort algorithm or approach.

AWS Region: (Oregon)west

#### **Sort in HADOOP:**

In Hadoop dataset produced from gensort of size 10GB and 100GB/1Tb has to be sorted which is achieved by implementing MapReduce code where the each line is divided into key and values. Each key is of first 10 bytes and remaining as values. Now this key value pairs are used in this sorting application.

AWS Region: (Oregon)west

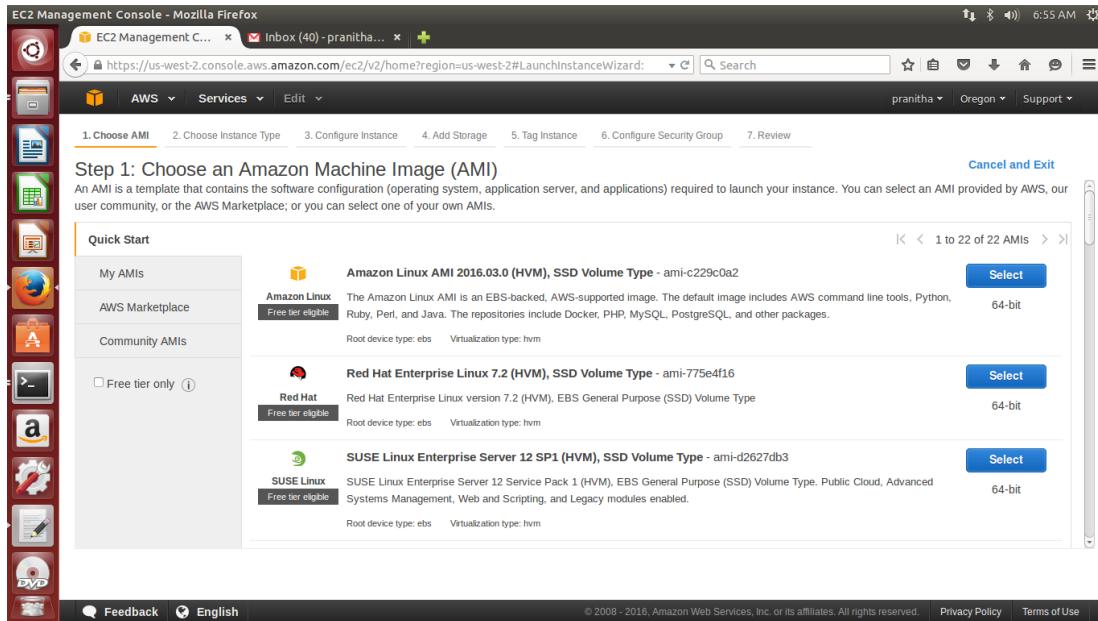
#### **Sort in SPARK:**

In spark dataset produced from gensort of size 10GB and 100GB/1Tb has to be sorted which is achieved by using some of the inbuilt functions line ‘sortByKey()’ where the each line is divided into key and values. Each key is of first 10 bytes and remaining as values. Now this key value pairs are used in this sorting application.

AWS Region: (N.Virginia)east

## Creating a AWS Instance:

- Log in to Amazon AWS console, under services select EC2
- Now launch an instance, that best AMI that suits the requirement



- Then choose the instance type which can be micro, small, medium, large Xlarge, 2Xlarge, 4Xlarge or 8Xlarge. For our assignment it is c3.large for 10GB data

**Step 2: Choose an Instance Type**

| General purpose | Compute optimized | CPU | RAM  | Storage       | Networking | Monitoring | Compute Optimized |
|-----------------|-------------------|-----|------|---------------|------------|------------|-------------------|
| m3.2xlarge      | c4.large          | 8   | 30   | 2 x 80 (SSD)  | Yes        | High       |                   |
|                 | c4.xlarge         | 2   | 3.75 | EBS only      | Yes        | Moderate   |                   |
|                 | c4.2xlarge        | 4   | 7.5  | EBS only      | Yes        | High       |                   |
|                 | c4.4xlarge        | 8   | 15   | EBS only      | Yes        | High       |                   |
|                 | c4.8xlarge        | 16  | 30   | EBS only      | Yes        | High       |                   |
|                 | c3.large          | 36  | 60   | EBS only      | Yes        | 10 Gigabit |                   |
|                 | c3.xlarge         | 2   | 3.75 | 2 x 16 (SSD)  | -          | Moderate   |                   |
|                 | c3.2xlarge        | 4   | 7.5  | 2 x 40 (SSD)  | Yes        | Moderate   |                   |
|                 | c3.4xlarge        | 8   | 15   | 2 x 80 (SSD)  | Yes        | High       |                   |
|                 | c3.8xlarge        | 16  | 30   | 2 x 160 (SSD) | Yes        | High       |                   |
|                 | c3.8xlarge        | 32  | 60   | 2 x 320 (SSD) | -          | 10 Gigabit |                   |

**Step 2: Choose an Instance Type**

Cancel Previous Review and Launch Next: Configure Instance Details

- Then we configure the instance i.e. we can choose the instance to be either spot instance or on Demand instance. In this assignment spot instance has been used.

**Step 3: Configure Instance Details**

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1      Launch into Auto Scaling Group

Purchasing option: Request Spot Instances

Current price: us-west-2a0.0182, us-west-2b0.0182, us-west-2c0.0183

Maximum price: \$ 0.3

Launch group: (Optional)

Request valid from: Any time

Request valid to: Any time

Persistent request:

Network: vpc-eaadb48f (172.31.0.0/16) (default)      Create new VPC

Subnet: No preference (default subnet in any Availability Zone)      Create new subnet

Cancel Previous Review and Launch Next: Add Storage

- Add additional storage to selected instance.

**Step 4: Add Storage**

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

| Volume Type | Device    | Snapshot      | Size (GiB) | Volume Type               | IOPS      | Delete on Termination               | Encrypted     |
|-------------|-----------|---------------|------------|---------------------------|-----------|-------------------------------------|---------------|
| Root        | /dev/xvda | snap-eef4cd4e | 8          | General Purpose SSD (GP2) | 24 / 3000 | <input checked="" type="checkbox"/> | Not Encrypted |

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel Previous Review and Launch Next: Tag Instance

- Configure the security group. Add TCP, UDP and ICMP protocols from the security group. This step is optional for 1 node .

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  Select an existing security group

Security group name:

Description:

| Type     | Protocol | Port Range | Source  |
|----------|----------|------------|---|
| SSH      | TCP      | 22         | Anywhere <input type="text" value="0.0.0.0/0"/> |
| All TCP  | TCP      | 0 - 65535  | Custom IP <input type="text"/>                  |
| All ICMP | ICMP     | 0 - 65535  | Custom IP <input type="text"/>                  |

Add Rule

**Warning**  
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous Review and Launch

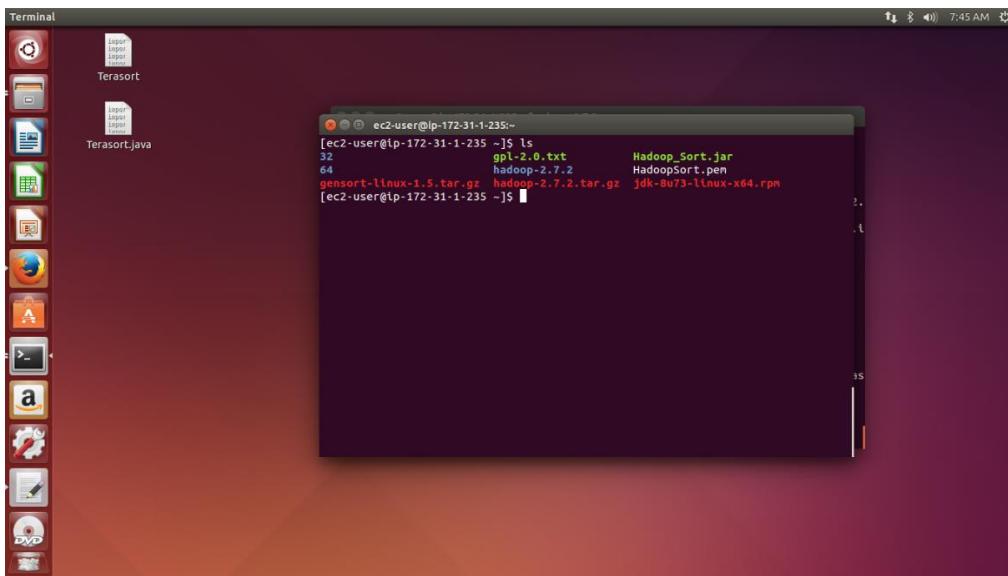
- Launch the spot instance once the request is accepted the instance will start

To connect to the node:

- Chmod 400 keypair<i>
- Ssh -I keypair<i> username@ip address.  
chmod 400 "HadoopSort.pem".  
ssh -i "HadoopSort.pem" ec2-user@ec2-54-191-132-210.us-west2.compute.amazonaws.com

After the connection is established:

- Update java using following commands  
wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u73-b02/jdk-8u73-linux-x64.rpm  
sudo rpm -ivh jdk-8u73-linux-x64.rpm
- Using wget command only download "hadoop-2.7.2.tar.gz" and un tar the file
- Also download "gensort-linux-1.5.tar.gz" and un tar it.
- Get the Hadoop executable MapReduce program jar and key pair file into the instance.



## 1. SharedMemory:

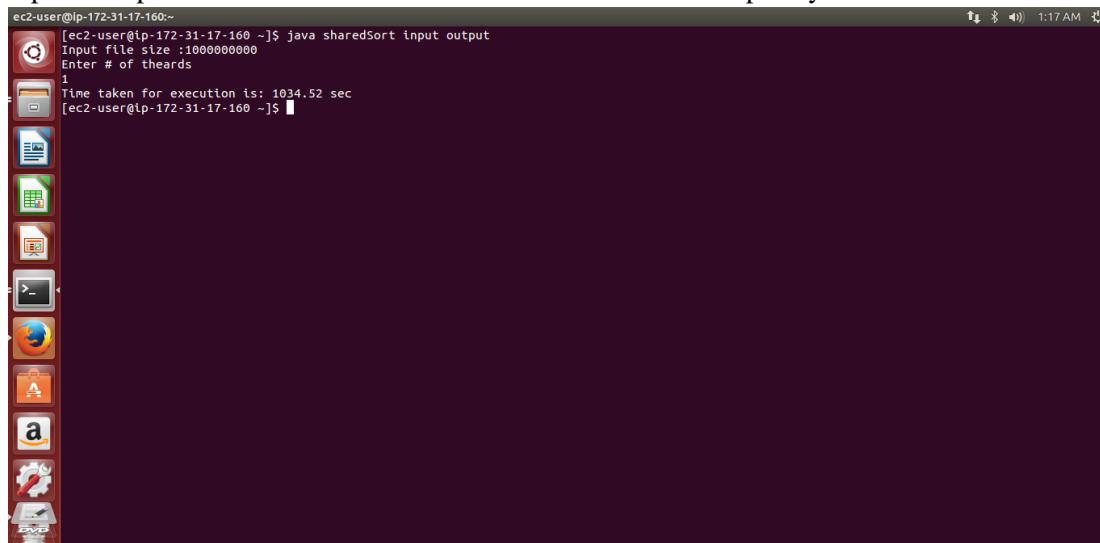
### System Configuration:

