

CS 553 Cloud Computing Programming Assignment 2

Abhishek Prakash Kumar (A20379845)
Swapnil Dharawat (A20379172)
Jagruti Vichare(A20378092)

Overview:

The objective of this assignment is to perform Terasort operation using C, Hadoop and Spark. The main goal behind this is to compare the results obtained from each of these methods and build a conclusion out of it.

Runtime Environment Settings

Amazon Web Services EC2

AMI - Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-45ead225

Hadoop 2.6.5

Spark 2.2.0

Java 1.8.0_151

Shared-Memory:

EC2 instance type : i3.large

Hadoop:

EC2 instance type : (master node)i3.large (slave node)i3.large

Spark:

EC2 instance type : (headnode)i3.large (worker node)i3.large

MPI: Open MPI 3.0.4, starcluster 1

Installation Steps of Virtual Cluster

1. 1 node

1) log in to the AWS, choose EC2

2) launch instance

3) choose Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-3d50120d

4) choose i3.large

5) choose spot instance

6) create a new security group, with rules: SSH, All TCP(Anywhere), All ICMP(Anywhere)

2. 1 node

1) choose i3.4xlarge

2) choose spot instance

3) create an image of instance established in 1st step.

4) use the same configuration

3. 8 nodes

1) finish the step of 1 node

2) create an image of instance established in 1 node

3) create another 7 nodes use the AMI just created

4) use the same configuration

→ Create an instance on Amazon AWS:

1. Choose Amazon Machine Image:

Step 1: Choose an Amazon Machine Image (AMI)

The screenshot shows the 'Choose an Amazon Machine Image (AMI)' step in the AWS console. Two AMIs are displayed:

- SUSE Linux Enterprise Server 12 Service Pack 3 (HVM), SSD Volume Type** (ami-3b5b0d03): SUSE Linux Enterprise Server 12 Service Pack 3 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled. Root device type: ebs, Virtualization type: hvm. 64-bit.
- Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** (ami-82f4dae7): Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>). Root device type: ebs, Virtualization type: hvm. 64-bit.

The Ubuntu AMI is selected, indicated by a blue highlight and a 'Select' button.

2. Choose Instance type

Step 2: Choose an Instance Type

<input type="checkbox"/>	Storage optimized	h1.xlarge	8	32	1 x 2000	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	h1.4xlarge	16	64	2 x 2000	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	h1.8xlarge	32	128	4 x 2000	Yes	10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	h1.16xlarge	64	256	8 x 2000	Yes	25 Gigabit	Yes
<input checked="" type="checkbox"/>	Storage optimized	i3.large	2	15.25	1 x 475 (SSD)	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	i3.xlarge	4	30.5	1 x 950 (SSD)	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	i3.2xlarge	8	61	1 x 1900 (SSD)	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	i3.4xlarge	16	122	2 x 1900 (SSD)	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	i3.8xlarge	32	244	4 x 1900 (SSD)	Yes	10 Gigabit	Yes
<input type="checkbox"/>	Storage optimized	i3.16xlarge	64	488	8 x 1900 (SSD)	Yes	25 Gigabit	Yes

Buttons: Cancel Previous **Review and Launch** Next: Configure Instance Details

3. Edit Storage

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 ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encrypted ⓘ
Root	/dev/sda1	snap-0bc297f6637ec77c6	8	General Purpose SSD (GP2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted
Instance Store 0	/dev/nvme*n1	N/A	N/A	N/A	N/A	N/A	N/A	Not Encrypted

Add New Volume

We have changed the storage to 1862GB

The screenshot is just an example

Instances created are:

aws

Services

Resource Groups

sdharawat

Ohio

Support

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

Name

Instance ID

Instance Type

Availability Zone

Instance State

Status Checks

Alarm Status

Public DNS

PA2_Swapnil_Abhishek_Jagruti

i-05c0840803a2813aa

i3.xlarge

us-east-2a

running

2/2 checks ...

None

ec2-52-15-6

PA2_Swapnil_Abhishek_Jagruti

i-0ac4efe533f96aaf7

i3.large

us-east-2a

running

2/2 checks ...

None

ec2-18-217-

i3.large

Instance: i-0ac4efe533f96aaf7 (PA2_Swapnil_Abhishek_Jagruti)		Public DNS: ec2-18-217-183-207.us-east-2.compute.amazonaws.com	
Description	Status Checks	Monitoring	Tags
Instance ID	i-0ac4efe533f96aaf7		Public DNS (IPv4)
Instance state	running		ec2-18-217-183-207.us-east-2.compute.amazonaws.com
Instance type	i3.large		IPv4 Public IP
Elastic IPs			18.217.183.207
Availability zone	us-east-2a		IPv6 IPs
Security groups	launch-wizard-4 . view inbound rules		-
Scheduled events	No scheduled events		Private DNS
AMI ID	ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20171121.1 (ami-82f4dae7)		ip-172-31-9-206.us-east-2.compute.internal
Platform	-		Private IPs
			172.31.9.206
			Secondary private IPs
			VPC ID
			vpc-d78d67bf
			Subnet ID
			subnet-a0cc33c8
			Network interfaces
			eth0

i3.4xlarge

Instance: i-05c0840803a2813aa (PA2_Swapnil_Abhishek_Jagruti)		Public DNS: ec2-52-15-63-135.us-east-2.compute.amazonaws.com	
Description	Status Checks	Monitoring	Tags
Instance ID	i-05c0840803a2813aa		Public DNS (IPv4)
Instance state	running		ec2-52-15-63-135.us-east-2.compute.amazonaws.com
Instance type	i3.4xlarge		IPv4 Public IP
Elastic IPs			52.15.63.135
Availability zone	us-east-2a		IPv6 IPs
Security groups	launch-wizard-5 . view inbound rules		-
Scheduled events	No scheduled events		Private DNS
AMI ID	ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20171121.1 (ami-82f4dae7)		ip-172-31-14-235.us-east-2.compute.internal
Platform	-		Private IPs
			172.31.14.235
			Secondary private IPs
			VPC ID
			vpc-d78d67bf
			Subnet ID
			subnet-a0cc33c8
			Network interfaces
			eth0

Connect to instance using ssh:

-
- I would like to connect with**
- ☒ A standalone SSH client
 - ☐ A Java SSH Client directly from my browser (Java required)
-

To access your instance:

1. Open an SSH client. (find out how to [connect using PuTTY](#))
2. Locate your private key file (Demo1.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 Demo1.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-52-15-63-135.us-east-2.compute.amazonaws.com
```

Example:

```
ssh -i "Demo1.pem" ubuntu@ec2-52-15-63-135.us-east-2.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

ssh -i "Keyname.pem" public DNS

For eg:

ssh -i "Demo1.pem" [ubuntu@ec2-18-217-183-207.us-east-2.compute.amazonaws.com](#)

Transfer file from local system to AWS instance:

scp -i Keyname.pem filename.ext [publicDNS:~/](#)

For eg:

scp -i Demo1.pem SparkSort.py [ubuntu@ec2-18-217-183-207.us-east-2.compute.amazonaws.com:~/](#)

```

Swapnil@DESKTOP-CAH45E8 MINGW64 /c/Studies/Cloud Computing/Assignment 2
$ ssh -i "Demo1.pem" ubuntu@ec2-52-15-63-135.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1041-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

13 packages can be updated.
8 updates are security updates.

Last login: Sun Dec  3 22:11:33 2017 from 208.59.146.145
ubuntu@ip-172-31-14-235:~$ |

```

Shared Memory Terasort:

We are making use of external sort algorithm for shared memory terasort.

We have programmed the code using C language.

We have generated data using Gensort

128Gb of data is being used as an input in this part

This is performed on virtual cluster1 of i3.large and i3.xlarge configurations.

I3.large

The screenshot displays the AWS Management Console interface. On the left, a navigation sidebar lists various services including EC2 Dashboard, INSTANCES, IMAGES, ELASTIC BLOCK STORE, and NETWORK & SECURITY. The main content area shows a table of EC2 instances with columns for Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, and Public DNS (IPv4). Two instances are listed: 'i3.large' (ID: i-0c2ba366008b31f32) and 'i3.xlarge' (ID: i-086c6438abb495698), both in a 'running' state. Below the table, the details for the selected 'i3.large' instance are shown, including its Instance ID, state, type, availability zone, security groups, and network interfaces.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
i3.large	i-0c2ba366008b31f32	i3.large	us-east-1d	running	2/2 checks ...	None	ec2-54-210-64-224.compute-1.amazonaws.com
i3.xlarge	i-086c6438abb495698	i3.xlarge	us-east-1d	running	2/2 checks ...	None	ec2-35-153-177-71.compute-1.amazonaws.com

Instance: i-0c2ba366008b31f32 (i3.large)			
Description	Status Checks	Monitoring	Tags
Instance ID	i-0c2ba366008b31f32		
Instance state	running		
Instance type	i3.large		
Elastic IPs			
Availability zone	us-east-1d		
Security groups	launch-wizard-6, view inbound rules		
Scheduled events	No scheduled events		
AMI ID	ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20171121.1 (ami-aa2ea6d0)		
Platform	-		
Public DNS (IPv4)	ec2-54-210-64-224.compute-1.amazonaws.com		
IPv4 Public IP	54.210.64.224		
IPv6 IPs	-		
Private DNS	ip-172-31-38-153.ec2.internal		
Private IPs	172.31.38.153		
Secondary private IPs			
VPC ID	vpc-9aafc6e2		
Subnet ID	subnet-4e8f5713		
Network interfaces	eth0		

i3.4xlarge

The screenshot shows the AWS Management Console interface for an EC2 instance. The instance is named **i-086c6438abb495698 (i3.4xlarge)** and is in the **us-east-1d** availability zone. It is currently in the **running** state. The console displays various attributes including Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, and Public DNS (IPv4). The instance is associated with the **launch-wizard-3** security group and the **ubuntu-xenial-16.04-amd64-server-20171121.1** AMI.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
i3.large	i-0c2ba366008b31f32	i3.large	us-east-1d	running	2/2 checks ...	None	ec2-54-210-64-224.con
i3.4xlarge	i-086c6438abb495698	i3.4xlarge	us-east-1d	running	2/2 checks ...	None	ec2-35-153-177-71.con

Instance: **i-086c6438abb495698 (i3.4xlarge)** Public DNS: **ec2-35-153-177-71.compute-1.amazonaws.com**

Description		Status Checks	Monitoring	Tags
Instance ID	i-086c6438abb495698			
Instance state	running			
Instance type	i3.4xlarge			
Elastic IPs				
Availability zone	us-east-1d			
Security groups	launch-wizard-3. view inbound rules			
Scheduled events	No scheduled events			
AMI ID	ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20171121.1 (ami-aa2ea6d0)			
Platform	-			
Public DNS (IPv4)	ec2-35-153-177-71.compute-1.amazonaws.com			
IPv4 Public IP	35.153.177.71			
IPv6 IPs	-			
Private DNS	ip-172-31-32-14.ec2.internal			
Private IPs	172.31.32.14			
Secondary private IPs				
VPC ID	vpc-9aafc6e2			
Subnet ID	subnet-4e8f5713			
Network interfaces	eth0			

Performance

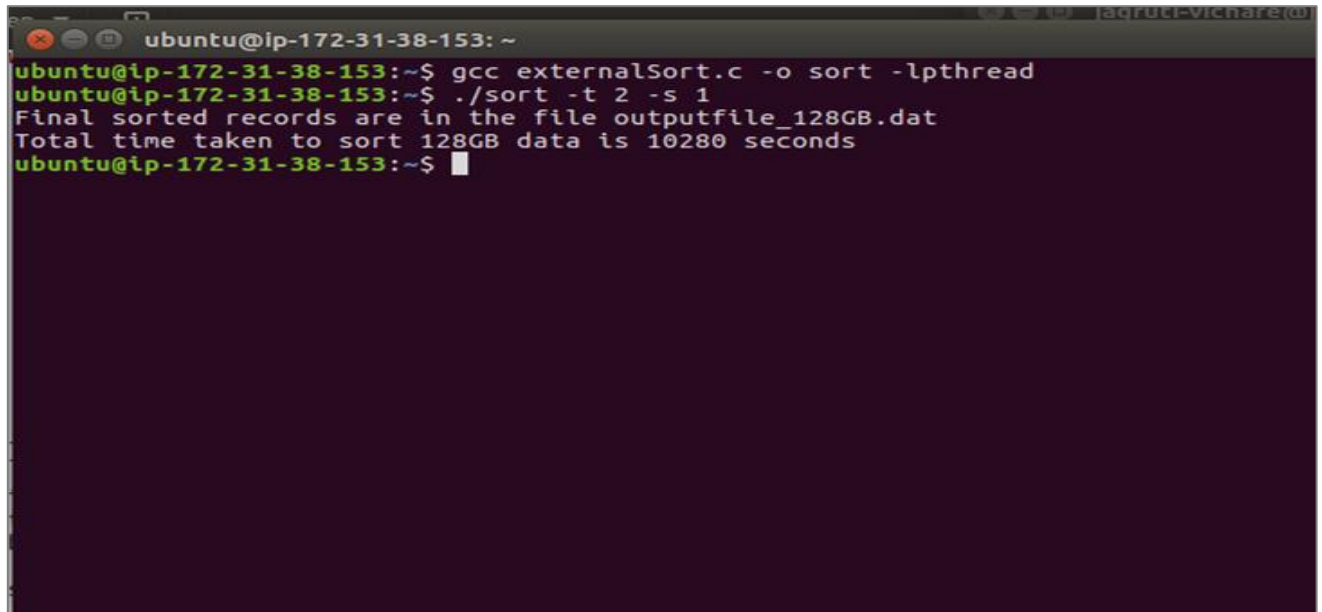
Externsl sort with 128GB dataset

Input File Size

```
ubuntu@ip-172-31-38-153: ~
ubuntu@ip-172-31-38-153:~$ ls -l inputfile_128GB.dat
-rwxrwxr-x 1 ubuntu ubuntu 128000000000 Dec  2 23:44 inputfile_128GB.dat
ubuntu@ip-172-31-38-153:~$
```

Here an input file of 128 GB is given. The shared memory terasort is supposed to take this dataset as an input to perform operations on it.

