

Assignment 2

Terasort on Amazon AWS

Team Members

Niket Patel (A20384264) npatel102@hawk.iit.edu

Purvank Patel (A20380792) ppatel104@hawk.iit.edu

Parth Hindia (A20384443) phindia@hawk.iit.edu

Introduction

Our aim for this assignment is to check performance of Terasort program using various approach such as Shared Memory External sort, Hadoop Terasort and Spark Terasort. We have generated 128GB and 1TB random datasets using gensort program and perform all the those three approach on amazon AWS i3.large, i3.4xlarge and 8 node i3.large cluster.

To perform the experiment we have used following environment setup on amazon AWS.

Operation System	Ubuntu 16.04
Java Version	OpenJDK-1.8.0
Hadoop Version	Hadoop-2.7.1
Spark Version	Spark-2.2.0

Configuration: 1

Virtual Cluster (1-Node, i3.large)

AWS I3.large configuration:

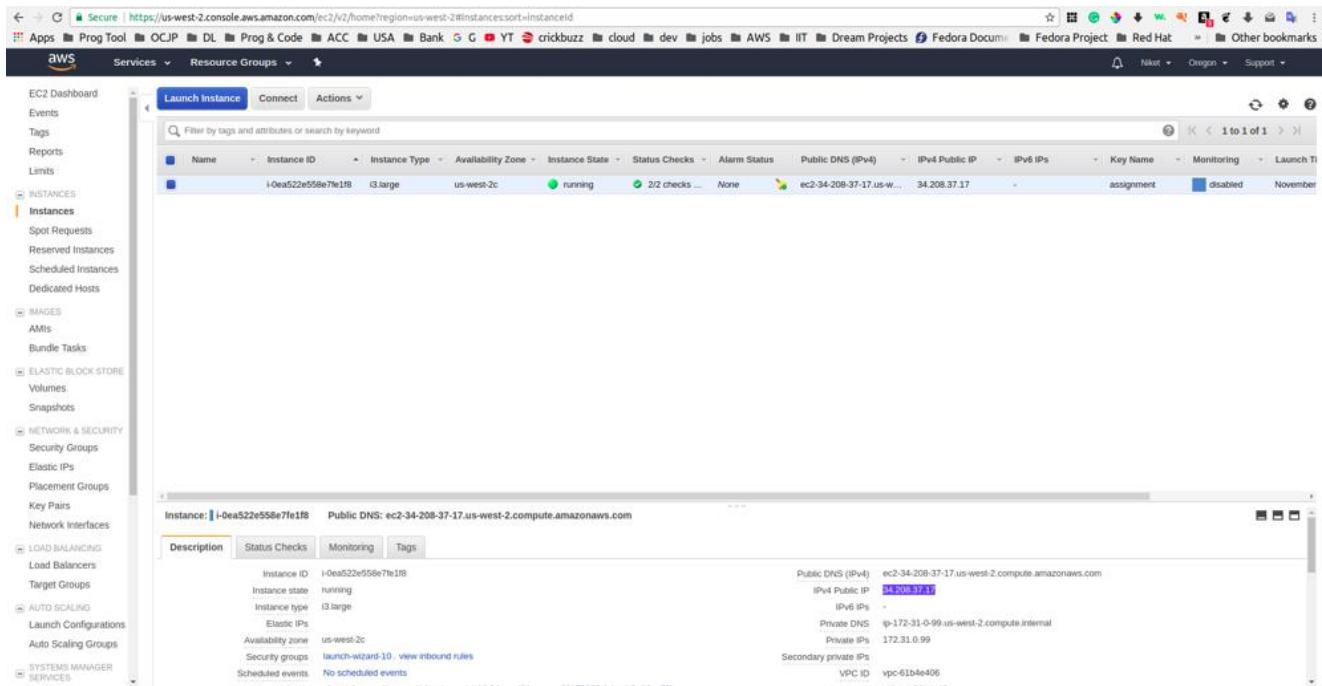
Instance Name	i3.large
vCPU count	2
Memory	15.25 GiB
Instance Storage (NVMe SSD)	0.475 TB
Price/Hour	\$0.15
Operating System	Ubuntu 16.04.3 LTS

To set up the environment and run the program follow the following steps respectively. We also can use automation script to simplify the test.

a. Shared Memory Terasort:

To perform shared Memory Terasort for 128 GB on i3.large instance, we have used Merge Sort as Main Sorting algorithm. To perform External Sort follow following steps.

1. Create i3.large instance on Amazon AWS.



The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with various services like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, AMIs, and more. The main area shows a table of instances. One instance is listed: Name: i-0ea522e558e7fe1f8, Instance ID: i-0ea522e558e7fe1f8, Instance Type: i3.large, Availability Zone: us-west-2c, Instance State: running, Status Checks: 2/2 checks, Alarm Status: None, Public DNS (IPv4): ec2-34-208-37-17.us-west-2.compute.amazonaws.com, IPv4 Public IP: 34.208.37.17, Key Name: assignment, Monitoring: disabled, Launch Time: November. Below the table, there's a detailed view of the selected instance (i-0ea522e558e7fe1f8) with tabs for Description, Status Checks, Monitoring, and Tags. The Description tab shows details like Instance ID, Instance state, Instance type, Elastic IPs, Availability zone, Security groups, and Scheduled events. The Status Checks tab shows 2/2 checks. The Monitoring tab shows monitoring is disabled. The Tags tab shows no tags assigned.

2. Upload ExternalSort.java file to the instance using following command.

```
$ scp -i assignment.pem ExternalSort.java ubuntu@ec2-52-40-13-239.us-west-2.compute.amazonaws.com
```

3. Generate 128GB file using following command.

```
$ cd /data
```

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

```
$ tar xvzf gensort-linux-1.5.tar.gz
$ cd 64/
$ ./gensort -a -t8 1374389534 sample128gb.txt
```

4. Now also create two directory in the same directory. Stmp and tmp directory using following command.

```
$ mkdir tmp
$ mkdir stmp
```

5. Now run the program using following command as a background service so that we can avoid any undesirable condition such as connection lost or connection broken error. It will also save result to nohup.out file.

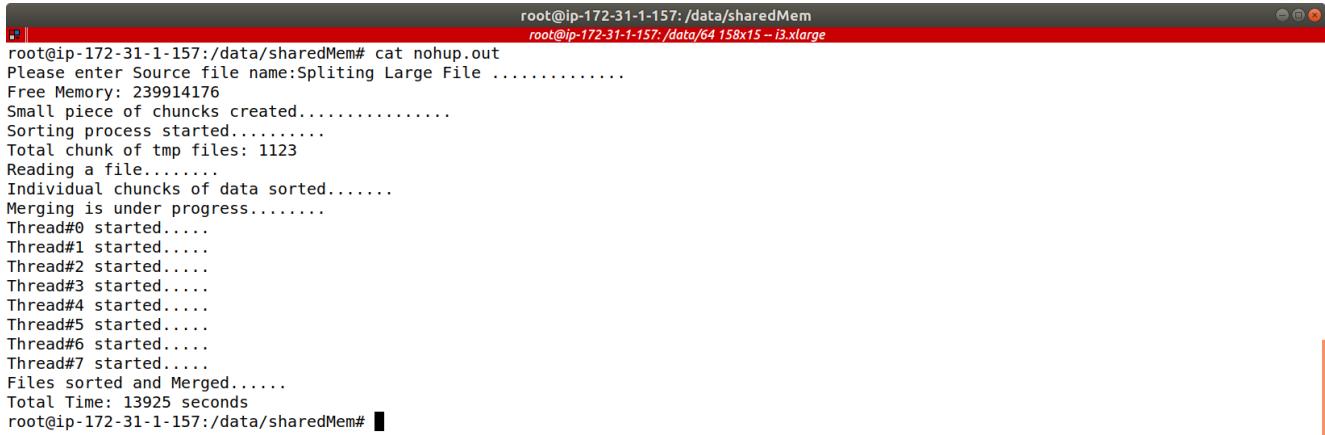
```
$ nohup java ExternalSort sample128gb.txt &
```

```
root@ip-172-31-1-157:/data/sharedMem
top - 19:01:15 up 4:51, 1 user, load average: 7.98, 7.96, 7.98
Tasks: 126 total, 1 running, 125 sleeping, 0 stopped, 0 zombie
%CPU(s): 68.8 us, 5.2 sy, 0.0 ni, 19.3 id, 6.7 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 15657368 total, 164764 free, 3246264 used, 12246340 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 12040452 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1955 root 20 0 6437884 3.003g 3276 S 193.8 20.1 419:51.00 java
 1 root 20 0 37700 3904 2140 S 0.0 0.0 0:03.03 systemd
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
 3 root 20 0 0 0 0 S 0.0 0.0 0:00.34 ksoftirqd/0
 5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H
 7 root 20 0 0 0 0 S 0.0 0.0 0:01.45 rcu_sched
 8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
 9 root rt 0 0 0 0 S 0.0 0.0 0:00.03 migration/0
10 root rt 0 0 0 0 S 0.0 0.0 0:00.06 watchdog/0
11 root rt 0 0 0 0 S 0.0 0.0 0:00.04 watchdog/1
12 root rt 0 0 0 0 S 0.0 0.0 0:00.03 migration/1
```

```
root@ip-172-31-1-157:/data/64# ls
gensort valsort
root@ip-172-31-1-157:/data/64# ./valsort /data/sharedMem/output.txt
Records: 1374388412
Checksum: 28f5eb798d9baaf6
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-1-157:/data/64#
```

Output generated by background service. nohup.out



```
root@ip-172-31-1-157:/data/sharedMem# cat nohup.out
Please enter Source file name:Spliting Large File .....
Free Memory: 239914176
Small piece of chuncks created.....
Sorting process started.....
Total chunk of tmp files: 1123
Reading a file.....
Individual chuncks of data sorted.....
Merging is under progress.....
Thread#0 started.....
Thread#1 started.....
Thread#2 started.....
Thread#3 started.....
Thread#4 started.....
Thread#5 started.....
Thread#6 started.....
Thread#7 started.....
Files sorted and Merged.....
Total Time: 13925 seconds
root@ip-172-31-1-157:/data/sharedMem#
```

Result to sort 128GB data using External Sort.

Data size	128 GB = 137438953472 Bytes
No. of Bytes Read (FILE)	692563476480
No. of Bytes Write (FILE)	277025390592
Time Taken by whole process	13925 seconds
Throughput	9.947 MB/s

Explanation:

Here the bottleneck is the merging the file as it need to load data from several sorted temporary file. To solve this problem and reduce the time to open and seek file pointer, we have used Priority Queue which will refer to the File Pointer and read the data from the file line by line and compare with other file.