JAVA VERSION: 1.8

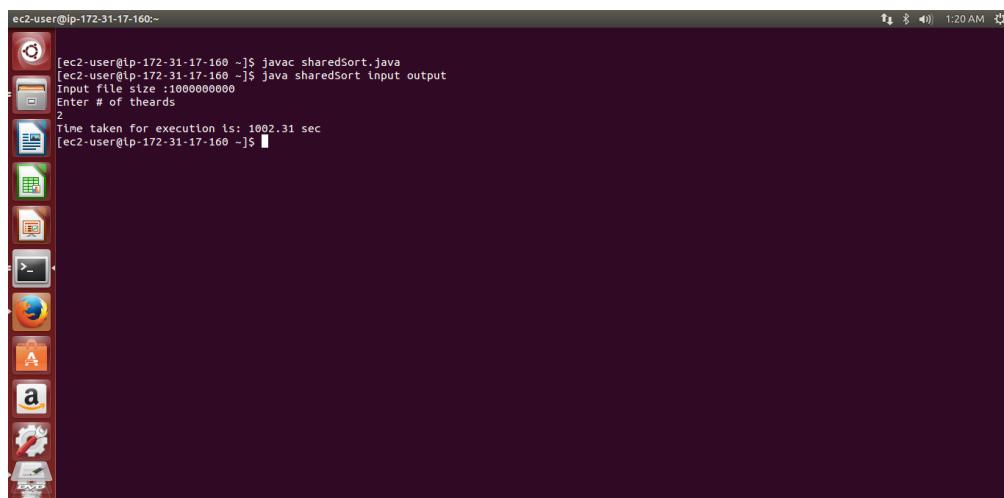
INSTANCE TYPE: c3. Large (10GB) UBUNTU

### Implementation:

- Given an input file, output file path, and number of threads to be implemented
- First the file size divides the file into equal parts depending on number of threads
- Now send input, output, fileptr, threads as arguments which will be used later inside the thread
- Now in thread start reading the input file line by line until it reaches the size or memory is full then write it into the temporary files, clear the map
- Do this for all files, then in merge sort all the files that are been spitted are sorted and stored and now are ready for merging.
- In merge file method from all files take minimum and write into the final file using seek function repeat this process until all files are done and delete all the temporary files



```
[ec2-user@ip-172-31-17-160:~]$ java sharedSort input output
Input file size :1000000000
Enter # of threads
1
Time taken for execution is: 1034.52 sec
[ec2-user@ip-172-31-17-160 ~]$
```



```
[ec2-user@ip-172-31-17-160:~]$ javac sharedSort.java
[ec2-user@ip-172-31-17-160:~]$ java sharedSort input output
Input file size :1000000000
Enter # of threads
2
Time taken for execution is: 1002.31 sec
[ec2-user@ip-172-31-17-160 ~]$
```

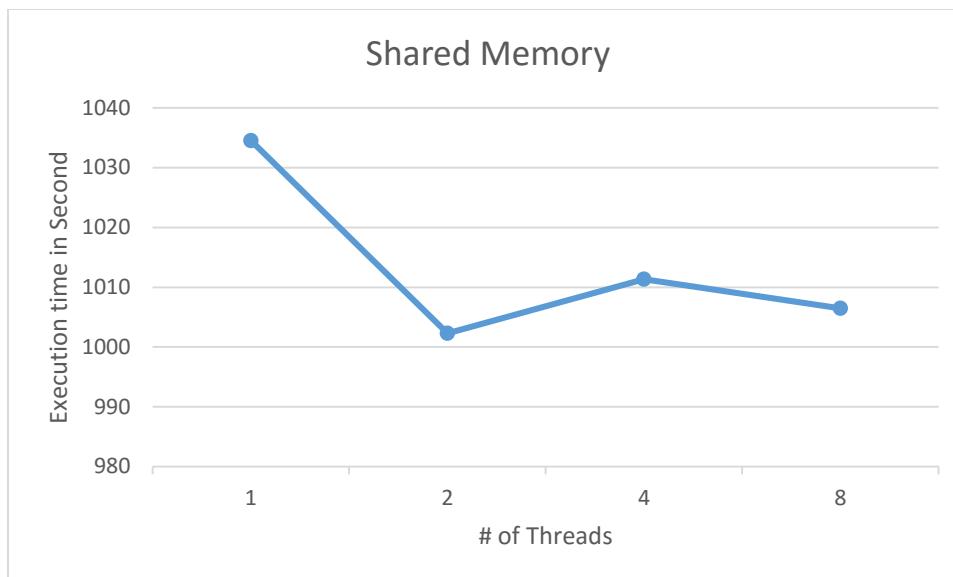
```
ec2-user@ip-172-31-17-160:~ [ec2-user@ip-172-31-17-160 ~]$ javac sharedSort.java
[ec2-user@ip-172-31-17-160 ~]$ java sharedSort input output
Input file size :10000000000
Enter # of threads
4
Time taken for execution is: 1011.34 sec
[ec2-user@ip-172-31-17-160 ~]$
```

```
ec2-user@ip-172-31-17-160:~ [ec2-user@ip-172-31-17-160 ~]$ javac sharedSort.java
[ec2-user@ip-172-31-17-160 ~]$ java sharedSort input output
Input file size :10000000000
Enter # of threads
8
Time taken for execution is: 1006.49 sec
[ec2-user@ip-172-31-17-160 ~]$
```

## Performance:

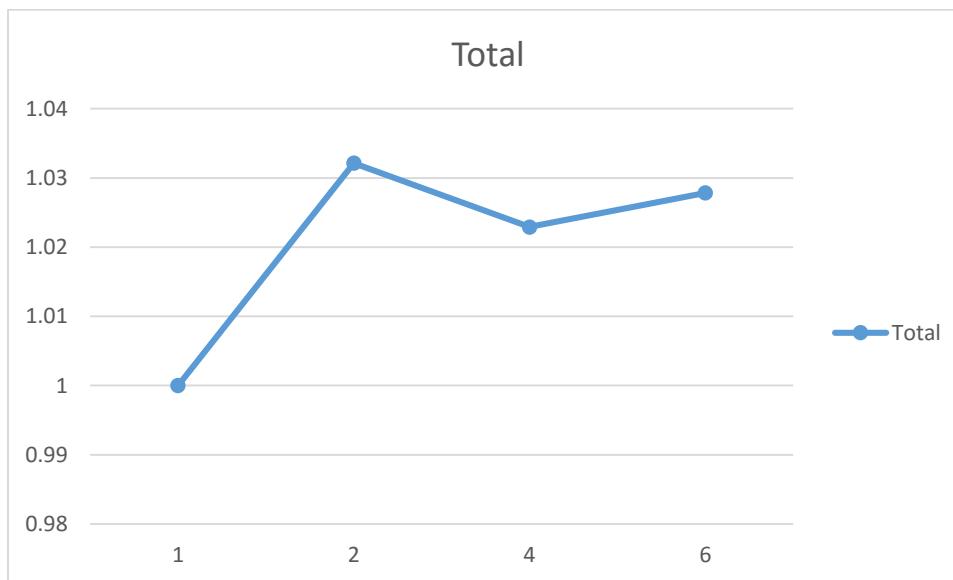
### 10GB dataset

| # of threads | Execution time in sec |
|--------------|-----------------------|
| 1            | 1034.52               |
| 2            | 1002.31               |
| 4            | 1011.34               |
| 8            | 1006.49               |



#### Shared Memory Speedup using 1 thread value

| # of thread | Execution time | 2 Thread value | Speed-up factor   |
|-------------|----------------|----------------|-------------------|
| 1           | <b>1034.52</b> | <b>1034.52</b> | <b>1</b>          |
| 2           | <b>1002.31</b> | <b>1034.52</b> | <b>1.03213577</b> |
| 4           | <b>1011.34</b> | <b>1034.52</b> | <b>1.02292009</b> |
| 8           | <b>1006.49</b> | <b>1034.52</b> | <b>1.02784926</b> |



For 1 node 10 GB data the best performance is achieved at 2 threads

## 2.Hadoop:

### System Configuration:

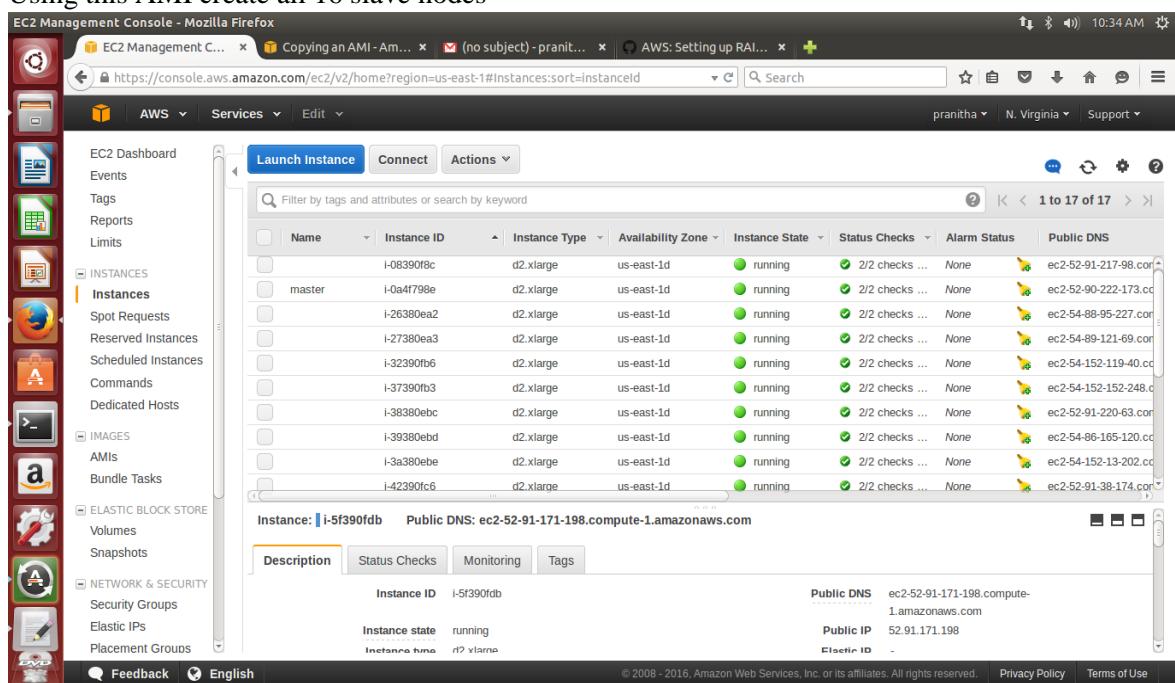
VERSION: Hadoop 2.7.2

INSTANCE TYPE: c3. Large (10GB&100GB) & d2.Xlarge(1TB) UBUNTU

### Hadoop and Aws Configuration:

#### For 16 Node AWS cluster:

- Create a instance with all files and configuration take an image of the instance which acts as a master to our 16 node cluster
- Using this AMI create all 16 slave nodes



The screenshot shows the AWS EC2 Management Console interface. On the left, there's a sidebar with various navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, Elastic Block Store, and Network & Security. The 'Instances' link is currently selected. The main area displays a table of 17 instances. The first instance, labeled 'master', is highlighted with a yellow background. The table columns include Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, and Public DNS. Below the table, there's a summary section for the master instance showing its Public DNS (ec2-52-91-171-198.compute-1.amazonaws.com), Public IP (52.91.171.198), and Elastic IP. At the bottom of the page, there are links for Feedback, English, Privacy Policy, and Terms of Use.

