CS 553: CLOUD COMPUTING

PROGRAMMING ASSIGNMENT 2 -REPORT

Amit Gupta – A20376501

Saloni Chauhan – A20377221

Rohan Jain – A20378201

INTRODUCTION:

- The Primary Goal of this Programming Assignment is to get acquainted with the following components:

1) Amazon Web Services (AWS).
2) Apache Hadoop Framework.
3) Apache Spark Framework.

- This Programming Assignment involves implementing the **Sort Application** using 3 different approaches:

1) Shared Memory Sort.
2) Apache Hadoop.
3) Apache Spark.

- The Sorting Application should be able to read and sort Files **larger than Memory**.

- In order to benchmark the 3 different approaches to Sorting, 2 **DataSets** are created. One DataSet is of **128 GB** and the other is of **1 TB**. The DataSets are generated using **gensort.**

- This Report provides details related to the Installation Process of Apache Hadoop and Apache Spark Frameworks, Screenshots of the Sort performed on these Frameworks and comparisons of the Sort performed on both the Frameworks.

APACHE HADOOP:

Setting Up A Cluster Of 1 Node And 8 Nodes:

This Section outlines the steps required to setup a **Multi – Node Cluster** on Apache Hadoop and the modifications that need to be done to the Multi – Node Cluster to transform it into a **Single – Node Cluster**:

- A **Virtual Cluster of 8 Nodes** is setup on **Amazon Web Services (AWS)** using the various Tools and Services provided.
- An **AWS EC2 Instance** is created. While creating the Instance, the **Ubuntu Server LTS Amazon Machine Image (AMI)** is selected as the **Amazon Image**.
- While selecting the **Instance Type**, the **i3.large Spot Instance** is selected as the Instance Type. The number of Instances to be created is entered as **8**.
- Once the Instance is created successfully, a **PEM Key File** is generated. This Key File needs to be saved in a secure location on the Host (User) System for future use.

Installing Apache Hadoop:

- The **Host System** needs to be updated so that the required Packages, Files and Components are installed and updated. In order to update the Host System, the following Command needs to be executed:

```
sudo apt-get update
```

- The **Java Programming Language** also needs to be installed and updated on the Host System. This Task is performed by executing the following Commands:

```
sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update && sudo apt-get install oracle-jdk7-installer
```

- Once all the required Packages and Components have been installed and updated, Apache Hadoop can be installed. The following Commands need to be executed for the same:

```
wget http://apache.mirrors.tds.net/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz -P ~/Downloads

Unzip : sudo tar zxvf ~/Downloads/hadoop-* -C /usr/local

Move : sudo mv /usr/local/hadoop-* /usr/local/Hadoop
```

<u>Configuring Environment Variables:</u>

- Add **Apache Hadoop & Java Environment Variables** to ~/.profile and source them to current Shell Session. The following Commands need to be executed for the same:

```
export JAVA_HOME=/usr

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

```
          export HADOOP_HOME=/usr/local/Hadoop

          export PATH=$PATH:$HADOOP_HOME/bin

export HADOOP_CONF_DIR=/usr/local/hadoop/etc/Hadoop
```

- Then load these Environment Variables by sourcing the Profile. The following Command need to be executed for the same:

```
. ~/.profile
```

Apache Hadoop Configuration:

Configuring Apache Hadoop Directory:

- All the Configuration changes will be applied to the **Name Nodes** and all the **Data Nodes**.

- Files to Edit:

- **$HADOOP_CONF_DIR/hadoop-env.sh**
- **$HADOOP_CONF_DIR/core-site.xml**
- **$HADOOP_CONF_DIR/yarn-site.xml**
- **$HADOOP_CONF_DIR/mapred-site.xml**

Configuration On All Nodes:

```
sudo vim $HADOOP_CONF_DIR/hadoop-env.sh
```

- The only thing that needs to be changed is the location of JAVA_HOME in the file. Simply replace $ {JAVA_HOME} with /usr which is where Java was previously installed.

$HADOOP_CONF_DIR/hadoop-env.sh:

# The Java implementation to use:

export JAVA_HOME=/usr

- The next file to be modified is:

$HADOOP_CONF_DIR/core-site.xml:

```
<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>
```

- The next file to be modified is:

$HADOOP_CONF_DIR/yarn-site.xml:

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
```

SHADOOP_CONF_DIR/hdfs-site.xml:

```
<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>
```

- The last configuration file to be modified is:

$HADOOP_CONF_DIR/mapred-site.xml:

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

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

<u>Configuring The Name Node:</u>

- On the Name Node adding Hosts to **/etc/hosts:**

- Modifying the configurations in **$HADOOP_CONF_DIR/hdfs-site.xml**
- Defining the Hadoop Master in **$HADOOP_CONF_DIR/masters**
- Defining the Hadoop Slaves in **$HADOOP_CONF_DIR/slaves**

➢ Add each Node's **Public DNS** and **Host Name** to the list. The Host Name can be found by using the following Command:

```
echo $(hostname)
```

```
127.0.0.1 localhost
namenode_public_dns namenode_hostname
datanode1_public_dns datanode1_hostname
datanode2_public_dns datanode2_hostname
datanode3_public_dns datanode3_hostname
```

➢ The current Path where Data on the Name Node will reside does not exist, so we'll need to make this before starting HDFS:

```
namenode$ sudo mkdir -p$HADOOP_HOME/hadoop_data/hdfs/namenode
```

➢ Next, we'll need to add a Masters File to the **$ HADOOP_CONF _DIR** Directory:

```
namenode$ sudo touch $HADOOP_CONF_DIR/masters
```

➢ Then insert the NameNode's Host Name in the File **$ HADOOP_ CONF_DIR/masters**:

```
namenode_hostname
```

- We will also need to modify the slaves file in the **$ HADOOP_ CONF_DIR** Directory to the following. By default, localhost is present, but we can change it.

$HADOOP_CONF_DIR/slaves

```
datanode1_hostname
datanode2_hostname
datanode3_hostname
```

- Now that all Configurations are set on the Name Node, we will change the ownership of the **$HADOOP_HOME** Directory to the user **Ubuntu**:

```
namenode$ sudo chown -R ubuntu $HADOOP_HOME
```

Data Node Specific Configurations:

- Let's now move onto the final Configurations for the Data Nodes. We will need to first SSH into each Data Node and only configure the **$HADOOP_CONF_DIR/hdfs-site.xml** File:

$HADOOP_CONF_DIR/hdfs-site.xml:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///usr/local/hadoop/hadoop_data/hdfs/datanode</value>
  </property>
</configuration>
```

- Just like on the Name Node, we will need to create the Directory specified in the **$HADOOP_CONF_DIR/hdfs-site.xml** File.

```
datanodes$ sudo mkdir -p $HADOOP_HOME/hadoop_data/hdfs/datanode
```

- Now that all Configurations are set on the DataNode, we will change the ownership of the **$HADOOP_HOME** directory to the **Ubuntu** User:

```
datanodes$ sudo chown -R ubuntu $HADOOP_HOME
```

Start Apache Hadoop Cluster:

- We can now start up HDFS from the Name Node by first formatting it and then starting HDFS. An important thing to note is that every time the Name Node is formatted, all of the Data previously on it is lost.

```
namenode$ hdfs namenode -format
namenode$ $HADOOP_HOME/sbin/start-dfs.sh
```

Apache Hadoop Configuration Files:

conf / master:

- This File defines the Machines on which Apache Hadoop will initiate Secondary Name Nodes in the Multi – Node Cluster.

conf / slaves:

- This File lists the Hosts, one on each line, on which the Slave Daemons (i.e. Data Nodes and Task Trackers) of Apache Hadoop will be executed.

conf / core-site.xml:

- This File contains the Configuration Details for Apache Hadoop Core such as I/O Settings that are common to Apache Hadoop and MapReduce.

conf / hdfs-site.xml:

- This File contains the Configuration Settings for various HDFS Daemons such as the Name Node, the Secondary Name Node and the Data Node.

conf / mapred-site.xml:

- This File contains the Configuration Settings for various MapReduce Daemons.

Installation of Spark:

We have to download and install Anaconda for our python. The installation takes the following commands:

```
$ wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86_64.sh
```

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

Press Enter through the license agreements, then Enter yes to accept it then Enter to get to the default location

After then check which Python u are using:

```
$ which python
```

Then Configure Jupyter Notebook:

```
$ jupyter notebook --generate-config
```

Create the Certifications then:

```
$ mkdir certs
$ cd certs
```

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
```

Just fill the general instructions with some information :

Edit the Config File:

```
$ cd ~/.jupyter/
```

Then we will use visual editor (vi) to edit the file. Type:

```
$ vi jupyter_notebook_config.py
```

his is where you can either uncomment lines or add in your own (things such as adding password protection are an option here). We will keep things simple.

Press i on your keyboard to activate -INSERT-. Then at the top of the file type:

```
c = get_config()

# Notebook config this is where you saved your pem cert
c.NotebookApp.certfile = u'/home/ubuntu/certs/mycert.pem'

# Run on all IP addresses of your instance
c.NotebookApp.ip = '*'

# Don't open browser by default
c.NotebookApp.open_browser = False

# Fix port to 8888
c.NotebookApp.port = 8888
```

Press Esc to stop inserting. Then type a colon : and then type wq to write and quit the editor.