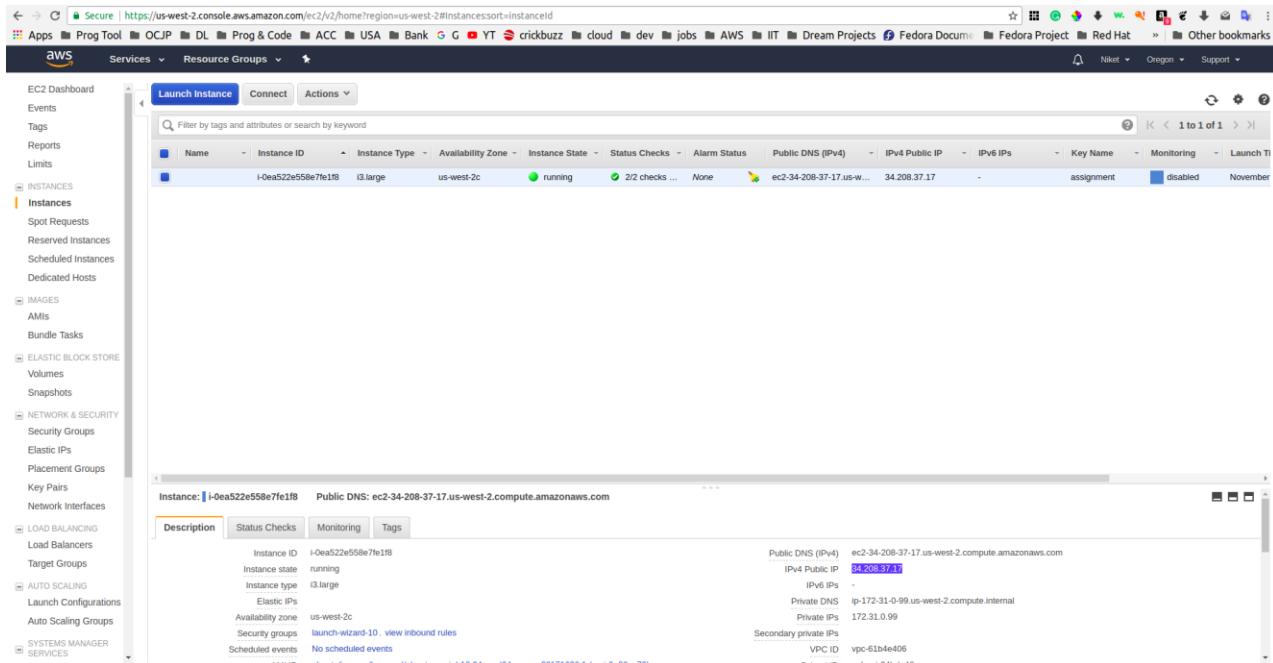
Difficulties:

As the connection with aws instance through terminal, several times my connection broken and my all process lost in between. To resolve this issue, I tried to run it as background service and used nohub command. This command will generate one nohup.out file which is the output of program which we used as logs.

b. Hadoop Terasort:

To perform the terasort hadoop program for 128GB on i3.large instance, follow the following steps.

1. Create i3.large instance on Amazon AWS.



The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various services like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, AMIs, and more. The main area shows a table of instances. One instance is selected, showing its details in a modal window. The instance ID is i-0ea522e558e7fe1f8, it's an i3.large type in us-west-2c, and it's currently running. The Public DNS is ec2-34-208-37-17.us-west-2.compute.amazonaws.com, and the Public IP is 34.208.37.1. The modal also displays other details like Security Groups, Network Interfaces, and VPC settings.

2. Mount the 0.475 SSD with our instance using following command.

```
$ sudo su  
$ sudo mkfs -t ext4 /dev/nvme0n1  
$ mkdir /data  
$ mount /dev/nvme0n1 /data  
$ chmod 777 /data
```

3. Install Hadoop from apache hadoop official website using following commands in /data directory where we have mounted our SSD.

We have used hadoop-2.7.4 version for our assignment.

```
$ cd /data  
$ wget https://archive.apache.org/dist/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz
```

```
$ tar xvzf hadoop-2.7.1.tar.gz  
$ rm -rf hadoop-2.7.1.tar.gz  
$ mv hadoop-2.7.1/ hadoop
```

4. After downloading Hadoop, install OpenJdk-8 and SSH using following command.

```
$ sudo apt-get update  
$ sudo apt-get install openjdk-8*  
$ sudo apt-get install ssh
```

5. Now go to /root directory and set the JAVA_HOME and HADOOP_HOME environment variable to .bashrc file.

```
$ sudo su  
$ cd /root  
$ vi .bashrc
```

And add the following lines at the end of the file.

```
export JAVA_HOME="/usr/lib/jvm/java-1.8.0-openjdk-amd64"  
export PATH="$PATH:$JAVA_HOME/bin"  
export HADOOP_HOME="/data/hadoop"  
export PATH="$PATH:$HADOOP_HOME/bin"  
export PATH="$PATH:$HADOOP_HOME/sbin"  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
export HADOOP_MAPRED_HOME=$HADOOP_HOME  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
export YARN_HOME=$HADOOP_HOME
```

```
export HADOOP_INSTALL=$HADOOP_HOME  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib/native"  
export HADOOP_CLASSPATH=/usr/lib/jvm/java-8-openjdk-amd64/lib/tools.jar
```

After updating .bashrc file, update the changes to the system using following command.

```
$ source .bashrc
```

6. Now create ssh free login to localhost using following command in the /root directory.

```
$ ssh-keygen -t rsa -P ""  
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys  
$ chmod 0600 ~/.ssh/authorized_keys
```

7. Now download TeraGen to /data directory and generate 128 GB file using following command.

```
$ cd /data  
$ wget http://www.ordinal.com/try.cgi/gensort-linux-1.5.tar.gz  
$ tar xvzf gensort-linux-1.5.tar.gz  
$ cd 64/  
$ ./gensort -a -t8 1374389534 sample128gb.txt
```

8. After completing above steps, logout from root and create hdfs directory and hdfs/namenode, hdfs/datanode directory in /data folder.

```
$ cd /data  
$ mkdir hdfs/  
$ mkdir hdfs/namenode  
$ mkdir hdfs/datanode
```

9. Now, configure hadoop environment by changing default property to following files.

a. `hadoop-env.sh`

Set `JAVA_HOME` variable in `hadoop-env.sh`

```
export JAVA_HOME="/usr/lib/jvm/java-1.8.0-openjdk-amd64"
```

b. `Core-site.xml`

Add following lines at the end of the file.

```
<configuration>  
  <property>  
    <name>fs.default.name</name>  
    <value>hdfs://localhost:9000</value>  
  </property>  
  <property>  
    <name>hadoop.tmp.dir</name>  
    <value>/data/tmp</value>  
  </property>  
</configuration>
```

c. `Hdfs-site.xml`

Add following lines at the end of the file.

```
<configuration>  
  <property>  
    <name>dfs.replication</name>  
    <value>1</value>  
  </property>  
  <property>  
    <name>dfs.namenode.name.dir</name>
```

```
<value>file:/data/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/data/hdfs/datanode</value>
</property>
<property>
  <name>dfs.blocksize</name>
  <value>1073741824</value>
</property>
</configuration>
```

d. Mapred-site.xml

In order to edit this file, use following command first.

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

Add following lines at the end of the file.

```
<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
<property>
  <name>mapreduce.job.reduces</name>
  <value>2</value>
</property>
</configuration>
```

e. Yarn-site.xml

Add following lines at the end of the file.

```
<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

10. Now format the namenode using following command.

```
$ hdfs namenode -format
```

11. Now run the dfs and yarn services using following command.

```
$ start-dfs.sh
```

```
$ start-yarn.sh
```

12. To check service running use following command.

```
$ jps
```

13. After completing and running the hadoop, upload the 'sample128gb.txt' file to the hadoop by setting 1024MB block size. By setting block size 1024MB will reduce I/O latency. Use following command to upload the file.

```
$ ssh localhost
```

```
$ cd /data
```

```
$ cd 64/
```

```
$ hdfs dfs -mkdir /input
```

```
$ hadoop fs -D dfs.block.size=536870912 -put sample128gb.txt /input
```

```

root@ip-172-31-0-99: /data/data
File Edit View Search Terminal Help
root@ip-172-31-0-99:~# ls
data hadoop spark
root@ip-172-31-0-99:~# ls
data hadoop spark
root@ip-172-31-0-99:~# cd data/
root@ip-172-31-0-99:~/data# ls
gensort valsrt
root@ip-172-31-0-99:~/data# cd ..
root@ip-172-31-0-99:~# ls
data hadoop spark
root@ip-172-31-0-99:~# cp -r data/ /data/
root@ip-172-31-0-99:~# cd /data/
root@ip-172-31-0-99:/data# ls
data hadoop lost+found spark
root@ip-172-31-0-99:/data# cd data/
root@ip-172-31-0-99:/data/data# ls
gensort valsrt
root@ip-172-31-0-99:/data/data# ./gensort -a -t4 1374389535 sample128gb.txt
root@ip-172-31-0-99:/data/data# ls -lh
total 129G
-rwxr-xr-x 1 root root 138K Nov 25 17:42 gensort
-rwxr-xr-x 1 root root 129G Nov 25 17:55 sample128gb.txt
-rwxr-xr-x 1 root root 132K Nov 25 17:42 valsrt
root@ip-172-31-0-99:/data/data#

```

14. To check number of blocks in the file, use the following command.

```
$ hdfs fsck /input/sample128gb.txt -blocks
```

```

root@ip-172-31-1-157: /data/data/64
File Edit View Search Terminal Help
root@ip-172-31-1-157:/data/data/64# hdfs fsck /input/sample128gb.txt -blocks
Connecting to namenode via http://localhost:50070/fsck?ugi=root&blocks=1&path=%2Finput%2Fsample128gb.txt
FSCK started by root (auth:SIMPLE) from /127.0.0.1 for path /input/sample128gb.txt at Sat Nov 25 21:27:39 UTC 2017
.Status: HEALTHY
Total size: 137438953500 B
Total dirs: 0
Total files: 1
Total symlinks: 0
Total blocks (validated): 129 (avg. block size 1065418244 B)
Minimally replicated blocks: 129 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Sat Nov 25 21:27:39 UTC 2017 in 7 milliseconds

The filesystem under path '/input/sample128gb.txt' is HEALTHY
root@ip-172-31-1-157:/data/data/64#

```

We can also check the result in web browser using following link.