### Configuration of Hadoop:

Files that have to be modified to complete the Hadoop setup

- ./bashrc
- Core-site.xml
- Hdfs-site.xml
- Hadoop-env.sh
- Mapred-site.xml
- Yarn-site.xml
- Slaves

#### 1. ~./bashrc:

```
#HADOOP VARIABLES START
```

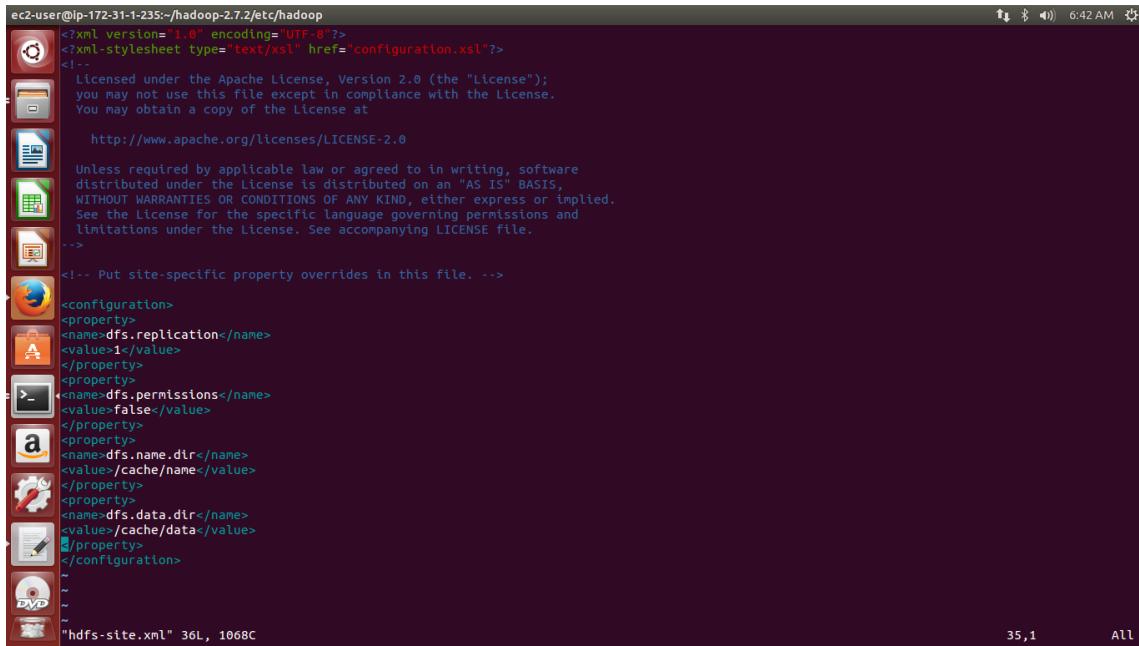
```
export JAVA_HOME=/usr/java/latest  
  
export HADOOP_INSTALL=/usr/ubuntu/install/hadoop  
  
export PATH=$PATH:$HADOOP_INSTALL/bin  
  
export PATH=$PATH:$HADOOP_INSTALL/sbin  
  
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL  
  
export HADOOP_COMMON_HOME=$HADOOP_INSTALL  
  
export HADOOP_HDFS_HOME=$HADOOP_INSTALL  
  
export YARN_HOME=$HADOOP_INSTALL  
  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native  
  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
```

## 2.core-site.xml

- Same configuration for both 1 node and 16 node cluster.
  - In 16 node cluster all the 16 slaves will have master configuration only

### 3.hdfs-site.xml

For 1 node

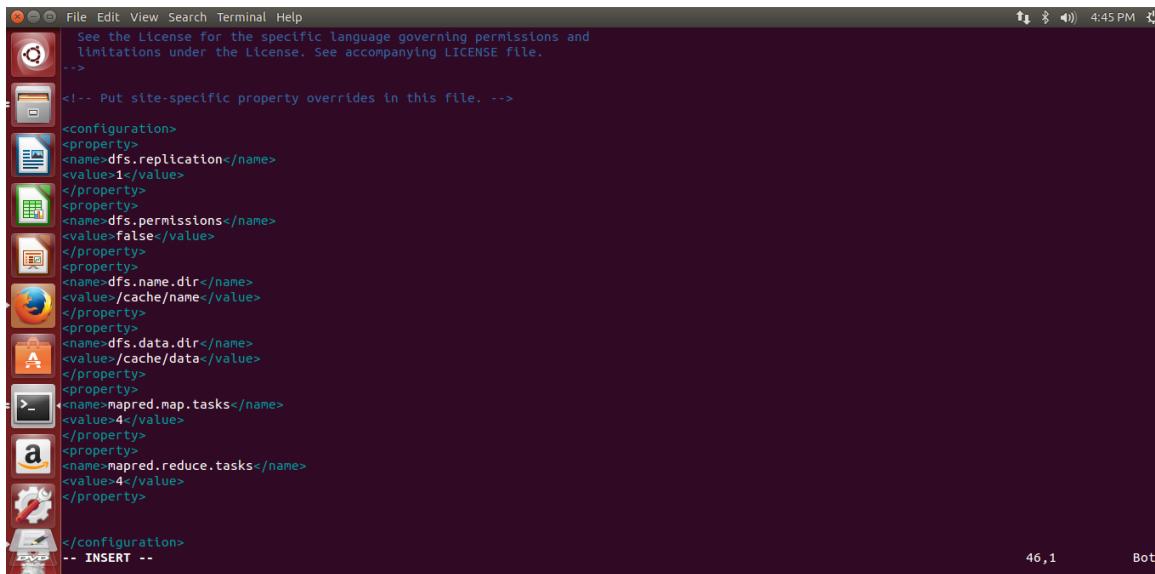


```
ec2-user@ip-172-31-1-235:~/hadoop-2.7.2/etc/hadoop
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.name.dir</name>
<value>/cache/name</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>/cache/data</value>
</property>
</configuration>
~
~
~
"hdfe-site.xml" 36L, 1068C
```

For 16 node



```
File Edit View Search Terminal Help
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.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.name.dir</name>
<value>/cache/name</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>/cache/data</value>
</property>
<property>
<name>mapred.map.tasks</name>
<value>4</value>
</property>
<property>
<name>mapred.reduce.tasks</name>
<value>4</value>
</property>
</configuration>
-- INSERT --
```

- Same configuration for both 1 node and 16 node cluster.
- '/cache' is the directory where the disk is mounted using following commands  
1 Node: can just add a EBSvolume and attach it to the instance

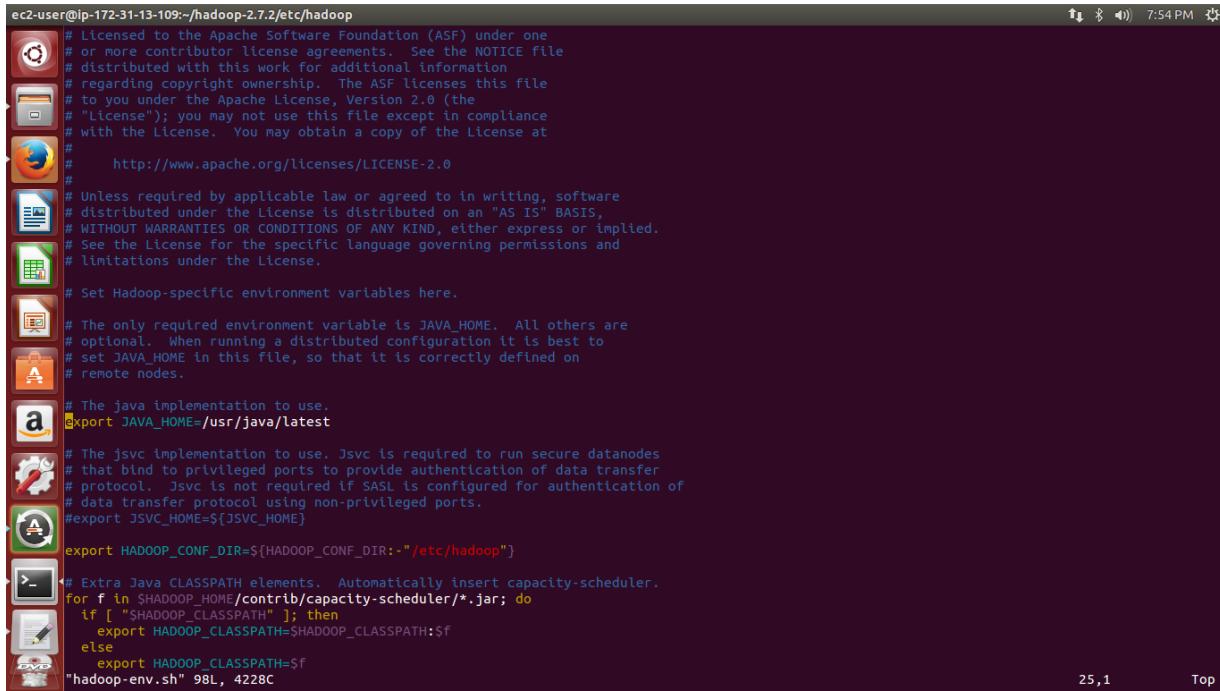
```
16 Node: sudo mdadm --create --verbose /dev/md0 --level=stripe --raid-devices=3 /dev/sdb  
/dev/sdc /dev/sdd
```

Followed by :

```
sudo mkfs.ext4 /dev/md0
```

```
sudo mount -t ext4 -o noatime,nodiratime,rw /dev/md0 /cache
```

#### 4.Hadoop-env.sh



```
ec2-user@ip-172-31-13-109:~/hadoop-2.7.2/etc/hadoop  
# Licensed to the Apache Software Foundation (ASF) under one  
# or more contributor license agreements. See the NOTICE file  
# distributed with this work for additional information  
# regarding copyright ownership. The ASF licenses this file  
# to you under the Apache License, Version 2.0 (the  
# "License"); you may not use this file except in compliance  
# with the License. You may obtain a copy of the License at  
#  
#     http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
  
# Set Hadoop-specific environment variables here.  
  
# The only required environment variable is JAVA_HOME. All others are  
# optional. When running a distributed configuration it is best to  
# set JAVA_HOME in this file, so that it is correctly defined on  
# remote nodes.  
  
# The java implementation to use.  
export JAVA_HOME=/usr/java/latest  
  
# 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}  
  
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}  
  
# Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.  
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do  
    if [ "$HADOOP_CLASSPATH" ]; then  
        export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f  
    else  
        export HADOOP_CLASSPATH=$f  
    fi  
done  
"hadoop-env.sh" 98L, 4228C
```

25,1

Top

#### 5.mapred-site

For 1 node

```

ec2-user@ip-172-31-1-235:~/hadoop-2.7.2/etc/hadoop
  <?xml version='1.0'?>
  <!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>mapreduce.jobtracker.address</name>
<value>hdfs://ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8021</value>
<description>The host and port that the MapReduce job tracker runs at. If "local" then jobs are run as single map and reduce task </description>
</property>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapred.map.tasks</name>
<value>4</value>
</property>
<property>
<name>mapred.reduce.tasks</name>
<value>1</value>
</property>
</configuration>
-
"mapred-site.xml" 38L, 1274C

```

For 16 node

```

File Edit View Search Terminal Help
  <!--
    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://ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8021</value>
<description>The host and port that the MapReduce job tracker runs at. If "local" then jobs are run as single map and reduce task </description>
</property>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
  <name>mapreduce.map.memory.mb</name>
  <value>4096</value>
</property>
<property>
  <name>mapreduce.reduce.memory.mb</name>
  <value>8192</value>
</property>
<property>
  <name>mapreduce.map.java.opts</name>
  <value>-Xmx3072m</value>
</property>
<property>
  <name>mapreduce.reduce.java.opts</name>
  <value>-Xmx6144m</value>
</property>
</configuration>
-- INSERT --

```

6.yarn-site.xml

For 1 node

```

ec2-user@ip-172-31-1-235:~/hadoop-2.7.2/etc/hadoop
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

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

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.

  ...
  <configuration>
    ...
    <!-- Site specific YARN configuration properties -->
    <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
    </property>
    <property>
      <name>yarn.resourcemanager.scheduler.address</name>
      <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8030</value>
    </property>
    <property>
      <name>yarn.resourcemanager.address</name>
      <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8032</value>
    </property>
    <property>
      <name>yarn.resourcemanager.webapp.address</name>
      <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8088</value>
    </property>
    <property>
      <name>yarn.resourcemanager.resource-tracker.address</name>
      <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8031</value>
    </property>
    <property>
      <name>yarn.resourcemanager.admin.address</name>
      <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8033</value>
    </property>
  </configuration>
  "yarn-site.xml" 42L, 1503C

