

Prog-2_Report

SHARED MEMORY:

1. Problem Description:

- To implement Shared Memory Sort on 10 GB file using multi-threading to take advantage of multi-cores and evaluate its performance in terms of time taken for sorting 10 GB file on Amazon EC2 c3.large instance.
- Shared Memory is the program name for performing sorting in parallel on single machine by sharing memory using multiple threads to take advantage of multiple cores.

2. Methodology:

- The large file of 10 GB was generated using gensort.
- In order to sort a large file efficiently, file was divided into smaller chunks of size 1024 each.
- Now to sort the file, each of these chunks are sorted and then merged back into original file.
- For sorting each chunk, first 10 characters of each line are taken as key and rest of them as value.
- These key value pairs are mapped into TreeMap, which maintains the key value pairs in a sorted manner.
- Once we get sorted data, key and value are merged into line and written back to the file.
- These chunks are then merged using MergeSort to get a sorted version of original 10 GB file.
- To evaluate the performance of shared memory, the program was multi-threaded and results were noted for 1, 2, 4 and 8 threads.

3. Runtime environment setup:

- This experiment was carried out on Amazon AWS.
- Amazon EC2 c3.large instance was launched on spot request and was configured with 10 GB of storage.
- Connection to the remote server was established using FileZilla for file transfer
- All the program files were kept on Amazon EC2 host using FileZilla.
- Instance was connected thru terminal of local host.
- Java environment setup on instance thru terminal was carried out using following commands.
 - Sudo apt-get update
 - Sudo apt-get install openjdk-7-jdk -y
 - Sudo apt-get install ssh -y
- To use ANT, it was downloaded using following command
 - Sudo apt-get install ant
- Now we have environment ready to start execution of our program.

4. Installation steps:

- Start Amazon Aws.
- Launch Amazon Ec2 c3 large instance with on spot request and configure it with 40 GB data storage. EBS volume will be mounted by this.
- Connect your instance with Local host using FileZilla to transfer the required files.
- Put the genSort Folder on Server host thru FileZilla.
- genSort Folder contains script that will download the gensort which will be used to generate 10 GB of data in a File named Input.txt.
- To generate this data, give following command from terminal by setting path cd gensort/64:
`./gensort -a 100000000 Input.txt`.
- Now we have Input.txt 10 GB file. This file will be given as input to shared memory program.
- Put files SharedMemory.java, propertiescontroller.java files which contains the code in a folder named src and build.xml file which is ANT file on server's root.
- Now give command run ant to run the shared memory program.
- The output will be shown on screen i.e. time taken for sorting 10 GB data with various no of threads.
- To check whether the output file generated was sorted or not we run valsor program on output file.
- Run this command to check it:
`./valsor Outputfile.txt`
- The same steps are followed by changing the number of threads in the code and readings are taken.

5. Difficulties:

- The only difficulty faced here was setting up enough memory so that program can run successfully for 10 GB of data.
- Initially as enough memory wasn't configured on instance the program stopped in between.

6. Versions:

- JAVA version: 7
- ANT version: 1.8.2

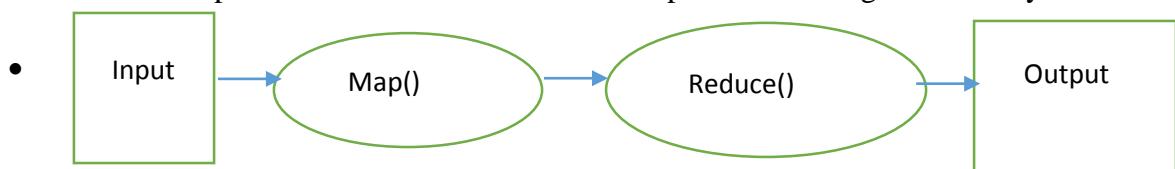
Hadoop:

1. Problem Description:

- Implement Hadoop Sort application, and evaluate its performance on 1 node and 16 nodes.
- Sort is kind of very basic and perfect example to understand the Map/Reduce programming model of Hadoop and its underlying system.
- Hadoop 1 node and 17 node cluster has to set up and then evaluation is to be carried out for 1 GB on 1 node and 10 GB on 17 nodes cluster.

2. Methodology:

- First part is to write Sorting Application for Hadoop.
- Map Reduce programming style of Hadoop consists of 3 main parts:
 - Mapper class: This is an identity class which simply parses each line of input using the TextInputFormat and emits first 10 bytes of each line as key and rest as value.
 - Reducer class: This is also an identity reducer class which simply emits every value for every key.
 - Driver class: This is the main driver program which creates a Hadoop job and configures it.
 - Comparator class: This class sorts the input in ascending order of key.



3. Runtime environment setup:

- First step would be download and setup java on instance.
- To install java use following commands:
 - Sudo apt-get update
 - Sudo apt-get install openjdk-7-jdk -y
 - Sudo apt-get install ssh -y
- Now install Hadoop using the install_hadoop_script.sh
- Allocate enough memory on AmazonEC2 instance around 500 GB.
- Also remember to turn off replication of Master node as replication in Hadoop is done around 3x which needs 3 times more memory.

4. Installation steps:

- Start Amazon AWS.
- Launch Amazon EC2 c3.large instance on spot request and add storage for 1 GB and 100 GB.