```
http://<public_id>:50070/fsck?ugi=root&blocks=1&path=%2Finput%2Fsample128gb.txt
```

```
.Status: HEALTHY
Total size: 137438953500 B
Total dirs: 0
Total files: 1
Total symlinks: 0
Total blocks (validated): 129 (avg. block size 1065418244 B)
Minimally replicated blocks: 129 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
```

15. Now run the Terasort Hadoop jar file to sort the 128GB data.

```
$ hadoop jar hadoop_demo-0.0.1-SNAPSHOT.jar assigment.hadoop_demo.Terasort /input /output
```

We can also trace the process using public ip and port number as follow.

```
$ http://52.40.13.239:8088/cluster
```

The screenshot shows a web browser window with the URL <http://52.40.13.239:8088/cluster>. The title bar of the browser says "RUNNING Applications". The main content area is titled "RUNNING Applications" and features a "hadoop" logo. On the left, there's a sidebar with a tree view of the cluster, showing sections for Cluster, Applications, and Scheduler. The Applications section lists statuses: NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED, and Scheduler. The Scheduler section shows metrics for Capacity Scheduler. The main table displays "Cluster Metrics" with the following data:

	Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	7	8 GB	8 GB	0 B	7	8	0	0	1	0	0	0	0

Below this is a "Scheduler Metrics" table for the Capacity Scheduler, showing scheduling resource type as [MEMORY] and minimum allocation as <memory:1024, vCores:1> and maximum allocation as <memory:8192, vCores:8>. A table below shows the details for the application "application_1512329171415_0001", which is running a TeraSort job on a single node with 13.1 large containers. The application was submitted at Sun Dec 3 13:44:50 -0600 2017 and is currently RUNNING.

MapReduce Job job_1512329171415_0001

Job Name: TeraSort on Hadoop i3.large
State: RUNNING
Uberized: false
Started: Sun Dec 03 19:44:56 UTC 2017
Elapsed: 2hrs, 14mins, 59sec

Attempt Number	Start Time	Node	Logs
1	Sun Dec 03 19:44:53 UTC 2017	ip-172-31-1-157.us-west-2.compute.internal:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map	<div style="width: 128px;"></div>	128	0	0	128
Reduce	<div style="width: 100px;"></div>	100	6	78	Successful

Attempt Type: Maps
Maps: 16
Reducers: 6

Reducer Phase.

Reduce Tasks for job_1512329171415_0001

Task	Progress	Status	State	Start Time	Finish Time	Elapsed Time
task_1512329171415_0001_r_000061	<div style="width: 100px;"></div>	reduce > reduce	RUNNING	Sun Dec 3 15:48:33 -0600 2017	N/A	2mins, 10sec
task_1512329171415_0001_r_000062	<div style="width: 100px;"></div>	reduce > reduce	RUNNING	Sun Dec 3 15:48:39 -0600 2017	N/A	2mins, 4sec
task_1512329171415_0001_r_000063	<div style="width: 100px;"></div>	reduce > reduce	RUNNING	Sun Dec 3 15:48:42 -0600 2017	N/A	2mins, 1sec
task_1512329171415_0001_r_000064	<div style="width: 100px;"></div>	reduce > reduce	RUNNING	Sun Dec 3 15:48:43 -0600 2017	N/A	2mins, 0sec
task_1512329171415_0001_r_000065	<div style="width: 100px;"></div>	reduce > reduce	RUNNING	Sun Dec 3 15:48:47 -0600 2017	N/A	1mins, 56sec
task_1512329171415_0001_r_000066	<div style="width: 100px;"></div>	reduce > reduce	RUNNING	Sun Dec 3 15:48:52 -0600 2017	N/A	1mins, 51sec

When mapreduce completes its job, we can see following screen on hadoop webpage.

Application application_1512329171415_0001

User: root
Name: TeraSort on Hadoop i3.large
Application Type: MAPREDUCE
Application Tags:
YarnApplicationState: FINISHED
FinalStatus Reported by AM: SUCCEEDED
Started: Sun Dec 03 19:44:50 +0000 2017
Elapsed: 2hrs, 27mins, 59sec
Tracking URL: History
Diagnostics:

Attempt ID	Started	Node	Logs
appattempt_1512329171415_0001_000001	Sun Dec 3 13:44:51 -0600 2017	http://ip-172-31-1-157.us-west-2.compute.internal:8042	Logs

We can also see the details on the terminal as follow.

```

root@ip-172-31-1-157:/home/ubuntu
root@ip-172-31-1-157:/data/64_158x15-i3.large
17/12/03 22:12:51 INFO mapreduce.Job: Job job_1512329171415_0001 completed successfully
17/12/03 22:12:51 INFO mapreduce.Job: Counters: 50
  File System Counters
    FILE: Number of bytes read=401327677313
    FILE: Number of bytes written=542527988326
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=137439375216
    HDFS: Number of bytes written=138813229612
    HDFS: Number of read operations=684
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=200
  Job Counters
    Killed reduce tasks=1
    Launched map tasks=128
    Launched reduce tasks=101
    Data-local map tasks=128
    Total time spent by all maps in occupied slots (ms)=21233521
    Total time spent by all reduces in occupied slots (ms)=31395729
    Total time spent by all map tasks (ms)=21233521
    Total time spent by all reduce tasks (ms)=31395729
    Total vcore-seconds taken by all map tasks=21233521
    Total vcore-seconds taken by all reduce tasks=31395729
    Total megabyte-seconds taken by all map tasks=21743125504
    Total megabyte-seconds taken by all reduce tasks=32149226496
  Map-Reduce Framework
    Map input records=1374388412
    Map output records=1374388412
    Map output bytes=138813229612
    Map output materialized bytes=141562083236
    Input split bytes=13824
    Combine input records=0
    Combine output records=0
    Reduce input groups=1374388412
    Reduce shuffle bytes=141562083236
    Reduce input records=1374388412
    Reduce output records=1374388412
    Spilled Records=5266991274
    Shuffled Maps =12800
    Failed Shuffles=0
    Merged Map outputs=12800
    Merged Map outputs=12800
    GC time elapsed (ms)=844914
    CPU time spent (ms)=14738030
    Physical memory (bytes) snapshot=66748776448
    Virtual memory (bytes) snapshot=450048360448
    Total committed heap usage (bytes)=46350729216
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=137439361392
  File Output Format Counters
    Bytes Written=138813229612
Total time to sort data on hadoop: 8883.314 seconds
root@ip-172-31-1-157:/home/ubuntu# 

```

Output files will be divided into 100 small size files. We can verify sorting results with valsort.

```

root@ip-172-31-1-157:/data/output128# ls
part-r-00000 part-r-00010 part-r-00020 part-r-00030 part-r-00040 part-r-00050 part-r-00060 part-r-00070 part-r-00080 part-r-00090 SUCCESS
part-r-00001 part-r-00011 part-r-00021 part-r-00031 part-r-00041 part-r-00051 part-r-00061 part-r-00071 part-r-00081 part-r-00091
part-r-00002 part-r-00012 part-r-00022 part-r-00032 part-r-00042 part-r-00052 part-r-00062 part-r-00072 part-r-00082 part-r-00092
part-r-00003 part-r-00013 part-r-00023 part-r-00033 part-r-00043 part-r-00053 part-r-00063 part-r-00073 part-r-00083 part-r-00093
part-r-00004 part-r-00014 part-r-00024 part-r-00034 part-r-00044 part-r-00054 part-r-00064 part-r-00074 part-r-00084 part-r-00094
part-r-00005 part-r-00015 part-r-00025 part-r-00035 part-r-00045 part-r-00055 part-r-00065 part-r-00075 part-r-00085 part-r-00095
part-r-00006 part-r-00016 part-r-00026 part-r-00036 part-r-00046 part-r-00056 part-r-00066 part-r-00076 part-r-00086 part-r-00096
part-r-00007 part-r-00017 part-r-00027 part-r-00037 part-r-00047 part-r-00057 part-r-00067 part-r-00077 part-r-00087 part-r-00097
part-r-00008 part-r-00018 part-r-00028 part-r-00038 part-r-00048 part-r-00058 part-r-00068 part-r-00078 part-r-00088 part-r-00098
part-r-00009 part-r-00019 part-r-00029 part-r-00039 part-r-00049 part-r-00059 part-r-00069 part-r-00079 part-r-00089 part-r-00099
root@ip-172-31-1-157:/data/output128# 

```

Validating sorting results.

```

File Edit View Search Terminal Help
root@ip-172-31-1-157:/data/64# ./valsort /data/output128/part-r-00000
Records: 13738553
Checksum: 68d90c70ca881a
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-1-157:/data/64# █

```

Results for 128GB *Hadoop Terasort*.

Data size	128 GB = 137438953472 Bytes
No of Mappers	128
No of Reducer	2
Total Bytes Read (HDFS)	137439375216
Total Bytes Write (HDFS)	138813229612
No of Read ops (HDFS)	684
No Of Write ops (HDFS)	200
Time Taken by whole process	8883.314 seconds
Throughput	15.471 MB/s

Explanation:

As here we have limited storage available and hadoop mapreduce also generates extra temporary files along with input files, consumes storages space and with increasing size of data, hadoop process slows down because it needs to clean up the junk files in order to save more data, which takes time and halts the mapreduce process for some amount of time.

Difficulties:

I have setup hadoop environment with 3 trial. During 1st trial, I have tried to run Terasort program with default configuration for 50GB dataset and note the time.

After this, I have update configuration like namenode and datanode path, tmp file path and jobtracker URI. I have noted that with this configuration I got improvisation than default configuration.

At the last I tried to change hadoop dfs.block.size property which highly improved my hadoop system performance as the file block

size changes from 128MB to 1024MB. It reduces total number of Mapper and reducer and eventually helped me to increase performance.

c. Spark Terasort:

To perform Spark Terasort, I assume we have already set up our hadoop environment and We have uploaded 128GB unsorted file in our hdfs system.

Follow the following steps to set up spark environment.

We have used spark apache spark-2.2.0 for our assignment.

1. Download spark-2.2.0 from apache spark website and move it to /data directory.

```
$ cd /data  
$ wget http://apache.claz.org/spark/spark-2.2.0/spark-2.2.0-bin-hadoop2.7.tgz  
$ tar xvzf spark-2.2.0-bin-hadoop2.7.tgz  
$ rm spark-2.2.0-bin-hadoop2.7.tgz  
$ mv spark-2.2.0-bin-hadoop2.7 spark
```

2. Now initiate spark-shell with following command.

```
$ spark-shell --conf spark.local.dir=/data/sparktmp --executor-memory 6g --num-executors 2 --driver-memory 4g
```

3. Write following scala Terasort program in spark shell.