```

42,1 Bot

for 16 node

```

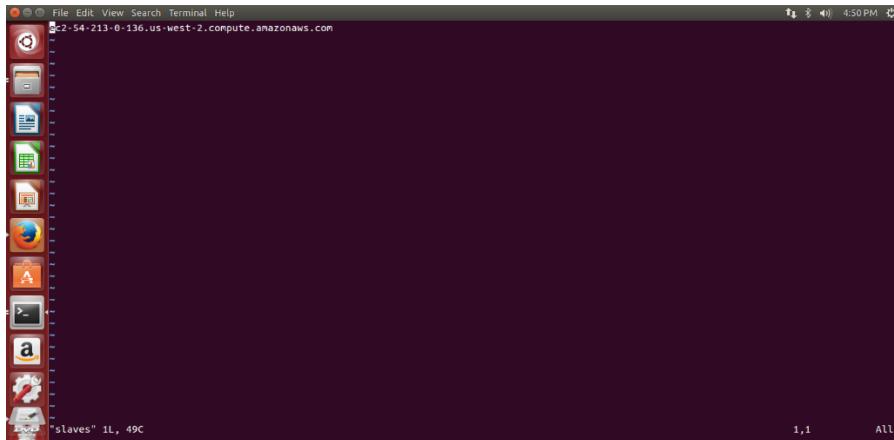
File Edit View Search Terminal Help
  ...
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8030</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8032</value>
  </property>
  <property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8088</value>
  </property>
  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8031</value>
  </property>
  <property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8033</value>
  </property>
  ...
  <property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
  </property>
  <property>
    <name>yarn.nodemanager.vmem-pmem-ratio</name>
    <value>4</value>
    <description>Ratio between virtual memory to physical memory when setting memory limits for containers</description>
  </property>
  ...
  </configuration>
  -- INSERT --

```

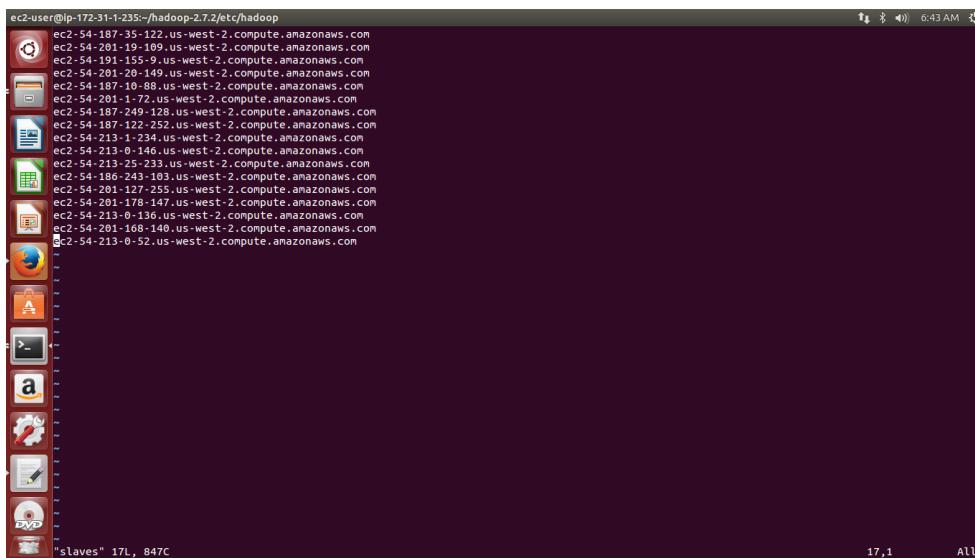
51,3 Bot

7.slaves

1 node and 16 slave node(each slave will have its own ip address)



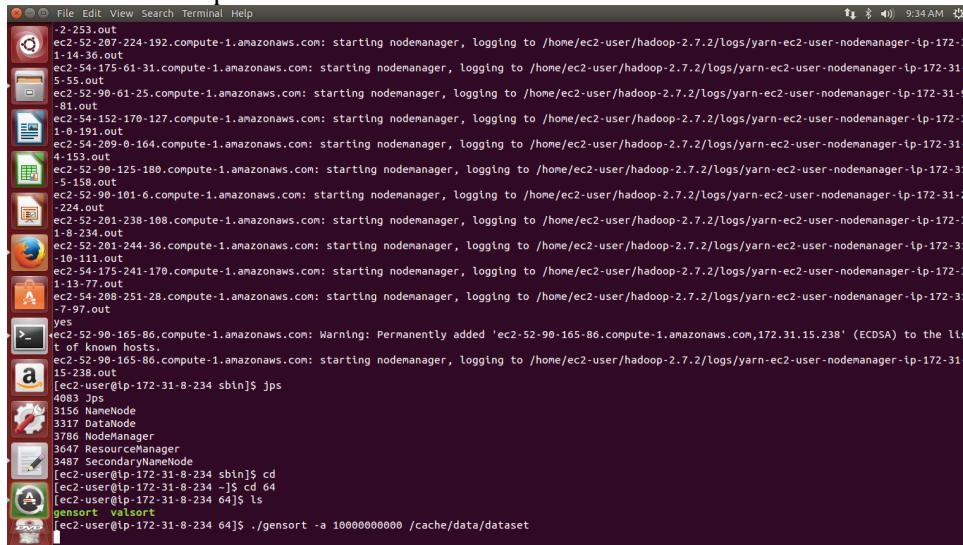
Master node of 16 node cluster



## Implementation:

1. Hadoop MapReduce has been implemented in java.
2. Launch the instance and do all necessary implementation stated above
3. Once the implementations are done, add storage depending on file size to be sorted
4. Get the jar file that has been implemented using SCP command from local system to instance.
5. Create a dataset of required size from gensort and put it in HDFS using put command
6. Now using the jar file start sorting the dataset
7. Hadoop Map Reduce: MapReduce programs are designed to compute large volumes of data in a parallel fashion. This requires dividing the workload across a two or single nodes.
8. Map Phase: In this phase input data is divided into input splits for analysis by map tasks running in parallel across the Hadoop cluster.
9. Reduce Phase: The results from map tasks are used as input to a set of parallel reduce tasks. The reduce tasks consolidate the data into final results.

10. In between this mapper and reducer phase there is a shuffle/sort phase where the data is sorted depending on the key values and is sent in to the mapper.
  11. The output is sent into the HDFS which can be accessed using get command
  12. This output value is checked using ./valsor in gensort
- 1) What is a Master node? What is a Slaves node?
    - The master node is the one which stores a lot of data and runs parallel computations on data, it controls all the slaves. It keeps track of all the slaves it controls namenode and datanode, it divides the work between the slaves.
    - The slave node has only information about itself it follows the commands of the master takes up the assigned task by the master, it does the work of storing data and running the computations
  - 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?
    - If ports are same it would interfere with the operations or computations of each other, that in shared environment, multiple processes will be running if they share same port it lead insecure operations along with it the operations are interfered and also will consume more time.
  - 3) How can we change the number of mappers and reducers from the configuration file?
    - It can be done by adding two properties mapred.map.tasks and mapred.reduce.task in mapred-site.xml



```

-2-253.out
ec2-52-207-224-192.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-3
1-14-36.out
ec2-54-175-61-31.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-31
5-158.out
ec2-52-90-61-25.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-31-9
81.out
ec2-54-152-170-127.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-3
1-0-191.out
ec2-54-209-0-164.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-31-4
4-153.out
ec2-52-90-125-188.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-31-5
-5-158.out
ec2-52-90-101-6.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-31-224.out
ec2-54-201-238-108.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-3
-8-234.out
ec2-52-201-244-36.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-31-10-111.out
ec2-54-175-241-170.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-3
1-13-77.out
ec2-54-208-251-28.compute-1.amazonaws.com: starting nodemanager, logging to /home/ec2-user/hadoop-2.7.2/logs/yarn-ec2-user-nodemanager-tp-172-31-7-97.out
yes
[ec2-user@ip-172-31-8-234 sbin]$ jps
4083 Jps
3156 NameNode
3317 DataNode
3786 NodeManager
3647 ResourceManager
3487 SecondaryNameNode
[ec2-user@ip-172-31-8-234 sbin]$ cd
[ec2-user@ip-172-31-8-234 ~]$ cd 64
[ec2-user@ip-172-31-8-234 64]$ ls
gensort valsor
[ec2-user@ip-172-31-8-234 64]$ ./gensort -a 10000000000 /cache/data/dataset

```

```

ec2-user@ip-172-31-8-222:/hadoop-2.7.2
[ec2-user@ip-172-31-8-222 ~]$ cd hadoop-2.7.2
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop dfs -ls /
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 1 items
-rw-r--r-- 1 ec2-user supergroup 100000000000 2016-03-27 04:55 /dataset
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar /ec2-user/Hadoop_Sort.jar /dataset /output
Not a valid JAR: /ec2-user/Hadoop_Sort.jar
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ cd
[ec2-user@ip-172-31-8-222 ~]$ ls
32 64 gensorTlinux-1.5.tar.gz gpl-2.0.txt hadoop-2.7.2 hadoop-2.7.2.tar.gz Hadoop_Sort.jar Sort.pem
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar /root/Hadoop_Sort.jar /dataset /output
Not a valid JAR: /root/Hadoop_Sort.jar
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar Hadoop_Sort.jar /dataset /output
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar /home/ec2-user/Hadoop_Sort.jar /dataset /output
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar /home/ec2-user/Hadoop_Sort.jar /dataset /output
16/03/27 05:00:55 INFO client.RMProxy: Connecting to ResourceManager at ec2-52-91-92-249.compute-1.amazonaws.com:172.31.8.222:8032
16/03/27 05:00:56 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/03/27 05:00:56 INFO Input.FileInputFormat: Total input paths to process : 1
16/03/27 05:00:57 INFO mapreduce.JobSubmissionмаркет: number of splits:75
16/03/27 05:00:57 INFO mapreduce.JobSubmissionмаркет: Submitting tokens for job: job_1459054015570_0001
16/03/27 05:00:58 INFO impl.YarnClientImpl: Submitted application application_1459054015570_0001
16/03/27 05:00:58 INFO mapreduce.Job: The url to track the job: http://ec2-52-91-92-249.compute-1.amazonaws.com:8088/proxy/application_1459054015570_0001
16/03/27 05:00:58 INFO mapreduce.Job: Job: Running job: job_1459054015570_0001 running in uber mode : false
16/03/27 05:01:07 INFO mapreduce.Job: map 0% reduce 0%
16/03/27 05:01:31 INFO mapreduce.Job: map 1% reduce 0%
16/03/27 05:01:34 INFO mapreduce.Job: map 2% reduce 0%
16/03/27 05:01:37 INFO mapreduce.Job: map 3% reduce 0%
16/03/27 05:01:42 INFO mapreduce.Job: map 4% reduce 0%
16/03/27 05:01:46 INFO mapreduce.Job: map 5% reduce 0%
16/03/27 05:02:02 INFO mapreduce.Job: map 6% reduce 0%
16/03/27 05:02:08 INFO mapreduce.Job: map 7% reduce 0%
16/03/27 05:02:12 INFO mapreduce.Job: map 8% reduce 0%
16/03/27 05:02:38 INFO mapreduce.Job: map 10% reduce 0%
16/03/27 05:02:41 INFO mapreduce.Job: map 11% reduce 0%

```

```

ec2-user@ip-172-31-8-222:/hadoop-2.7.2
Total time spent by all maps in occupied slots (ms)=4615336
Total time spent by all reduces in occupied slots (ms)=982777
Total time spent by all map tasks (ms)=4615336
Total time spent by all reduce tasks (ms)=982777
Total vcore-milliseconds taken by all map tasks=4615336
Total vcore-milliseconds taken by all reduce tasks=982777
Total megabyte-milliseconds taken by all map tasks=4726104064
Total megabyte-milliseconds taken by all reduce tasks=1006363648
Map-Reduce Framework
  Map Input records=1000000000
  Map output records=1000000000
  Map output bytes=9000000000
  Map output materialized bytes=101000000450
  Input split bytes=9375
  Combine input records=0
  Combine output records=0
  Failed map outputs=0
  Failed reduce outputs=0
  Reduce shuffle bytes=101000000450
  Reduce input records=1000000000
  Reduce output records=1000000000
  Spilled Records=396636763
  Shuffled Maps =75
  Failed Shuffles=0
  Merged Map outputs=75
  GC time elapsed (ms)=39541
  DU time spent (ms)=1223240
  Physical memory (bytes) snapshot=188880479232
  Virtual memory (bytes) snapshot=160216629248
  Total committed heap usage (bytes)=15486943232
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=10000303104
  File Output Format Counters
    Bytes Written=990000000000
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$