128 GB sort Result

A terminal window with a dark background and light-colored text. The window title is 'ubuntu@ip-172-31-38-153: ~'. The terminal shows the following commands and output:

```
ubuntu@ip-172-31-38-153:~$ gcc externalSort.c -o sort -lpthread
ubuntu@ip-172-31-38-153:~$ ./sort -t 2 -s 1
Final sorted records are in the file outputfile_128GB.dat
Total time taken to sort 128GB data is 10280 seconds
ubuntu@ip-172-31-38-153:~$
```

The screenshot above displays the sorting result of 128 GB of dataset

As we can see the details are as follows:

The dataset has been sorted and stored in the output file

The total time taken for sorting is 10280 seconds.

This time shows that shared memory terasort performs good with the small datasets in one node.

Validation of result using Valsort

A terminal window with a dark background and light-colored text. The prompt is 'ubuntu@ip-172-31-38-153: ~'. The command './valsort outputfile_128GB.dat' has been executed. The output shows 'Records: 1280000000', 'Checksum: 2f987241eb', 'Duplicate keys: 0', and 'SUCCESS - all records are in order.' followed by a new prompt.

```
ubuntu@ip-172-31-38-153: ~  
ubuntu@ip-172-31-38-153:~$ ./valsort outputfile_128GB.dat  
Records: 1280000000  
Checksum: 2f987241eb  
Duplicate keys: 0  
SUCCESS - all records are in order.  
ubuntu@ip-172-31-38-153:~$
```

The screenshot above shows that a total of 1280000000 records are created which has to be validated.

The validation is performed after the sorting of data to check if the files are in place or not. This step ensures that no data is lost in the process of sorting

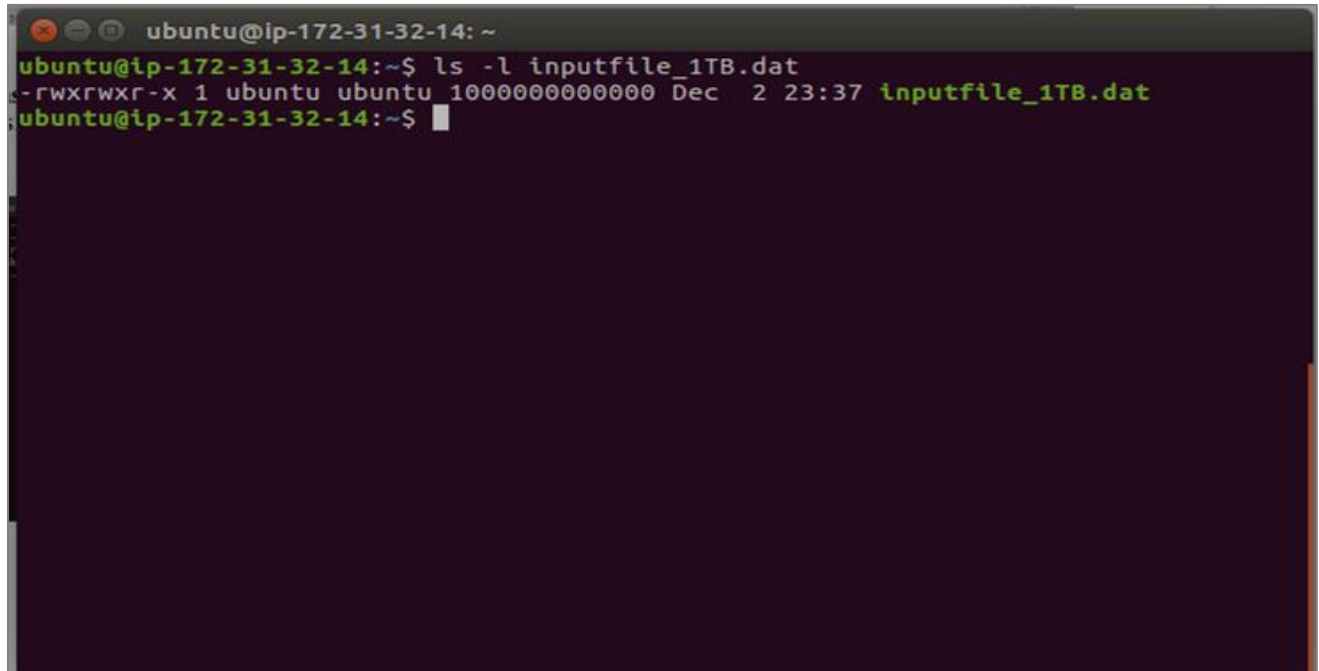
As we can see in the image that a checksum is generated and it shows that it has no duplicate keys.

After this the code displays a message saying that all records are found to be in order and that the sorting was successful.

Similarly we will carry out all the operations for 1TB of dataset with same steps just the different dataset.

External Sort with 1 TB Dataset

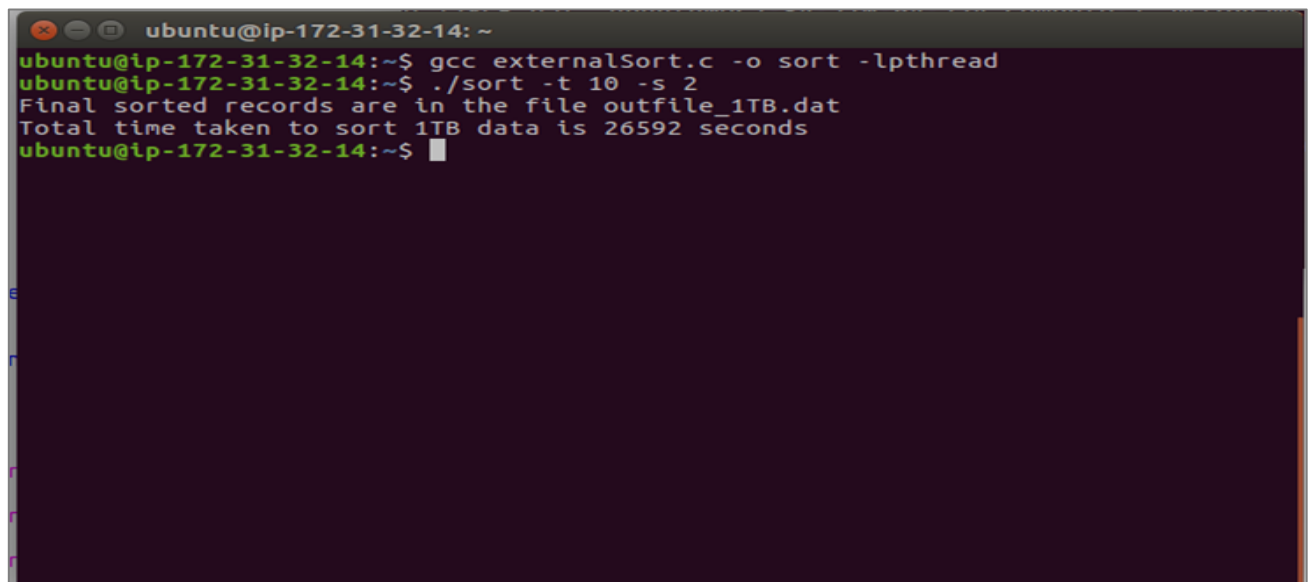
Input File Size



```
ubuntu@ip-172-31-32-14: ~  
ubuntu@ip-172-31-32-14:~$ ls -l inputfile_1TB.dat  
-rwxrwxr-x 1 ubuntu ubuntu 1000000000000 Dec  2 23:37 inputfile_1TB.dat  
ubuntu@ip-172-31-32-14:~$
```

As we can see from the screenshot, it shows that 1 TB of dataset is provided to the code as input.

1 TB sort Result



```
ubuntu@ip-172-31-32-14: ~  
ubuntu@ip-172-31-32-14:~$ gcc externalSort.c -o sort -lpthread  
ubuntu@ip-172-31-32-14:~$ ./sort -t 10 -s 2  
Final sorted records are in the file outfile_1TB.dat  
Total time taken to sort 1TB data is 26592 seconds  
ubuntu@ip-172-31-32-14:~$
```

The screenshot above shows that the result has been generated for the 1TB dataset and the data has been sorted and stored in an output file

The total time taken for the sorting process to complete is 26592 seconds.
That still is a good time because it is computed on a single node
It shows that the time taken for sorting is 7.3 hours.

Validation of Result using Valsort



```
ubuntu@lp-172-31-32-14: ~  
ubuntu@lp-172-31-32-14:~$ ./valsort outfile_1TB.dat  
Records: 10000000000  
Checksum: 2f987241eb  
Duplicate keys: 0  
SUCCESS - all records are in order  
ubuntu@lp-172-31-32-14:~$
```

The image in the screenshot shows that the number of records generated is 10000000000 and has no duplicate keys.

The command displays a success message at the end which says that all the records are found to be in order and also that the operation is completed as desired.

Hadoop Terasort:

Referred this :

➔ <https://janzhou.org/2014/how-to-compile-hadoop.html>

How to compile and generate jar files:

```

root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# ls
container-executor  Hadoop.java  input128  rcc  yarn.cmd
hadoop             hdfs         mapred     test-container-executor
hadoop.cmd         hdfs.cmd    mapred.cmd yarn
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# javac -classpath ${HADOOP_CLASSPATH} Hadoop.java
javac: no source files
Usage: javac <options> <source files>
use -help for a list of possible options
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin#
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# cd ..
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5# bin/hadoop com.sun.tools.javac.Main bin/Hadoop.p.java
/usr/lib/jvm/java-8-oracle/jre/bin/java
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5# jar -cvf bin/Hadoop.jar bin/Hadoop
bin/Hadoop : no such file or directory
added manifest
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5# cd bin/
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# ls
container-executor  Hadoop.jar  hdfs  mapred.cmd  yarn.cmd
hadoop             Hadoop.java  hdfs.cmd  rcc
Hadoop.class       Hadoop$SortingMapper.class  input128  test-container-executor
hadoop.cmd         Hadoop$SortingReducer.class  mapred    yarn
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# rm Hadoop.jar
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# ls
container-executor  Hadoop.java  hdfs.cmd  rcc
hadoop             Hadoop$SortingMapper.class  input128  test-container-executor
Hadoop.class       Hadoop$SortingReducer.class  mapred    yarn
hadoop.cmd         hdfs  mapred.cmd  yarn.cmd
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# jar cvf Hadoop.jar Hadoop.class Hadoop$SortingMapper.class Hadoop$SortingReducer.class
added manifest
adding: Hadoop.class(in = 1925) (out= 1025)(deflated 46%)
adding: Hadoop$SortingMapper.class(in = 1558) (out= 644)(deflated 58%)
adding: Hadoop$SortingReducer.class(in = 1644) (out= 704)(deflated 57%)
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# ls
container-executor  Hadoop.jar  hdfs  mapred.cmd  yarn.cmd
hadoop             Hadoop.java  hdfs.cmd  rcc
Hadoop.class       Hadoop$SortingMapper.class  input128  test-container-executor
hadoop.cmd         Hadoop$SortingReducer.class  mapred    yarn
root@ip-172-31-14-235:/home/ubuntu/hadoop-2.6.5/bin# |

```

Hadoop Setup on i3.large and i3.4xlarge:

1. Login to new EC2 instance i3.large or i3.4xlarge(or any other instance you create)
2. Login as root user and install all base packages:
sudo su
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
3. Check the java version just to be sure you've installed proper version for java:
java -version
4. Download latest stable Hadoop using wget

```
wget http://www.trieuvan.com/apache/hadoop/common/hadoop-2.6.5/hadoop-2.6.5.tar.gz
```

```
tar xzf hadoop-2.6.5.tar.gz
```

5. Create a directory to store all data for hadoop.
mkdir hadoopdata
6. Add the Hadoop related environment variables in your bash file.
Vi ~/.bashrc

Copy and paste these environment variables.

```
export HADOOP_HOME=/home/ubuntu/hadoop-2.6.5  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"  
export JAVA_HOME=/usr/lib/jvm/java-8-oracle  
PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

7. Refresh the bash settings by using this command:

```
source ~/.bashrc
```

8. We'll setup hadoop for password less ssh access:

```
ssh-keygen -t rsa -P ""  
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

check password less ssh access to localhost:

```
ssh localhost
```

and then exit.