```
Scala> val lines = sc.textFile("hdfs://localhost:9000/input/sample128gb.txt")  
scala> val rdd1 = lines.map(x=>(x.slice(0,10), x.slice(10,x.length)))  
scala> val rdd2 = rdd1.sortBy(_.value)  
scala> rdd2.map(k => k._1+k._2).saveAsTextFile("hdfs://localhost:9000/output")
```

4. After submitting Spark job, we can trace it through our local webpage:

`http://<public_ip>:4040`

Details for Job 1

Status: RUNNING

Active Stages: 1

Completed Stages: 1

- ▶ Event Timeline
- ▶ DAG Visualization

Active Stages (1)

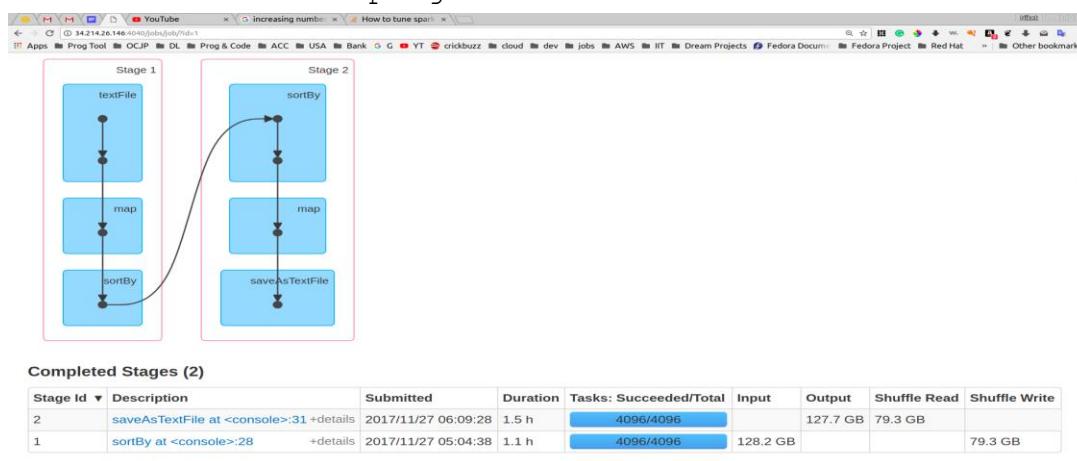
Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	saveAsTextFile at <console>:31 +details (kill)	2017/11/27 03:29:05	52 s	42/4096	1341.0 MB	763.5 MB		

Completed Stages (1)

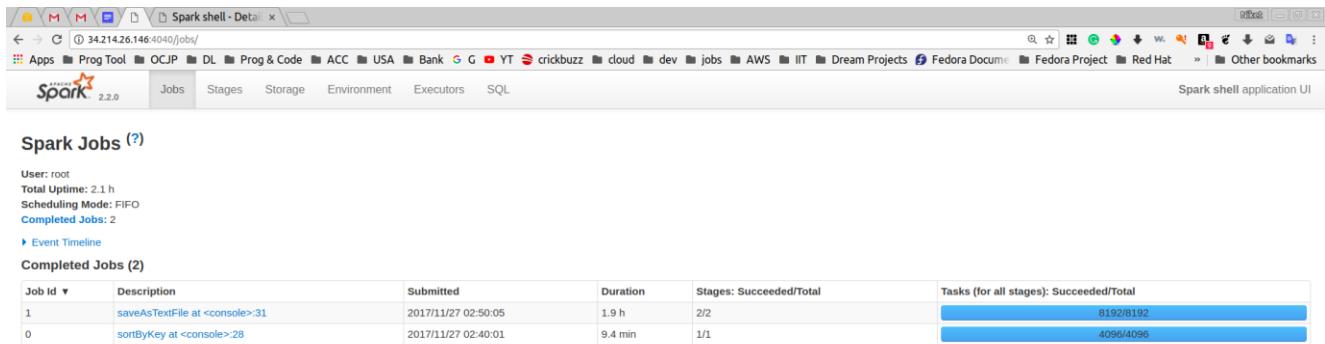
Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	map at <console>:26 +details	2017/11/27 02:50:05	39 min	4096/4096	128.2 GB			72.7 GB

34.214.26.146:4040/jobs/job?id=1&activeStage.sort=Submitted&activeStage.pageSize=100#active

DAG Visualization of our program.



When Spark completes its execution.



The screenshot shows the Spark shell application UI with the title "Spark shell - Details". It displays the "Completed Jobs (2)" section. The table shows two completed jobs:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	saveAsTextFile at <console>:31	2017/11/27 02:50:05	1.9 h	2/2	8192/8192
0	sortByKey at <console>:28	2017/11/27 02:40:01	9.4 min	1/1	4096/4096

Results for 128GB [actually 129GB generated (overhead bytes per line)]

Data size	128 GB = 137438953472 Bytes
Total Bytes Read (HDFS)	137439487516
Total Bytes Write (HDFS)	137438953500
Time Taken by whole process	10080 seconds
Time Taken to sort the data	9.4 min = 564 seconds
Throughput	13.63 MB/s

Configuration: 2

Virtual Cluster (1-Node, i3.4xlarge)

AWS I3.4xlarge configuration:

Instance Name	i3.4xlarge
vCPU count	16
Memory	122 GiB
Instance Storage (NVMe SSD)	3.8 TB
Price/Hour	\$1.248
Operating System	Ubuntu 16.04.3 LTS

Launch the instance i3.4xlarge.

The screenshot shows the AWS EC2 Management Console in Mozilla Firefox. The left sidebar is collapsed, and the main area displays a table of instances. A single row is selected, representing an i3.4xlarge instance with the ID i-0343ecf02b2a37350. The instance is running in the us-west-2c availability zone. The Public DNS is ec2-34-215-5-206.us-west-2.compute.amazonaws.com, and the IPv4 Public IP is 34.215.5.206. The instance has a key name c2 and was launched in December. Below the table, a detailed view of the selected instance is shown in a card. The card includes tabs for Description, Status Checks, Monitoring, and Tags. The Description tab shows the following details:

Attribute	Value
Instance ID	i-0343ecf02b2a37350
Instance state	running
Instance type	i3.4xlarge
Elastic IPs	-
Availability zone	us-west-2c
Security groups	niket-all-traffic, view inbound rules
Scheduled events	No scheduled events

On the right side of the card, there is a list of network and connectivity details:

Attribute	Value
Public DNS (IPv4)	ec2-34-215-5-206.us-west-2.compute.amazonaws.com
IPv4 Public IP	34.215.5.206
IPv6 IPs	-
Private DNS	ip-172-31-3-26.us-west-2.compute.internal
Private IPs	172.31.3.26
Secondary private IPs	-
VPC ID	vpc-1fc0dc79

a. Shared Memory Sort

To perform shared Memory Terasort on i3.4xlarge follow the same steps as we performed for i3.large Terasort and generate 1TB data using gensort.

```
$ ./gensort -a -t12 10995116277 sample1tb.txt
```

Use the following command to run it as background service.

```
$ nohup java ExternalSort sample1tb.txt &
```

```
ubuntu@ip-172-31-1-208: ~
File Edit View Search Terminal Help
top - 05:25:18 up 1:34, 3 users, load average: 3.30, 1.91, 1.41
Tasks: 235 total, 1 running, 234 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.8 us, 0.5 sy, 0.0 ni, 93.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 12582787+total, 38612836 free, 20979120 used, 66235920 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 10401057+avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
17640 root 20 0 34.471g 0.019t 16656 S 101.0 16.4 9:13.11 java
  1 root 20 0 37752 5784 3960 S 0.0 0.0 0:02.81 systemd
  2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
  3 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0
  5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H
  7 root 20 0 0 0 0 S 0.0 0.0 0:01.45 rcu_sched
  8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
  9 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
 10 root rt 0 0 0 0 S 0.0 0.0 0:00.02 watchdog/0
 11 root rt 0 0 0 0 S 0.0 0.0 0:00.00 watchdog/1
 12 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/1
 13 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/1
 15 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/1:0H
 16 root rt 0 0 0 0 S 0.0 0.0 0:00.00 watchdog/2
 17 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/2
 18 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/2
```

Output of Program.

```
root@ip-172-31-1-208: /data/sharedMem
File Edit View Search Terminal Help
4.0K nohup.out
1.0T output.txt
32K stamp
root@ip-172-31-1-208:/data/sharedMem# cat nohup.out
Please enter Source file name:Spliting Large File .....
Free Memory: 1919313496
Small piece of chuncks created.....
Sorting process started.....
Total chunk of tmp files: 1123
Reading a file.....
Individual chuncks of data sorted.....
Merging is under progress.....
Thread#0 started.....
Thread#1 started.....
Thread#2 started.....
Thread#3 started.....
Thread#4 started.....
Thread#5 started.....
Thread#6 started.....
Thread#7 started.....
Thread#8 started.....
Thread#9 started.....
Thread#10 started.....
Thread#11 started.....
Thread#12 started.....
Thread#13 started.....
Files sorted and Merged.....
Total Time: 45485 seconds
root@ip-172-31-1-208:/data/sharedMem#
```

- Result using valsort.

```
root@ip-172-31-1-208:/data/64# cat nohup.out
Records: 10995115155
Checksum: 147ada84970251286
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-1-208:/data/64#
```

Result to sort 1 TB data using External Sort.

Data size	1 TB = 1099511627776 Bytes
No. of Bytes Read (FILE)	5.4975581e+12
No. of Bytes Write (FILE)	2199023255552
Time Taken by whole process	45485 seconds
Throughput	24.17 MB/s

b. Hadoop Terasort

We will have 2 1.7 TB SSD attached with our i3.4xlarge instance. To utilize these disk, we will create RAID 0 disk array.

To create RAID 0 array, follow the steps.

```
$ Sudo su
$ sudo apt-get install mdadm
$ mdadm --create /dev/md0 --run --level=0 --name=RAID --force --
  raid-device=2 /dev/nvme0n1 /dev/nvme1n1
$ mkfs.ext4 -L RAID /dev/md0
$ mkdir /data
$ mount LABEL=RAID /data
$ chmod 777 /data
```

Follow configuration-1 hadoop installation steps to install and run Hadoop program.

To create 1TB data use following command.

```
$ ./gensort -a -t12 10995116277 sample1tb.txt
```

Load the data to hdfs using following command.

```
$ hdfs dfs -copyFromLocal sample1tb.txt /input
```

Execute the Terasort Jar file using following command.