```

```

ec2-user@ip-172-31-1-235:/hadoop-2.7.2
16/03/30 18:17:25 INFO mapreduce.Job: map 74% reduce 16%
16/03/30 18:18:53 INFO mapreduce.Job: map 75% reduce 16%
16/03/30 18:19:05 INFO mapreduce.Job: map 75% reduce 17%
16/03/30 18:20:25 INFO mapreduce.Job: map 76% reduce 17%
16/03/30 18:21:54 INFO mapreduce.Job: map 77% reduce 17%
16/03/30 18:23:23 INFO mapreduce.Job: map 78% reduce 17%
16/03/30 18:24:57 INFO mapreduce.Job: map 79% reduce 17%
16/03/30 18:25:22 INFO mapreduce.Job: map 79% reduce 18%
16/03/30 18:26:47 INFO mapreduce.Job: map 80% reduce 18%
16/03/30 18:28:08 INFO mapreduce.Job: map 81% reduce 18%
16/03/30 18:29:26 INFO mapreduce.Job: map 82% reduce 18%
16/03/30 18:30:56 INFO mapreduce.Job: map 83% reduce 18%
16/03/30 18:31:37 INFO mapreduce.Job: map 83% reduce 19%
16/03/30 18:32:25 INFO mapreduce.Job: map 84% reduce 19%
16/03/30 18:33:53 INFO mapreduce.Job: map 85% reduce 19%
16/03/30 18:35:21 INFO mapreduce.Job: map 86% reduce 19%
16/03/30 18:36:53 INFO mapreduce.Job: map 87% reduce 19%
16/03/30 18:37:57 INFO mapreduce.Job: map 87% reduce 20%
16/03/30 18:38:23 INFO mapreduce.Job: map 88% reduce 20%
16/03/30 18:39:55 INFO mapreduce.Job: map 89% reduce 20%
16/03/30 18:41:22 INFO mapreduce.Job: map 90% reduce 20%
16/03/30 18:42:57 INFO mapreduce.Job: map 91% reduce 20%
16/03/30 18:44:32 INFO mapreduce.Job: map 92% reduce 21%
16/03/30 18:44:58 INFO mapreduce.Job: map 93% reduce 21%
16/03/30 18:45:57 INFO mapreduce.Job: map 94% reduce 21%
16/03/30 18:47:26 INFO mapreduce.Job: map 94% reduce 21%
16/03/30 18:48:56 INFO mapreduce.Job: map 95% reduce 21%
16/03/30 18:50:17 INFO mapreduce.Job: map 95% reduce 22%
16/03/30 18:50:27 INFO mapreduce.Job: map 96% reduce 22%
16/03/30 18:51:53 INFO mapreduce.Job: map 97% reduce 22%
16/03/30 18:53:23 INFO mapreduce.Job: map 98% reduce 22%
16/03/30 18:54:56 INFO mapreduce.Job: map 99% reduce 22%
16/03/30 18:56:27 INFO mapreduce.Job: map 100% reduce 22%
16/03/30 18:56:38 INFO mapreduce.Job: map 100% reduce 23%

```

```
ec2-user@ip-172-31-1-235:~/hadoop-2.7.2
16/03/31 00:55:34 INFO mapreduce.Job: map 100% reduce 92%
16/03/31 00:55:34 INFO mapreduce.Job: map 100% reduce 93%
16/03/31 00:57:33 INFO mapreduce.Job: map 100% reduce 94%
16/03/31 00:59:32 INFO mapreduce.Job: map 100% reduce 95%
16/03/31 01:01:29 INFO mapreduce.Job: map 100% reduce 96%
16/03/31 01:03:27 INFO mapreduce.Job: map 100% reduce 97%
16/03/31 01:06:24 INFO mapreduce.Job: map 100% reduce 98%
16/03/31 01:09:25 INFO mapreduce.Job: map 100% reduce 99%
16/03/31 01:11:20 INFO mapreduce.Job: Job job_1450371506174_0001 completed successfully
16/03/31 01:11:29 INFO mapreduce.Job: Counters: 51
  File System Counters
    FILE: Number of bytes read=3026732087790
    FILE: Number of bytes written=4037517156353
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=1000031513634
    HDFS: Number of bytes written=9900000000000
    HDFS: Number of read operations=22365
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=8
  Job Counters
    Killed map tasks=2
    Launched map tasks=7453
    Launched reduce tasks=4
    Data-local map tasks=1733
    Rack-local map tasks=5720
    Total time spent by all map tasks in occupied slots (ms)=447937984
    Total time spent by all map tasks (ms)=432963976
    Total time spent by all reduce tasks (ms)=54128497
    Total vcore-milliseconds taken by all map tasks=11984496
    Total vcore-milliseconds taken by all reduce tasks=54120497
    Total megabyte-milliseconds taken by all map tasks=458688495616
```

```
ec2-user@ip-172-31-1-235:~/hadoop-2.7.2
  Total megabyte-milliseconds taken by all map tasks=458688495616
  Total megabyte-milliseconds taken by all reduce tasks=443355111424
Map-Reduce Framework
  Map input records=100000000000
  Map output records=100000000000
  Map output bytes=990000000000
  Map output materialized bytes=1010000178824
  Input split bytes=998434
  Combine input records=0
  Combine output records=0
  Reduce input groups=100000000000
  Reduce shuffle bytes=1010000178824
  Reduce input records=100000000000
  Reduce output records=100000000000
  Spilled Records=39966729135
  Shuffled Maps =29804
  Failed Shuffles=0
  Merged Map outputs=29804
  GC time elapsed (ms)=1410745
  CPU time spent (ms)=110929690
  Physical memory (bytes) snapshot=5875777015808
  Virtual memory (bytes) snapshot=39106445713408
  Total committed heap usage (bytes)=6471886569472
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=10000030515200
File Output Format Counters
  Bytes Written=9900000000000
[ec2-user@ip-172-31-1-235 hadoop-2.7.2]$
```

All Applications - Mozilla Firefox

EC2 Management C... (no subject) - pranit... All Applications

ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8088/cluster

**hadoop**

## All Applications

**Cluster Metrics**

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | Vcores Used | Vcores Total | Vcores Reserved | Active Nodes | Decommissioned Nodes |
|----------------|--------------|--------------|----------------|--------------------|-------------|--------------|-----------------|-------------|--------------|-----------------|--------------|----------------------|
| 1              | 0            | 1            | 0              | 2                  | 10 GB       | 136 GB       | 0 B             | 2           | 136          | 0               | 17           | 0                    |

**Scheduler Metrics**

| Scheduler Type     | Scheduling Resource Type | Minimum Allocation      | Maximum Allocation      |
|--------------------|--------------------------|-------------------------|-------------------------|
| Capacity Scheduler | [MEMORY]                 | <memory:1024, vCores:1> | <memory:8192, vCores:1> |

Show 20 entries

| ID                             | User     | Name | Application Type | Queue   | StartTime                      | FinishTime | State   | FinalStatus | Progress |
|--------------------------------|----------|------|------------------|---------|--------------------------------|------------|---------|-------------|----------|
| application_1459371596174_0001 | ec2-user | Sort | MAPREDUCE        | default | Wed Mar 30 16:01:19 -0500 2016 | N/A        | RUNNING | UNDEFINED   |          |

Showing 1 to 1 of 1 entries

All Applications - Mozilla Firefox

EC2 Management C... (no subject) - pranit... All Applications

ec2-54-187-35-122.us-west-2.compute.amazonaws.com:8088/cluster

**hadoop**

## All Applications

**Cluster Metrics**

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | Vcores Used | Vcores Total | Vcores Reserved | Active Nodes | Decommissioned Nodes |
|----------------|--------------|--------------|----------------|--------------------|-------------|--------------|-----------------|-------------|--------------|-----------------|--------------|----------------------|
| 1              | 0            | 0            | 1              | 0                  | 0 B         | 136 GB       | 0 B             | 0           | 136          | 0               | 17           | 0                    |

**Scheduler Metrics**

| Scheduler Type     | Scheduling Resource Type | Minimum Allocation      | Maximum Allocation      |
|--------------------|--------------------------|-------------------------|-------------------------|
| Capacity Scheduler | [MEMORY]                 | <memory:1024, vCores:1> | <memory:8192, vCores:1> |

Show 20 entries

| ID                             | User     | Name | Application Type | Queue   | StartTime                      | FinishTime                     | State    | FinalStatus | Progress |
|--------------------------------|----------|------|------------------|---------|--------------------------------|--------------------------------|----------|-------------|----------|
| application_1459371596174_0001 | ec2-user | Sort | MAPREDUCE        | default | Wed Mar 30 16:01:19 -0500 2016 | Wed Mar 30 20:11:28 -0500 2016 | FINISHED | SUCCEEDED   |          |

Showing 1 to 1 of 1 entries

## Performance:

Time taken to sort 10GB :736.8 sec

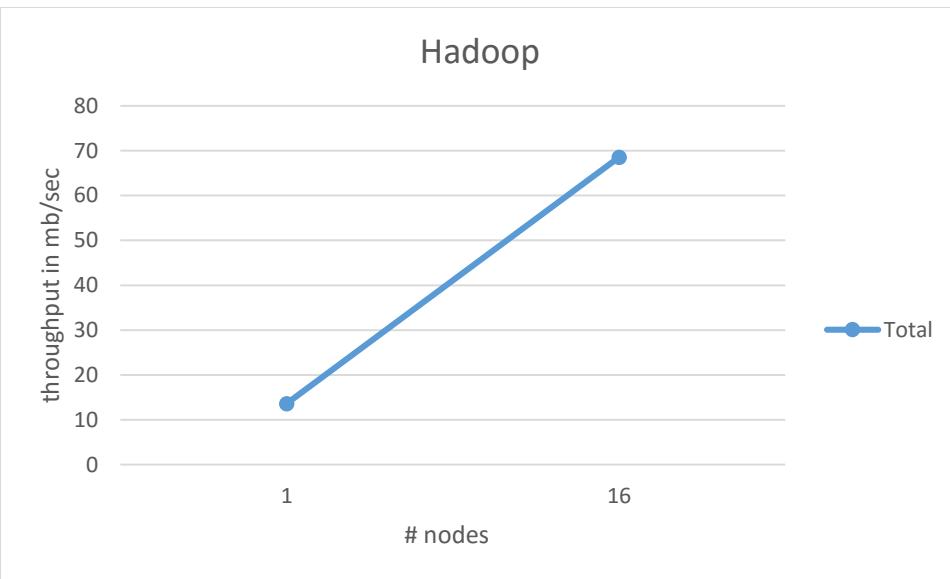
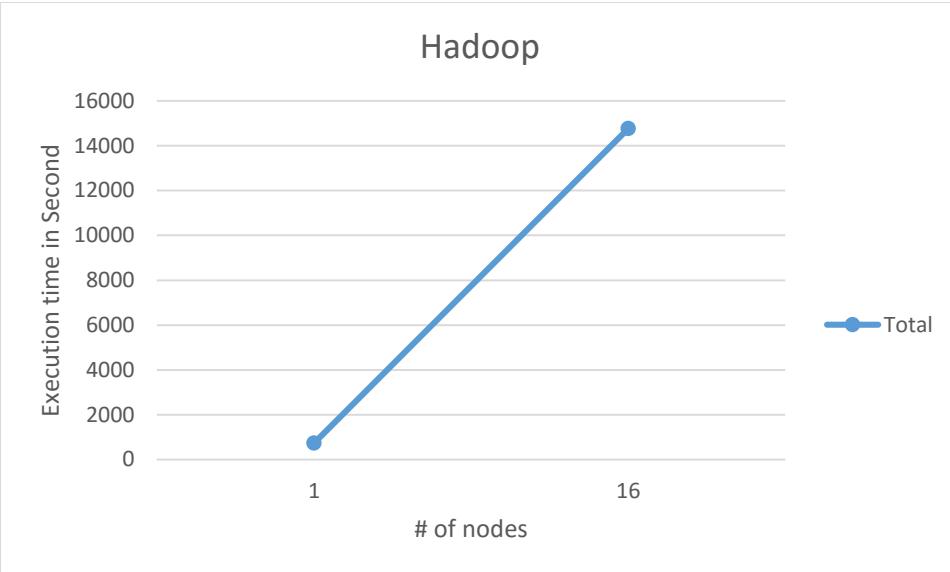
100GB: 1459.23 sec