9. Now, we've got to set the hadoop config files, has to be changed in order.

The files changed are:

- a. core-site.xml

```
<configuration>  
<property>  
<name>hadoop.tmp.dir</name>  
<value>/home/ubuntu/hadooptmp/hadoop-${user.name}</value>  
<description>A base for other temporary directories.</description>  
</property>  
<property>  
<name>fs.default.name</name>  
<value>hdfs://localhost:9000</value>  
</property></configuration>
```

- b. Hadoop-env.sh

```
# The java implementation to use.  
export JAVA_HOME=/usr/lib/jvm/java-8-oracle  
  
# 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"}
```

c. yarn.site.xml

```
<!-- Site specific YARN configuration properties -->
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

d. hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/ubuntu/hadoopdata/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/ubuntu/hadoopdata/hdfs/datanode</value>
  </property>
</configuration>
```

e. mapred-site.xml

```
<configuration>
  <property>

    <name>mapred.job.tracker</name>
    <value>localhost:9001</value>
  </property>

</configuration>
```

10. Now we format HDFS filesystem via NameNode:

hdfs namenode -format

11. Start all services:

start-all.sh

We can check for hadoop processes/daemons running on hadoop with Java Virtual Machine Process Status Tool

Command:

jps

Hadoop with 128GB dataset:

```
64
anaconda3
Anaconda3-4.1.1-Linux-x86_64.sh
certs
demo1
ubuntu@ip-172-31-9-206:~$ HadoopTests/
-bash: HadoopTests/: Is a directory
ubuntu@ip-172-31-9-206:~$ cd HadoopTests/
ubuntu@ip-172-31-9-206:~/HadoopTests$ ls
hadoop.jar      SortHadoop.class      WordCount.jar
hadoop-sort.jar SortHadoop$SortingMapper.class
input128        SortHadoop$SortingReducer.class
ubuntu@ip-172-31-9-206:~/HadoopTests$ ls -li
total 125000036
264252 -rw-r--r-- 1 root root 1496 Dec 2 23:31 hadoop.jar
31744003 -rw-r--r-- 1 root root 3154 Dec 3 00:18 hadoop-sort.jar
2304274 -rwxrwxr-x 1 ubuntu ubuntu 128000000000 Dec 2 08:29 input128
264251 -rw-r--r-- 1 root root 1941 Dec 2 23:26 SortHadoop.class
256053 -rw-r--r-- 1 root root 1570 Dec 2 23:26 SortHadoop$Sorting
Mapper.class
264250 -rw-r--r-- 1 root root 1656 Dec 2 23:26 SortHadoop$Sorting
Reducer.class
31744002 -rw-r--r-- 1 root root 342 Dec 3 00:17 WordCount.jar
ubuntu@ip-172-31-9-206:~/HadoopTests$ |
```


Hadoop with 1TB Dataset:

```
Mapper.class
 264250 -rw-r--r-- 1 root root          1656 Dec  2 23:26 SortHadoop$Sorting
Reducer.class
31744002 -rw-r--r-- 1 root root          342 Dec  3 00:17 WordCount.jar
ubuntu@ip-172-31-9-206:~/HadoopTests$ cd ..
ubuntu@ip-172-31-9-206:~$ ls
32                                gensort-linux-1.5.tar.gz  HadoopTests
64                                gpl-2.0.txt             hadooptmp
anaconda3                       hadoop-2.6.5            input
Anaconda3-4.1.1-Linux-x86_64.sh hadoop-2.6.5.tar.gz     pyTeraSort.py
certs                            hadoopdata              SparkSort.py
demo1                            HadoopSort.java
ubuntu@ip-172-31-9-206:~$ cd 64/
ubuntu@ip-172-31-9-206:~/64$ ls
externalSort.c  gensort  input1Tb  sort  spark_sort.py  valsort
ubuntu@ip-172-31-9-206:~/64$ ls -li
total 97656572
 264242 -rw-rw-r-- 1 ubuntu ubuntu          8499 Dec  2 07:46 externalSort.c
2304273 -rwxrwxr-x 1 ubuntu ubuntu       141045 Mar 17  2013 gensort
2304278 -rwxrwxr-x 1 ubuntu ubuntu 1000000000000 Dec  2 08:45 input1Tb
2304275 -rwxrwxr-x 1 ubuntu ubuntu        18968 Dec  2 07:49 sort
 264245 -rw----- 1 ubuntu ubuntu          759 Dec  2 08:12 spark_sort.py
2304272 -rwxrwxr-x 1 ubuntu ubuntu       134558 Mar 17  2013 valsort
ubuntu@ip-172-31-9-206:~/64$ |
```

Performance

Hadoop Sort Execution

For 128 GB dataset

```
ubuntu@ip-172-31-14-235:~$ cd hadoop-2.6.5/bin/
ubuntu@ip-172-31-14-235:~/hadoop-2.6.5/bin$ ls
container-executor  input
hadoop              input128
Hadoop.class        input1TB
hadoop.cmd          mapred
Hadoop.jar          mapred.cmd
Hadoop.java         rcc
Hadoop$SortingMapper.class  test-container-executor
Hadoop$SortingReducer.class  yarn
hdfs                yarn.cmd
hdfs.cmd
ubuntu@ip-172-31-14-235:~/hadoop-2.6.5/bin$ hadoop jar Hadoop.jar Hadoop input128 output128|
```

```

INFO input.FileInputFormat: Total input paths to process : 1
INFO mapreduce.JobSubmitter: number of splits:75
INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459023141669_0001
INFO impl.YarnClientImpl: Submitted application application_1459023141669_0001
INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1459023141669_0001/
INFO mapreduce.Job: Running job: job_1459023141669_0001
INFO mapreduce.Job: Job job_1459023141669_0001 running in uber mode : false
INFO mapreduce.Job:  map 0% reduce 0%
INFO mapreduce.Job:  map 1% reduce 0%
INFO mapreduce.Job:  map 2% reduce 0%
INFO mapreduce.Job:  map 3% reduce 0%
INFO mapreduce.Job:  map 4% reduce 0%
INFO mapreduce.Job:  map 5% reduce 0%
INFO mapreduce.Job:  map 6% reduce 0%
INFO mapreduce.Job:  map 7% reduce 0%
INFO mapreduce.Job:  map 8% reduce 0%
INFO mapreduce.Job:  map 9% reduce 0%
INFO mapreduce.Job:  map 10% reduce 0%
INFO mapreduce.Job:  map 11% reduce 0%
INFO mapreduce.Job:  map 12% reduce 0%
INFO mapreduce.Job:  map 13% reduce 0%
INFO mapreduce.Job:  map 14% reduce 0%
INFO mapreduce.Job:  map 15% reduce 0%
INFO mapreduce.Job:  map 16% reduce 0%
INFO mapreduce.Job:  map 17% reduce 0%
INFO mapreduce.Job:  map 18% reduce 0%
INFO mapreduce.Job:  map 19% reduce 0%
INFO mapreduce.Job:  map 20% reduce 0%
INFO mapreduce.Job:  map 21% reduce 0%
INFO mapreduce.Job:  map 22% reduce 1%
INFO mapreduce.Job:  map 22% reduce 2%
INFO mapreduce.Job:  map 22% reduce 3%
INFO mapreduce.Job:  map 22% reduce 4%
INFO mapreduce.Job:  map 24% reduce 4%
INFO mapreduce.Job:  map 24% reduce 5%
INFO mapreduce.Job:  map 25% reduce 5%

```

```

INFO mapreduce.Job:  map 46% reduce 15%
INFO mapreduce.Job:  map 47% reduce 15%
INFO mapreduce.Job:  map 49% reduce 15%
INFO mapreduce.Job:  map 50% reduce 15%
INFO mapreduce.Job:  map 51% reduce 15%
INFO mapreduce.Job:  map 52% reduce 15%
INFO mapreduce.Job:  map 53% reduce 15%
INFO mapreduce.Job:  map 54% reduce 15%
INFO mapreduce.Job:  map 55% reduce 15%
INFO mapreduce.Job:  map 56% reduce 15%
INFO mapreduce.Job:  map 57% reduce 15%
INFO mapreduce.Job:  map 58% reduce 15%
INFO mapreduce.Job:  map 58% reduce 16%
INFO mapreduce.Job:  map 59% reduce 16%
INFO mapreduce.Job:  map 60% reduce 16%
INFO mapreduce.Job:  map 61% reduce 16%
INFO mapreduce.Job:  map 62% reduce 16%
INFO mapreduce.Job:  map 63% reduce 16%
INFO mapreduce.Job:  map 64% reduce 16%
INFO mapreduce.Job:  map 64% reduce 17%
INFO mapreduce.Job:  map 65% reduce 17%
INFO mapreduce.Job:  map 65% reduce 18%
INFO mapreduce.Job:  map 65% reduce 19%
INFO mapreduce.Job:  map 65% reduce 20%
INFO mapreduce.Job:  map 66% reduce 20%
INFO mapreduce.Job:  map 67% reduce 20%
INFO mapreduce.Job:  map 68% reduce 20%
INFO mapreduce.Job:  map 69% reduce 20%
INFO mapreduce.Job:  map 69% reduce 21%
INFO mapreduce.Job:  map 69% reduce 22%
INFO mapreduce.Job:  map 69% reduce 23%
INFO mapreduce.Job:  map 70% reduce 23%
INFO mapreduce.Job:  map 71% reduce 23%
INFO mapreduce.Job:  map 72% reduce 23%
INFO mapreduce.Job:  map 73% reduce 23%
INFO mapreduce.Job:  map 74% reduce 23%
INFO mapreduce.Job:  map 75% reduce 23%
INFO mapreduce.Job:  map 76% reduce 23%
INFO mapreduce.Job:  map 77% reduce 23%

```

```
INFO mapreduce.Job: map 100% reduce 76%
INFO mapreduce.Job: map 100% reduce 77%
INFO mapreduce.Job: map 100% reduce 78%
INFO mapreduce.Job: map 100% reduce 79%
INFO mapreduce.Job: map 100% reduce 80%
INFO mapreduce.Job: map 100% reduce 81%
INFO mapreduce.Job: map 100% reduce 82%
INFO mapreduce.Job: map 100% reduce 83%
INFO mapreduce.Job: map 100% reduce 84%
INFO mapreduce.Job: map 100% reduce 85%
INFO mapreduce.Job: map 100% reduce 86%
INFO mapreduce.Job: map 100% reduce 87%
INFO mapreduce.Job: map 100% reduce 88%
INFO mapreduce.Job: map 100% reduce 89%
INFO mapreduce.Job: map 100% reduce 90%
INFO mapreduce.Job: map 100% reduce 91%
INFO mapreduce.Job: map 100% reduce 92%
INFO mapreduce.Job: map 100% reduce 93%
INFO mapreduce.Job: map 100% reduce 94%
INFO mapreduce.Job: map 100% reduce 95%
INFO mapreduce.Job: map 100% reduce 96%
INFO mapreduce.Job: map 100% reduce 97%
INFO mapreduce.Job: map 100% reduce 98%
INFO mapreduce.Job: map 100% reduce 99%
INFO mapreduce.Job: map 100% reduce 100%
INFO mapreduce.Job: Job job_1459023141669_0001 completed successfully
```

Output:

Head -5 output 128

```
!$%&'()*+,-./0123456789:;<=>?@A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` { | } ~ ! $ % & ' ( ) * + , - . / : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` { | } ~
```

The above screenshots are a collection of operations performed in the Hadoop Map Reduce method. As the theory says Hadoop is a method which performs sorting in many steps, It reads data from cluster, perform an operation, write results to the cluster, read updated data from the cluster, perform next operation, write next result to cluster. So the whole series of operations can be seen in our screenshot in the form of percentages.