```
$ hadoop jar hadoop_demo-0.0.1-SNAPSHOT.jar assigment.hadoop_demo.Terasort /input /output
```

The process will be traced using hadoop UI portal.

The screenshot shows the Hadoop User Interface (UI) for the 'All Applications' view. The top navigation bar includes links for EC2 Management, scripts - Google Dr, PA2 Report - Google, All Applications, Application applic, and MapReduce Job. The main title is 'All Applications'. On the left, there's a sidebar with 'Cluster Metrics' (About, Nodes, Node Labels, Applications), 'Scheduler Metrics' (Capacity Scheduler, Scheduler Type, Scheduling Resource Type, Minimum Allocation, Maximum Allocation, Maximum Cluster Application Priority), and 'Tools' (Scheduler, Tools). The main content area displays a table of application details. One row is highlighted, representing a TeraSort job:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes	ApplicationMaster
application_1512005009455_0001	root	TeraSort on Hadoop	MAPREDUCE	default	0	Wed Nov 29 20:33:38 -0600 2017	N/A	RUNNING	UNDEFINED	7	8192	100.0	100.0				0	

At the bottom, it says 'Showing 1 to 1 of 1 entries' and has navigation links for First, Previous, Next, and Last.

The running process.

Job Overview

Job Name: TeraSort on Hadoop i3.large
State: RUNNING
Uberized: false
Started: Sat Dec 02 05:11:31 UTC 2017
Elapsed: 35mins, 11sec

Attempt Number	Start Time	Node	Logs
1	Sat Dec 02 05:11:29 UTC 2017	ip-172-31-3-26.us-west-2.compute.internal:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map	<div style="width: 1024px;"></div>	1024	775	5	244
Reduce	<div style="width: 100px;"></div>	100	99	1	0

Attempt Type	New	Running	Failed	Killed	Successful
Maps	775	5	0	0	244
Reduces	99	1	0	0	0

Windows Taskbar: Type here to search, Start button, File Explorer, Mail, Edge, Firefox, Google Chrome, File Manager, Task View, Taskbar icons, 11:47 PM, 12/1/2017.

Note: The name displaying here in the screenshot is referring to the name provided in `Terasort.java` file.

Application Overview

User: root
Name: TeraSort on Hadoop i3.large
Application type: MAPREDUCE
Application Tags:
YarnApplicationState: FINISHED
FinalStatus Reported by AM: SUCCEEDED
Started: Sat Dec 02 05:11:27 +0000 2017
Elapsed: 5hrs, 21sec
Tracking URL: History
Diagnostics:

Application Metrics

Total Resource Preempted: <memory:0, vCores:0>
Total Number of Non-AM Containers Preempted: 0
Total Number of AM Containers Preempted: 0
Resource Preempted from Current Attempt: <memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt: 0
Aggregate Resource Allocation: 147000914 MB-seconds, 124974 vcore-seconds

Show 20 ▾ entries	Search:		
Attempt ID	Started	Node	Logs
appattempt_1512187649386_0001_000001	Fri Dec 1 23:11:27 -0600 2017	http://ip-172-31-3-26.us-west-2.compute.internal:8042	Logs

Showing 1 to 1 of 1 entries

Windows Taskbar: Type here to search, Start button, File Explorer, Mail, Edge, Firefox, Google Chrome, File Manager, Task View, Taskbar icons, 8:19 AM, 12/2/2017.

Screenshot of a web browser showing the Hadoop cluster management interface at 35.160.81.224:8088/cluster. The title bar says "All Applications". The page displays "All Applications" with a Hadoop logo. On the left, there's a sidebar with "Cluster Metrics" and "Scheduler Metrics" sections, and a "Tools" section. The main area shows a table of application metrics and a detailed view of a single application named "application_1512187649386_0001". The application details table includes columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, and Tracking UI. The status is "SUCCEEDED". Below the table, it says "Showing 1 to 1 of 1 entries". At the bottom of the page, there's a search bar and navigation links.

After Completing process .

```
root@ip-172-31-3-26:/home/ubuntu
root@ip-172-31-3-26:/home/ubuntu 158x41
Counters: 49
File System Counters
FILE: Number of bytes read=4096648974240
FILE: Number of bytes written=5216339944692
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=1099515926452
HDFS: Number of bytes written=1098375069600
HDFS: Number of read operations=3369
HDFS: Number of large read operations=0
HDFS: Number of write operations=199
Job Counters
Launched map tasks=1024
Launched reduce tasks=100
Data-local map tasks=1024
Total time spent by all maps in occupied slots (ms)=44646946
Total time spent by all reduces in occupied slots (ms)=61196180
Total time spent by all map tasks (ms)=44646946
Total time spent by all reduce tasks (ms)=61196180
Total vcore-seconds taken by all map tasks=44646946
Total vcore-seconds taken by all reduce tasks=61196180
Total megabyte-seconds taken by all map tasks=45718472704
Total megabyte-seconds taken by all reduce tasks=62664888320
Map-Reduce Framework
Map input records=10995116277
Map output records=10995116277
Map output bytes=1099511627700
Map output materialized bytes=1121502474654
Input split bytes=108544
Combine input records=0
Combine output records=0
Reduce input groups=10983760690
Reduce shuffle bytes=1121502474654
Reduce input records=10983760690
Reduce output records=10983760690
Spilled Records=51029224524
Shuffled Maps =102400
Failed Shuffles=0
Merged Map outputs=102400
GC time elapsed (ms)=4595515
CPU time spent (ms)=110342470
```

```

CPU time spent (ms)=110342470
Physical memory (bytes) snapshot=341006716928
Virtual memory (bytes) snapshot=2271661056000
Total committed heap usage (bytes)=231906738176
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Reads=109515817908
File Output Format Counters
  Bytes Written=1098376070100

```

```

root@ip-172-31-3-26: /data
File Edit View Search Terminal Help
0 updates are security updates.

Last login: Sat Dec 2 08:34:58 2017 from 69.47.179.48
ubuntu@ip-172-31-3-26:~$ clear
ubuntu@ip-172-31-3-26:~$ ls
hadoop_demo-0.0.1-SNAPSHOT.jar
ubuntu@ip-172-31-3-26:~$ sudo su
root@ip-172-31-3-26:/home/ubuntu#
hadoop_demo-0.0.1-SNAPSHOT.jar
root@ip-172-31-3-26:/home/ubuntu# cd /data/
root@ip-172-31-3-26:/data# ls
datagen hadoop_hadoop-2.7.1.tar.gz hdfs lost+found tmp
root@ip-172-31-3-26:/data# hadoop job -list
DEPRECATED: Use of this script to execute mapred command is deprecated.
Instead use the mapred command for it.

17/12/02 14:23:28 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Total jobs:0
      JobId          State        StartTime      UserName      Queue      Priority      UsedContainers    RsvdContainers    UsedMem
      RsvdMem NeededMem
root@ip-172-31-3-26:/data# hadoop job -list all
DEPRECATED: Use of this script to execute mapred command is deprecated.
Instead use the mapred command for it.

17/12/02 14:24:46 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Total jobs:1
      JobId          State        StartTime      UserName      Queue      Priority      UsedContainers    RsvdContainers    UsedMem
      RsvdMem NeededMem
job_1512187649386_0001 SUCCEEDED      1512191487864      root      default      NORMAL      N/A            N/A            N/A
      N/A           N/A      http://ip-172-31-3-26:8088/proxy/application_1512187649386_0001/
root@ip-172-31-3-26:/data# |

```

Validating data using ./valsor

```

root@ip-172-31-3-26:/data/datagen/64#
File Edit View Search Terminal Help
root@ip-172-31-3-26:/data/datagen/64# ./valsor /data/output/part-r-00000
Records: 109955710
Checksum: 34703aaabe67458
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-3-26:/data/datagen/64# |

```

Results for 1 TB Data.

Data size	1 TB = 1,099,511,627,776 Bytes
No of Mappers	1024
No of Reducer	16
Total Bytes Read (HDFS)	1099515926452

Total Bytes Write (HDFS)	1098375069600
No of Read ops (HDFS)	3369
No Of Write ops (HDFS)	200
Time Taken by whole process	18021 seconds
Throughput	61.01 MB/s

c. Spark Terasort

To run spark on i3.4xlarge instance, follow the above mentioned steps to download the spark and use the following code to run spark.

1. Launch i3.4xlarge instance on AWS.

The screenshot shows the AWS EC2 Management Console interface. On the left, there's a navigation sidebar with options like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Images, AMIs, Bundle Tasks, Elastic Block Store, Volumes, Snapshots, Network & Security, Security Groups, Load Balancing, Target Groups, and Auto Scaling. The 'Instances' section is currently selected. The main pane displays a table of instances. One row is highlighted, showing details for an i3.4xlarge instance with the ID i-0343ecf02b2a37350. The instance is running in the us-west-2c availability zone. Its public DNS is ec2-34-215-5-206.us-west-2.compute.amazonaws.com and its private IP is 34.215.5.206. It has two elastic IPs assigned. The monitoring status is disabled. A 'Description' tab is open, showing the instance ID, state, type, availability zone, security group (nikeit-all-traffic), and scheduled events. Other tabs include 'Status Checks' and 'Tags'. At the bottom of the instance card, there are icons for stopping, terminating, or launching the instance.

2. Now initiate spark-shell with following command.

```
$ spark-shell --conf spark.local.dir=/data/sparktmp --executor-memory 12g --num-executors 2 --driver-memory 4g
```

2. Write following scala Terasort program in spark shell.