FOR 1 NODE VIRTUAL CLUSTER:

- Connect your instance to the local host by following commands on your terminal:
 - Chmod 400 pem filename (E.g. Hadoop.pem)
 - Ssh -i filename.pem ec2_public_dns
- Connect your local host to server host using FileZilla by giving master public DNS for transferring files to the Server.
- Put the script for installing Hadoop and Java on Server Host thru FileZilla.
- Run this script to install Hadoop and Java.
- On installing Hadoop, go to etc folder inside Hadoop folder.
- Change following files: Replace with your public dns in following files
 - Core-site.xml:
 - Mapred-site.xml:
 - Slaves:
 - Yarn-site.xml
- Now generate 10 GB of data using gensort. For this, put gensort folder on server thru FileZilla. Now run the script for downloading and generate 10 gb data file using following command:
 - ./gensort –a 1000000000 Input.txt.
- Now change entry in host file of your master. Replace the private ip and public dns of local host with that of your master node. For this perform following steps:
 - Cd /etc
 - Sudo nano hosts
 - Now edit this file as mentioned above.
- The cluster setup is done. You can now start the nodes on Hadoop and put file in the HDFS with following commands:
 - Cd Hadoop-2.7/bin
 - ./Hadoop namenode –format
 - Cd Hadoop-2.7/sbin
 - ./start-all.sh
 - Cd Hadoop-2.7/bin
 - ./Hadoop fs –put /Input.txt /src
 - ./Hadoop dfsadmin –safemode leave
 - Cd Hadoop-2.7
 - Bin/Hadoop jar ~/Hadoop_Sort /src Output_Files will start the sorting application.
- To get output files on local disk perform following commands:
 - Bin/Hadoop fs –get Output_Files /output

- Now go to output:
 - Cd /output
 - Head -10 part-r-00000
 - Tail -10 part-r-00000
- Here for 10 GB we get only 1 output file name part-r-00000 as we have used one reducer.
- To check if file was sorted give following command from gensort folder:
 - ./valsrt ~/output/part-r-00000

FOR 17 NODE VIRTUAL CLUSTER:

- Connect your instance to the local host by following commands on your terminal:
 - Chmod 400 pem filename (E.g. Hadoop.pem)
 - Ssh -i filename.pem ec2_public_dns
- Connect your local host to server host using FileZilla by giving master public DNS for transferring files to the Server.
- Put the script for installing Hadoop and Java on Server Host thru FileZilla.
- Run this script to install Hadoop and Java.
- On installing Hadoop, go to etc folder inside Hadoop folder.
- Change following files: Replace with your public dns in following files
 - Core-site.xml:
 - Mapred-site.xml:
 - Slaves:
 - Yarn-site.xml
- Create an AMI image of this instance and launch 16 instances for this image same way as we created an instance in the beginning.
- Now add the public DNS of all the slaves in the Hadoop/etc/slaves.xml file.
- Now generate 100 GB of data using gensort. For this, put gensort folder on server thru FileZilla. Now run the script for downloading and generate 100 gb data file using following command:
 - ./gensort -a 1000000000 Input.txt
- Now change entry in host file of your master. Replace the private ip and public dns of local host with that of your master node as well as of all 16 slaves. And for each of 16 slaves host file add private n public ip of that node and master. For this perform following steps:
 - Cd /etc
 - Sudo nano hosts
 - Now edit this file as mentioned above.
- A software named terminator can be used to open multiple terminals of 17 instances and broadcast instructions for all the instances.
- To bind master with slaves an RSA is generated and all 17 instances are binded with that common key.

- To create key give following command:
 - Ssh-keygen -t rsa
- To bind them together give following commands:
 - Ssh-copy-id -i ~/.ssh/id_rsa.pub ec2_master@ip
 - Ssh-copy-id -i ~/.ssh/id_rsa.pub ec2_slave1@ip
 - Ssh-copy-id -i ~/.ssh/id_rsa.pub ec2_slave16@ip
- The cluster setup is done. You can now start the nodes on Hadoop and put file in the HDFS with following commands:
 - Cd Hadoop-2.7/bin
 - ./Hadoop namenode –format
 - Cd Hadoop-2.7/sbin
 - ./start-all.sh
 - Cd Hadoop-2.7/bin
 - ./Hadoop fs –put /Input.txt /src
 - ./Hadoop dfsadmin –safemode leave
 - Cd Hadoop-2.7
 - Bin/Hadoop jar ~/Hadoop_Sort /src Output_Files will start the sorting application.
- To get output files on local disk perform following commands:
 - Bin/Hadoop fs –get Output_Files /output
- Now go to output:
 - Cd /output
 - Head -10 part-r-00000
 - Tail -10 part-r-00039
- Here for 100 GB we get 40 output files named from part-r-00000 to part-r-00039.
- So we have to concatenate all these parts to get original sorted file:
 - Cat * > /output/*
- To check if file was sorted give following command from gensor sort folder:
 - ./valsort ~/output/Output.txt

5. Difficulties:

- The most challenging part here was changing all the entries in host files of 17 instances and configuring the 4 files of Hadoop/etc.
- The cluster didn't start once as I had made a mistake in the host files.
- Also, if enough memory isn't provided the sorting couldn't be performed.
- Initially I had given only 100 GB on instance but got error in between that not enough memory as DFS needs more space while sorting.

6. Versions:

- **Hadoop:** 2.7.2
- **JAVA:** 7

FOLLOWING MODIFICATIONS WERE MADE TO FILES FOR MULTIPLE NODES:

- i. **Slaves.xml:**

Added 17 entries for all 17 instances- 1 master n 17 slaves public DNS.

ii. Core-site.xml:

Add following property in this file.