The whole sorting process is divided in percentages from 0 to 100.

The last screenshot displays the output of the dataset sorted by Hadoop.

For 1 TB Dataset

```
ubuntu@ip-172-31-14-235:~$ ls
32                                gensort-linux-1.5.tar.gz  hadoop-2.6.5.tar.gz
64                                gpl-2.0.txt              hadoopdata
Anaconda3-4.1.1-Linux-x86_64.sh  hadoop-2.6.5             hadoop.tmp
ubuntu@ip-172-31-14-235:~$ cd hadoop-2.6.5/
ubuntu@ip-172-31-14-235:~/hadoop-2.6.5$ cd bin/
ubuntu@ip-172-31-14-235:~/hadoop-2.6.5/bin$ ls
container-executor      input
hadoop                  input128
Hadoop.class            input1TB
hadoop.cmd              mapred
Hadoop.jar              mapred.cmd
Hadoop.java             rcc
Hadoop$SortingMapper.class test-container-executor
Hadoop$SortingReducer.class yarn
hdfs                    yarn.cmd
hdfs.cmd
ubuntu@ip-172-31-14-235:~/hadoop-2.6.5/bin$ hadoop jar Hadoop.jar Hadoop input1TB output1TB|
```

```
INFO input.FileInputFormat: Total input paths to process : 1
INFO mapreduce.JobSubmitter: number of splits:75
INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459023141669_0001
INFO impl.YarnClientImpl: Submitted application application_1459023141669_0001
INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1459023141669_0001/
INFO mapreduce.Job: Running job: job_1459023141669_0001
INFO mapreduce.Job: Job job_1459023141669_0001 running in uber mode : false
INFO mapreduce.Job:  map 0% reduce 0%
INFO mapreduce.Job:  map 1% reduce 0%
INFO mapreduce.Job:  map 2% reduce 0%
INFO mapreduce.Job:  map 3% reduce 0%
INFO mapreduce.Job:  map 4% reduce 0%
INFO mapreduce.Job:  map 5% reduce 0%
INFO mapreduce.Job:  map 6% reduce 0%
INFO mapreduce.Job:  map 7% reduce 0%
INFO mapreduce.Job:  map 8% reduce 0%
INFO mapreduce.Job:  map 9% reduce 0%
INFO mapreduce.Job:  map 10% reduce 0%
INFO mapreduce.Job:  map 11% reduce 0%
INFO mapreduce.Job:  map 12% reduce 0%
INFO mapreduce.Job:  map 13% reduce 0%
INFO mapreduce.Job:  map 14% reduce 0%
INFO mapreduce.Job:  map 15% reduce 0%
INFO mapreduce.Job:  map 16% reduce 0%
INFO mapreduce.Job:  map 17% reduce 0%
INFO mapreduce.Job:  map 18% reduce 0%
INFO mapreduce.Job:  map 19% reduce 0%
INFO mapreduce.Job:  map 20% reduce 0%
INFO mapreduce.Job:  map 21% reduce 0%
INFO mapreduce.Job:  map 22% reduce 1%
INFO mapreduce.Job:  map 22% reduce 2%
INFO mapreduce.Job:  map 22% reduce 3%
INFO mapreduce.Job:  map 22% reduce 4%
INFO mapreduce.Job:  map 24% reduce 4%
INFO mapreduce.Job:  map 24% reduce 5%
INFO mapreduce.Job:  map 25% reduce 5%
```



```

INFO mapreduce.Job: map 46% reduce 15%
INFO mapreduce.Job: map 47% reduce 15%
INFO mapreduce.Job: map 49% reduce 15%
INFO mapreduce.Job: map 50% reduce 15%
INFO mapreduce.Job: map 51% reduce 15%
INFO mapreduce.Job: map 52% reduce 15%
INFO mapreduce.Job: map 53% reduce 15%
INFO mapreduce.Job: map 54% reduce 15%
INFO mapreduce.Job: map 55% reduce 15%
INFO mapreduce.Job: map 56% reduce 15%
INFO mapreduce.Job: map 57% reduce 15%
INFO mapreduce.Job: map 58% reduce 15%
INFO mapreduce.Job: map 58% reduce 16%
INFO mapreduce.Job: map 59% reduce 16%
INFO mapreduce.Job: map 60% reduce 16%
INFO mapreduce.Job: map 61% reduce 16%
INFO mapreduce.Job: map 62% reduce 16%
INFO mapreduce.Job: map 63% reduce 16%
INFO mapreduce.Job: map 64% reduce 16%
INFO mapreduce.Job: map 64% reduce 17%
INFO mapreduce.Job: map 65% reduce 17%
INFO mapreduce.Job: map 65% reduce 18%
INFO mapreduce.Job: map 65% reduce 19%
INFO mapreduce.Job: map 65% reduce 20%
INFO mapreduce.Job: map 66% reduce 20%
INFO mapreduce.Job: map 67% reduce 20%
INFO mapreduce.Job: map 68% reduce 20%
INFO mapreduce.Job: map 69% reduce 20%
INFO mapreduce.Job: map 69% reduce 21%
INFO mapreduce.Job: map 69% reduce 22%
INFO mapreduce.Job: map 69% reduce 23%
INFO mapreduce.Job: map 70% reduce 23%
INFO mapreduce.Job: map 71% reduce 23%
INFO mapreduce.Job: map 72% reduce 23%
INFO mapreduce.Job: map 73% reduce 23%
INFO mapreduce.Job: map 74% reduce 23%
INFO mapreduce.Job: map 75% reduce 23%
INFO mapreduce.Job: map 76% reduce 23%
INFO mapreduce.Job: map 77% reduce 23%

```

```

INFO mapreduce.Job: map 100% reduce 76%
INFO mapreduce.Job: map 100% reduce 77%
INFO mapreduce.Job: map 100% reduce 78%
INFO mapreduce.Job: map 100% reduce 79%
INFO mapreduce.Job: map 100% reduce 80%
INFO mapreduce.Job: map 100% reduce 81%
INFO mapreduce.Job: map 100% reduce 82%
INFO mapreduce.Job: map 100% reduce 83%
INFO mapreduce.Job: map 100% reduce 84%
INFO mapreduce.Job: map 100% reduce 85%
INFO mapreduce.Job: map 100% reduce 86%
INFO mapreduce.Job: map 100% reduce 87%
INFO mapreduce.Job: map 100% reduce 88%
INFO mapreduce.Job: map 100% reduce 89%
INFO mapreduce.Job: map 100% reduce 90%
INFO mapreduce.Job: map 100% reduce 91%
INFO mapreduce.Job: map 100% reduce 92%
INFO mapreduce.Job: map 100% reduce 93%
INFO mapreduce.Job: map 100% reduce 94%
INFO mapreduce.Job: map 100% reduce 95%
INFO mapreduce.Job: map 100% reduce 96%
INFO mapreduce.Job: map 100% reduce 97%
INFO mapreduce.Job: map 100% reduce 98%
INFO mapreduce.Job: map 100% reduce 99%
INFO mapreduce.Job: map 100% reduce 100%
INFO mapreduce.Job: Job job_1459023141669_0001 completed successfully

```

Head -5 output 1 TB

Compute Time: 618 Minutes (10.3 hours)

Private Ips are stored in etc/hadoop/slaves file for slave node and etc/hadoop/masters file for master node

Slave7-172.31.8.141

22

<input type="checkbox"/>	Name	Instance ID	Inst	Availability Zone	Instance State	Status Checks	Alarm Status
<input type="checkbox"/>	Master	i-00557d4d1ca5ccf5b	i3...	us-west-1b	running	2/2 checks ...	None
<input checked="" type="checkbox"/>	Slave3	i-0434d8834a0a1aa...	i3...	us-west-1c	running	2/2 checks ...	None
<input type="checkbox"/>	Ignore	i-044e22fc4c457e981	i3.l...	us-west-1c	terminated		None
<input type="checkbox"/>	Slave5	i-08ba20615964cd5ff	i3...	us-west-1b	running	2/2 checks ...	None
<input type="checkbox"/>	Slave6	i-0a904907e2f6ddf9a	i3.l...	us-west-1c	running	2/2 checks ...	None

Instance: **i-0434d8834a0a1aa94 (Slave3)** Public DNS: [ec2-13-56-151-164.us-west-1.compute.amazonaws.com](#)

Description	Status Checks	Monitoring	Tags
Instance ID	i-0434d8834a0a1aa94	Public DNS (IPv4)	ec2-13-56-151-164.us-west-1.compute.amazonaws.com
Instance state	running	IPv4 Public IP	13.56.151.164
Instance type	i3.xlarge	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-28-110.us-west-1.compute.internal
Availability zone	us-west-1c	Private IPs	172.31.28.110
Security groups	launch-wizard-5 . view inbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-eb071f8f

```
INFO mapreduce.Job: Running job: job_1459040042109_0001
INFO mapreduce.Job: Job job_1459040042109_0001 running in uber mode :

INFO mapreduce.Job:  map 0% reduce 0%
INFO mapreduce.Job:  map 1% reduce 0%
INFO mapreduce.Job:  map 2% reduce 0%
INFO mapreduce.Job:  map 3% reduce 0%
INFO mapreduce.Job:  map 4% reduce 0%
INFO mapreduce.Job:  map 5% reduce 0%
INFO mapreduce.Job:  map 6% reduce 0%
INFO mapreduce.Job:  map 7% reduce 0%
INFO mapreduce.Job:  map 8% reduce 0%
INFO mapreduce.Job:  map 9% reduce 0%
INFO mapreduce.Job:  map 10% reduce 0%
INFO mapreduce.Job:  map 11% reduce 0%
INFO mapreduce.Job:  map 12% reduce 0%
INFO mapreduce.Job:  map 13% reduce 0%
INFO mapreduce.Job:  map 14% reduce 0%
INFO mapreduce.Job:  map 15% reduce 0%
INFO mapreduce.Job:  map 16% reduce 0%
INFO mapreduce.Job:  map 17% reduce 0%
INFO mapreduce.Job:  map 18% reduce 0%
INFO mapreduce.Job:  map 19% reduce 0%
INFO mapreduce.Job:  map 20% reduce 0%
INFO mapreduce.Job:  map 21% reduce 0%
INFO mapreduce.Job:  map 22% reduce 0%
INFO mapreduce.Job:  map 23% reduce 0%
INFO mapreduce.Job:  map 24% reduce 0%
```

```
INFO mapreduce.Job: map 95% reduce 6%
INFO mapreduce.Job: map 96% reduce 6%
INFO mapreduce.Job: map 97% reduce 6%
INFO mapreduce.Job: map 98% reduce 6%
INFO mapreduce.Job: map 99% reduce 6%
INFO mapreduce.Job: map 100% reduce 6%
INFO mapreduce.Job: map 100% reduce 7%
INFO mapreduce.Job: map 100% reduce 8%
INFO mapreduce.Job: map 100% reduce 9%
INFO mapreduce.Job: map 100% reduce 10%
INFO mapreduce.Job: map 100% reduce 11%
INFO mapreduce.Job: map 100% reduce 12%
INFO mapreduce.Job: map 100% reduce 13%
INFO mapreduce.Job: map 100% reduce 14%
INFO mapreduce.Job: map 100% reduce 15%
INFO mapreduce.Job: map 95% reduce 0%
INFO mapreduce.Job: map 96% reduce 0%
INFO mapreduce.Job: map 96% reduce 1%
INFO mapreduce.Job: map 97% reduce 1%
INFO mapreduce.Job: map 98% reduce 1%
INFO mapreduce.Job: map 99% reduce 1%
INFO mapreduce.Job: map 99% reduce 2%
INFO mapreduce.Job: map 100% reduce 2%
INFO mapreduce.Job: map 100% reduce 3%
INFO mapreduce.Job: map 100% reduce 4%
INFO mapreduce.Job: map 100% reduce 5%
```

Here in the above screenshot you can see that Maps 100% but Reduces 15% and gets into that loop again which was a problem, I first thought it was something wrong with the code, but I just had to insert the numbers of mappers and reducers in Mapred-site.xml in etc/hadoop and then it just worked.