```
Scala> val lines = sc.textFile("hdfs://localhost:9000/input/part50.txt")
scala> val rdd1 = lines.map(x=>(x.slice(0,10), x.slice(10,x.length)))
scala> val rdd2 = rdd1.sortBy(_._1)
scala> rdd2.map(k => k._1+k._2).saveAsTextFile("hdfs://localhost:9000/output50")
```

3. After submitting Spark job, we can trace it through our local webpage:

```
http://<public_ip>:4040
```

Stage:1

User: root
Total Uptime: 4.1 min
Scheduling Mode: FIFO
Active Jobs: 1

Event Timeline

Active Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	sortBy at <console>:28 (kill)	2017/12/02 17:11:57	1.1 min	0/1	48/1024



Stage:2

The screenshot shows the Apache Spark 2.2.0 UI interface. At the top, there's a navigation bar with various icons and links. Below it, the main header says "Spark shell - Details for Job 1". The main content area is titled "Details for Job 1" and shows the following information:

- Status: RUNNING
- Active Stages:** 1
- Pending Stages:** 1
- Event Timeline**
- DAG Visualization**

Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	sortBy at <console>:28	+details (kill)	2017/12/02 17:28:15	43 s	0/1024			

Pending Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	saveAsTextFile at <console>:31	+details	Unknown	Unknown	0/1024			



Results for 1 TB data.

Data size	1 TB = 1,099,511,627,776 Bytes
Total Bytes Read (HDFS)	1099515926452
Total Bytes Write (HDFS)	1098375069600
Time Taken by whole process	25920 seconds
Time Taken to sort the data	15 min = 900 seconds
Throughput	42.45 MB/s

Configuration: 3

Virtual Cluster (8-Node, i3.large)

AWS I3.large configuration:

Instance Name	i3.large
vCPU count	2
Memory	15.25 GiB
Instance Storage (NVMe SSD)	0.475 TB
Price/Hour	\$0.15
Operating System	Ubuntu 16.04.3 LTS

a. Hadoop 8 Node Terasort.

To perform Terasort on hadoop 8 node cluster, we need to create 8 VM machine of i3.large configuration.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name	Monitoring	Launch Time
Master	i-0311d3dff41253131	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-18-217-114-249.us-east-2.compute.amazonaws.com	18.217.114.249	-	cluster	disabled	December
Slave-1	i-06651ddadeb03e8...	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-18-217-49-93.us-e...	18.217.49.93	-	cluster	disabled	December
Slave-2	i-071ef116296e5441	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-18-216-150-43.us...	18.216.150.43	-	cluster	disabled	December
Slave-3	i-098ee97085b6c536	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-18-217-188-14.us...	18.217.188.14	-	cluster	disabled	December
Slave-4	i-0ae6091efdf9b41bb	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-18-220-66-55.us-e...	18.220.66.55	-	cluster	disabled	December
Slave-5	i-0ae6c8736b26b9440	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-18-221-251-65.us...	18.221.251.65	-	cluster	disabled	December
Slave-6	i-0afea48974e4d0f6b	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-18-217-76-65.us-e...	18.217.76.65	-	cluster	disabled	December
Slave-7	i-0d764e36705e163...	i3.large	us-east-2c	running	2/2 checks ...	None	ec2-52-14-155-237.us...	52.14.155.237	-	cluster	disabled	December

Follow the following steps to create cluster using this 8 node.

1. Upload all autonomous script from Autonomous Script folder using following command to each node.

```
$ scp -i cluster.pem autonomus.xz ubuntu@ec2-18-217-114-249.us-east-2.compute.amazonaws.com
```

2. Run Mount.sh script to mount 475 GB hard disk with each running node.

```
$ ./Mount.sh
```

3. Run install-required-package.sh script from each node to install java and ssh.

```
$ ./install-required-package.sh
```

4. Add private ip node_name pair to /etc/hosts file and save it.

```
$ vi /etc/hosts
```

```
172.31.33.117 master
172.31.46.18 slave1
172.31.42.104 slave2
172.31.45.111 slave3
172.31.46.173 slave4
172.31.41.130 slave5
172.31.33.251 slave6
172.31.44.75 slave7
```

5. Now create public private key pair using generate-ssh.sh script from master node and copy public key to each slave node authorized_keys file residing in .ssh folder

```
$ ssh-keygen -t rsa -P ""
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

6. Now, download hadoop-2.7.1 using Hadoop-Download.sh script from each node.

```
$ ./Hadoop-Download.sh
```

Now update following files to the /data/hadoop/etc/hadoop/ directory.

a. Hadoop-env.sh

```
export JAVA_HOME="/usr/lib/jvm/java-1.8.0-openjdk-amd64"
```

b. Core-site.xml

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

```
<name>hadoop.tmp.dir</name>
<value>/data/tmp</value>
</property>
</configuration>
```

c. Hdfs-site.xml

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.blocksize</name>
  <value>1073741824</value>
</property>
</configuration>
```

d. Mapred-site.xml

In order to edit this file, use following command first.

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

Add following lines at the end of the file.

```
<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
<property>
  <name>mapreduce.job.reduces</name>
  <value>16</value>
</property>
</configuration>
```

e. Yarn-site.xml

Add following lines at the end of the file.

```
<configuration>
```

```
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>master:54311</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

f. slaves

```
master
slave1
slave2
slave3
slave4
slave5
slave6
slave7
```

7. Copy this configuration to all the other node using following script.

```
$ for i in `cat /data/hadoop/etc/hadoop/slaves`; do \
> echo $i; rsync -avxP --exclude=logs /data/hadoop/
$i:/data/hadoop/; \
> done
```

```

root@ip-172-31-33-117:/data/hadoop/etc/hadoop
File Edit View Search Terminal Help
 909 100% 887.70kB/s 0:00:00 (xfr#6, ir-chk=1016/1069)

sent 179,185 bytes received 1,122 bytes 360,614.00 bytes/sec
total size is 329,421,112 speedup is 1,827.00
slave6
sending incremental file list
etc/hadoop/
etc/hadoop/core-site.xml
 960 100% 253.91kB/s 0:00:00 (xfr#1, ir-chk=1042/1069)
etc/hadoop/hadoop-env.sh
 4,249 100% 4.05MB/s 0:00:00 (xfr#2, ir-chk=1040/1069)
etc/hadoop/hdfs-site.xml
 1,163 100% 1.11MB/s 0:00:00 (xfr#3, ir-chk=1036/1069)
etc/hadoop/mapred-site.xml
 1,031 100% 1006.84kB/s 0:00:00 (xfr#4, ir-chk=1023/1069)
etc/hadoop/slaves
 56 100% 54.69kB/s 0:00:00 (xfr#5, ir-chk=1021/1069)
etc/hadoop/yarn-site.xml
 909 100% 887.70kB/s 0:00:00 (xfr#6, ir-chk=1016/1069)

sent 179,165 bytes received 1,102 bytes 120,178.00 bytes/sec
total size is 329,421,112 speedup is 1,827.41
slave7
sending incremental file list
etc/hadoop/
etc/hadoop/core-site.xml
 960 100% 253.91kB/s 0:00:00 (xfr#1, ir-chk=1042/1069)
etc/hadoop/hadoop-env.sh
 4,249 100% 4.05MB/s 0:00:00 (xfr#2, ir-chk=1040/1069)
etc/hadoop/hdfs-site.xml
 1,163 100% 1.11MB/s 0:00:00 (xfr#3, ir-chk=1036/1069)
etc/hadoop/mapred-site.xml
 1,031 100% 1006.84kB/s 0:00:00 (xfr#4, ir-chk=1023/1069)
etc/hadoop/slaves
 56 100% 54.69kB/s 0:00:00 (xfr#5, ir-chk=1021/1069)
etc/hadoop/yarn-site.xml
 909 100% 887.70kB/s 0:00:00 (xfr#6, ir-chk=1016/1069)

sent 179,185 bytes received 1,122 bytes 360,614.00 bytes/sec
total size is 329,421,112 speedup is 1,827.00
root@ip-172-31-33-117:/data/hadoop/etc/hadoop#

```

8. Export JAVA_HOME and HADOOP_HOME path using bashrc.sh script.

```
$ ./Mount.sh
```

9. Now use following command to format the namenode from master and run the daemons.

```
$ hadoop namenode -format
$ hdfs start-dfs.sh
$ hdfs start-yarn.sh
```

```

root@ip-172-31-33-117:~#
File Edit View Search Terminal Help
Decommission Status : Normal
Configured Capacity: 467278299136 (435.19 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 24183726080 (22.52 GB)
DFS Remaining: 443094548480 (412.66 GB)
DFS Used%: 0.00%
DFS Remaining%: 94.82%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Mon Dec 04 17:00:34 UTC 2017

root@ip-172-31-33-117:~# start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /data/hadoop/logs/hadoop-root-namenode-ip-172-31-33-117.out
slave1: starting datanode, logging to /data/hadoop/logs/hadoop-root-datanode-ip-172-31-33-117.out
slave2: starting datanode, logging to /data/hadoop/logs/hadoop-root-datanode-ip-172-31-42-104.out
slave3: starting datanode, logging to /data/hadoop/logs/hadoop-root-datanode-ip-172-31-45-111.out
slave4: starting datanode, logging to /data/hadoop/logs/hadoop-root-datanode-ip-172-31-46-173.out
slave5: starting datanode, logging to /data/hadoop/logs/hadoop-root-datanode-ip-172-31-41-138.out
slave6: starting datanode, logging to /data/hadoop/logs/hadoop-root-datanode-ip-172-31-33-251.out
slave7: starting datanode, logging to /data/hadoop/logs/hadoop-root-datanode-ip-172-31-44-75.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /data/hadoop/logs/hadoop-root-secondarynamenode-ip-172-31-33-117.out
starting yarn daemons
starting resourcemanager, logging to /data/hadoop/logs/yarn-root-resourcemanager-ip-172-31-33-117.out
slave7: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-44-75.out
slave3: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-45-111.out
slave5: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-41-130.out
slave6: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-33-251.out
slave1: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-46-18.out
slave2: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-42-104.out
slave4: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-46-173.out
master: starting nodemanager, logging to /data/hadoop/logs/yarn-root-nodemanager-ip-172-31-33-117.out
root@ip-172-31-33-117:~# 

```

10. Now generate 128gb file using gensort using Gensort-128gb.sh script and load the data to the hdfs. This process will add 1 TB data to hdfs.

```

$ ./Gensort-128gb.sh
$ hdfs dfs -mkdir /input
$ hdfs dfs -put part1.txt /input

```

11. Now run the Hadoop program using following command.

```

$ hadoop jar hadoop_demo-0.0.1-SNAPSHOT.jar assigment.hadoop_demo.Terasort /input /output

```

We can also trace the process using public ip of master node and port number 8088.

- 8 node JPS screenshot [ip of each node is visible on each terminal header.]

```

root@ip-172-31-44-75: /home/ubuntu
root@ip-172-31-33-117: /home/ubuntu 78x9 master
4429 YarnChild
2638 SecondaryNameNode
4334 YarnChild
4399 YarnChild
4112 YarnChild
2289 NameNode
2451 DataNode
4148 YarnChild
4184 YarnChild
root@ip-172-31-42-104: /home/ubuntu 78x9 slave2
23393 YarnChild
23253 YarnChild
23303 YarnChild
22038 DataNode
22887 YarnChild
23208 YarnChild
22169 NodeManager
23532 Jps
23373 YarnChild
23133 YarnChild
root@ip-172-31-42-104:/home/ubuntu# 

root@ip-172-31-46-18: /home/ubuntu 78x9 slave1
23811 YarnChild
23988 YarnChild
23861 YarnChild
24025 YarnChild
23883 YarnChild
23803 YarnChild
22637 DataNode
root@ip-172-31-46-18:/data/hadoop/etc/hadoop# 

root@ip-172-31-45-111: /home/ubuntu 78x9 slave3
23281 YarnChild
23315 YarnChild
22036 DataNode
23093 YarnChild
23141 YarnChild
23446 Jps
23383 YarnChild
22167 NodeManager
23336 YarnChild
23199 YarnChild
root@ip-172-31-45-111:/data# 

root@ip-172-31-45-111: /data# 
root@ip-172-31-45-111:/data# 

root@ip-172-31-41-130: /home/ubuntu 78x9 slave5
23267 Jps
22963 YarnChild
22835 YarnChild
23012 YarnChild
21911 NodeManager
22664 YarnChild
23161 YarnChild
root@ip-172-31-41-130:/home/ubuntu# 

root@ip-172-31-33-251: /home/ubuntu 78x9 slave6
5703 YarnChild
5655 YarnChild
3883 MRAppMaster
3565 DataNode
5726 YarnChild
5342 YarnChild
3695 NodeManager
root@ip-172-31-33-251:/data# 

root@ip-172-31-44-75: /home/ubuntu 78x8
24216 Jps
23995 YarnChild
23894 YarnChild
22763 DataNode
23996 YarnChild
22894 NodeManager
24079 YarnChild
root@ip-172-31-33-251:/data# 

root@ip-172-31-44-75:/home/ubuntu# 

```

The screenshot shows the Hadoop MapReduce Job Overview page. The job name is "TeraSort on Hadoop i3.large" and its state is "RUNNING". The job was started on Mon Dec 04 23:44:01 UTC 2017 and has been running for 2hrs, 37mins, 2sec. The ApplicationMaster table shows one attempt with 1024 Map tasks and 54 Reduce tasks. The Task Type table shows 1024 Running Map tasks and 54 Running Reduce tasks. The Attempt Type table shows 87 Failed Map tasks and 5 Failed Reduce tasks.