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/chaud/hadoop</value>
  <description>base location for other hdfs directories.</description>
</property>
```

iii. Hdfs-site.xml:

No change as such

iv. Mapred-site.xml: Add following property

```
<property>
  <name>mapreduce.job.reduces</name>
  <value>40</value>
</property>
```

v. Yarn-site.xml: Add following property

```
<property>
  <name>yarn.nodemanager.disk-health-checker.max-disk-utilization-per-
disk-percentage</name>
  <value>98.5</value>
  <description>checks all the datanodes until 98.5 percent of the disk
utilization has reached</description>
</property>
```

⊕ What is master node ? What is slave node ?

- Master nodes oversee the following key operations that comprise Hadoop: storing data in the Hadoop Distributed File System (HDFS) and running parallel computations on that data using MapReduce. The NameNode coordinates the data storage function (with the HDFS), while the JobTracker oversees and coordinates the parallel processing of data using MapReduce.
- Worker nodes make up the majority of virtual machines and perform the job of storing the data and running computations. Each worker node runs both a DataNode and TaskTracker service that communicates with, and receives instructions from their master nodes. The TaskTracker service is subordinate to the JobTracker, and the DataNode service is subordinate to the NameNode.

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

- The configuration files contain ip address with different port numbers because they are assigned for different services such as resource manager as resource tracker, webapp, admin, scheduler.

- Suppose we assign same port number for all the services then each of these services won't be able to work properly as the port number will be already occupied and so that service won't be able to start on that port.

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

- We can change number of mappers and reducers from mapred-site file by adding following property.

```
<property>
  <name>mapreduce.job.reduces</name>
  <value>40</value>
</property>
```

SPARK:

1. Problem Description:

- Implement Spark Sort Application. Setup cluster of 1 node and 17 nodes for 10 GB and 100 GB respectively Perform the sort on 10 GB and 100 GB for 1 node and 17 nodes on Spark.
- Apache Spark is a lightning-fast cluster computing designed for fast computation. It was built on top of Hadoop MapReduce and it extends the MapReduce model to efficiently use more types of computations.

2. Methodology:

- At the core of Spark is the notion of a **Resilient Distributed Dataset** (RDD), which is an immutable collection of objects that is partitioned and distributed across multiple physical nodes of a YARN cluster and that can be operated in parallel.
- Typically, RDDs are instantiated by loading data from a shared filesystem, HDFS, HBase, or any data source offering a Hadoop InputFormat on a YARN cluster.
- Sort Application for spark is developed in Scala which comes from Scalable Language.
- As sort uses HDFS for storing files, first the file is taken from HDFS and loaded into memory of spark.
- Then mapping is done in terms of key and values pairs.
- Then Sorting is performed on keys.
- After sorting, the lines are written to output file.
- Timing is noted for the sorting done on 10 GB and 100 GB data.

3. Runtime Environment Setup:

- To setup environment for spark, first we have to download spark with its latest version.
- Compared to Hadoop, configuration as well as setup for spark is very easy and less complicated.
- Once spark is downloaded follow the steps mentioned in Installation steps and you're done with setup.

4. Installation Steps:

1 NODE VIRTUAL CLUSTER

- First login to Amazon AWS account.
- Get your AWS Access key and Secret Access Key id in a safe place.
- Also, store your pem file in a safe place.
- To create an instance from terminal, follow following commands:
 - export AWS_ACCESS_KEY_ID=...
 - export AWS_SECRET_ACCESS_KEY=...
 - chmod 400 Spark.pem
- Go to the folder where you downloaded the spark
 - cd/spark.../ec2
- Launch spark using following command.
 - ./spark-ec2 -k Spark -i /home/anuradha/Desktop/Spark/Spark.pem -s 1 -t c3.large --spot-price=0.15 --ebs-vol-size=50 launch spark
- Login to spark using following command.
 - ./spark-ec2 -k Spark -i /home/anuradha/Desktop/Spark/Spark.pem login spark
- Generate 10 GB file using following command.
 - ./gensort -a 10000000 /achaud/Input
- Put data into HDFS by this command.
 - ./hadoop fs -Ddfs.replication=1 -put /vol0/Input /root/ephemeral-hdfs/bin/Input
- Sort the data using following command.
 - ./spark-shell -i /root/sort.scala
- Now to get the output file which is currently in HDFS use following commands.
 - cd ephemeral-hdfs/bin
 - ./hadoop fs -copyToLocal /root /root/output
- For 10 GB 7 parts will be created.
 - head -10 part-r-00000
 - tail -10 part-r-00007

17 NODE VIRTUAL CLUSTER:

- For 17 node cluster same steps are to be followed except you have to now generate 100 GB of data using gensort and while launching give following command:
 - ./spark-ec2 -k Spark -i /home/anuradha/Desktop/Spark/Spark.pem -s 16 -t c3.large --spot-price=0.15 --ebs-vol-size=500 launch spark

5. Difficulties:

- Due to insufficient memory, program stopped. Then mounted ebs volume to provide enough memory.

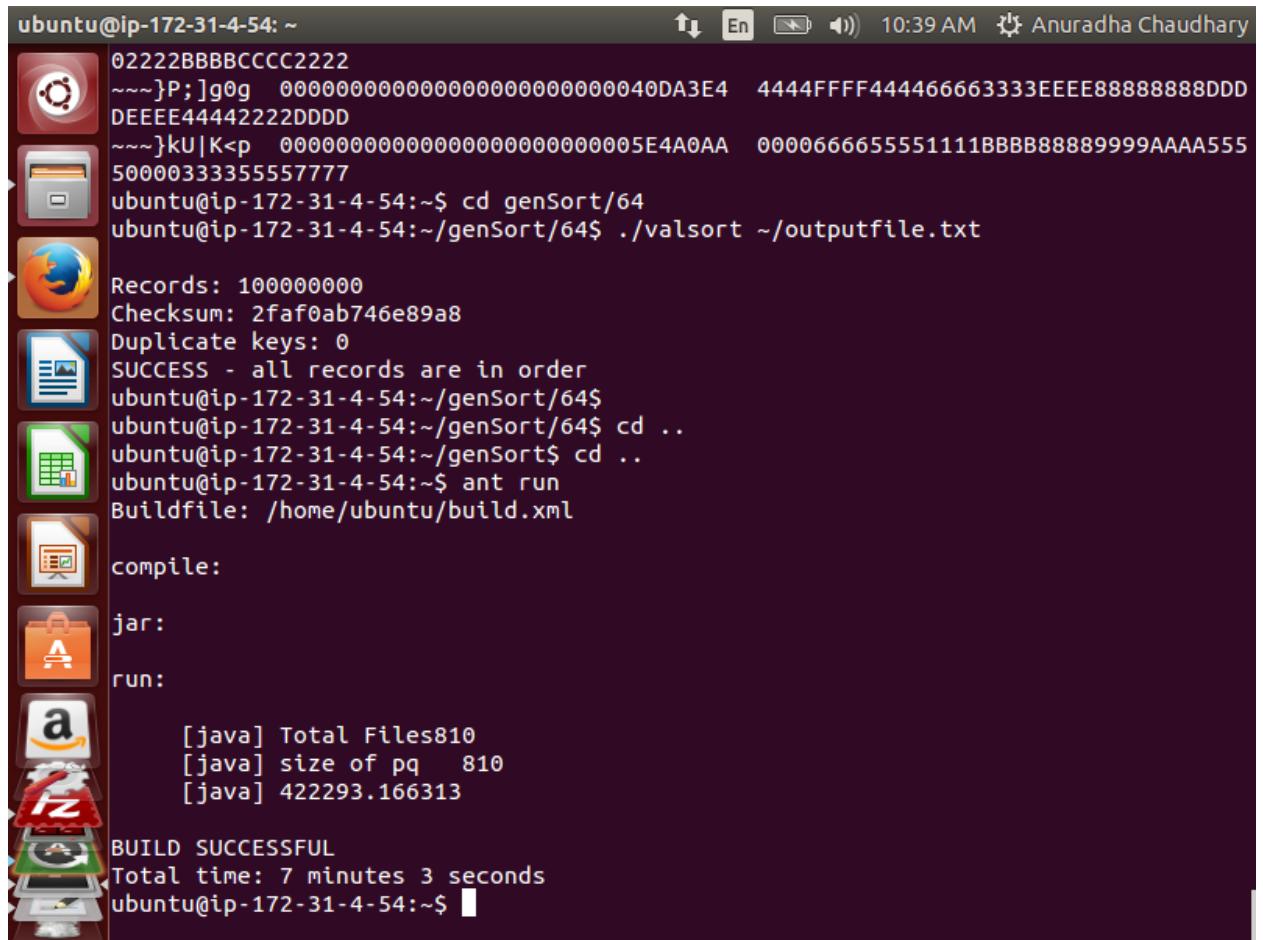
6. Versions:

- **JAVA: 7**
- **SPARK: 1.6.1**

PERFORMANCE:

1. SHARED MEMORY:

A. Time taken for sorting with 1 thread



The screenshot shows a terminal window on an Ubuntu desktop environment. The terminal output is as follows:

```
ubuntu@ip-172-31-4-54: ~
02222BBBBCCCC2222
~~~}P; ]g0g 0000000000000000000000000000000040DA3E4 4444FFFF444466663333EEEE88888888DDD
DEEEE44442222DDDD
~~~}kU|K<p 000000000000000000000000000000005E4A0AA 0000666655551111BBBB88889999AAAA555
50000333355557777
ubuntu@ip-172-31-4-54:~$ cd genSort/64
ubuntu@ip-172-31-4-54:~/genSort/64$ ./valsort ~/outputfile.txt

Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-4-54:~/genSort/64$
ubuntu@ip-172-31-4-54:~/genSort/64$ cd ..
ubuntu@ip-172-31-4-54:~/genSort$ cd ..
ubuntu@ip-172-31-4-54:~$ ant run
Buildfile: /home/ubuntu/build.xml

compile:

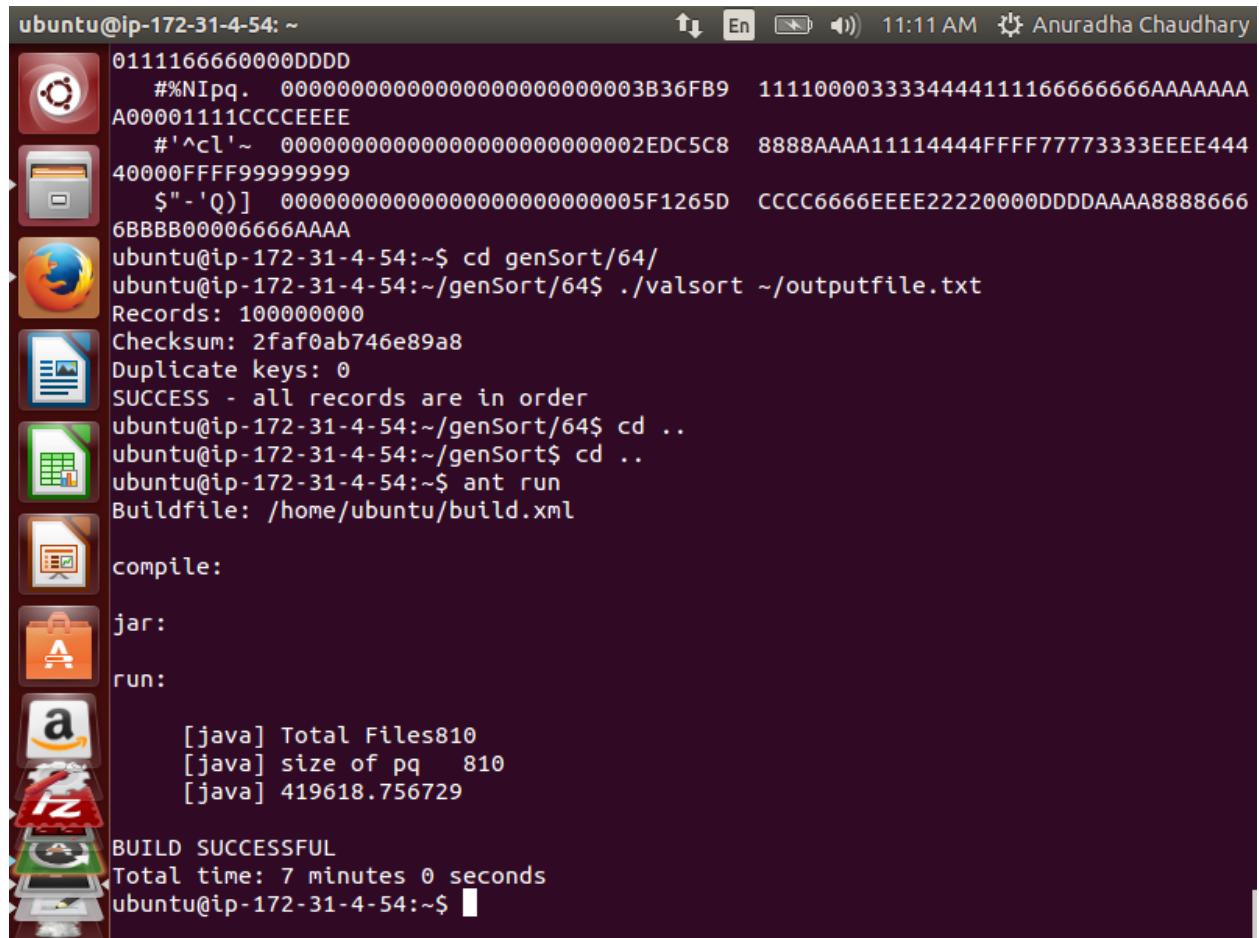
jar:

run:

[java] Total Files810
[java] size of pq 810
[java] 422293.166313

BUILD SUCCESSFUL
Total time: 7 minutes 3 seconds
ubuntu@ip-172-31-4-54:~$
```

B. Time taken for sorting with 2 threads:



ubuntu@ip-172-31-4-54: ~

01111666600000DDDD
#%NIpq. 00000000000000000000000000000003B36FB9 1111000033334444111166666666AAAAAA
A00001111CCCCEEE
#'^cl'~ 00000000000000000000000000000002EDC5C8 8888AAAA11114444FFFF77773333EEEE444
40000FFFF99999999
\$"- 'Q)] 00000000000000000000000000000005F1265D CCCC6666EEEE22220000DDDDAAAA8888666
6BBBBB00006666AAAA
ubuntu@ip-172-31-4-54:~\$ cd genSort/64/
ubuntu@ip-172-31-4-54:~/genSort/64\$./valsrt ~/outputfile.txt
Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-4-54:~/genSort/64\$ cd ..
ubuntu@ip-172-31-4-54:~/genSort\$ cd ..
ubuntu@ip-172-31-4-54:~/genSort\$ ant run
Buildfile: /home/ubuntu/build.xml

compile:

jar:

run:

[java] Total Files810
[java] size of pq 810
[java] 419618.756729

BUILD SUCCESSFUL
Total time: 7 minutes 0 seconds
ubuntu@ip-172-31-4-54:~\$ █

C. Time taken for sorting 4 threads:

```
ubuntu@ip-172-31-4-54: ~          En  10:11 AM  Anuradha Chaudhary
64/valsort
64/gensort
./install-gensort.sh: line 3: cd: /root/64: Permission denied
./install-gensort.sh: line 4: ./gensort: No such file or directory
mv: cannot stat 'inputfile.txt': No such file or directory
ubuntu@ip-172-31-4-54:~/genSort$ cd 64
ubuntu@ip-172-31-4-54:~/genSort/64$ ./gensort -a 100000000 inputfile.txt
ubuntu@ip-172-31-4-54:~/genSort/64$ ant run
Buildfile: build.xml does not exist!
Build failed
ubuntu@ip-172-31-4-54:~/genSort/64$ cd ..
ubuntu@ip-172-31-4-54:~/genSort$ cd ..
ubuntu@ip-172-31-4-54:~$ ant run
Buildfile: /home/ubuntu/build.xml