Check the next screenshot!


```
INFO mapreduce.Job:  map 100% reduce 75%
INFO mapreduce.Job:  map 100% reduce 76%
INFO mapreduce.Job:  map 100% reduce 77%
INFO mapreduce.Job:  map 100% reduce 78%
INFO mapreduce.Job:  map 100% reduce 79%
INFO mapreduce.Job:  map 100% reduce 80%
INFO mapreduce.Job:  map 100% reduce 81%
INFO mapreduce.Job:  map 100% reduce 82%
INFO mapreduce.Job:  map 100% reduce 83%
INFO mapreduce.Job:  map 100% reduce 84%
INFO mapreduce.Job:  map 100% reduce 85%
INFO mapreduce.Job:  map 100% reduce 86%
INFO mapreduce.Job:  map 100% reduce 87%
INFO mapreduce.Job:  map 100% reduce 88%
INFO mapreduce.Job:  map 100% reduce 89%
INFO mapreduce.Job:  map 100% reduce 90%
INFO mapreduce.Job:  map 100% reduce 91%
INFO mapreduce.Job:  map 100% reduce 92%
INFO mapreduce.Job:  map 100% reduce 93%
INFO mapreduce.Job:  map 100% reduce 94%
INFO mapreduce.Job:  map 100% reduce 95%
INFO mapreduce.Job:  map 100% reduce 96%
INFO mapreduce.Job:  map 100% reduce 97%
INFO mapreduce.Job:  map 100% reduce 98%
INFO mapreduce.Job:  map 100% reduce 99%
INFO mapreduce.Job:  map 100% reduce 100%
```

And the time elapsed(Compute time) comes out to be: 477.53 minutes(7.95hrs)

Spark Terasort:

Spark Setup on i3.large and i3.4xlarge:

1. Download and Install python :

```
wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86\_64.sh
```

```
bash Anaconda3-4.1.1-Linux-x86_64.sh
```

2. To know what version has been installed:

```
which python
```

3. Installing Scala:

```
sudo apt-get install scala
```

To check whether it has been installed properly or not:

```
scala -version
```

4. Install Spark:

```
wget http://archive.apache.org/dist/spark/spark-2.0.0/spark-2.0.0-bin-hadoop2.7.tgz
```

```
sudo tar -zxvf spark-2.0.0-bin-hadoop2.7.tgz
```

5. We've got to let python know where spark is:

```
export PATH=$SPARK_HOME:$PATH
```

```
export PYTHONPATH=$SPARK_HOME/python:$PYTHONPATH
```

Sparksort on 128 GB of dataset:

```

root@ip-172-31-9-206:/home/ubuntu# ls
32                                gensort-linux-1.5.tar.gz  HadoopTests
64                                gp1-2.0.txt              hadooptmp
anaconda3                        hadoop-2.6.5             input
Anaconda3-4.1.1-Linux-x86_64.sh  hadoop-2.6.5.tar.gz      pyTeraSort.py
certs                            hadoopdata
demo1                            HadoopSort.java
root@ip-172-31-9-206:/home/ubuntu# cd HadoopTests/
root@ip-172-31-9-206:/home/ubuntu/HadoopTests# ls
hadoop.jar      SortHadoop.class          SparkSort.py
hadoop-sort.jar SortHadoop$SortingMapper.class WordCount.jar
input128        SortHadoop$SortingReducer.class
root@ip-172-31-9-206:/home/ubuntu/HadoopTests# python SparkSort.py input128 outp
ut128
The program 'python' can be found in the following packages:
* python-minimal
* python3
Try: apt install <selected package>
root@ip-172-31-9-206:/home/ubuntu/HadoopTests# exit
exit
ubuntu@ip-172-31-9-206:~$ cd HadoopTests/
ubuntu@ip-172-31-9-206:~/HadoopTests$ ls
hadoop.jar      SortHadoop.class          SparkSort.py
hadoop-sort.jar SortHadoop$SortingMapper.class WordCount.jar
input128        SortHadoop$SortingReducer.class
ubuntu@ip-172-31-9-206:~/HadoopTests$ python SparkSort.py input128 output128
Input file path is...input128
Output file path is...output128
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
17/12/03 20:52:18 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
17/12/03 20:52:19 WARN Utils: Service 'SparkUI' could not bind on port 4040. Att
empting port 4041.
[Stage 0:>                                     (61 + 1) / 3815]|

```

Stage 0 of execution:

```

ubuntu@ip-172-31-9-206:~/HadoopTests$ python SparkSort.py input128 output128
Input file path is...input128
Output file path is...output128
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
17/12/03 20:52:18 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
17/12/03 20:52:19 WARN Utils: Service 'SparkUI' could not bind on port 4040. Att
empting port 4041.
[Stage 0:====>                                (324 + 1) / 3815]|

```

```

ubuntu@ip-172-31-9-206:~/HadoopTests$ python SparkSort.py input128 output128
Input file path is...input128
Output file path is...output128
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/12/03 20:52:18 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/12/03 20:52:19 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
[Stage 0:=====> (1722 + 1) / 3815]

```

Stage 1 of execution:

```

ubuntu@ip-172-31-9-206:~/HadoopTests$ python SparkSort.py input128 output128
Input file path is...input128
Output file path is...output128
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/12/03 20:52:18 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/12/03 20:52:19 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
[Stage 0:=====> (2282 + 1) / 3815]

```

This shows that the number of parts in which the dataset was divided is 3815

The operation is divided in 3 stages basically because the operation is performed on the whole dataset at once.

First stage reads the data from the dataset, the second stage performs the sorting operation on the data and then the third operation stored the processed data back in the output folder.

Sparksort on 1 TB dataset:

```
ubuntu@ip-172-31-9-206:~$ ls
32                                gensort-linux-1.5.tar.gz  HadoopTests
64                                gpl-2.0.txt              hadooptmp
anaconda3                       hadoop-2.6.5             input
Anaconda3-4.1.1-Linux-x86_64.sh hadoop-2.6.5.tar.gz      pyTeraSort.py
certs                           hadoopdata               SparkSort.py
demo1                           HadoopSort.java

ubuntu@ip-172-31-9-206:~$ mv SparkSort.py 64/
ubuntu@ip-172-31-9-206:~$ ls'
> ^C
ubuntu@ip-172-31-9-206:~$ ls
32                                gensort-linux-1.5.tar.gz  HadoopTests
64                                gpl-2.0.txt              hadooptmp
anaconda3                       hadoop-2.6.5             input
Anaconda3-4.1.1-Linux-x86_64.sh hadoop-2.6.5.tar.gz      pyTeraSort.py
certs                           hadoopdata               SparkSort.py
demo1                           HadoopSort.java

ubuntu@ip-172-31-9-206:~$ cd 64/
ubuntu@ip-172-31-9-206:~/64$ ls
externalSort.c  gensort  input1Tb  sort  spark_sort.py  SparkSort.py  valsort
```

Stage 0 of execution:

```
ubuntu@ip-172-31-9-206:~/64$ python SparkSort.py input1Tb output1Tb
Input file path is...input1Tb
Output file path is...output1Tb
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/12/03 20:47:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[Stage 0:=====>                                (359 + 1) / 2981]
```

```
ubuntu@ip-172-31-9-206:~/64$ python SparkSort.py input1Tb output1TB
Input file path is...input1Tb
Output file path is...output1TB
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/12/03 20:47:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[Stage 0:=====> (561 + 1) / 2981]|
```

```
ubuntu@ip-172-31-9-206:~/64$ python SparkSort.py input1Tb output1TB
Input file path is...input1Tb
Output file path is...output1TB
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/12/03 20:47:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[Stage 0:=====> (2503 + 1) / 2981]|
```

Stage 1 of execution:

```
ubuntu@ip-172-31-9-206:~/64$ python SparkSort.py input1Tb output1TB
Input file path is...input1Tb
Output file path is...output1TB
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/12/03 20:47:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[Stage 1:=> (103 + 1) / 2981]|
```


Stage 2 of Execution:

```

    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
sorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
    at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
    at py4j.Gateway.invoke(Gateway.java:280)
    at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
    at py4j.commands.CallCommand.execute(CallCommand.java:79)
    at py4j.GatewayConnection.run(GatewayConnection.java:214)
    at java.lang.Thread.run(Thread.java:748)

ubuntu@ip-172-31-9-206:~/64$ python spark_sort.py input1Tb output
Using spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/12/02 22:22:05 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
17/12/02 22:22:07 WARN Utils: Service 'SparkUI' could not bind on port 4040. Att
empting port 4041.
Path of input file ->input1Tb
Path of output file ->output
[Stage 2:=====> (2365 + 1) / 2981]\

```

The screenshot above displays a collection of different operations and all of them connects to one operation of sorting.

As said above the operations are divided in three stages that collectively gives us an output and stores in the desired format in the folder.

1) Performance:

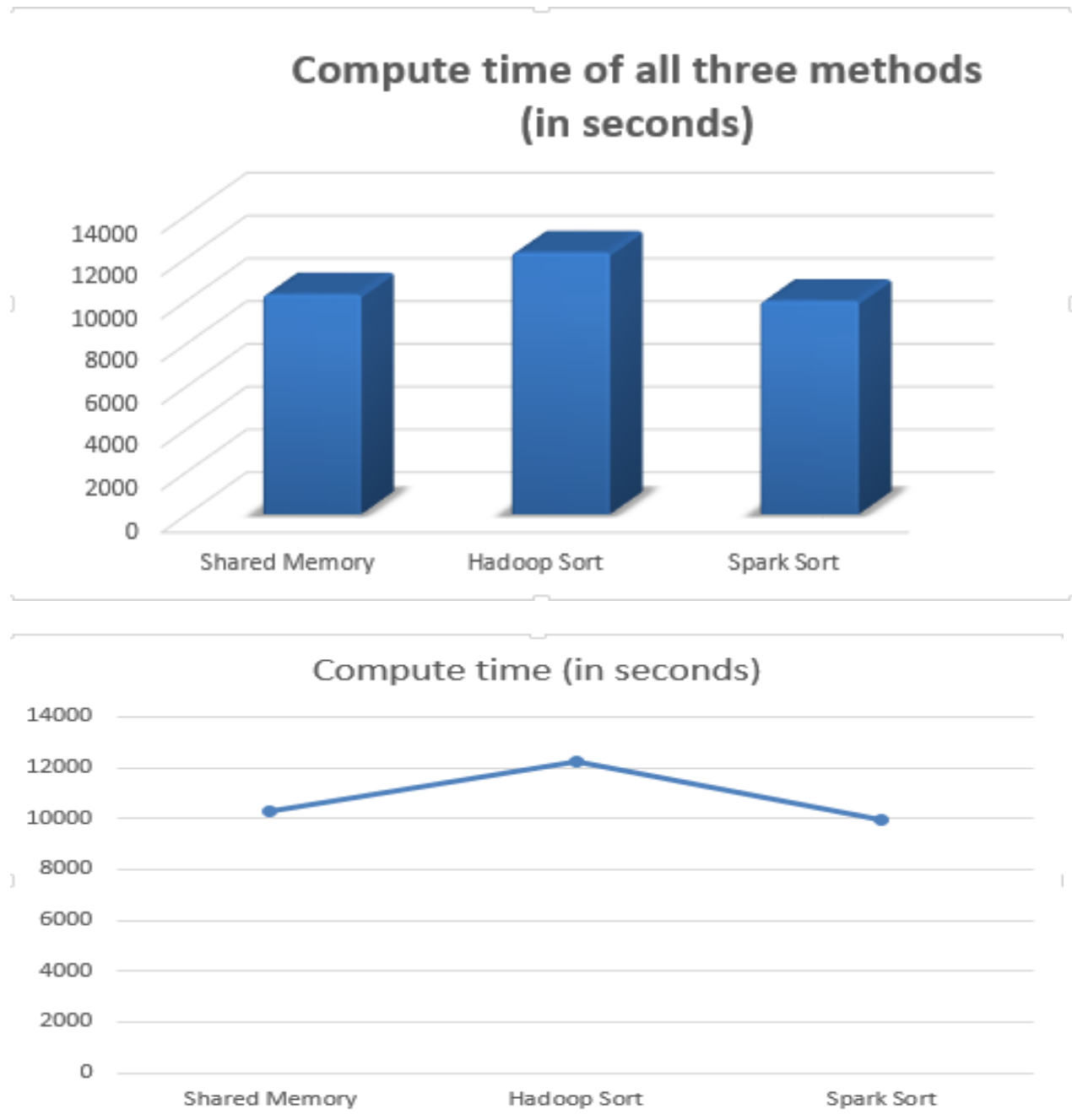
Table 1: Performance evaluation of TeraSort