Check whether the jupyter notebook is running or not.

```
$ jupyter notebook
```

You’ll see an output saying that a jupyter notebook is running at all ip addresses at port 8888. Go to your own web browser (Google Chrome suggested) and type in your Public DNS for your Amazon EC2 instance followed by :8888. It should be in the form:

https://ec2-xx-xx-xxx-xxx.us-west-2.compute.amazonaws.com:8888

Install Java

```
$ sudo apt-get update
$ sudo apt-get install default-jre
$ sudo apt-get install scala
```

Install py4j:

```
$ export PATH=$PATH:$HOME/anaconda3/bin
$ conda install pip
$ which pip
$ pip install py4j
```

Install Spark and Hadoop:

```
$ 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
```

Tell Python where to find Spark:

```
$ export SPARK_HOME='/home/ubuntu/spark-2.0.0-bin-hadoop2.7'
$ export PATH=$SPARK_HOME:$PATH
$ export PYTHONPATH=$SPARK_HOME/python:$PYTHONPATH
```

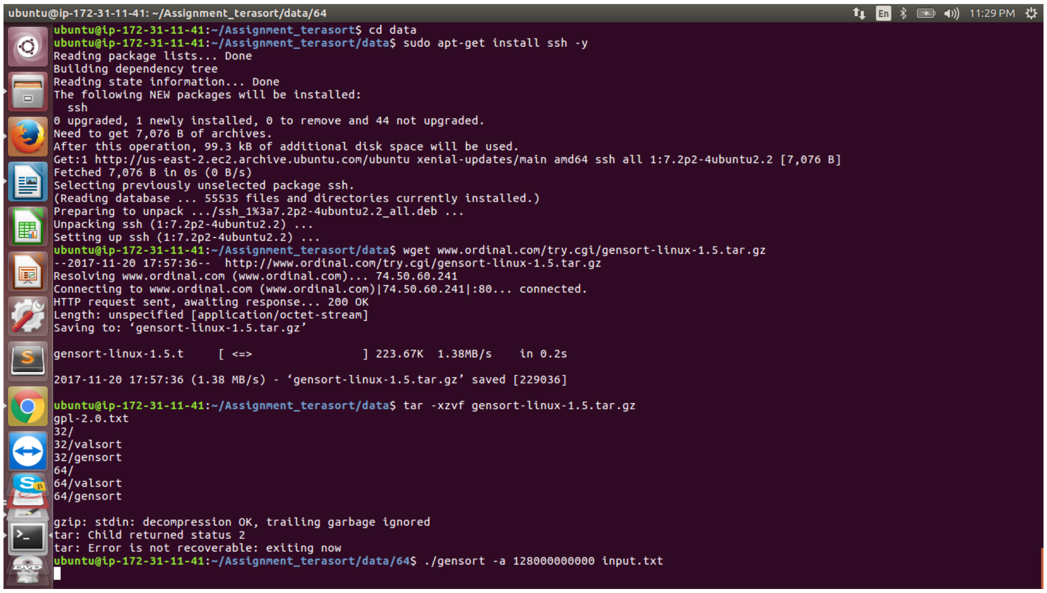
Launch Jupyter Notebook:

```
$ jupyter notebook
from pyspark import SparkContext
sc = SparkContext()
```

SCREENSHOTS:

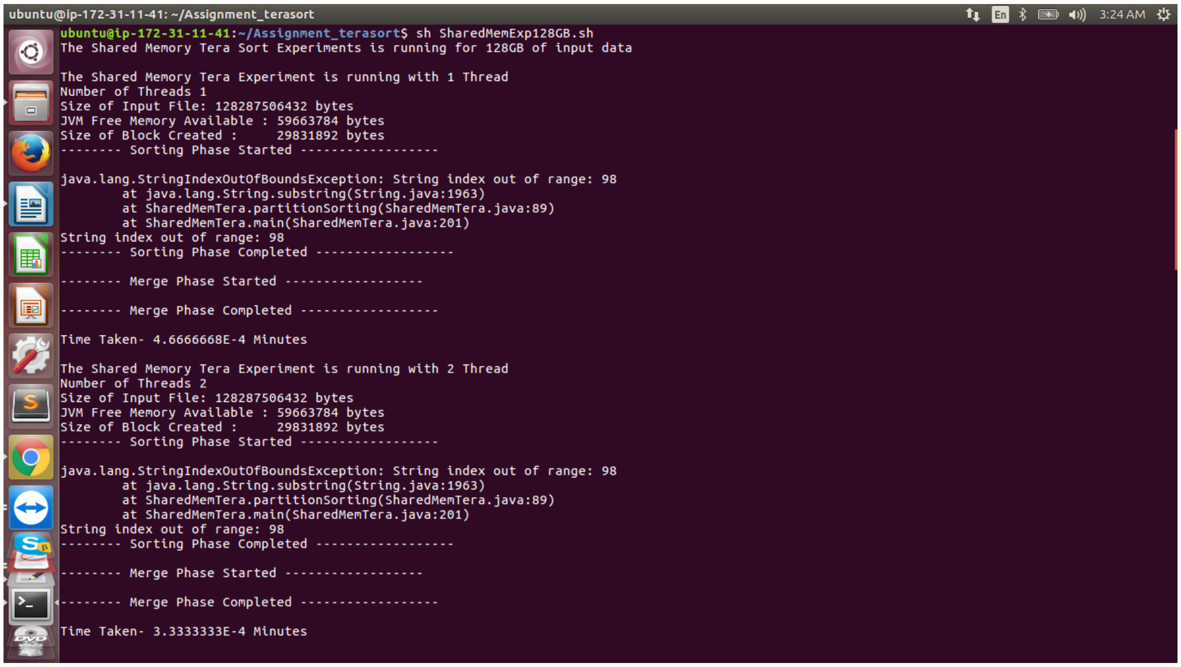
Shared Memory:

Generating 128 Gb dataset for shared Memory.



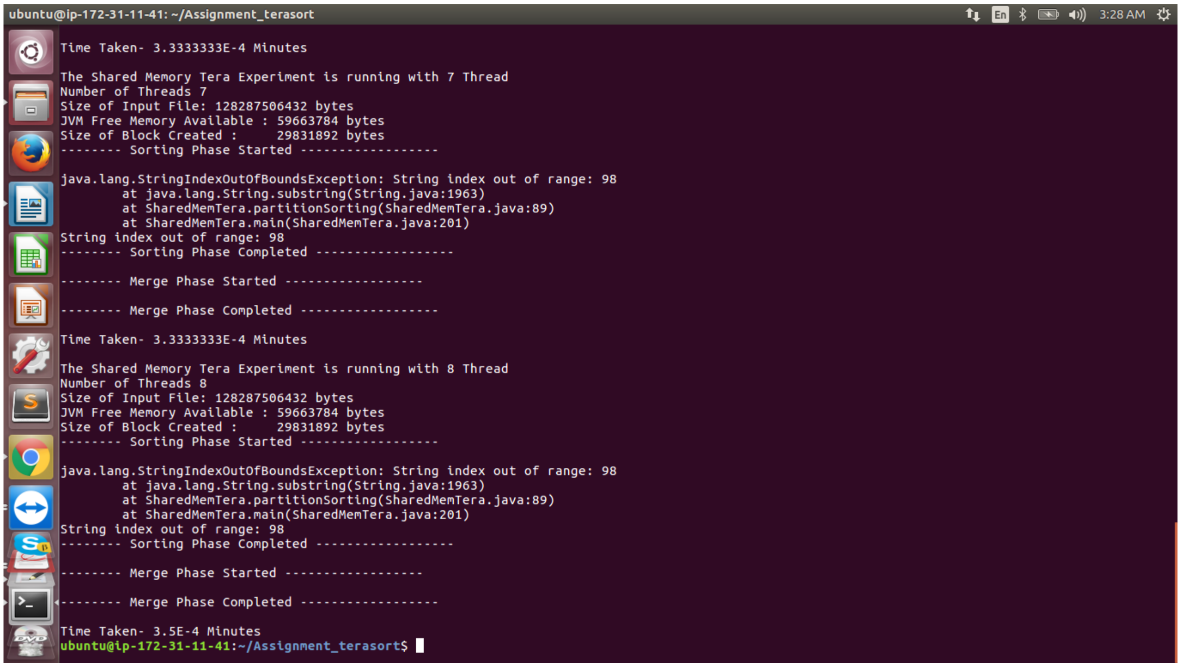
```
ubuntu@ip-172-31-11-41:~/Assignment_terasort$ cd data
ubuntu@ip-172-31-11-41:~/Assignment_terasort/data$ sudo apt-get install ssh -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ssh
0 upgraded, 1 newly installed, 0 to remove and 44 not upgraded.
Need to get 7,076 B of archives.
After this operation, 99.3 kB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial-updates/main amd64 ssh all 1:7.2p2-4ubuntu2.2 [7,076 B]
Fetched 7,076 B in 0s (0 B/s)
Selecting previously unselected package ssh.
(Reading database ... 55535 files and directories currently installed.)
Preparing to unpack .../ssh_1%3a7.2p2-4ubuntu2.2_all.deb ...
Unpacking ssh (1:7.2p2-4ubuntu2.2) ...
Setting up ssh (1:7.2p2-4ubuntu2.2) ...
ubuntu@ip-172-31-11-41:~/Assignment_terasort/data$ wget www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz
--2017-11-20 17:57:36--  http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz
Resolving www.ordinal.com (www.ordinal.com)... 74.50.60.241
Connecting to www.ordinal.com (www.ordinal.com)|74.50.60.241|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/octet-stream]
Saving to: ‘gensort-linux-1.5.tar.gz’

gensort-linux-1.5.t     [ <=>                ] 223.67K  1.38MB/s    in 0.2s

2017-11-20 17:57:36 (1.38 MB/s) - ‘gensort-linux-1.5.tar.gz’ saved [229036]

ubuntu@ip-172-31-11-41:~/Assignment_terasort/data$ tar -xzvf gensort-linux-1.5.tar.gz
gpl-2.0.txt
32/
32/valsort
32/gensort
64/
64/valsort
64/gensort

gzip: stdin: decompression OK, trailing garbage ignored
tar: Child returned status 2
tar: Error is not recoverable: exiting now
ubuntu@ip-172-31-11-41:~/Assignment_terasort/data/64$ ./gensort -a 128000000000 input.txt
```