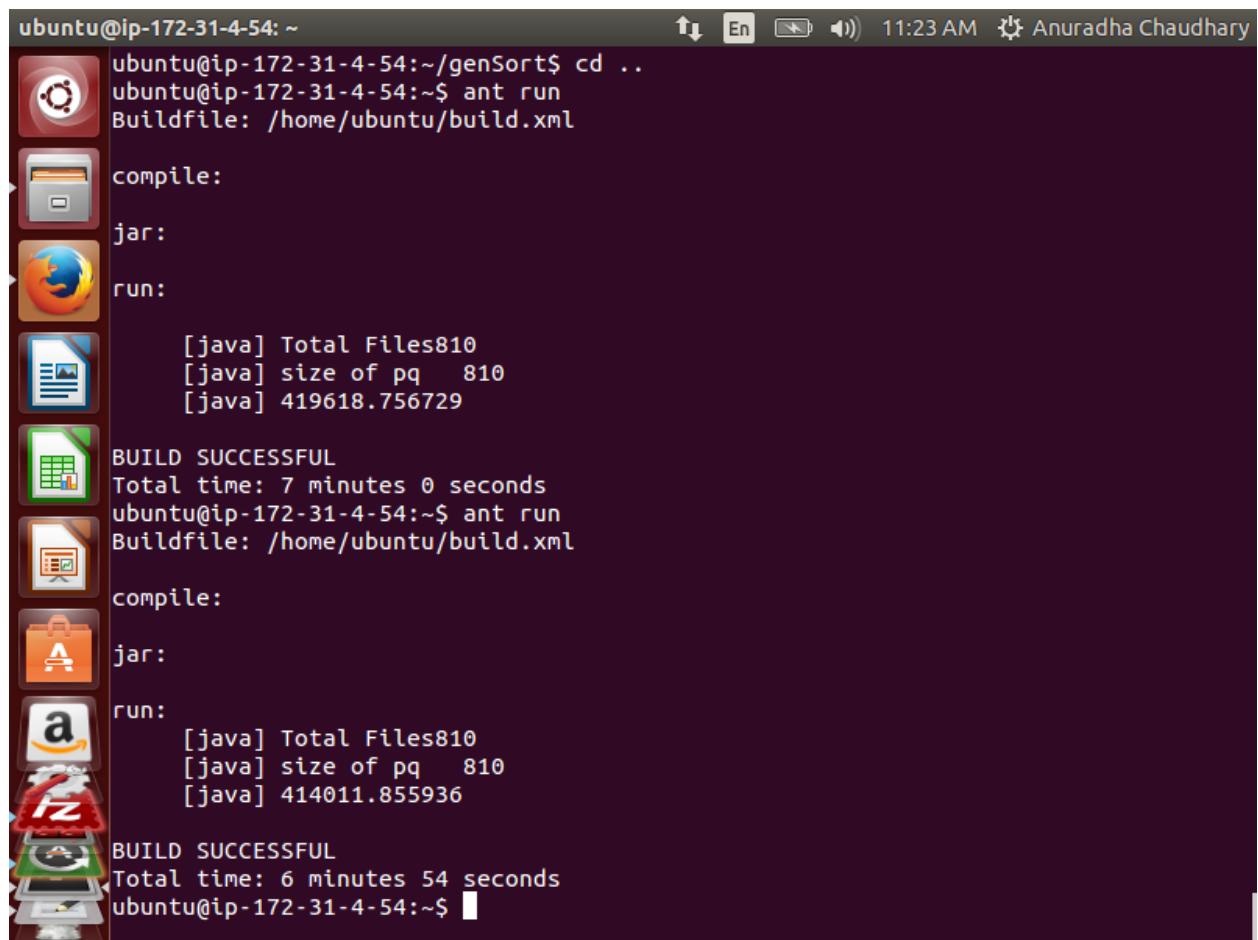
compile:
[mkdir] Created dir: /home/ubuntu/build/classes
[javac] Compiling 5 source files to /home/ubuntu/build/classes

jar:
[mkdir] Created dir: /home/ubuntu/build/jar
[jar] Building jar: /home/ubuntu/build/jar/SharedMemory.jar

run:
[java] Total Files810
[java] size of pq 810
[java] 415493.204899

BUILD SUCCESSFUL
Total time: 6 minutes 57 seconds
ubuntu@ip-172-31-4-54:~$
```

D. Time taken for sorting with 8 threads:



The screenshot shows a terminal window on an Ubuntu desktop. The terminal output is as follows:

```
ubuntu@ip-172-31-4-54: ~
ubuntu@ip-172-31-4-54:~/genSort$ cd ..
ubuntu@ip-172-31-4-54:~$ ant run
Buildfile: /home/ubuntu/build.xml

compile:
jar:
run:
[java] Total Files810
[java] size of pq 810
[java] 419618.756729

BUILD SUCCESSFUL
Total time: 7 minutes 0 seconds
ubuntu@ip-172-31-4-54:~$ ant run
Buildfile: /home/ubuntu/build.xml

compile:
jar:
run:
[java] Total Files810
[java] size of pq 810
[java] 414011.855936

BUILD SUCCESSFUL
Total time: 6 minutes 54 seconds
ubuntu@ip-172-31-4-54:~$
```

As it can be seen from above screenshots, we get least time for sorting 10 GB of data when 8 threads are running in parallel.

E. Output of valsrt:

```
ubuntu@ip-172-31-4-54: ~/genSort/64
6BBBB00006666AAAA
ubuntu@ip-172-31-4-54:~$ tail -10 outputfile.txt
~~~uq2k#=U 00000000000000000000000000000000C06745 99991111DDDD22221110000FFFFEEEEFFF
F33337777CCCC2222
~~~v/0&Qnm 000000000000000000000000000000004709701 CCCC88883333FFFF0000000000009999111
1FFFFF777744446666
~~~yK0l:gE 000000000000000000000000000000002048B4F CCCC11114444888822226666BBBB8888555
57777EEEEBBBB0000
~~~yK^H.il 00000000000000000000000000000000463D004 44440000FFFF3333999944447777DDDDFFF
FAAAA11118888DDDD
~~~yL;C'XE 000000000000000000000000000000005B0D211 2222EEEE3333000022221111CCCCFFFF555
577774444BBBBB6666
~~~zbA_ Tt 000000000000000000000000000000007F9F4F BBBBCCCC666655559999FFFF8888AAAA111
16666AAAABBBB0000
~~~ze0^FEg 00000000000000000000000000000001E06130 4444CCCCBBBB99992222888855558888CCC
CFFFF000011111111
~~~}GxjWHI 00000000000000000000000000000000CA1345 777711118888AAAAAAA22221111BBBB000
02222BBBBCCCC2222
~~~}P;]g0g 0000000000000000000000000000000040DA3E4 4444FFFF444466663333EEE88888888DDD
DEEEE44442222DDDD
~~~}kU|K<p 00000000000000000000000000000005E4A0AA 0000666655551111BBBB88889999AAAA555
50000333355557777
ubuntu@ip-172-31-4-54:~$ cd genSort/64
ubuntu@ip-172-31-4-54:~/genSort/64$ ./valsrt ~/outputfile.txt

Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-4-54:~/genSort/64$
ubuntu@ip-172-31-4-54:~/genSort/64$
```

F. First 10 lines of sort file:

```
ubuntu@ip-172-31-4-54: ~          En  10:21 AM  Anuradha Chaudhary
[ java] 415493.204899
BUILD SUCCESSFUL
Total time: 6 minutes 57 seconds
ubuntu@ip-172-31-4-54:~$ unix2dos outputfile.txt
unix2dos: converting file outputfile.txt to DOS format ...
ubuntu@ip-172-31-4-54:~$ head -10 outputfile.txt
"0!uve 000000000000000000000000000000001228D4 77778888000022224444DDDDDDDEEEE000
00000CCCC7777DDDD
PMd32= 000000000000000000000000000003440CC1 FFFFEEEE6666CCCCBBBB99993335555DDD
DDDDDD777788886666
^3C0], 00000000000000000000000000000158C5C5 5555AAAA9999EEEE888822229999CCCCDDD
D666655554442222
!&S3/] ] 000000000000000000000000000002145D78 8888BBBBDDDD1111CCCC55556666BBBB111
1EEEEEDDDD22229999
!,=U#,9 0000000000000000000000000000019072E3 33332222FFFFBBBB0000FFFFAAAA6666555
53333DDDD3333CCCC
!0f[ITd 000000000000000000000000000003CAAB4B 9999FFFF55553337777CCCC4444BBBB777
7EEEEBBBBDDDD4444
!f6Suy2 000000000000000000000000000003ABFD84 EEEE55555556666AAAA5555BBBBDDDD000
0111166660000DDDD
#%NIPq. 000000000000000000000000000003B36FB9 111100003333444411116666666AAAAAAA
A00001111CCCCEEE
#'^cl'~ 000000000000000000000000000002EDC5C8 8888AAAA11114444FFFF7777333EEEE444
40000FFFF9999999
$"-Q)] 000000000000000000000000000005F1265D CCCC6666EEEE22220000DDDAAAA8888666
6BBBBB00006666AAAA
ubuntu@ip-172-31-4-54:~$ tail -10 outputfile.txt
~~~uq2k#=U 0000000000000000000000000002C06745 99991111DDDD22221110000FFFFEEEEFFF
```

G. Last 10 lines of a sorted file:

```
ubuntu@ip-172-31-4-54: ~
7EEEEBBBBDDDD4444
!f6Suy2 00000000000000000000000000000003ABFD84 EEEE55555556666AAAA5555BBBBDDDD000
0111166660000DDDD
#%NIpq. 000000000000000000000000000003B36FB9 111100003333444411116666666AAAAAAA
A0000111CCCCEEE
#'^cl'~ 000000000000000000000000000002EDC5C8 8888AAAA11114444FFFF77773333EEEE444
40000FFFF9999999
$"-Q)] 000000000000000000000000000005F1265D CCCC6666EEEE22220000DDDAAAA8888666
6BBBBB00006666AAAA
ubuntu@ip-172-31-4-54:~$ tail -10 outputfile.txt
~~~uq2k#=U 000000000000000000000000000002C06745 99991111DDDD222211110000FFFFEEEEFFF
F33337777CCCC2222
~~~v/0&Qnm 000000000000000000000000000004709701 CCCC88883333FFFF000000000009999111
1FFFFF777744446666
~~~yK0l:gE 000000000000000000000000000002048B4F CCCC11114444888822226666BBBB8888555
57777EEEEBBBB0000
~~~yK^H.il 00000000000000000000000000000463D004 44440000FFFF3333999944447777DDDDFFF
FAAAA11118888DDDD
~~~yL;C'XE 000000000000000000000000000005B0D211 2222EEEE3333000022221111CCCCFFFF555
577774444BBBB6666
~~~zbA_ Tt 00000000000000000000000000000007F9F4F BBBBCCCC666655559999FFFF8888AAAA111
16666AAAABBBB0000
~~~ze0^FEg 000000000000000000000000000001E06130 4444CCCCBBBB99992222888855558888CCC
CFFFF00001111111
~~~}GxjWH! 00000000000000000000000000000CA1345 777711118888AAAAAAA22221111BBBB000
02222BBBBCCCC2222
~~~}P;]g0g 0000000000000000000000000000040DA3E4 4444FFFF444466663333EEEE88888888DDD
DEEEE44442222DDDD
~~~}kU|K<p 000000000000000000000000000005E4A0AA 0000666655551111BBBB88889999AAAA555
50000333355557777
ubuntu@ip-172-31-4-54:~$
```

2. HADOOP:

A. Namenode started for sorting 10 GB of data on 1 Node Virtual cluster.

The screenshot shows a Mozilla Firefox browser window titled "Namenode information - Mozilla Firefox". The address bar displays the URL "46.eu-west-1.compute.amazonaws.com:5007". The main content area is titled "Overview" and shows the IP address "ec2-52-30-191-146.eu-west-1.compute.amazonaws.com:9000" in parentheses. Below this, there is a table with the following data:

Started:	Sun Mar 27 17:39:20 UTC 2016
Version:	2.7.2, rb165c4fe8a74265c792ce23f546c64604acf0e41
Compiled:	2016-01-26T00:08Z by jenkins from (detached from b165c4f)
Cluster ID:	CID-1875c3ac-7b99-4d6b-8db9-e559500b102b
Block Pool ID:	BP-716632415-172.31.6.207-1459100293376

Below the table, the section "Summary" is visible, containing the following information:

- Security is off.
- Safemode is off.
- 20 files and directories, 153 blocks = 173 total filesystem object(s).
- Heap Memory used 119.48 MB of 165.5 MB Heap Memory. Max Heap Memory is 889 MB.
- Non Heap Memory used 33.79 MB of 34.94 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity:	62.86 GB
DFS Used:	18.77 GB (29.87%)
Non DFS Used:	13.67 GB