Experiment (instance/dataset)	Shared Memory Terasort	Hadoop Terasort	Spark Terasort	MPI Terasort
Compute Time (sec) [1xi3.large 128GB]	10280 sec	12235 sec	9972 sec	-
Data Read (GB) [1xi3.large 128GB]	125 GB	120 GB	122 GB	-
Data Write (GB) [1xi3.large 128GB]	126 GB	120 GB	122 GB	-
I/O Throughput (MB/sec) [1xi3.large 128GB]	112.412 MB/sec	85.7 MB/sec	105.15 MB/sec	-
Compute Time (sec) [1xi3.4xlarge 1TB]	26592 sec	37080 sec	25391 sec	-
Data Read (GB) [1xi3.4xlarge 1TB]	920 GB	920 GB	920 GB	-
Data Write (GB) [1xi3.4xlarge 1TB]	920 GB	920 GB	920 GB	-
I/O Throughput (MB/sec) [1xi3.4xlarge 1TB]	315.45 MB/sec	226 MB/sec	330 Mb/sec	-
Compute Time (sec) [8xi3.large 1TB]	N/A	28652 sec	24649 sec	-
Data Read (GB) [8xi3.large 1TB]	N/A	920 GB	920 GB	-
Data Write (GB) [8xi3.large 1TB]	N/A	920 GB	920GB	-
I/O Throughput (MB/sec) [8xi3.large 1TB]	N/A	263.04 MB/sec	305.75 MB/sec	-
Speedup (weak scale)	2.80619	1.163	0.926	-
Efficiency (weak scale)	2.58677	1.290	1.030	-

Performance Report:

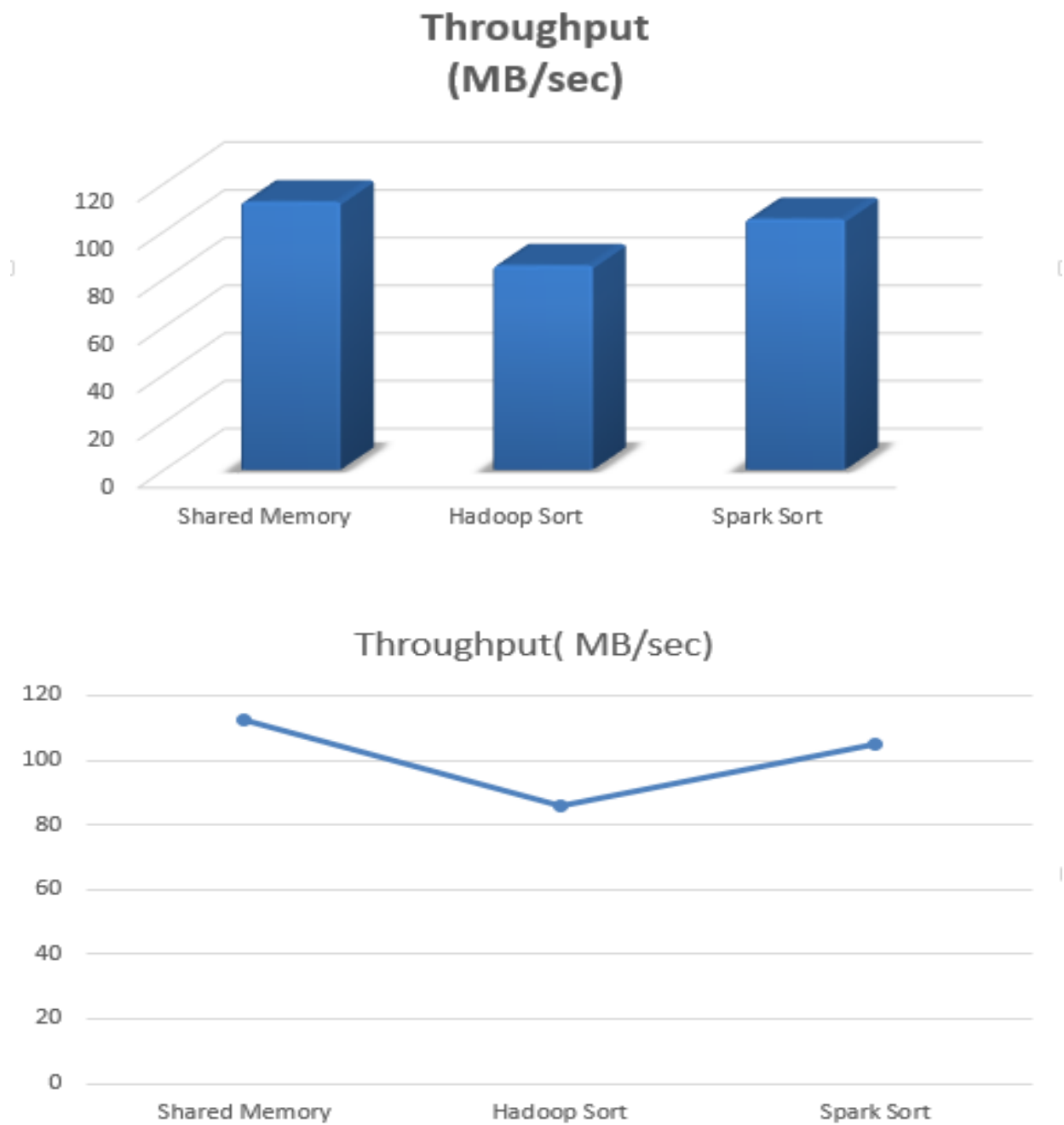
For 128 GB

Compute time comparison



It is visible from the graph that Spark Sort takes the least time of all and after that the shared memory sort is efficient. Hadoop sort is not at all efficient to be run on a single node. And that shared memory performs as good as spark on a small dataset and when it comes to the single node then the performance of shared memory is very good comparatively.

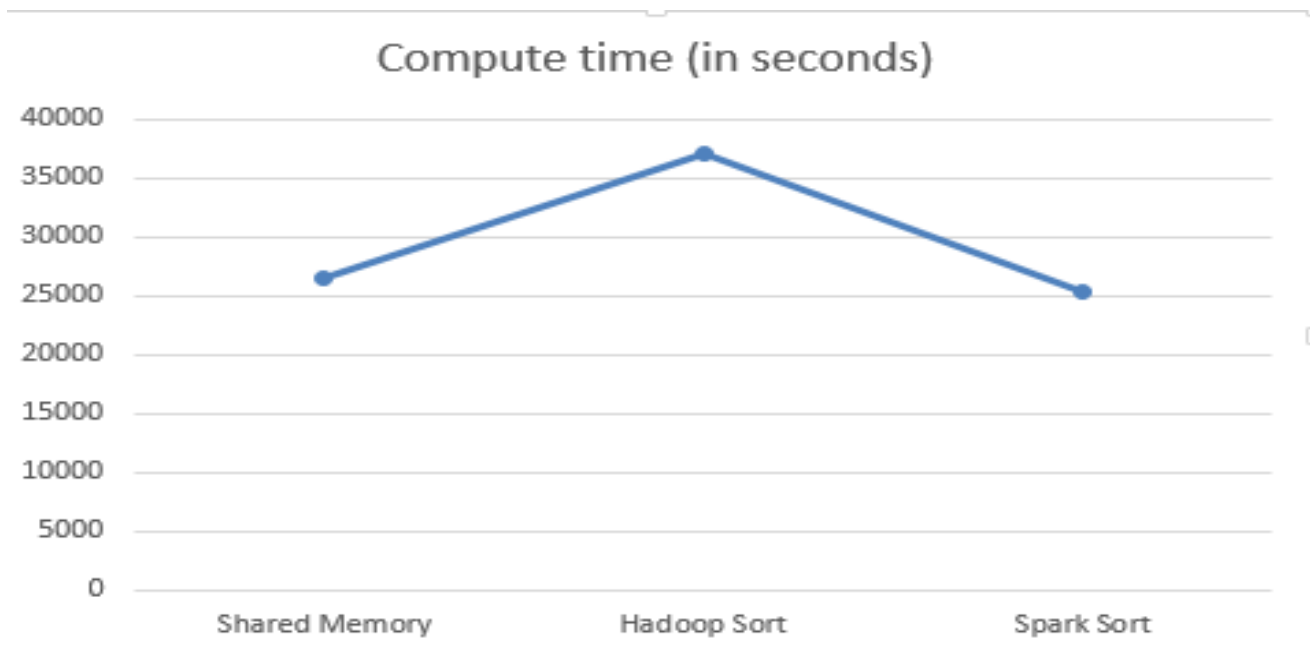
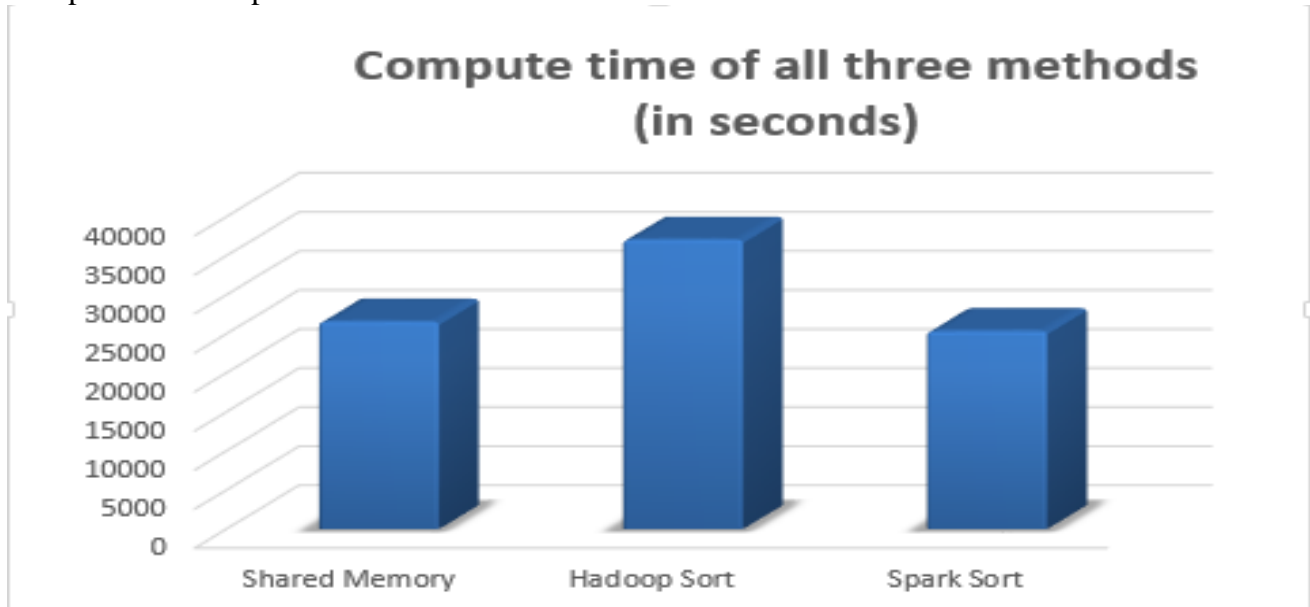
Throughput Comparison



As we can see from the graph, the throughput of shared memory sorting is the best amongst all. Since the sorting is done on a single node so the shared memory outperforms all other sorting methods.