1TB: 14769 sec

Time taken to sort 10GB :13.5722041 mb/sec

100GB: 68.5292928 mb/sec

1TB: 677.093913 mb/sec



## 3.Spark:

### System Configuration:

VERSION: spark-1.6.0-bin-hadoop2.6

INSTANCE TYPE: c3. Large (10GB&100GB) & d2.Xlarge(1TB) UBUNTU

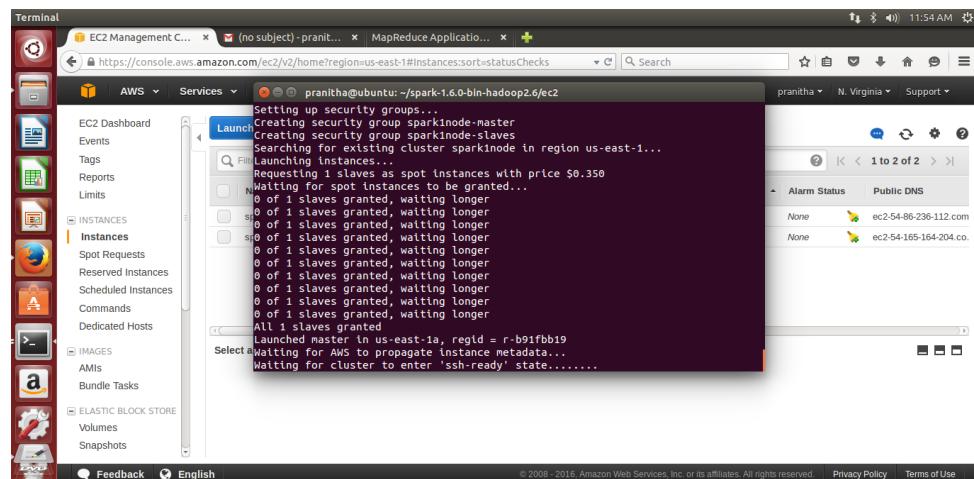
JDK:1.8

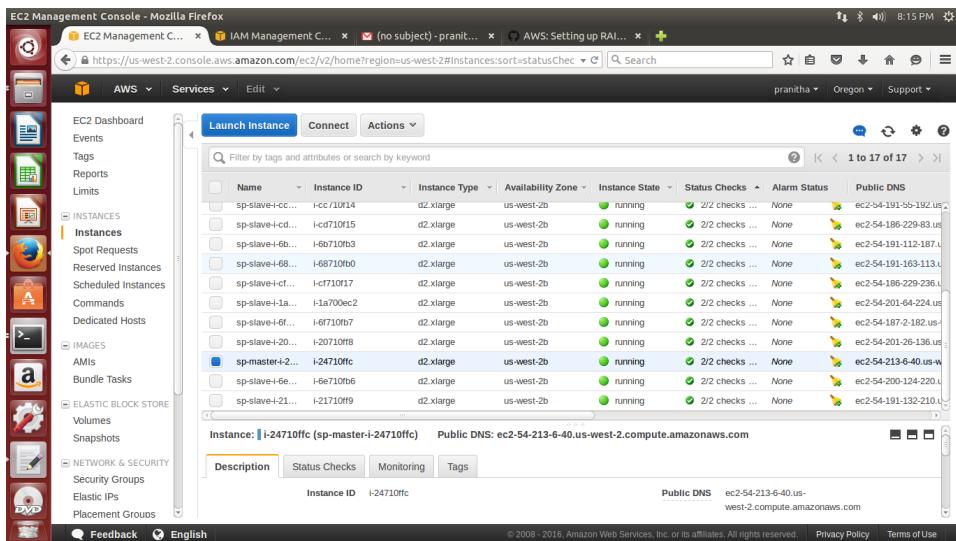
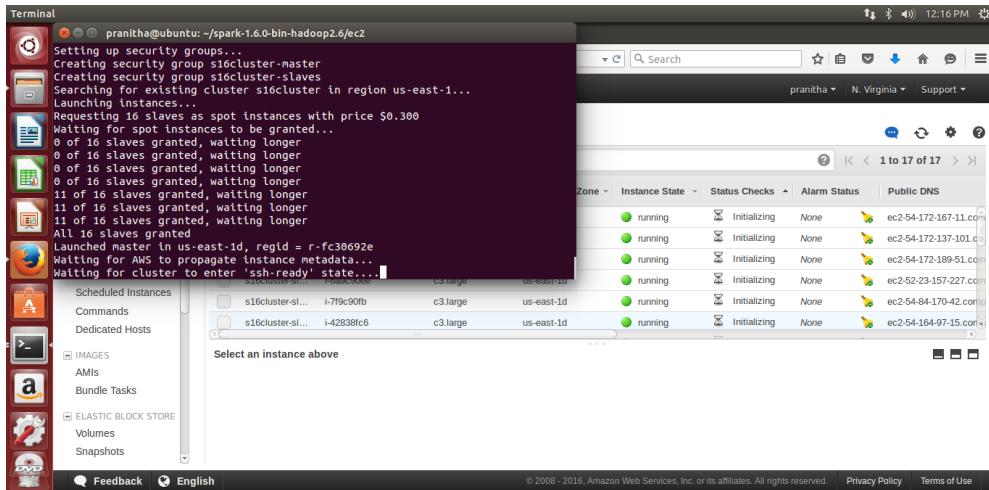
### Spark Configuration:

To implement spark in aws instance there are a set of commands that create appropriate setup for us to directly implement on it

- export AWS\_ACCESS\_KEY\_ID=<Access key id>
- export AWS\_SECRET\_ACCESS\_KEY=<secret acces key values>
- ./spark-ec2 -k <key pair name> -i <keypair.pem> -s <no of slaves> -t <instance-type> -r us-west-2 --spot-price=0.2 launch <cluster-name>
- ./spark-ec2 -k <keypair> -i <keypair.pem> login <cluster-name>

```
AWSAccessKeyId=AKIAJCOSENBNX  
AWSSecretKey=RJvLRL0+HLH593  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6/ec2  
pranitha@ubuntu:~$ cd spark-1.6.0-bin-hadoop2.6  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ cd  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ cd  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ cd  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ ls  
bin  conf  ec2  lib  licenses  python  README.md  sbin  
CHANGES.txt  data  examples  LICENSE  NOTICE  R  RELEASE  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ cd ec2  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ export AWS_ACCESS_KEY_ID=AKIAJCOSENBNX  
OSNBXXFHGDNQ  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ export AWS_SECRET_ACCESS_KEY=RJvLRL0+HLH593  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ ./spark-ec2 launch -k sparkinode -i sparkinode.pem -s 1 -t c3.8xlarge -r 0.2  
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6$ ./spark-ec2 login sparkinode
```





```

pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6/ec2
ec2-54-187-43-12.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-187-43-12.us-west-2.compute.amazonaws.com,54.187.43.12' (ECDSA) to the list of known hosts.
ec2-54-201-184-1.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-201-184-1.us-west-2.compute.amazonaws.com,54.201.184.1' (ECDSA) to the list of known hosts.
ec2-54-201-184-1.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-201-184-1.us-west-2.compute.amazonaws.com,54.201.184.1' (ECDSA) to the list of known hosts.
ec2-54-191-105-88.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-191-105-88.us-west-2.compute.amazonaws.com,54.191.105.88' (ECDSA) to the list of known hosts.
ec2-54-191-155-192.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-191-155-192.us-west-2.compute.amazonaws.com,54.191.155.192' (ECDSA) to the list of known hosts.
ec2-54-186-229-83.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-186-229-83.us-west-2.compute.amazonaws.com,54.186.229.83' (ECDSA) to the list of known hosts.
ec2-54-191-112-187.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-191-112-187.us-west-2.compute.amazonaws.com,54.191.112.187' (ECDSA) to the list of known hosts.
ec2-54-191-163-113.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-191-163-113.us-west-2.compute.amazonaws.com,54.191.163.113' (ECDSA) to the list of known hosts.
ec2-54-186-229-236.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-186-229-236.us-west-2.compute.amazonaws.com,54.186.229.236' (ECDSA) to the list of known hosts.
ec2-54-186-224.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-186-224.us-west-2.compute.amazonaws.com,54.186.224' (ECDSA) to the list of known hosts.
ec2-54-187-2-182.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-187-2-182.us-west-2.compute.amazonaws.com,54.187.2.182' (ECDSA) to the list of known hosts.
ec2-54-201-26-136.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-201-26-136.us-west-2.compute.amazonaws.com,54.201.26.136' (ECDSA) to the list of known hosts.
ec2-54-191-132-210.us-west-2.compute.amazonaws.com
Warning: Permanently added 'ec2-54-191-132-210.us-west-2.compute.amazonaws.com,54.191.132.210' (ECDSA) to the list of known hosts.
Cloning spark-ec2 scripts from https://github.com/amplab/spark-ec2/tree/branch-1.5 on master...
Warning: Permanently added 'ec2-54-213-6-40.us-west-2.compute.amazonaws.com,54.213.6.40' (ECDSA) to the list of known hosts.
Cloning into 'spark-ec2'...
remote: Counting objects: 2072, done.
remote: Total 2072 (delta 0), reused 0 (delta 0), pack-reused 2072
Receiving objects: 100% (2072/2072), 356.17 KB | 0 bytes/s, done.
Resolving deltas: 100% (789/789), done.
Checking connectivity... done.
Connection to ec2-54-213-6-40.us-west-2.compute.amazonaws.com closed.
Deploying files to master...

```

```

pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6/ec2
Starting GANGLIA gmond: [ OK ]
Connection to ec2-54-172-114-148.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [ OK ]
Connection to ec2-54-86-162-186.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [ OK ]
Connection to ec2-52-207-230-236.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [ OK ]
Connection to ec2-52-23-224-141.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [ OK ]
Connection to ec2-54-152-238-88.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [ OK ]
Connection to ec2-54-89-162-141.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmond: [ OK ]
Connection to ec2-52-90-134-55.compute-1.amazonaws.com closed.
Shutting down GANGLIA gmond: [FAILED]
Starting GANGLIA gmetad: [ OK ]
Connection to ec2-52-201-238-203.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]
[timimg] ganglia setup: 00h 00m 24s
Connection to ec2-54-174-129-217.compute-1.amazonaws.com closed.
Spark standalone cluster started at http://ec2-54-174-129-217.compute-1.amazonaws.com:8080
Ganglia started at http://ec2-54-174-129-217.compute-1.amazonaws.com:5080/ganglia
Done!
pranitha@ubuntu:~/spark-1.6.0-bin-hadoop2.6/ec2$
```

- Once master and slaves are all up running, open master instance from where all slave nodes can be operated
- To transfer files like jar, gensort etc. use SCP command and sent it to master.
- Check java version if its below 1.8 update using following commands
  - `wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u73-b02/jdk-8u73-linux-x64.rpm`
  - `sudo rpm -ivh jdk-8u73-linux-x64.rpm`
  - `export JAVA_HOME=/usr/java/latest`