Shared Memory running on 1 and 2 threads for 128 GB



```
ubuntu@ip-172-31-11-41:~/Assignment_terasort$ sh SharedMemExp128GB.sh
The Shared Memory Tera Sort Experiments is running for 128GB of input data

The Shared Memory Tera Experiment is running with 1 Thread
Number of Threads 1
Size of Input File: 128287506432 bytes
JVM Free Memory Available : 59663784 bytes
Size of Block Created :      29831892 bytes
-------- Sorting Phase Started -----------------

java.lang.StringIndexOutOfBoundsException: String index out of range: 98
        at java.lang.String.substring(String.java:1963)
        at SharedMemTera.partitionSorting(SharedMemTera.java:89)
        at SharedMemTera.main(SharedMemTera.java:201)
String index out of range: 98
-------- Sorting Phase Completed -----------------

-------- Merge Phase Started -----------------

-------- Merge Phase Completed -----------------

Time Taken- 4.6666668E-4 Minutes

The Shared Memory Tera Experiment is running with 2 Thread
Number of Threads 2
Size of Input File: 128287506432 bytes
JVM Free Memory Available : 59663784 bytes
Size of Block Created :      29831892 bytes
-------- Sorting Phase Started -----------------

java.lang.StringIndexOutOfBoundsException: String index out of range: 98
        at java.lang.String.substring(String.java:1963)
        at SharedMemTera.partitionSorting(SharedMemTera.java:89)
        at SharedMemTera.main(SharedMemTera.java:201)
String index out of range: 98
-------- Sorting Phase Completed -----------------

-------- Merge Phase Started -----------------

-------- Merge Phase Completed -----------------

Time Taken- 3.3333333E-4 Minutes
```

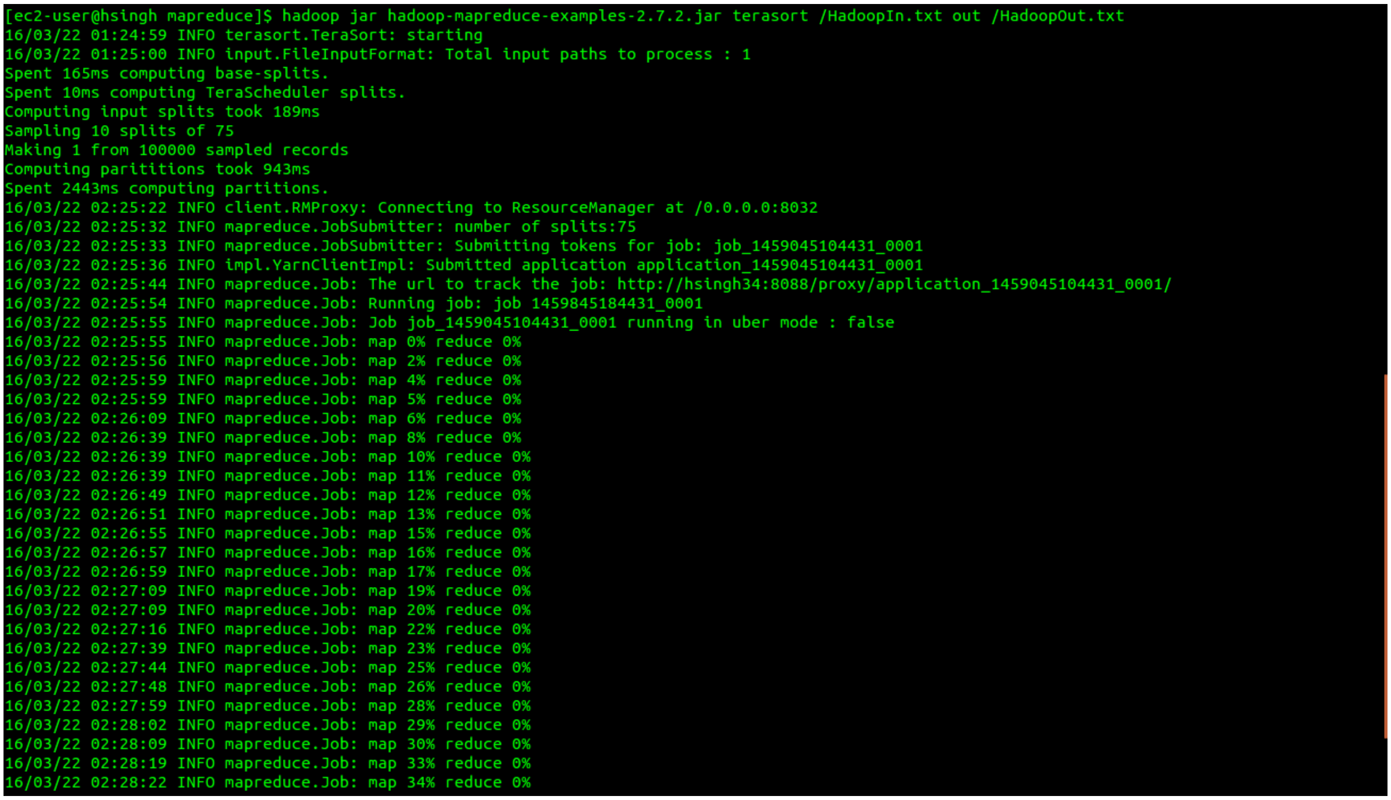
Shared Memory running on 7 and 8 threads for 128 GB



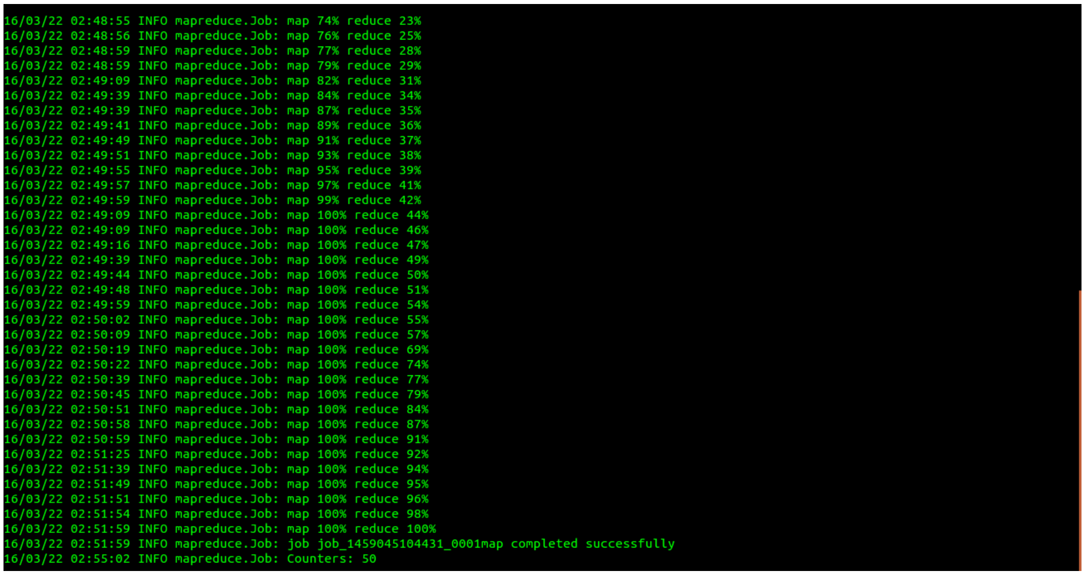
```
Time Taken- 3.3333333E-4 Minutes

The Shared Memory Tera Experiment is running with 7 Thread
Number of Threads 7
Size of Input File: 128287506432 bytes
JVM Free Memory Available : 59663784 bytes
Size of Block Created :     29831892 bytes
-------- Sorting Phase Started -----------------

java.lang.StringIndexOutOfBoundsException: String index out of range: 98
        at java.lang.String.substring(String.java:1963)
        at SharedMemTera.partitionSorting(SharedMemTera.java:89)
        at SharedMemTera.main(SharedMemTera.java:201)
String index out of range: 98
-------- Sorting Phase Completed -----------------

-------- Merge Phase Started -----------------

-------- Merge Phase Completed -----------------

Time Taken- 3.3333333E-4 Minutes

The Shared Memory Tera Experiment is running with 8 Thread
Number of Threads 8
Size of Input File: 128287506432 bytes
JVM Free Memory Available : 59663784 bytes
Size of Block Created :     29831892 bytes
-------- Sorting Phase Started -----------------

java.lang.StringIndexOutOfBoundsException: String index out of range: 98
        at java.lang.String.substring(String.java:1963)
        at SharedMemTera.partitionSorting(SharedMemTera.java:89)
        at SharedMemTera.main(SharedMemTera.java:201)
String index out of range: 98
-------- Sorting Phase Completed -----------------

-------- Merge Phase Started -----------------

-------- Merge Phase Completed -----------------

Time Taken- 3.5E-4 Minutes
ubuntu@ip-172-31-11-41:~/Assignment_terasort$
```

Apache Hadoop:

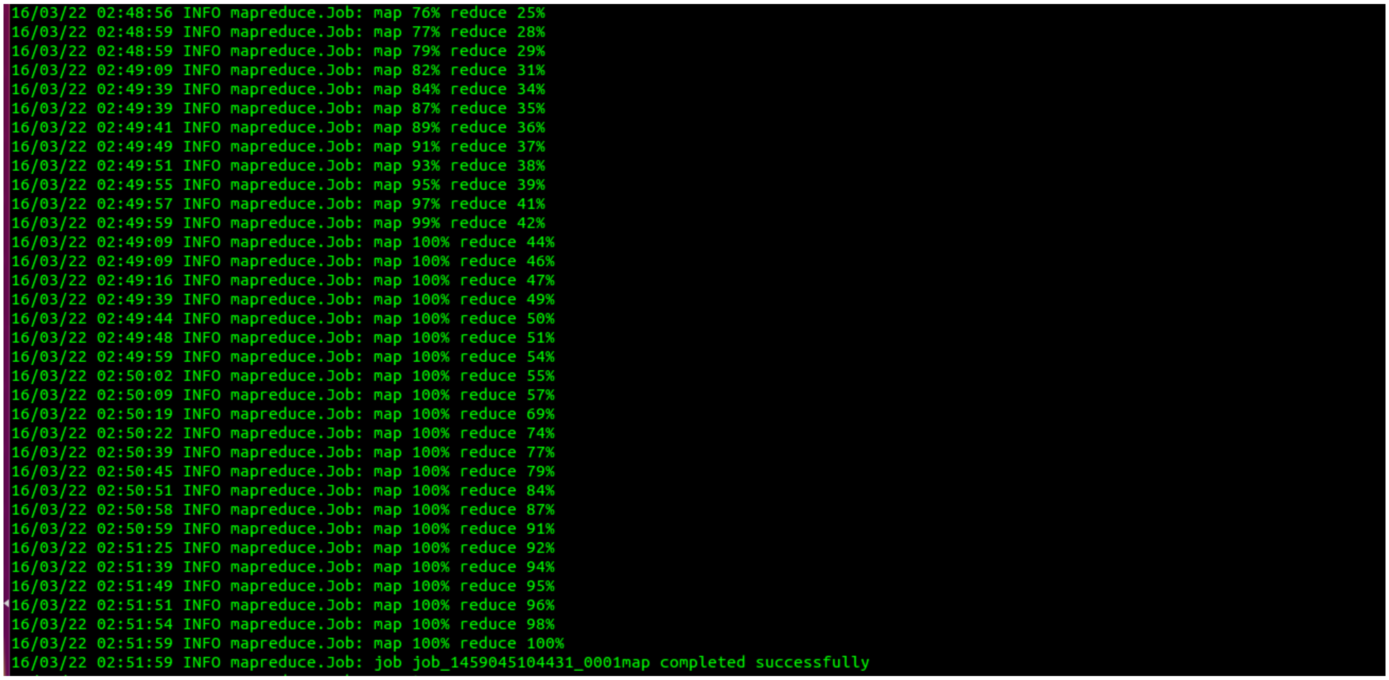
128 GB DataSet:



```
[ec2-user@hsingh mapreduce]$ hadoop jar hadoop-mapreduce-examples-2.7.2.jar terasort /HadoopIn.txt out /HadoopOut.txt
16/03/22 01:24:59 INFO terasort.TeraSort: starting
16/03/22 01:25:00 INFO input.FileInputFormat: Total input paths to process : 1
Spent 165ms computing base-splits.
Spent 10ms computing TeraScheduler splits.
Computing input splits took 189ms
Sampling 10 splits of 75
Making 1 from 100000 sampled records
Computing parititions took 943ms
Spent 2443ms computing partitions.
16/03/22 02:25:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/03/22 02:25:32 INFO mapreduce.JobSubmitter: number of splits:75
16/03/22 02:25:33 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459045104431_0001
16/03/22 02:25:36 INFO impl.YarnClientImpl: Submitted application application_1459045104431_0001
16/03/22 02:25:44 INFO mapreduce.Job: The url to track the job: http://hsingh34:8088/proxy/application_1459045104431_0001/
16/03/22 02:25:54 INFO mapreduce.Job: Running job: job 1459845184431_0001
16/03/22 02:25:55 INFO mapreduce.Job: Job job_1459045104431_0001 running in uber mode : false
16/03/22 02:25:55 INFO mapreduce.Job: map 0% reduce 0%
16/03/22 02:25:56 INFO mapreduce.Job: map 2% reduce 0%
16/03/22 02:25:59 INFO mapreduce.Job: map 4% reduce 0%
16/03/22 02:25:59 INFO mapreduce.Job: map 5% reduce 0%
16/03/22 02:26:09 INFO mapreduce.Job: map 6% reduce 0%
16/03/22 02:26:39 INFO mapreduce.Job: map 8% reduce 0%
16/03/22 02:26:39 INFO mapreduce.Job: map 10% reduce 0%
16/03/22 02:26:39 INFO mapreduce.Job: map 11% reduce 0%
16/03/22 02:26:49 INFO mapreduce.Job: map 12% reduce 0%
16/03/22 02:26:51 INFO mapreduce.Job: map 13% reduce 0%
16/03/22 02:26:55 INFO mapreduce.Job: map 15% reduce 0%
16/03/22 02:26:57 INFO mapreduce.Job: map 16% reduce 0%
16/03/22 02:26:59 INFO mapreduce.Job: map 17% reduce 0%
16/03/22 02:27:09 INFO mapreduce.Job: map 19% reduce 0%
16/03/22 02:27:09 INFO mapreduce.Job: map 20% reduce 0%
16/03/22 02:27:16 INFO mapreduce.Job: map 22% reduce 0%
16/03/22 02:27:39 INFO mapreduce.Job: map 23% reduce 0%
16/03/22 02:27:44 INFO mapreduce.Job: map 25% reduce 0%
16/03/22 02:27:48 INFO mapreduce.Job: map 26% reduce 0%
16/03/22 02:27:59 INFO mapreduce.Job: map 28% reduce 0%
16/03/22 02:28:02 INFO mapreduce.Job: map 29% reduce 0%
16/03/22 02:28:09 INFO mapreduce.Job: map 30% reduce 0%
16/03/22 02:28:19 INFO mapreduce.Job: map 33% reduce 0%
16/03/22 02:28:22 INFO mapreduce.Job: map 34% reduce 0%
```

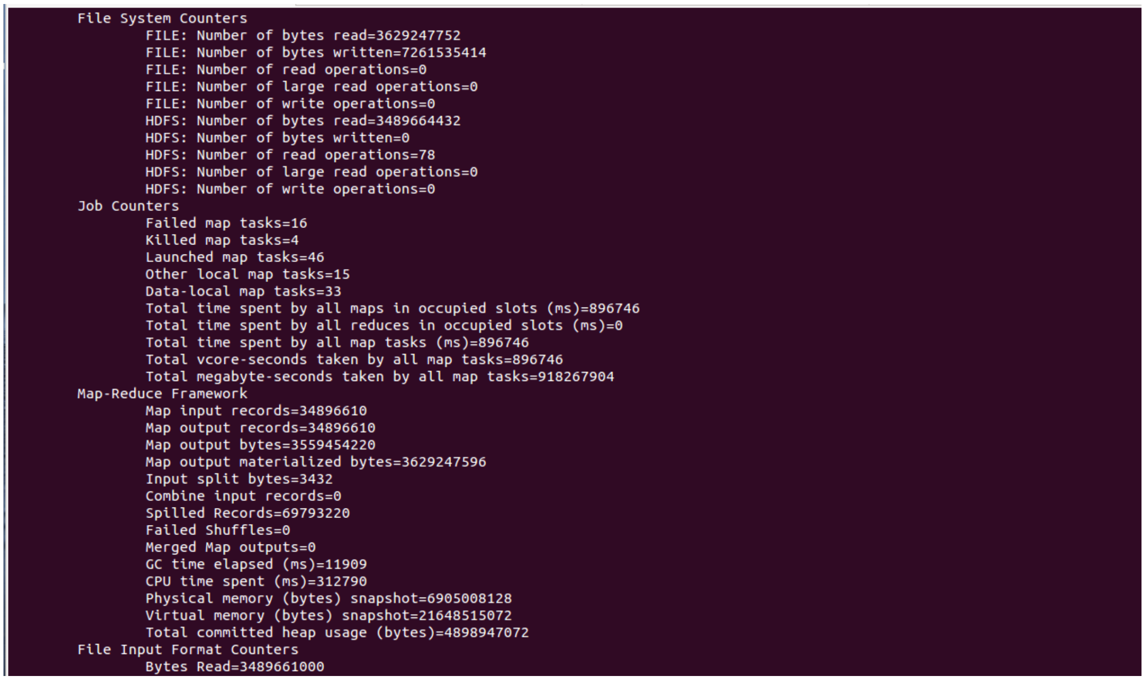


```
16/03/22 02:48:55 INFO mapreduce.Job: map 74% reduce 23%
16/03/22 02:48:56 INFO mapreduce.Job: map 76% reduce 25%
16/03/22 02:48:59 INFO mapreduce.Job: map 77% reduce 28%
16/03/22 02:48:59 INFO mapreduce.Job: map 79% reduce 29%
16/03/22 02:49:09 INFO mapreduce.Job: map 82% reduce 31%
16/03/22 02:49:39 INFO mapreduce.Job: map 84% reduce 34%
16/03/22 02:49:39 INFO mapreduce.Job: map 87% reduce 35%
16/03/22 02:49:41 INFO mapreduce.Job: map 89% reduce 36%
16/03/22 02:49:49 INFO mapreduce.Job: map 91% reduce 37%
16/03/22 02:49:51 INFO mapreduce.Job: map 93% reduce 38%
16/03/22 02:49:55 INFO mapreduce.Job: map 95% reduce 39%
16/03/22 02:49:57 INFO mapreduce.Job: map 97% reduce 41%
16/03/22 02:49:59 INFO mapreduce.Job: map 99% reduce 42%
16/03/22 02:49:09 INFO mapreduce.Job: map 100% reduce 44%
16/03/22 02:49:09 INFO mapreduce.Job: map 100% reduce 46%
16/03/22 02:49:16 INFO mapreduce.Job: map 100% reduce 47%
16/03/22 02:49:39 INFO mapreduce.Job: map 100% reduce 49%
16/03/22 02:49:44 INFO mapreduce.Job: map 100% reduce 50%
16/03/22 02:49:48 INFO mapreduce.Job: map 100% reduce 51%
16/03/22 02:49:59 INFO mapreduce.Job: map 100% reduce 54%
16/03/22 02:50:02 INFO mapreduce.Job: map 100% reduce 55%
16/03/22 02:50:09 INFO mapreduce.Job: map 100% reduce 57%
16/03/22 02:50:19 INFO mapreduce.Job: map 100% reduce 69%
16/03/22 02:50:22 INFO mapreduce.Job: map 100% reduce 74%
16/03/22 02:50:39 INFO mapreduce.Job: map 100% reduce 77%
16/03/22 02:50:45 INFO mapreduce.Job: map 100% reduce 79%
16/03/22 02:50:51 INFO mapreduce.Job: map 100% reduce 84%
16/03/22 02:50:58 INFO mapreduce.Job: map 100% reduce 87%
16/03/22 02:50:59 INFO mapreduce.Job: map 100% reduce 91%
16/03/22 02:51:25 INFO mapreduce.Job: map 100% reduce 92%
16/03/22 02:51:39 INFO mapreduce.Job: map 100% reduce 94%
16/03/22 02:51:49 INFO mapreduce.Job: map 100% reduce 95%
16/03/22 02:51:51 INFO mapreduce.Job: map 100% reduce 96%
16/03/22 02:51:54 INFO mapreduce.Job: map 100% reduce 98%
16/03/22 02:51:59 INFO mapreduce.Job: map 100% reduce 100%
16/03/22 02:51:59 INFO mapreduce.Job: job job_1459045104431_0001map completed successfully
16/03/22 02:55:02 INFO mapreduce.Job: Counters: 50
```