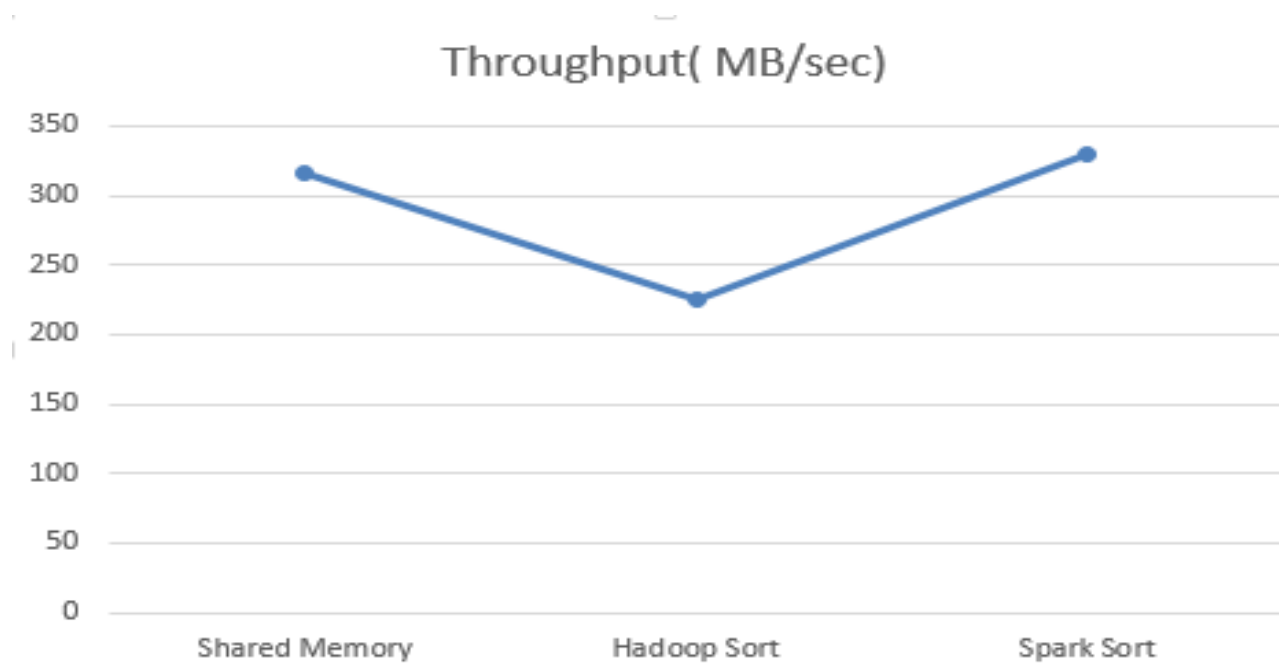
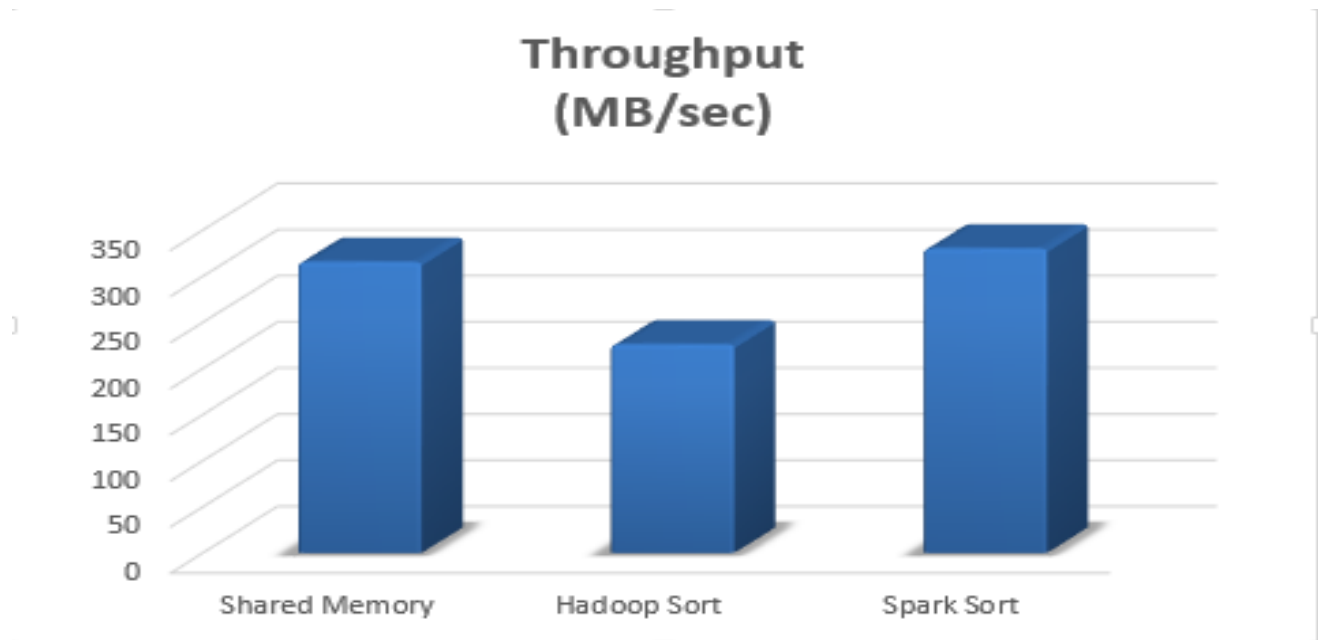
For 1 TB Dataset: (instance type i3.4xlarge)

Compute time comparison:



The Compute time for spark is the least in this case as seen from the graph.
The spark sort is efficient for performing sorting operations on bigger datasets with higher number of nodes.

Throughput comparison:

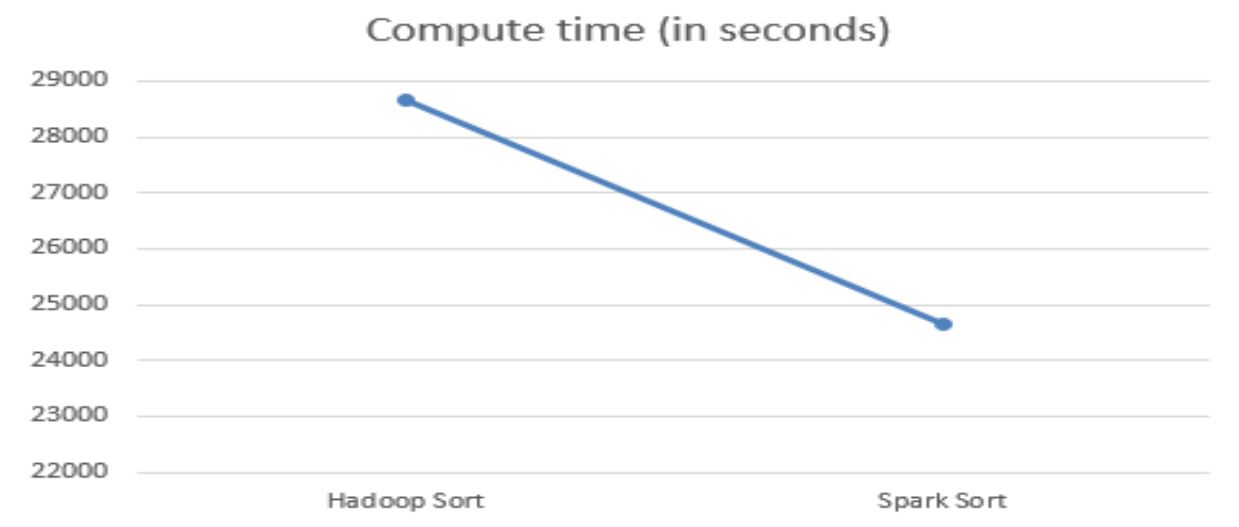
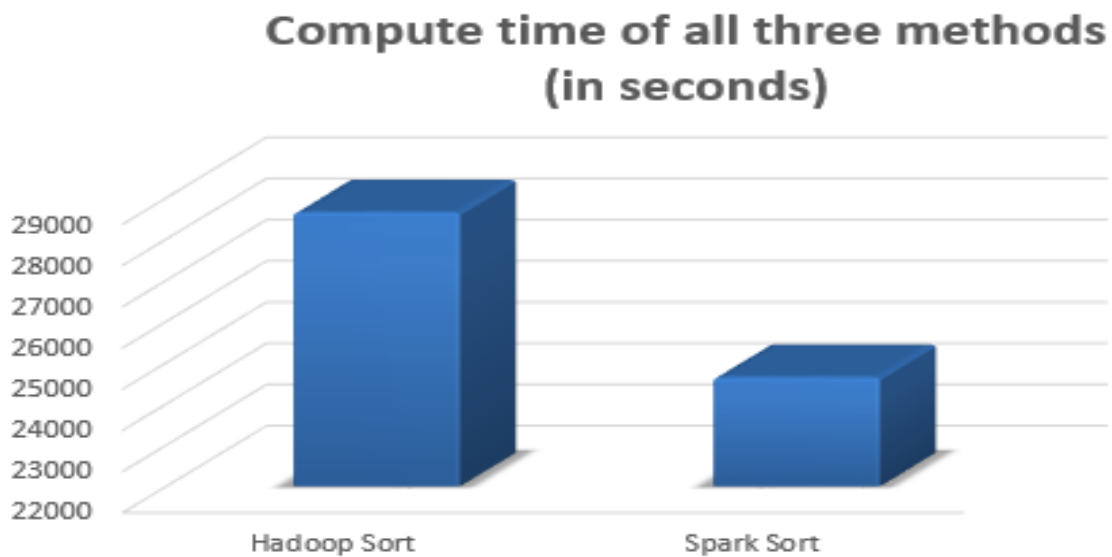


The throughput of Spark Sort is the best as it performs the fastest sorting amongst all. The Shared memory method has the second best throughput.

For 8 nodes in one cluster for instance type (i3.large)

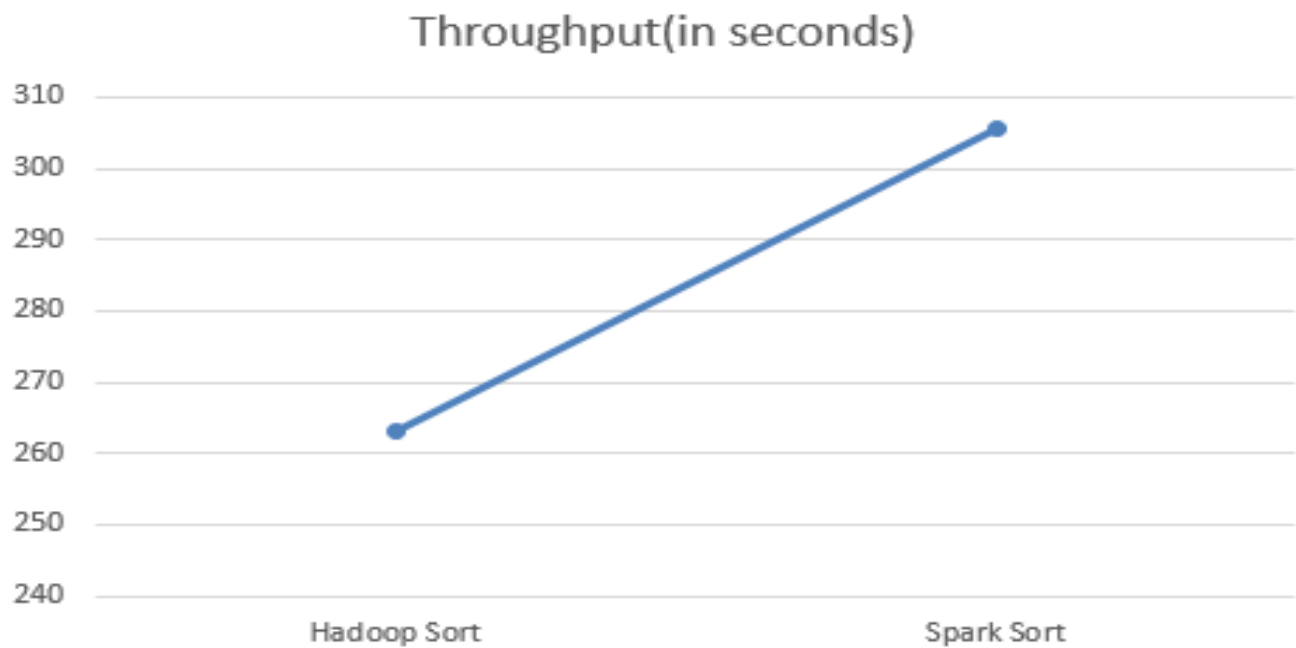
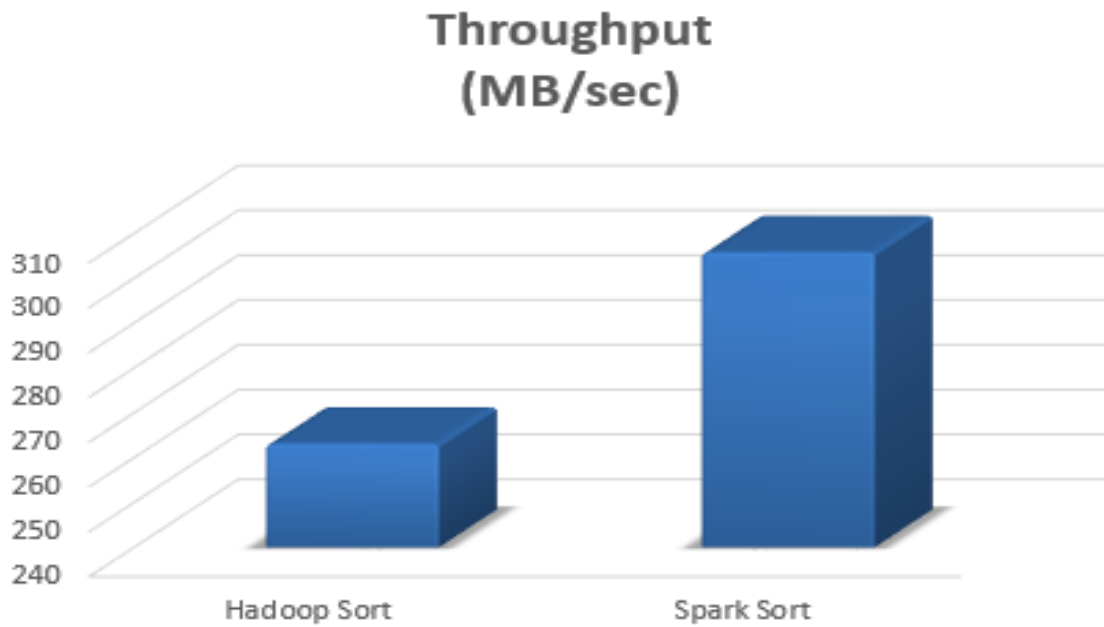
Comparison of Hadoop and Spark Sorting:

Compute time Comparison:



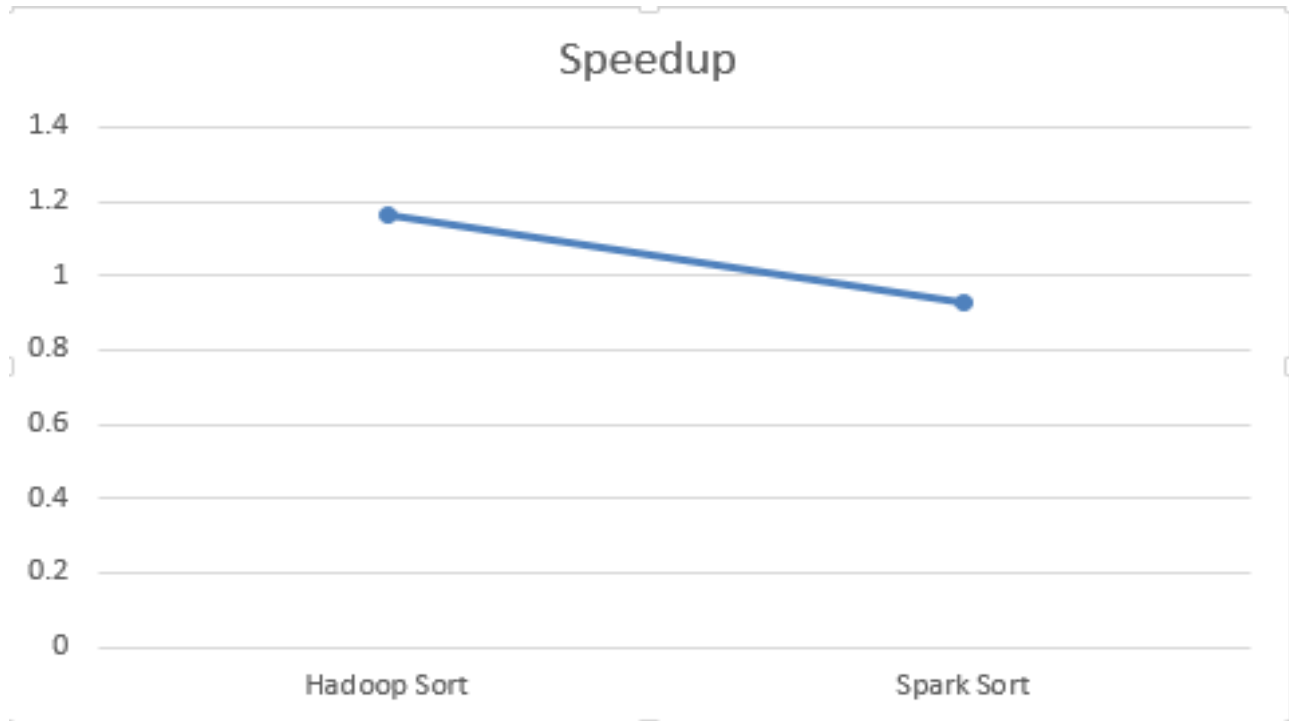
The compute time taken by Spark is considerably less than that taken by Hadoop.

Throughput comparison:

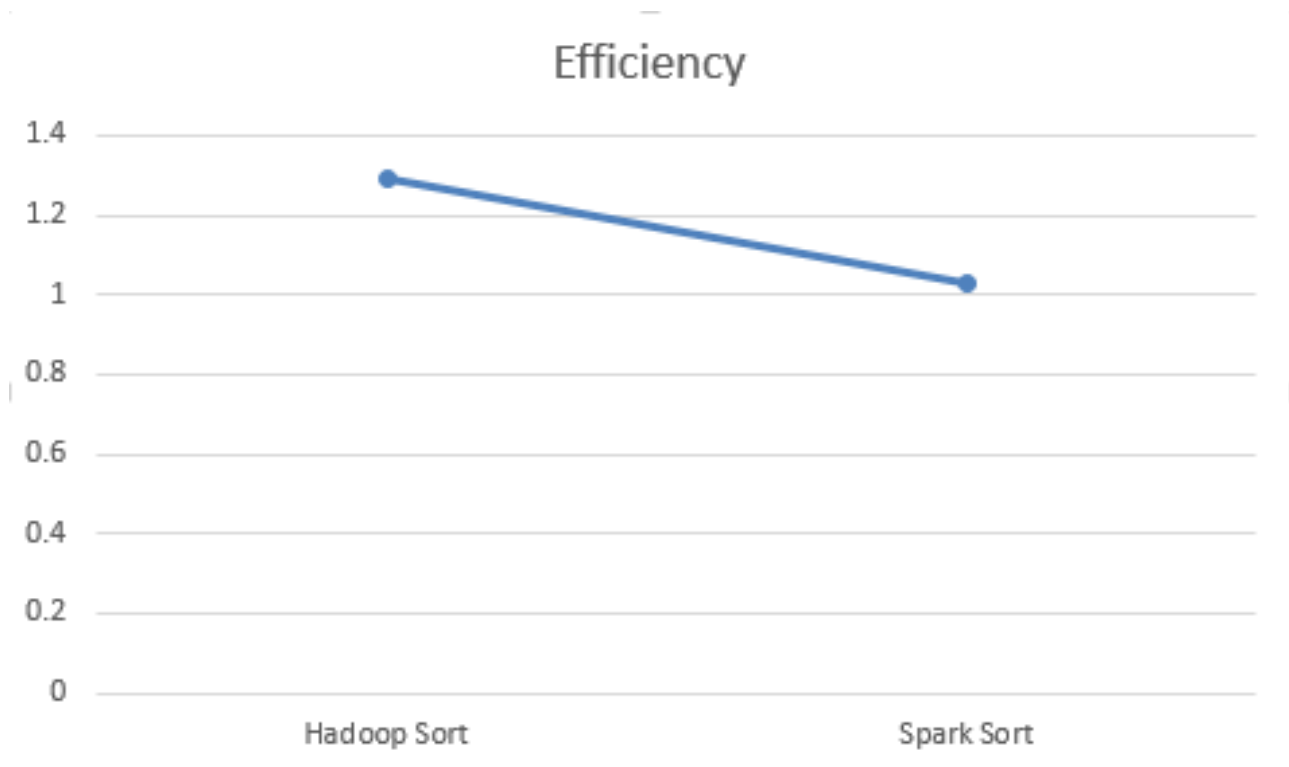


The Throughput of Spark is greater than Hadoop even in this type of setup.

Speedup for Hadoop and Spark:



Efficiency for Hadoop and Spark:



What conclusions can you draw?

The only conclusion that we can come to is that spark is the best sorting technique used in the whole assignment. And terasorting can be performed with all these sorting algorithms.

Which seems to be best at 1 node scale?

Performing sorting operations at 1 node is technically preferable with shared memory terasort, because with this method we are saved from setting up a cluster of nodes. Perform sorting of large datasets is quite difficult and time taking in a 1 node setup but it is the best option for relatively small datasets.

How about 8 nodes?

At 8 nodes we have a master and slave kind of distribution and Spark performs better than all other in this setup as it uses RDD (Resilient Distributed Datasets) and has better and updated architecture for data processing when compared to Hadoop. Spark can be as much as 10 times faster than MapReduce for batch processing and up to 100 times faster for in-memory analytics.

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

Spark is clearly the one to be most efficient to work with at 100 node scale or more. The reason for this is when compared with Hadoop MapReduce it operates in steps, the workflow for MapReduce is: “read data from cluster, perform an operation, write results to the cluster, read updated data from the cluster, perform next operation, write next result to cluster” and so on..... Spark on the other hand completes full data analytics operations in-memory and in near real-time: “Read data from the cluster, perform all of the requisite analytic operations, write results to the cluster, done”.

How about 1000 node scales?

As I have stated earlier Spark will be the better option for this 1000 node setup because it is more efficient single step algorithm and its results gets better for bigger datasets with a large distribution of nodes. From our calculations also we can see that the performance of Spark keeps on improving as the number of nodes keeps on increasing.

Compare your results with those from the Sort Benchmark [9], specifically the winners in 2013 and 2014 who used Hadoop and Spark.

In 2013, according to Daytona GraySort, Hadoop was the winner with the sorting speed of 1.42 TB/min as it calculated 102.5 TB in 4,328 seconds with 2100 nodes of distributed systems. In 2014, according to Daytona GraySort, Apache Spark was the winner with the sorting speed of 4.27 TB/min and it calculated 100 TB in 1,378 seconds with only 207 Amazon EC2 nodes.

Also, what can you learn from the CloudSort benchmark?

CloudSort benchmark is something that makes use of resources available at public cloud to perform sorting operations on the data. CloudSort proposes using the cloud to define a benchmark that measures the efficiency of external sort from a total-cost of ownership perspective. The CloudSort Benchmark is summarized as follows:

1. Sort a fixed number (currently 1012 , ~100TB) of randomly permuted 100-byte records with 10-byte keys
2. The sort must start with the input on a non-ephemeral, persistent store and finish with output on a nonephemeral, persistent store.
3. All operations must be performed on a commercially available public cloud.
4. The winner is the system with the minimum cost prorated for the duration of the sort.

In all CloudSort benchmark is one of the best public cloud benchmark system we have and it takes in consideration all the factors like cost and speed and efficiency for the calculation of benchmark.

Conclusion:

We have learned from this assignment that Spark performs better than Hadoop and Shared Memory. Spark performs better in the distributed environment and huge datasets because of the special features of the algorithm that it performs all the operations collectively and not in steps.

Ideally the shared memory should perform better on the single node irrespective of the size of the dataset. The shared memory proves to be true in the results from our programs.

We also learned the application of AWS and how to create the master and slave node in cluster for performing calculation of huge datasets.

Implementing MPI was tough and was taking us a lot of time to give output so couldn't include it in the report.

Though Spark was the fastest to sort the data we can see that Hadoop can also perform well for bigger datasets in cluster computing.

REFERENCE:

- [1] <https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/terasort/package-summary.html>
- [2] <http://ant.apache.org/bindownload.cgi>
- [3] <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- [4] http://hadoop.apache.org/docs/current1/mapred_tutorial.html
- [5] <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html>
- [6] <http://spark.apache.org/downloads.html>
- [7] <http://spark.apache.org/docs/latest/cluster-overview.html>
- [8] <http://www.ordinal.com/gensort.html>
- [9] <http://sortbenchmark.org>
- [10] http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf
- [11] <http://www.srccodes.com/p/article/45/run-hadoop-wordcount-mapreduce-example-windows>
- [12] <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Usage>
- [13] <https://sagarruchandani.wordpress.com/2015/08/01/hadoop-setting-up-hadoop-2-6-0-single-node-on-aws-ec2-ubuntu-ami/>
- [14] <https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297>