### Implementation:

- Spark sort has been implemented in java.
- Launch the instance and do all necessary implementation stated above
- Once the implementations are done, add storage depending on file size to be sorted
- Get the jar file that has been implemented using SCP command from local system to instance.
- Create a dataset of required size from gensort and put it in HDFS using put command
- Now using the jar file start sorting the dataset
- In the java program the file is divided into key and value RDD pair.
- The sortbykey() function is specific is used to sort the values by key
- The output is sent into the HDFS which can be accessed using get command
- This output value is checked using ./valsor in gensort

```
pranitha@ubuntu:~  
16/03/26 15:36:23 INFO storage.MemoryStore: Block broadcast_4_piece0 stored as bytes in memory (estimated size 8.1 KB, free 81.7 KB)  
16/03/26 15:36:23 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on localhost:37395 (size: 8.1 KB, free: 511.1 MB)  
16/03/26 15:36:23 INFO spark.SparkContext: Created broadcast 4 from broadcast at DAGScheduler.scala:1066  
16/03/26 15:36:23 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 3 (MapPartitionsRDD[1] at saveAsTextFile at Sort.java:61)  
16/03/26 15:36:23 INFO Scheduler: TaskSchedulerImpl: Adding task set 3.0 with 1 tasks  
16/03/26 15:36:23 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 3.0 (TID 24, localhost, partition 0, NODE_LOCAL, 2218 bytes)  
16/03/26 15:36:23 INFO executor.Executor: Running task 0.0 in stage 3.0 (TID 24)  
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Getting 8 non-empty blocks out of 8 blocks  
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms  
16/03/26 15:36:48 INFO mapred.FileOutputCommitter: Saved output of task 'attempt_201603261534_0003_m_000000_24' to hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/output  
16/03/26 15:36:48 INFO mapred.SparkHadoopAppRedUtil: attempt_201603261534_0003_m_000000_24: Committed  
16/03/26 15:36:48 INFO executor.Executor: Finished task 0.0 in stage 3.0 (TID 24), 1165 bytes result sent to driver  
16/03/26 15:36:48 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 3.0 (TID 24) in 25202 ms on localhost (1/1)  
16/03/26 15:36:48 INFO scheduler.TaskschedulerImpl: Removed Taskset 3.0, whose tasks have all completed, from pool  
16/03/26 15:36:48 INFO scheduler.DAGScheduler: Job 1 finished: saveAsTextFile at Sort.java:61 took 110.076747 s  
16/03/26 15:36:48 INFO spark.SparkContext: Invoking stop() from shutdown hook  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/metrics/json,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/stage/kll,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/api,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/static,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/threaddump/json,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/threaddump,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/environment/json,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/environment,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/rdd/json,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/rdd,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/json,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/jobs/job/json,null)  
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/jobs/job,null)
```

```
pranitha@ubuntu:~  
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Got job 0 (sortByKey at Sort.java:56) with 8 output partitions  
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Final stage: ResultStage 0 (sortByKey at Sort.java:56)  
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Parents of final stage: List()  
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Missing parents: List()  
16/03/26 15:34:43 INFO scheduler.DAGScheduler: Submitting ResultStage 0 (MapPartitionsRDD[4] at sortByKey at Sort.java:56), which has no missing parents  
16/03/26 15:34:43 INFO storage.MemoryStore: Block broadcast_1 stored as values in memory (estimated size 4.1 KB, free 54.8 KB)  
16/03/26 15:34:43 INFO storage.MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 2.3 KB, free 57.1 KB)  
16/03/26 15:34:43 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on localhost:37395 (size: 2.3 KB, free: 511.1 MB)  
16/03/26 15:34:43 INFO spark.SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1066  
16/03/26 15:34:43 INFO scheduler.TaskSchedulerImpl: Submitting 8 missing tasks from ResultStage 0 (MapPartitionsRDD[4] at sortByKey at Sort.java:56)  
16/03/26 15:34:43 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, partition 0, ANY, 2227 bytes)  
16/03/26 15:34:43 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)  
16/03/26 15:34:43 INFO executor.Executor: Fetching http://172.31.23.245:50337/jars/sparkSort.jar with timestamp 1459006481723  
16/03/26 15:34:43 INFO util.Utils: Fetching http://172.31.23.245:50337/jars/SparkSort.jar to /mnt/spark/spark-0c03a1ca-e899-421a-9515-dc6a96102a  
fc/userFiles-8843b726-a807-4587-9af8-55f1e8db8d74/fetchfileTemp08872056735648350.tmp  
16/03/26 15:34:43 INFO executor.Executor: Adding file:/mnt/spark/spark-0c03a1ca-e899-421a-9515-dc6a96102a/func/userFiles-8843b726-a807-4587-9af8-55  
f1e8db8d74/SparkSort.jar to class loader  
16/03/26 15:34:43 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:0+134217728  
16/03/26 15:34:45 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 3068 bytes result sent to driver  
16/03/26 15:34:45 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, localhost, partition 1, ANY, 2227 bytes)  
16/03/26 15:34:45 INFO executor.Executor: Running task 1.0 in stage 0.0 (TID 1)  
16/03/26 15:34:45 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:134217728+134217728  
16/03/26 15:34:46 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 2789 ms on localhost (1/8)  
16/03/26 15:34:46 INFO executor.Executor: Flinshed task 1.0 in stage 0.0 (TID 1). 3068 bytes result sent to driver  
16/03/26 15:34:46 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 0.0 (TID 2, localhost, partition 2, ANY, 2227 bytes)  
16/03/26 15:34:46 INFO executor.Executor: Running task 2.0 in stage 0.0 (TID 2)  
16/03/26 15:34:46 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:208435456+134217728  
16/03/26 15:34:46 INFO scheduler.TaskSetManager: Flinshed task 1.0 in stage 0.0 (TID 1) in 959 ms on localhost (2/8)  
16/03/26 15:34:46 INFO executor.Executor: Flinshed task 2.0 in stage 0.0 (TID 2). 3068 bytes result sent to driver  
16/03/26 15:34:48 INFO scheduler.TaskSetManager: Starting task 3.0 in stage 0.0 (TID 3, localhost, partition 3, ANY, 2227 bytes)  
16/03/26 15:34:48 INFO executor.Executor: Running task 3.0 in stage 0.0 (TID 3)  
16/03/26 15:34:48 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:402653184+134217728  
16/03/26 15:34:48 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 1601 ms on localhost (3/8)  
16/03/26 15:34:50 INFO executor.Executor: Flinshed task 3.0 in stage 0.0 (TID 3). 3068 bytes result sent to driver  
16/03/26 15:34:50 INFO scheduler.TaskSetManager: Starting task 4.0 in stage 0.0 (TID 4, localhost, partition 4, ANY, 2227 bytes)  
16/03/26 15:34:50 INFO executor.Executor: Running task 4.0 in stage 0.0 (TID 4)  
16/03/26 15:34:50 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:536870912+134217728  
16/03/26 15:34:50 INFO scheduler.TaskSetManager: Flinshed task 3.0 in stage 0.0 (TID 3) in 2104 ms on localhost (4/8)
```

```

pranitha@ubuntu: ~
16/03/26 15:52:02 INFO scheduler.DAGScheduler: ResultStage 3 (saveAsTextFile at Sort.java:61) finished in 25.209 s
16/03/26 15:52:02 INFO scheduler.DAGScheduler: Job 1 finished saveAsTextFile at Sort.java:61, took 111.283124 s
16/03/26 15:52:02 INFO handler.ContextHandler: Stopped o.s.j.s.ServletContextHandler(/metrics/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/stage/kill,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/api/null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/_,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/static,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/threaddump/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/threaddump,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/environment/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/rdd/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/rdd,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/pool/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/stage/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/_/null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/jobs/job/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/jobs/job,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/jobs/json,null)
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/jobs,null)
16/03/26 15:52:02 INFO util.SparkUI: Stopped Spark web UI at http://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:4040
16/03/26 15:52:02 INFO spark.MapOutputTrackerMasterEndpoint: MapoutputTrackerMasterEndpoint stopped!
16/03/26 15:52:02 INFO storage.BlockManagerInfo: Storage BlockManagerInfo: Memory 81.7 MB free
16/03/26 15:52:02 INFO storage.BlockManager: Memory 81.7 MB free
16/03/26 15:52:02 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
16/03/26 15:52:02 INFO scheduler.OutputCommitCoordinatorOutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/03/26 15:52:02 INFO util.ShutdownHookManager: Shutdown hook called
c1-9b2-8c62ac51b0e
16/03/26 15:52:02 INFO util.ShutdownHookManager: Deleting directory /mnt/spark/spark-703f408d-a4dd-414d-b50f-993eca8cf90f
16/03/26 15:52:02 INFO util.ShutdownHookManager: Deleting directory /mnt/spark/spark-73878714-639c-4694-bdb5-93747b860313
root@ip-172-31-23-245: spark$
```

```

pranitha@ubuntu: ~
16/03/26 15:36:23 INFO storage.MemoryStore: Block broadcast_4_piece0 stored as bytes in memory (estimated size 8.1 KB, free 81.7 KB)
16/03/26 15:36:23 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on localhost:37795 (size: 8.1 KB, free: 511.1 MB)
16/03/26 15:36:23 INFO spark.SparkContext: Created broadcast 4 from broadcast at DAGScheduler.scala:1006
16/03/26 15:36:23 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 3 (MapPartitionsRDD[11] at saveAsTextFile at Sort.java:61)
16/03/26 15:36:23 INFO scheduler.TaskSchedulerImpl: Adding task set 3.0 with 1 tasks
16/03/26 15:36:23 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 3.0 (TID 24, localhost, partition 0,NODE_LOCAL, 2218 bytes)
16/03/26 15:36:23 INFO executor.Executor: Running task 0.0 in stage 3.0 (TID 24)
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Getting 8 non-empty blocks out of 8 blocks
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Started 8 block fetches in 1 ms
16/03/26 15:36:48 INFO output.FileOutputCommitter: Saved output of task attempt_201603261534_0003_m_000000_24' to hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/output
16/03/26 15:36:48 INFO mapred.SparkHadoopMapRedUtil: attempt_201603261534_0003_m_000000_24: Committed
16/03/26 15:36:48 INFO executor.Executor: Finished task 0.0 in stage 3.0 (TID 24). 1165 bytes sent to driver
16/03/26 15:36:48 INFO scheduler.TaskSetManager: Flinshed task 0.0 in stage 3.0 (TID 24) in 25202 ms on localhost (1/1)
16/03/26 15:36:48 INFO scheduler.DAGScheduler: Removed TaskSet 3.0, whose tasks have all completed, from pool
16/03/26 15:36:48 INFO scheduler.DAGScheduler: ResultStage 3 (saveAsTextfile at Sort.java:61) finished in 25.203 s
16/03/26 15:36:48 INFO scheduler.TaskSchedulerImpl: Removed 1 dead partitions at Sort.java:61, took 110.076747 s
16/03/26 15:36:48 INFO spark.SparkContext: Invoking stop() from shutdown hook
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/metrics/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/stage/kill,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/api,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/_,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/static,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/threaddump/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/threaddump,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/executors,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/environment/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/rdd/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/rdd,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/storage,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/pool/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/stage/json,null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/stages/null)
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler(/jobs/job/json,null)
```

The screenshot shows a desktop environment with a terminal window and a file manager window.

**Terminal Window:**

```

pranitha@ubuntu: ~
root@ip-172-31-23-245: ~]$ ./valsort output
Records: 10000000
Checksum: 4c48a891c779d5
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-23-245: ~]$
```

**File Manager Window:**

The file manager window displays the contents of the current directory, which includes:

- Desktop
- Documents
- Downloads
- eclipse
- Pictures
- Public
- sp
- spark-1.6-bin-hadoop2.6
- dataset
- Examples
- rootkey.csv
- Sort\_Spark.pem
- Hadoop\_Sort.jar
- spark-1.6-bin-hadoop2.6.tgz
- Spark.pem
- SparkSort.jar
- SparkSort.pem

A tooltip at the bottom right of the file manager window says: "commands selected (593 bytes)".