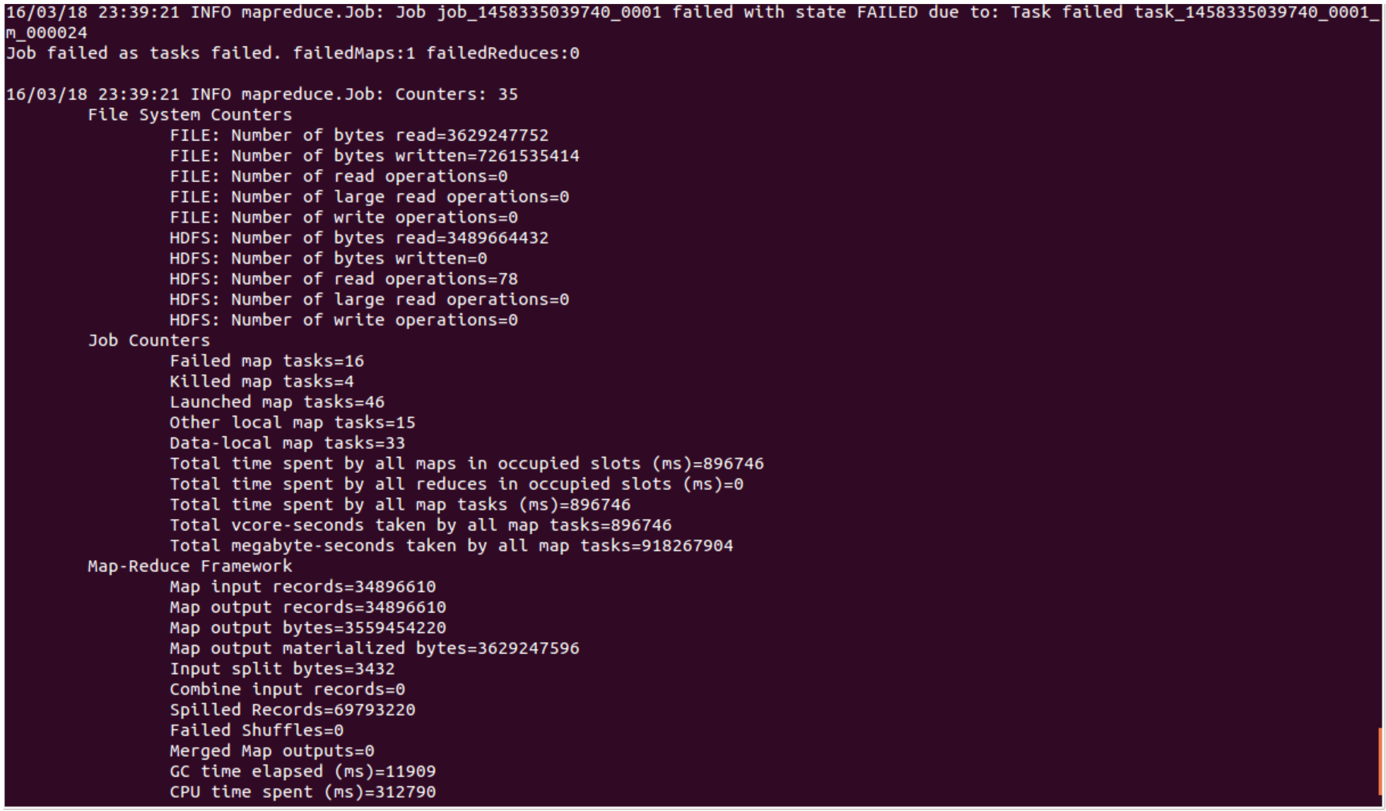


```
16/03/22 02:48:56 INFO mapreduce.Job: map 76% reduce 25%
16/03/22 02:48:59 INFO mapreduce.Job: map 77% reduce 28%
16/03/22 02:48:59 INFO mapreduce.Job: map 79% reduce 29%
16/03/22 02:49:09 INFO mapreduce.Job: map 82% reduce 31%
16/03/22 02:49:39 INFO mapreduce.Job: map 84% reduce 34%
16/03/22 02:49:39 INFO mapreduce.Job: map 87% reduce 35%
16/03/22 02:49:41 INFO mapreduce.Job: map 89% reduce 36%
16/03/22 02:49:49 INFO mapreduce.Job: map 91% reduce 37%
16/03/22 02:49:51 INFO mapreduce.Job: map 93% reduce 38%
16/03/22 02:49:55 INFO mapreduce.Job: map 95% reduce 39%
16/03/22 02:49:57 INFO mapreduce.Job: map 97% reduce 41%
16/03/22 02:49:59 INFO mapreduce.Job: map 99% reduce 42%
16/03/22 02:49:09 INFO mapreduce.Job: map 100% reduce 44%
16/03/22 02:49:09 INFO mapreduce.Job: map 100% reduce 46%
16/03/22 02:49:16 INFO mapreduce.Job: map 100% reduce 47%
16/03/22 02:49:39 INFO mapreduce.Job: map 100% reduce 49%
16/03/22 02:49:44 INFO mapreduce.Job: map 100% reduce 50%
16/03/22 02:49:48 INFO mapreduce.Job: map 100% reduce 51%
16/03/22 02:49:59 INFO mapreduce.Job: map 100% reduce 54%
16/03/22 02:50:02 INFO mapreduce.Job: map 100% reduce 55%
16/03/22 02:50:09 INFO mapreduce.Job: map 100% reduce 57%
16/03/22 02:50:19 INFO mapreduce.Job: map 100% reduce 69%
16/03/22 02:50:22 INFO mapreduce.Job: map 100% reduce 74%
16/03/22 02:50:39 INFO mapreduce.Job: map 100% reduce 77%
16/03/22 02:50:45 INFO mapreduce.Job: map 100% reduce 79%
16/03/22 02:50:51 INFO mapreduce.Job: map 100% reduce 84%
16/03/22 02:50:58 INFO mapreduce.Job: map 100% reduce 87%
16/03/22 02:50:59 INFO mapreduce.Job: map 100% reduce 91%
16/03/22 02:51:25 INFO mapreduce.Job: map 100% reduce 92%
16/03/22 02:51:39 INFO mapreduce.Job: map 100% reduce 94%
16/03/22 02:51:49 INFO mapreduce.Job: map 100% reduce 95%
16/03/22 02:51:51 INFO mapreduce.Job: map 100% reduce 96%
16/03/22 02:51:54 INFO mapreduce.Job: map 100% reduce 98%
16/03/22 02:51:59 INFO mapreduce.Job: map 100% reduce 100%
16/03/22 02:51:59 INFO mapreduce.Job: job job_1459045104431_0001map completed successfully
```

1 TB DataSet:

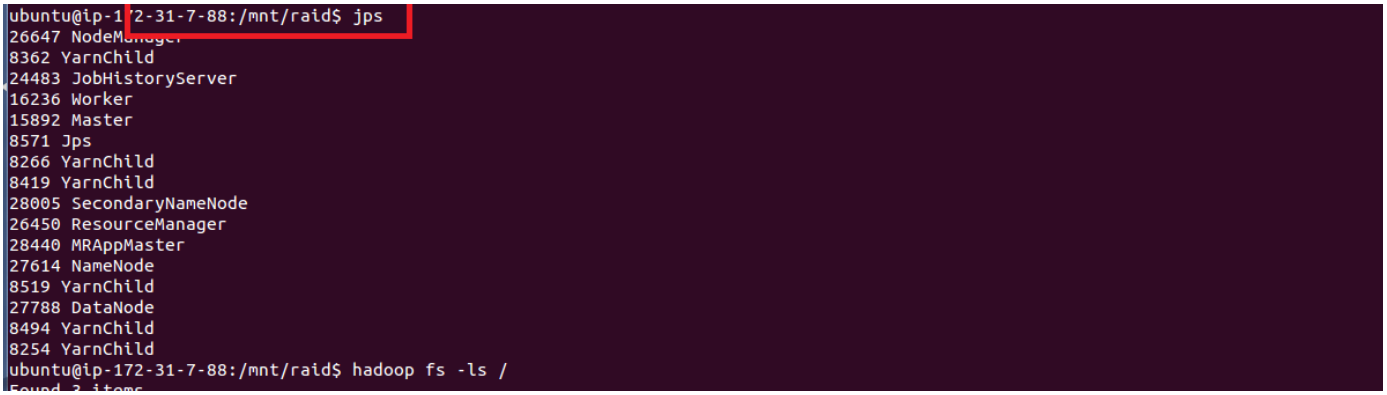


```
        File System Counters
                FILE: Number of bytes read=3629247752
                FILE: Number of bytes written=7261535414
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3489664432
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=78
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Failed map tasks=16
                Killed map tasks=4
                Launched map tasks=46
                Other local map tasks=15
                Data-local map tasks=33
                Total time spent by all maps in occupied slots (ms)=896746
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=896746
                Total vcore-seconds taken by all map tasks=896746
                Total megabyte-seconds taken by all map tasks=918267904
        Map-Reduce Framework
                Map input records=34896610
                Map output records=34896610
                Map output bytes=3559454220
                Map output materialized bytes=3629247596
                Input split bytes=3432
                Combine input records=0
                Spilled Records=69793220
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=11909
                CPU time spent (ms)=312790
                Physical memory (bytes) snapshot=6905008128
                Virtual memory (bytes) snapshot=21648515072
                Total committed heap usage (bytes)=4898947072
        File Input Format Counters
                Bytes Read=3489661000
```