When Job completes.

```

root@ip-172-31-33-117:/home/ubuntu
niket@niket: ~/cloud/cluster
root@ip-172-31-33-117:/home/ubuntu 158x40

File System Counters
FILE: Number of bytes read=2360546272788
FILE: Number of bytes written=3059634345178
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=687197431320
HDFS: Number of bytes written=137438953400
HDFS: Number of read operations=2220
HDFS: Number of large read operations=0
HDFS: Number of write operations=200
Job Counters
Killed map tasks=86
Killed reduce tasks=6
Launched map tasks=1024
Launched reduce tasks=106
Data-local map tasks=237
Rack-local map tasks=489
Total time spent by all maps in occupied slots (ms)=172442513
Total time spent by all reduces in occupied slots (ms)=248683356
Total time spent by all map tasks (ms)=172442513
Total time spent by all reduce tasks (ms)=248683356
Total vcore-seconds taken by all map tasks=172442513
Total vcore-seconds taken by all reduce tasks=248683356
Total megabyte-seconds taken by all map tasks=176581133312
Total megabyte-seconds taken by all reduce tasks=254651756544
Map-Reduce Framework
Map input records=6871947670
Map output records=6871947670
Map output bytes=687194767000
Map output materialized bytes=700939046340
Input split bytes=63360
Combine input records=0
Combine output records=0
Reduce input groups=1374389534
Reduce shuffle bytes=700939046340
Reduce input records=6871947670
Reduce output records=1374389534
Spilled Records=29995499949
Shuffled Maps =64000

```

```

root@ip-172-31-33-117:/home/ubuntu
niket@niket:~/cloud/cluster
root@ip-172-31-33-117:/home/ubuntu 158x40

Total time spent by all map tasks (ms)=172442513
Total time spent by all reduce tasks (ms)=248683356
Total vcore-seconds taken by all map tasks=172442513
Total vcore-seconds taken by all reduce tasks=248683356
Total megabyte-seconds taken by all map tasks=176581133312
Total megabyte-seconds taken by all reduce tasks=254651756544
Map-Reduce Framework
  Map input records=6871947670
  Map output records=6871947670
  Map output bytes=687194767000
  Map output materialized bytes=700939046340
  Input split bytes=63360
  Combine input records=0
  Reduce input groups=1374389534
  Reduce shuffle bytes=700939046340
  Reduce input records=6871947670
  Reduce output records=1374389534
  Spilled Records=29995499949
  Shuffled Maps =64000
  Failed Shuffles=0
  Merged Map outputs=64000
  GC time elapsed (ms)=4968861
  CPU time spent (ms)=75414100
  Physical memory (bytes) snapshot=208915197952
  Virtual memory (bytes) snapshot=1456594714624
  Total committed heap usage (bytes)=145270243328
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=687197367960
File Output Format Counters
  Bytes Written=137438953400
Total time to sort data on hadoop: 17280.32 seconds
root@ip-172-31-33-117:/home/ubuntu#

```

- Validating data using ./valsor

```

root@ip-172-31-3-26:/data/datagen/64
File Edit View Search Terminal Help
root@ip-172-31-3-26:/data/datagen/64# ./valsor /data/output/part-r-00000
Records: 109955710
Checksum: 34703aaabe67458
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-3-26:/data/datagen/64#

```

Results for 1 TB Data.

Data size	1 TB = 1,099,511,627,776 Bytes
No of Mappers	1024
No of Reducer	16
Total Bytes Read (HDFS)	1099515926452
Total Bytes Write (HDFS)	1098375069600
No of Read ops (HDFS)	3369
No Of Write ops (HDFS)	200

Time Taken by whole process	17280 seconds
Throughput	63.62 MB/s

Difficulties:

To run the hadoop program over 8 node cluster, I haven't used any extra storage. So when the any slave node is running out of memory, nodemanager needs to clean the disk space by cleaning temporary file. Which was increasing time taken by mapper task. But to solve this, I used extra space using EBS and perform the Terasort. Which eventually gave me result almost similar to i3.4xlarge instance.

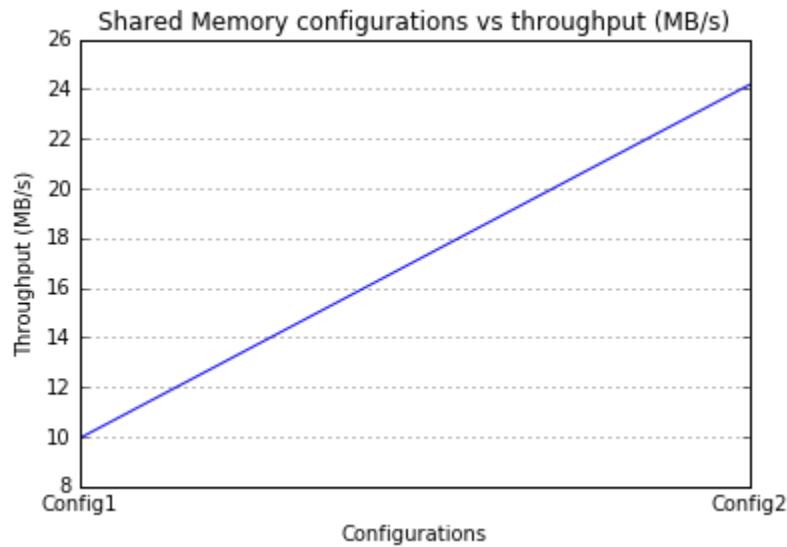
- **Conclusion**

Experiment	Shared Memory Terasort	Hadoop Terasort	Spark Terasort
Compute Time (sec) [1xi3.large 128GB]	13925 seconds	8883.314 seconds	10080 seconds > 9.4 min [To sort data]
Data Read (GB) [1xi3.large 128GB]	645 GB	256 GB	256 GB
Data Write (GB) [1xi3.large 128GB]	384 GB	256 GB	256 GB
I/O Throughput (MB/sec) [1xi3.large 128GB]	9.947 MB/s	15.471 MB/s	13.63 MB/s
Compute Time (sec) [1xi3.4xlarge 1TB]	45485 seconds	18021 seconds	25920 seconds > 15 MIN [To sort data]
Data Read (GB) [1xi3.4xlarge 1TB]	5120 GB	2048 GB	2048 GB
Data Write (GB) [1xi3.4xlarge 1TB]	3072 GB	2048 GB	2048 GB
I/O Throughput (MB/s) [1xi3.4xlarge 1TB]	24.17 MB/s	61.01 MB/s	42.45 MB/s

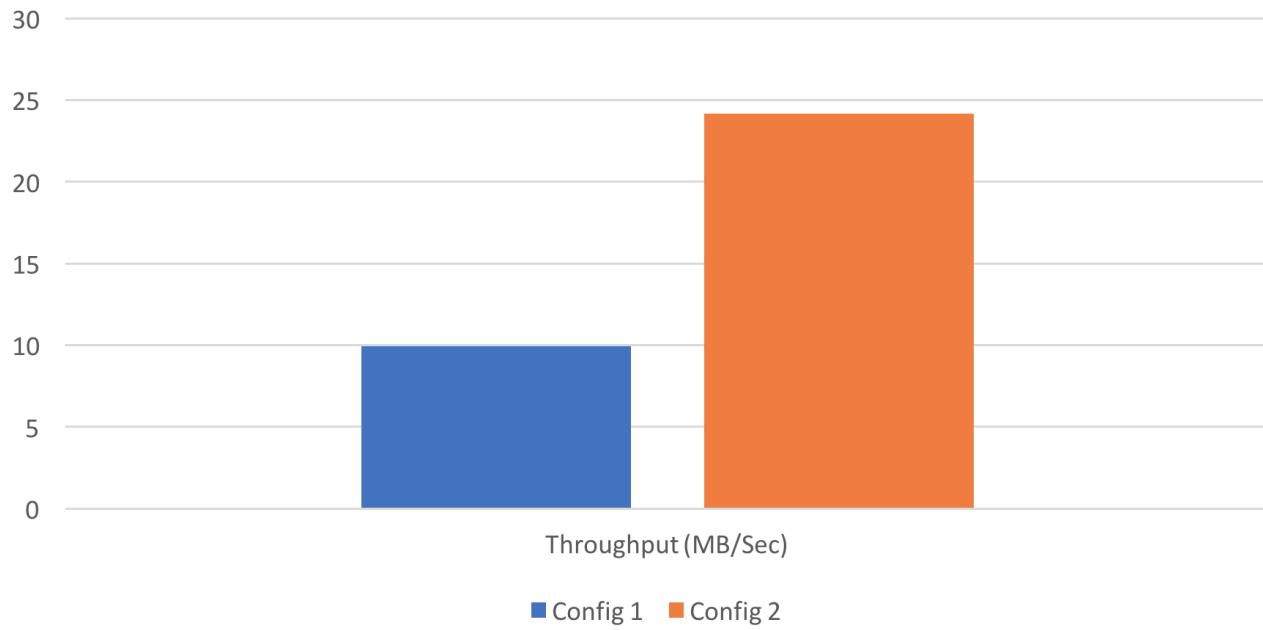
Compute Time (sec) [8xi3.large 1TB]	N/A	17280 seconds	---
Data Read (GB) [8xi3.large 1TB]	N/A	2048 GB	---
Data Write (GB) [8xi3.large 1TB]	N/A	2048 GB	---
I/O Throughput (MB/sec) [8xi3.large 1TB]	N/A	63.62 MB/s	---
Speedup (weak scale)	N/A	4.11	3.11
Efficiency (weak scale)	N/A	0.51375	0.38875

Here we have performed Terasort configuration for all three different configuration and with different environment. Following graph shows the performance for each configuration using their respective I/O throughput.