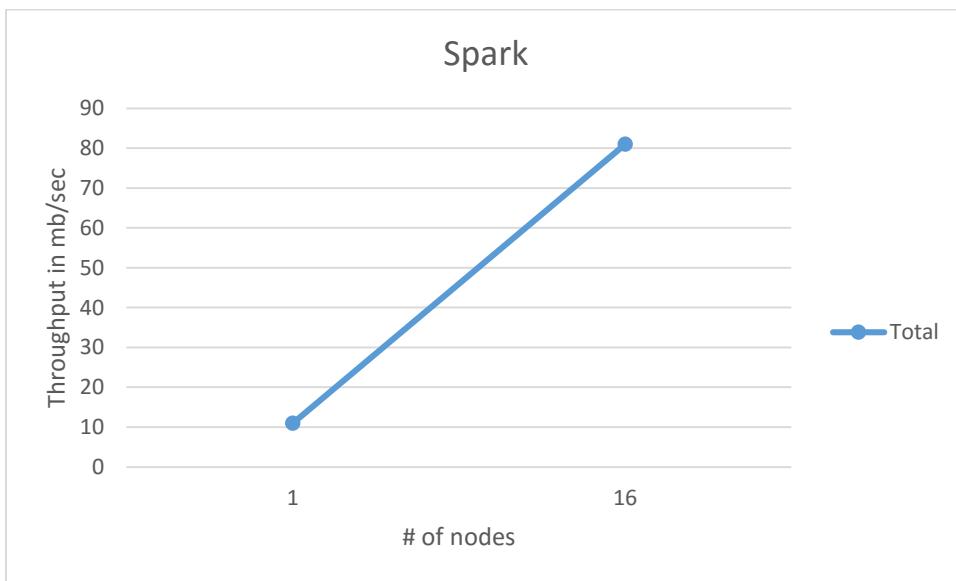
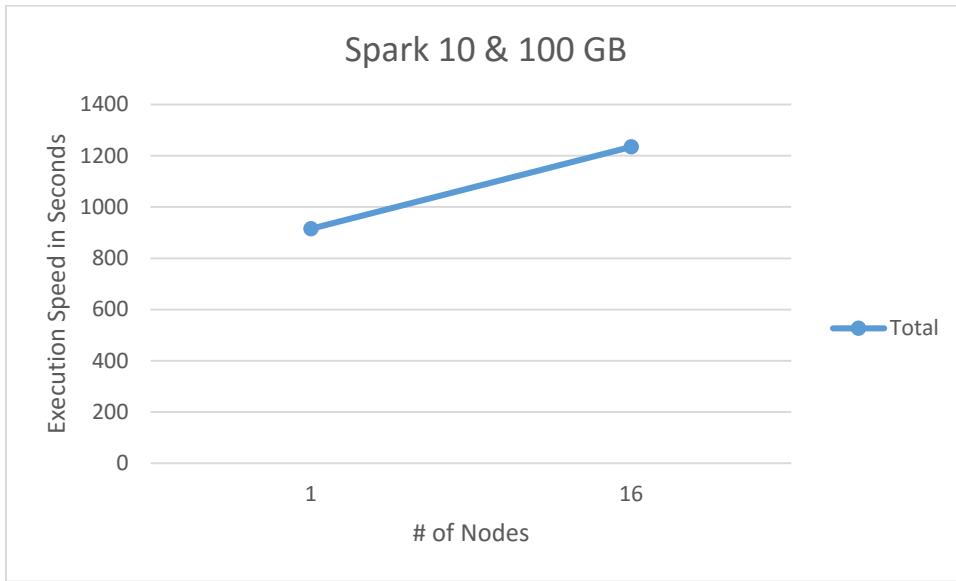
## Performance:

Time taken to sort 10GB:915.6 sec

100GB:1235.34 sec

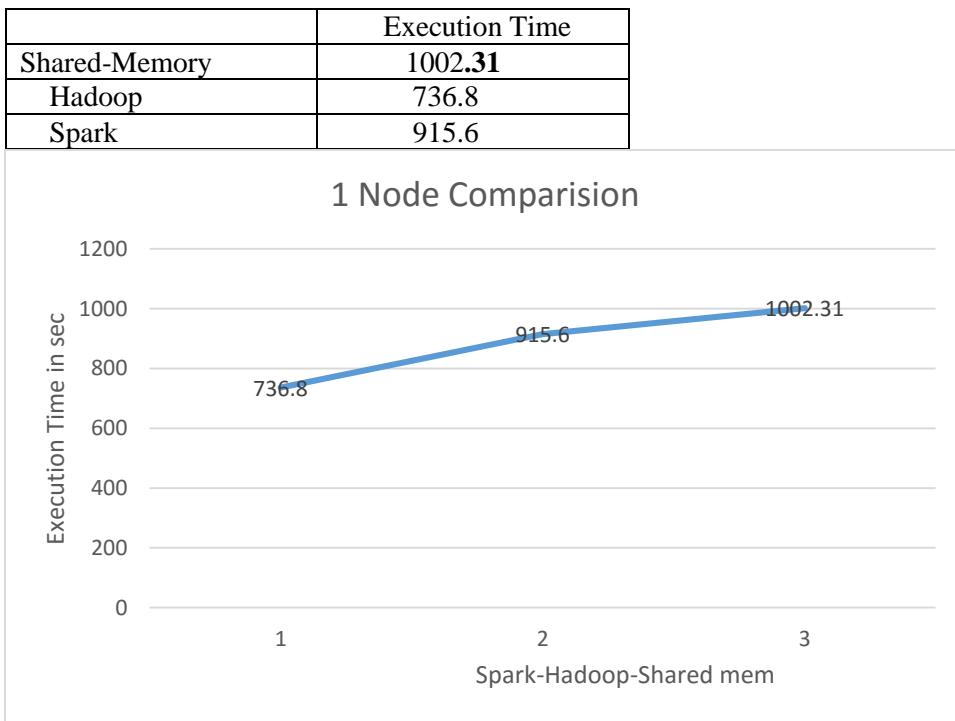
Throughput 10GB:10.9217999 mb/sec

100GB:80.9493743 mb/sec



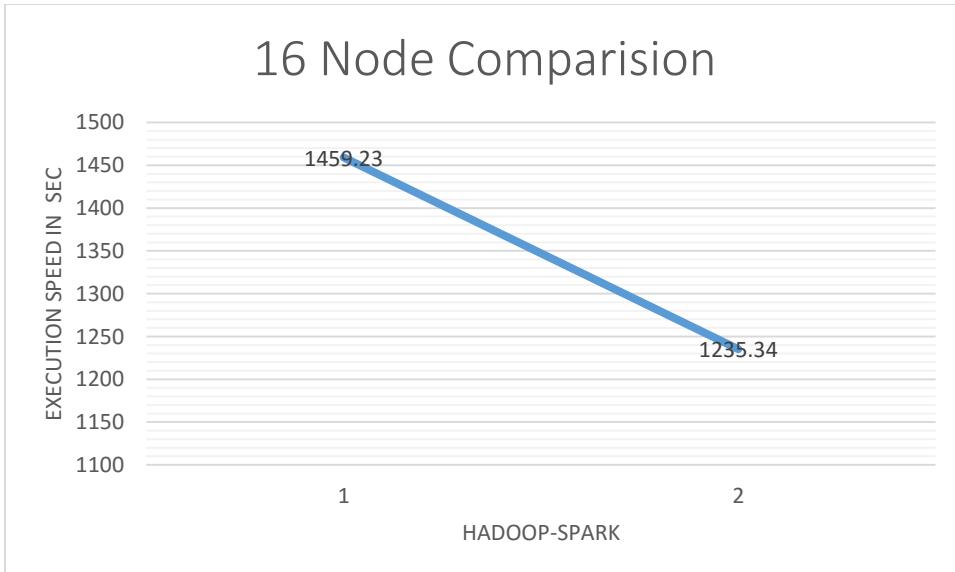
### Comparison of performances:

For 10GB



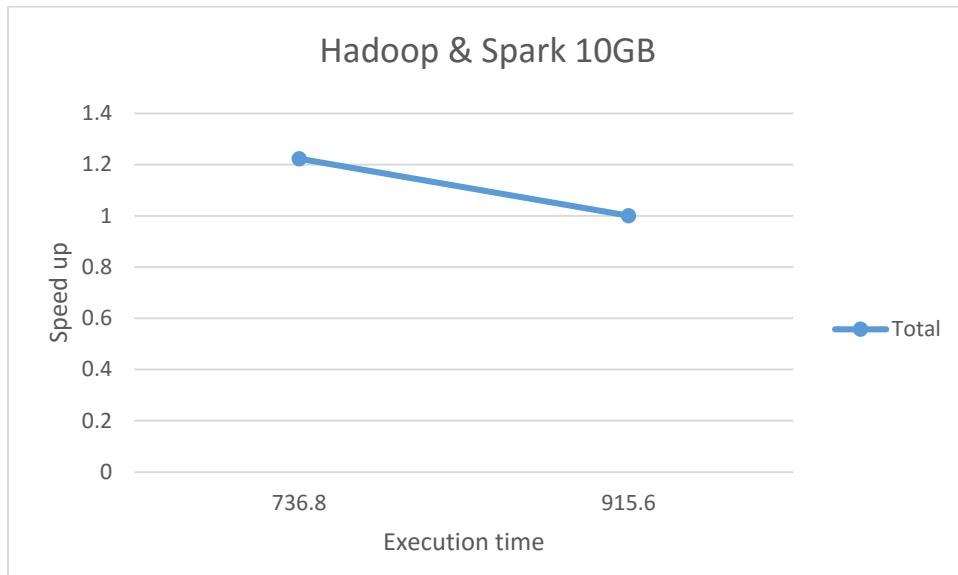
For 100GB

|        | Execution Time |
|--------|----------------|
| Hadoop | 1459.23        |
| Spark  | 1235.34        |



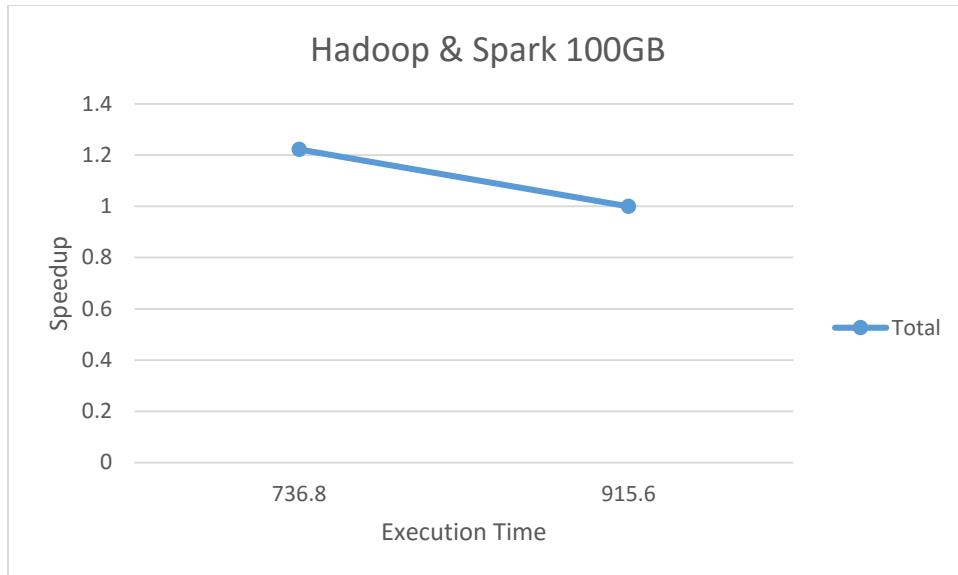
### Hadoop and spark speedup for 10 GB

|        | Execution Time | Spark Execution Time | Speed up   |
|--------|----------------|----------------------|------------|
| Hadoop | 736.8          | 915.6                | 1.22267101 |
| Spark  | 915.6          | 915.6                | 1          |



### Hadoop and spark speedup for 100GB

|        | Execution Time | Hadoop Execution time | Speedup    |
|--------|----------------|-----------------------|------------|
| Hadoop | 1459.23        | 1459.23               | 1          |
| Spark  | 1235.34        | 1459.23               | 1.18123755 |



What Conclusion are drawn?

- The best performance for smaller data is achieved using Hadoop
- The best performance for larger data is achieved using Spark

What seems to be best at 1node scale?

- Hadoop

What seems to be best at 100 node scale?

- Spark