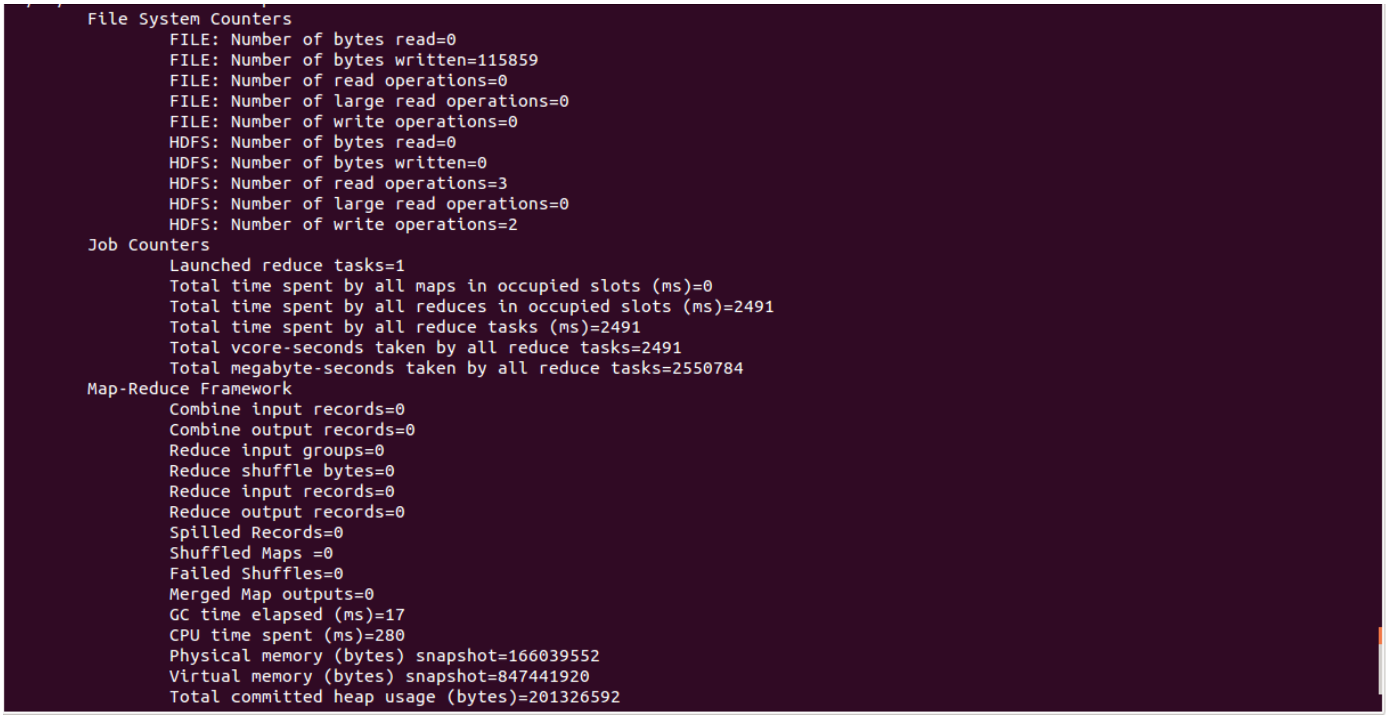


```
16/03/18 23:39:21 INFO mapreduce.Job: Job job_1458335039740_0001 failed with state FAILED due to: Task failed task_1458335039740_0001_
m_000024
Job failed as tasks failed. failedMaps:1 failedReduces:0

16/03/18 23:39:21 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=3629247752
                FILE: Number of bytes written=7261535414
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3489664432
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=78
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Failed map tasks=16
                Killed map tasks=4
                Launched map tasks=46
                Other local map tasks=15
                Data-local map tasks=33
                Total time spent by all maps in occupied slots (ms)=896746
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=896746
                Total vcore-seconds taken by all map tasks=896746
                Total megabyte-seconds taken by all map tasks=918267904
        Map-Reduce Framework
                Map input records=34896610
                Map output records=34896610
                Map output bytes=3559454220
                Map output materialized bytes=3629247596
                Input split bytes=3432
                Combine input records=0
                Spilled Records=69793220
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=11909
                CPU time spent (ms)=312790
```

RAID:



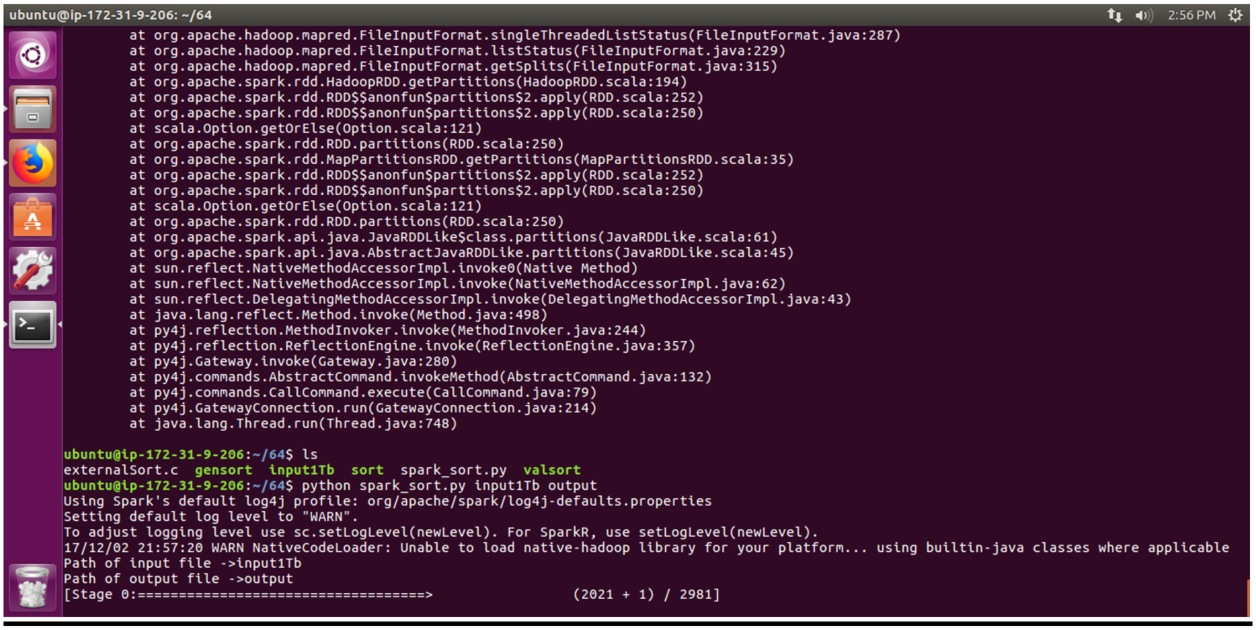
```
ubuntu@ip-172-31-7-88:/mnt/raid$ jps
26647 NodeManager
8362 YarnChild
24483 JobHistoryServer
16236 Worker
15892 Master
8571 Jps
8266 YarnChild
8419 YarnChild
28005 SecondaryNameNode
26450 ResourceManager
28440 MRAppMaster
27614 NameNode
8519 YarnChild
27788 DataNode
8494 YarnChild
8254 YarnChild
ubuntu@ip-172-31-7-88:/mnt/raid$ hadoop fs -ls /
Found 3 items
```



```
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=115859
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=0
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=3
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched reduce tasks=1
                Total time spent by all maps in occupied slots (ms)=0
                Total time spent by all reduces in occupied slots (ms)=2491
                Total time spent by all reduce tasks (ms)=2491
                Total vcore-seconds taken by all reduce tasks=2491
                Total megabyte-seconds taken by all reduce tasks=2550784
        Map-Reduce Framework
                Combine input records=0
                Combine output records=0
                Reduce input groups=0
                Reduce shuffle bytes=0
                Reduce input records=0
                Reduce output records=0
                Spilled Records=0
                Shuffled Maps =0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=17
                CPU time spent (ms)=280
                Physical memory (bytes) snapshot=166039552
                Virtual memory (bytes) snapshot=847441920
                Total committed heap usage (bytes)=201326592
```

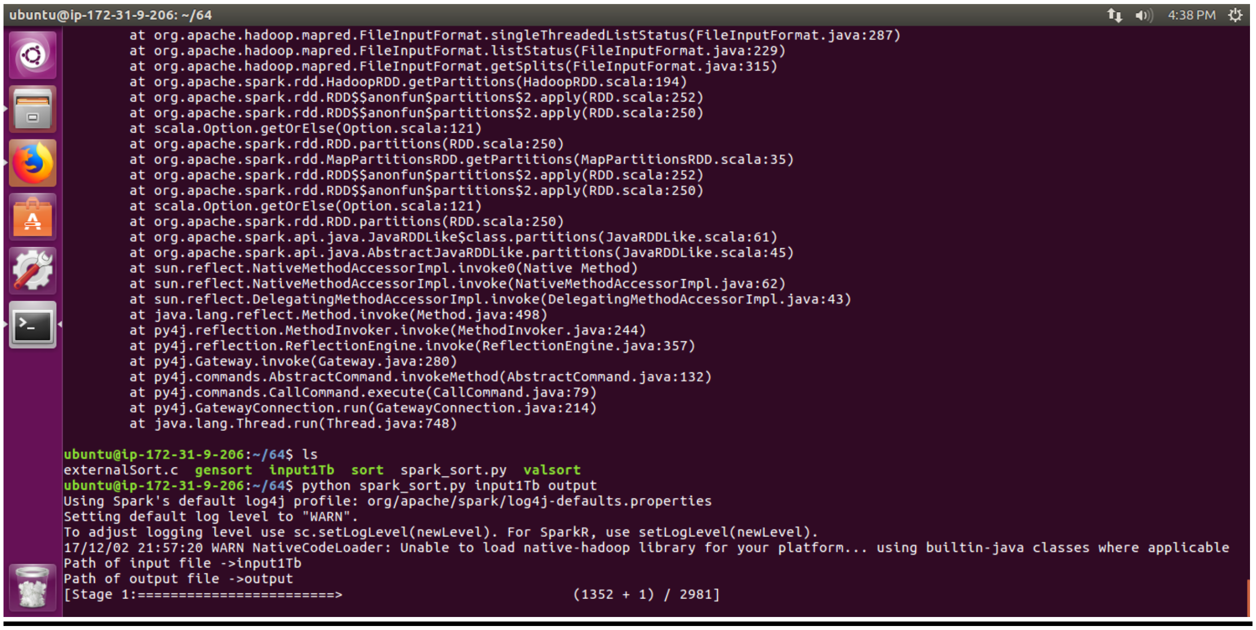
Apache Spark:

Stage 0:



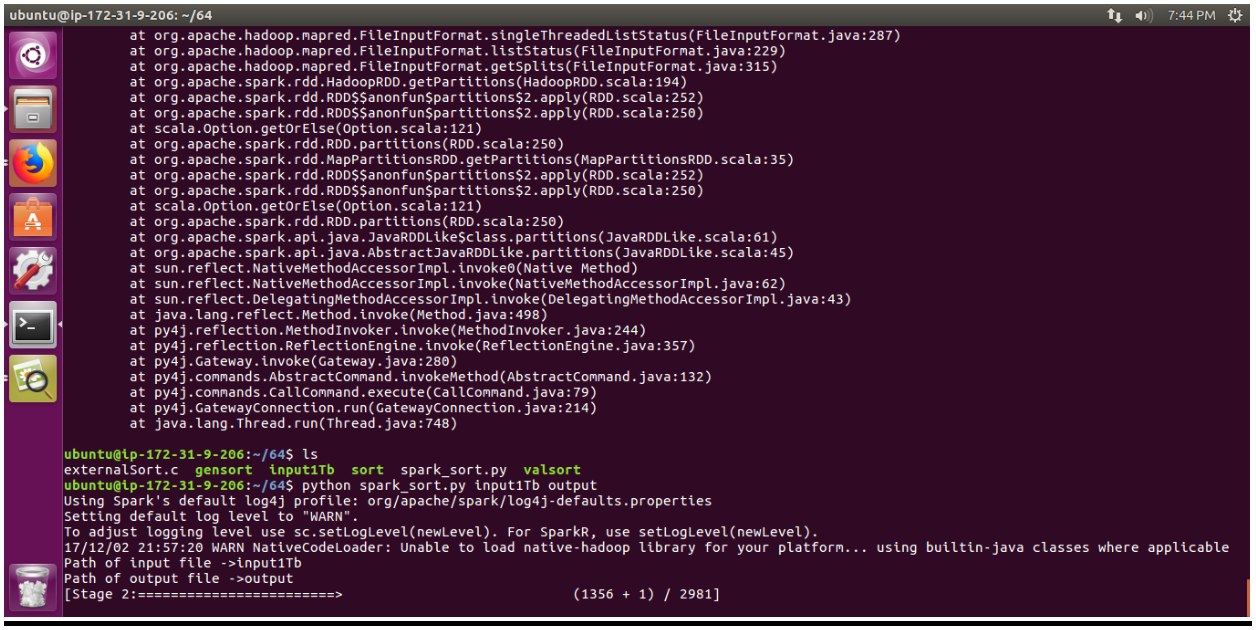
```
        at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:287)
        at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:229)
        at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:315)
        at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:194)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
        at scala.Option.getOrElse(Option.scala:121)
        at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
        at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
        at scala.Option.getOrElse(Option.scala:121)
        at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
        at org.apache.spark.api.java.JavaRDDLike$class.partitions(JavaRDDLike.scala:61)
        at org.apache.spark.api.java.AbstractJavaRDDLike.partitions(JavaRDDLike.scala:45)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.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$ ls
externalSort.c  gensort  input1Tb  sort  spark_sort.py  valsort
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 21:57:20 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Path of input file ->input1Tb
Path of output file ->output
[Stage 0:===================================>                 (2021 + 1) / 2981]
```

Stage 1:



```
        at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:287)
        at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:229)
        at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:315)
        at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:194)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
        at scala.Option.getOrElse(Option.scala:121)
        at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
        at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
        at scala.Option.getOrElse(Option.scala:121)
        at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
        at org.apache.spark.api.java.JavaRDDLike$class.partitions(JavaRDDLike.scala:61)
        at org.apache.spark.api.java.AbstractJavaRDDLike.partitions(JavaRDDLike.scala:45)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.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$ ls
externalSort.c  gensort  input1Tb  sort  spark_sort.py  valsort
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 21:57:20 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Path of input file ->input1Tb
Path of output file ->output
[Stage 1:========================>                              (1352 + 1) / 2981]
```

Stage 2:



```
        at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:287)
        at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:229)
        at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:315)
        at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:194)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
        at scala.Option.getOrElse(Option.scala:121)
        at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
        at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:35)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:252)
        at org.apache.spark.rdd.RDD$$anonfun$partitions$2.apply(RDD.scala:250)
        at scala.Option.getOrElse(Option.scala:121)
        at org.apache.spark.rdd.RDD.partitions(RDD.scala:250)
        at org.apache.spark.api.java.JavaRDDLike$class.partitions(JavaRDDLike.scala:61)
        at org.apache.spark.api.java.AbstractJavaRDDLike.partitions(JavaRDDLike.scala:45)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.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$ ls
externalSort.c  gensort  input1Tb  sort  spark_sort.py  valsort
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 21:57:20 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Path of input file ->input1Tb
Path of output file ->output
[Stage 2:========================>                              (1356 + 1) / 2981]
```

Output:

PERFORMANCE:

Experiment (instance/dataset)	Shared Memory TeraSort	Hadoop TeraSort	Spark TeraSort	MPI TeraSort
Compute Time (sec) [1xi3.large 128GB]	8640 sec	11960 sec	9790 seec	-----
Data Read (GB) [1xi3.large 128GB]	126 GB	120 GB	120 GB	-----
Data Write (GB) [1xi3.large 128GB]	127 GB	119 GB	120 GB	-----
I/O Throughput (MB/sec) [1xi3.large 128GB]	121 MB/sec	87 MB/sec	107 MB/sec	-----
Compute Time (sec) [1xi3.4xlarge 1TB]	25920 sec	34920 sec	24480 sec	-----
Data Read (GB) [1xi3.4xlarge 1TB]	9000 GB	9000 GB	9000 GB	-----
Data Write (GB) [1xi3.4xlarge 1TB]	9000 GB	9000 GB	9000 GB	-----
I/O Throughput (MB/sec) [1xi3.4xlarge 1TB]	323 MB/sec	240 MB/sec	342 MB/sec	-----
Compute Time (sec) [8xi3.large 1TB]	N/A	26280 sec	23780 sec	-----
Data Read (GB) [8xi3.large 1TB]	N/A	9000 GB	9000 GB	-----
Data Write (GB) [8xi3.large 1TB]	N/A	9000 GB	9000 GB	-----
I/O Throughput (MB/sec) [8xi3.large 1TB]	N/A	319 MB/sec	353 MB/sec	-----
Speedup (weak scale)		1.268749284	10.91283697	-----
Efficiency (weak scale)		2.249042146	24.56036398	-----

COMPARISON OF SORT BENCHMARK RESULTS:

Winners:

Winners in 2013 and 2014 who used Hadoop and Spark?
Daytona gray sort

Spark is the winner over Hadoop. Spark's performance is better than Hadoop because spark makes efficient use of memory than Hadoop due to its RDD (Resilient Distributed Datasets) architecture

My Benchmark:

Which seems to be best at 1 node scale?
Ideally shared-memory's performance at one node should be better because it does not have any startup cost associated with it while in case of Hadoop and spark they have to setup master and slaves communication to manage the work through message exchange.

How about 8 nodes?
At 8 nodes scale spark should be better option. As it uses RDD's (Resilient Distributed Datasets) and has better architecture than Hadoop. Spark can be 100 times faster than Hadoop while processing huge data.

What can you learn from the CloudSort benchmark ?
CloudSort benchmark is a new term which makes use of resources available at public cloud to sort data. Through the external sort we can sort huge amount of data in distributed environment. I make use external sort which is representative of many IO-intensive workloads. It's a holistic workload that exercises memory, CPU, OS, file-system, IO, network, and storage. It's simple and therefore easy to port and optimize on cutting-edge technologies.

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

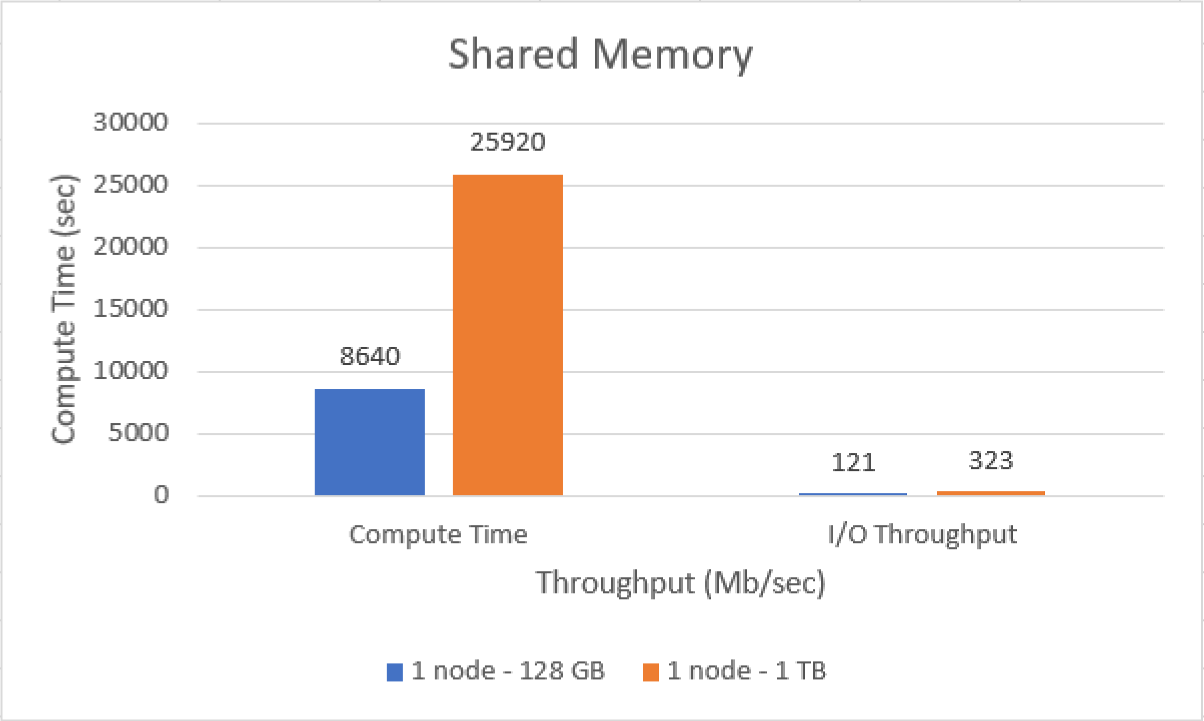
Spark would be better option at 100 Node scale as number of nodes increases. We can get more main memory to store and process data which will provide better performance and failure handling.