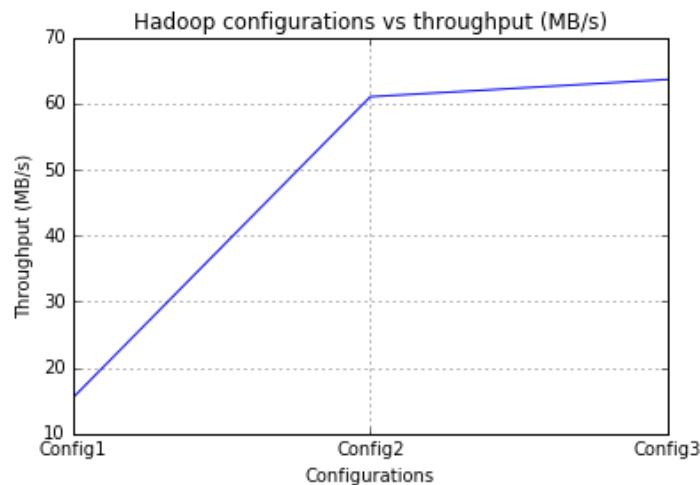
- Shared Memory Terasort

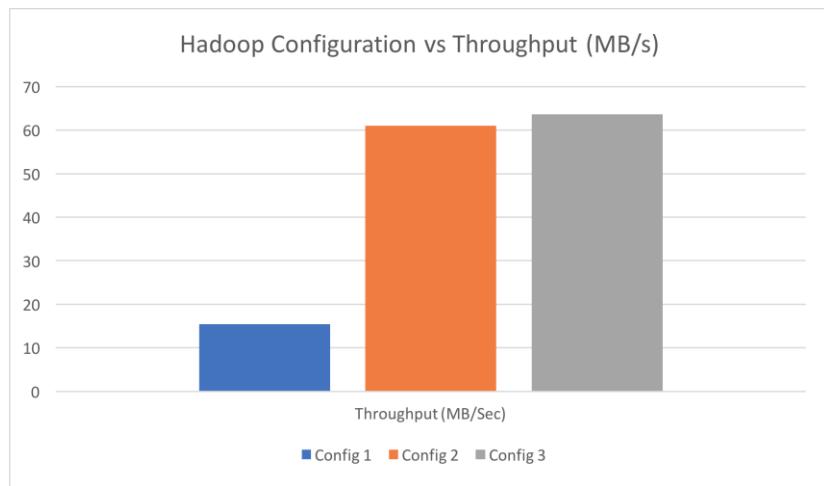


Shared Memory Configuration vs Throughput (MB/s)

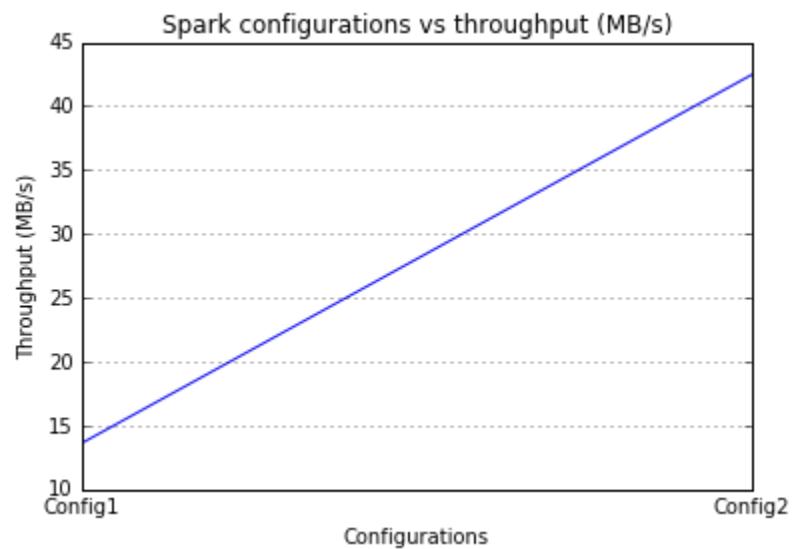


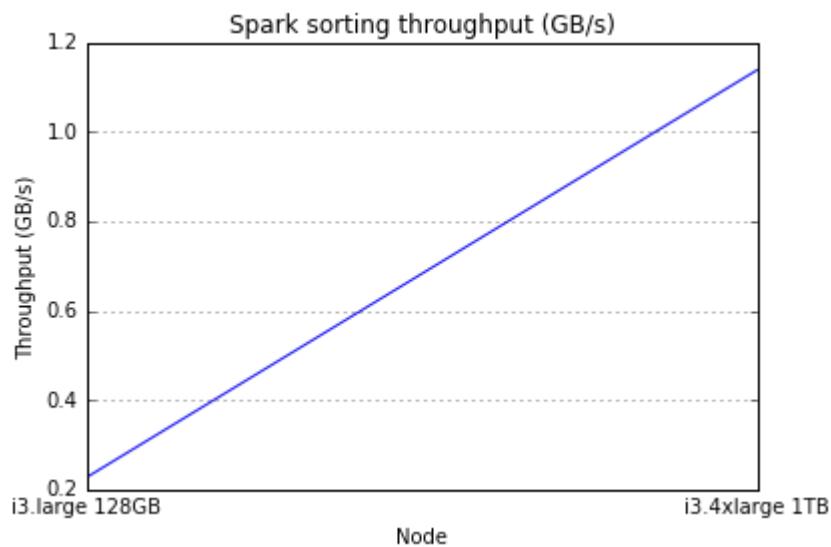
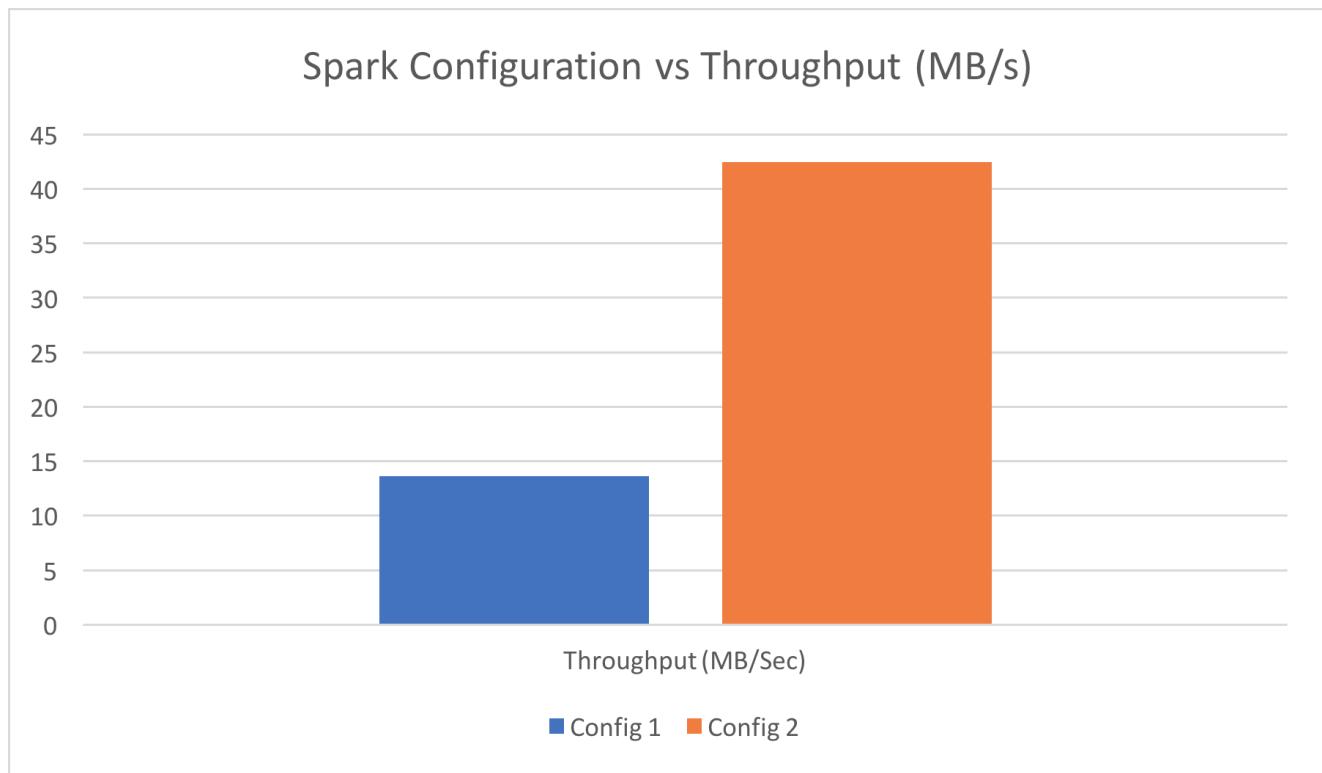
- Hadoop Terasort





- Spark Terasort





Best Performance:

To sort the data, spark is best at all number of node as it is performing in memory sorting, spark always wins. The major bottleneck for spark is disk i/o.

Comparing with 2013 & 2014 Sort Benchmark Winner using data available from <http://sortbenchmark.org/> .

Year 2013 Hadoop

Benchmark configuration: (2013)

System: 2100 nodes x (2.2.3Ghz hexcore Xeon E5-2630, 64 GB memory, 12x3TB disks)

Throughput: 4328 seconds to sort 102.5 TB data. I.e. 23.68 GB/Sec

If we consider it is linear scaling and find throughput for 8 node cluster it will give us throughput around 90.20 MB/sec.

By comparing our 8 node cluster result with this, we can say that our Cluster performance is almost 70% of the standard benchmark.

Year 2014 Spark

Benchmark configuration:

System: 207 nodes x(32 vCores - 2.5Ghz Intel Xeon E5-2670 v2, 244GB memory, 8x800 GB SSD

Throughput: 1406 seconds to sort 100 TB data. I.e. 71.12 GB/Sec

If we consider it is linear scaling and find throughput for 1 node cluster it will give us throughput around 343.59 MB/sec.

CloudSort Benchmark

The paper presents a benchmark called cloudsor t for IO-intensive workloads, that calculates a minimum cost of sorting a fixed number of records for any public cloud. It is a total-cost-of-ownership benchmark based on an external sort. It is better than other benchmarks in terms of Accessibility, Affordability, and Auditability.