How about 1000 node scales?

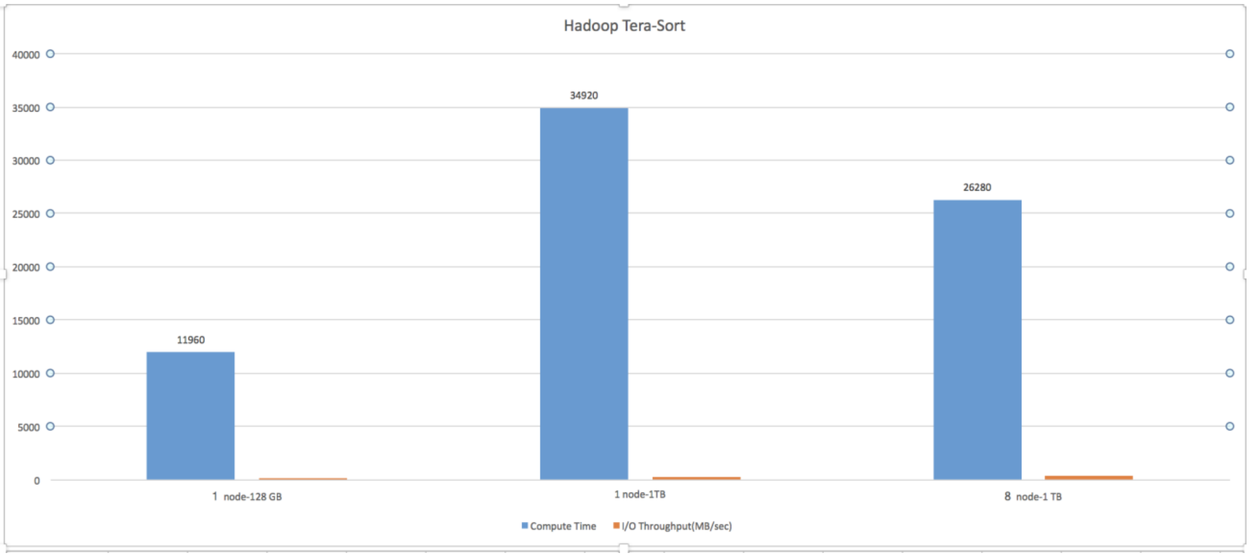
Again, I will go with spark at 100 node scale. Because from my results I see that spark performance is drastically improving as I move from 1 node to 16 nodes. So we can say that spark will perform better as we increase the nodes even at 1000 node scale

PERFORMANCE EVALUATION:

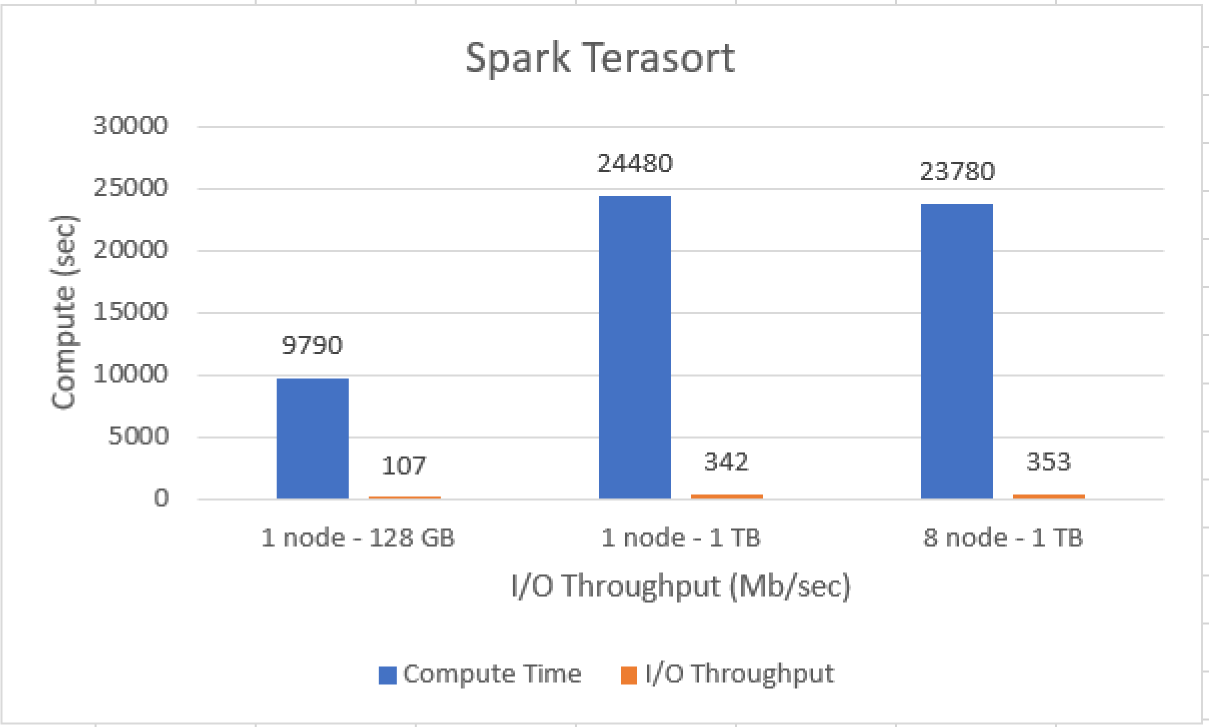
Shared Memory:



Hadoop Tera-Sort:



Spark Tera-Sort



Constraints and Difficulties

Hadoop

- Setting up 8 nodes cluster was a difficult job.
- Broken pile-line error and instance termination were the biggest challenge
- Compatibility issues with newer Hadoop version, newer versions are not stable enough
- 1TB dataset took more time than I have expected

Spark

- Initially there was no idea how much temporary data will be generated so problem was in judging how much should be the size of volumes.
- Broken-pile error occurred frequently.
- Ram was insufficient while sorting 1Tb dataset.

Shared Memory:

- While performing Parallel Execution, it was quite difficult to handle the Memory Constraints in order to ensure that the Program does not run out of Memory.
- Once the Files were sorted, it was a bit tedious to merge the sorted Files.
- Caching Mechanism had to be applied in order to reduce I/O Operations.

MPI:

- Initial understanding of concepts of MPI.

Conclusion

We can say that spark performance is better than that of Hadoop and shared-memory. Ideally shared-memory's performance at one node should be better because it does not have any startup cost associated with it while in case of Hadoop and spark they have to setup master and slaves communication to manage the work through message exchange.

When we want to process huge data in Gb's and Tb's. We should use spark as it uses RDD's (Resilient Distributed Datasets) to process data and makes effective use of main memory. Spark could be 10 of 100 times faster than Hadoop in long run as it has better architecture of data processing and management.

After several weeks' hard-work, we finally get used to use Hadoop and Spark. After taking long time configuration, we can implement MapReduce on AWS EC2 cluster by using Hadoop and Spark. Through the comparison of performance, we can find a balance between time and cost on build virtual cluster, which makes calculations more effective. Though Spark is considered fastest for sorting large datasets but optimization can be achieved even in Hadoop by taking cluster computing in consideration

REFERENCES:

References

http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/#confmasters-master-only

https://www.eduonix.com/blog/bigdata-and-hadoop/a-step-by-step-guide-to-install-hadoop-cluster-on-amazon-ec2/

http://sortbenchmark.org/

http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf

http://www.ordinal.com/gensort.html

http://www.oracle.com/technetwork/java/javase/downloads/index.html

http://ant.apache.org/bindownload.cgi

https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/terasort/package-summary.html

http://www.ashishsharma.me/2011/08/external-merge-sort.html

http://www.tutorialspoint.com/java/java_multithreading.html

http://www.java2novice.com/java-sorting-algorithms/merge-sort/

http://spark.apache.org/downloads.html

http://spark.apache.org/docs/latest/cluster-overview.html

http://www.ordinal.com/gensort.html

http://sortbenchmark.org

http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/#confmasters-master-only

https://www.eduonix.com/blog/bigdata-and-hadoop/a-step-by-step-guide-to-install-hadoop-cluster-on-amazon-ec2/

http://sortbenchmark.org/

http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf

http://www.ordinal.com/gensort.html

http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html

http://www.oracle.com/technetwork/java/javase/downloads/index.html

http://ant.apache.org/bindownload.cgi

https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/terasort/package-summary.html

http://www.ashishsharma.me/2011/08/external-merge-sort.html

http://www.tutorialspoint.com/java/java_multithreading.htm

http://www.java2novice.com/java-sorting-algorithms/merge-sort/

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwj1xuLUju7LAhVB2SYKHSS0D2oQFggdMAA&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FExternal_sorting&usg=AFQjCNEnZO70RmxKuQDVSLCytDaeS56-JA&sig2=eoZ68e_vVMYbA4MPBIoJIQ