B. Namenode started for sorting 10 GB of data on 1 Node Virtual cluster.

Namenode information - Mozilla Firefox

EC2 Managem... HDFS fetch co... Namenode infor... All Applications

46.eu-west-1.compute.amazonaws.com:5007 Google

Summary

Security is off.
Safemode is off.
20 files and directories, 153 blocks = 173 total filesystem object(s).
Heap Memory used 119.48 MB of 165.5 MB Heap Memory. Max Heap Memory is 889 MB.
Non Heap Memory used 33.79 MB of 34.94 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity:	62.86 GB
DFS Used:	18.77 GB (29.87%)
Non DFS Used:	13.67 GB
DFS Remaining:	30.41 GB (48.38%)
Block Pool Used:	18.77 GB (29.87%)
DataNodes usages% (Min/Median/Max/stdDev):	29.87% / 29.87% / 29.87% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	75
Number of Blocks Pending Deletion	0
Block Deletion Start Time	3/27/2016, 12:39:20 PM

C. Namenode started for sorting 10 GB of data on 1 Node Virtual cluster.

Namenode information - Mozilla Firefox

EC2 Managem... x HDFS fetch co... x Namenode infor... x All Applications x +

46.eu-west-1.compute.amazonaws.com:5007 Google

Dead Nodes 0 (Decommissioned: 0)
Decommissioning Nodes 0
Total Datanode Volume Failures 0 (0 B)
Number of Under-Replicated Blocks 75
Number of Blocks Pending Deletion 0
Block Deletion Start Time 3/27/2016, 12:39:20 PM

NameNode Journal Status

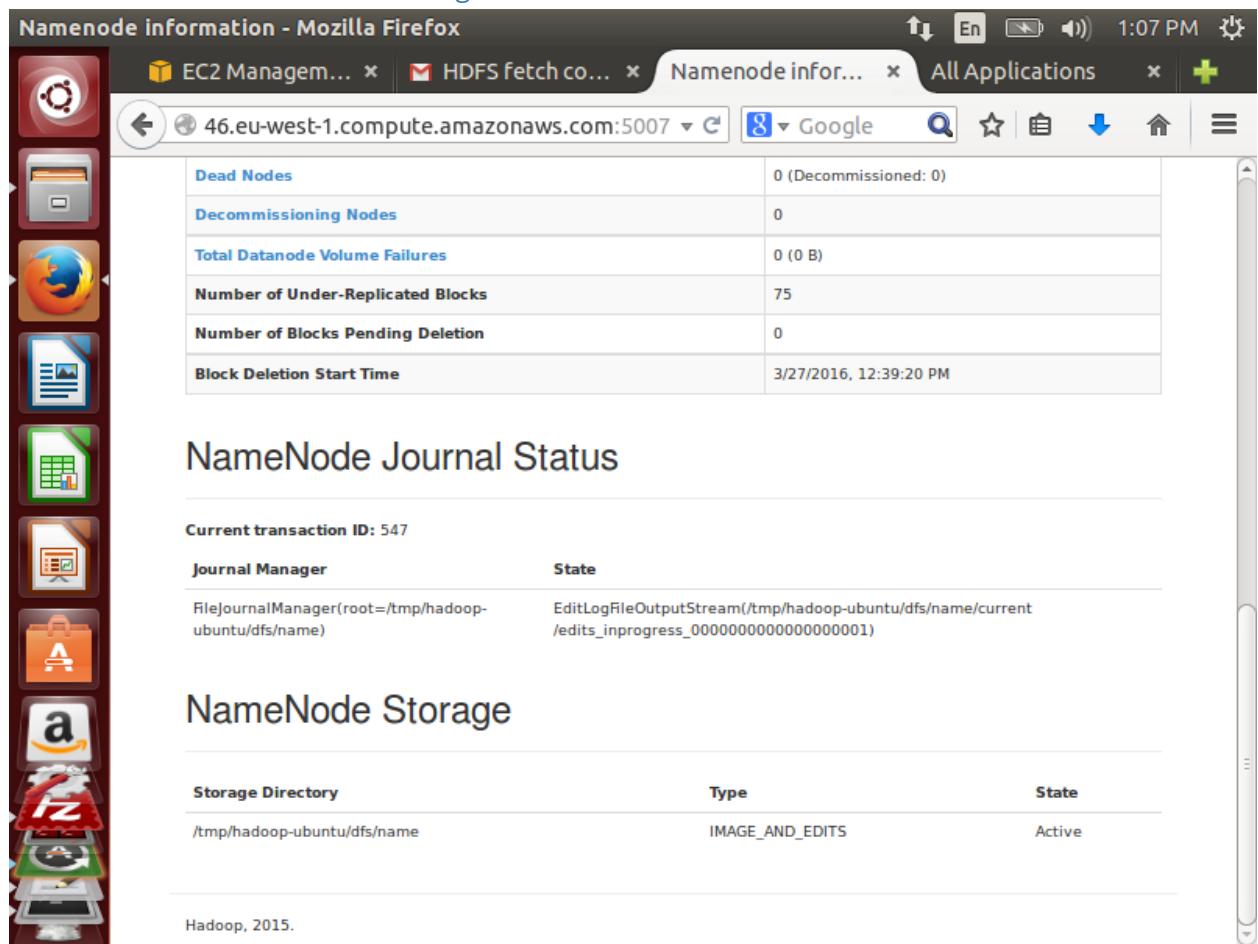
Current transaction ID: 547

Journal Manager	State
FileJournalManager(root=/tmp/hadoop-ubuntu/dfs/name)	EditLogFileOutputStream(/tmp/hadoop-ubuntu/dfs/name/current /edits_inprogress_0000000000000001)

NameNode Storage

Storage Directory	Type	State
/tmp/hadoop-ubuntu/dfs/name	IMAGE_AND_EDITS	Active

Hadoop, 2015.



D. Application status on job completion

The screenshot shows the Hadoop application status page within a Firefox browser. The URL in the address bar is 46.eu-west-1.compute.amazonaws.com:9014. The main content area is titled "All Applications". It displays two tables: "Cluster Metrics" and "Scheduler Metrics".

Cluster Metrics

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

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Below these tables is a search and filter interface:

- Show: 20 entries
- Search: (empty)
- Columns: ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, Tracking UI, Blacklisted Nodes.

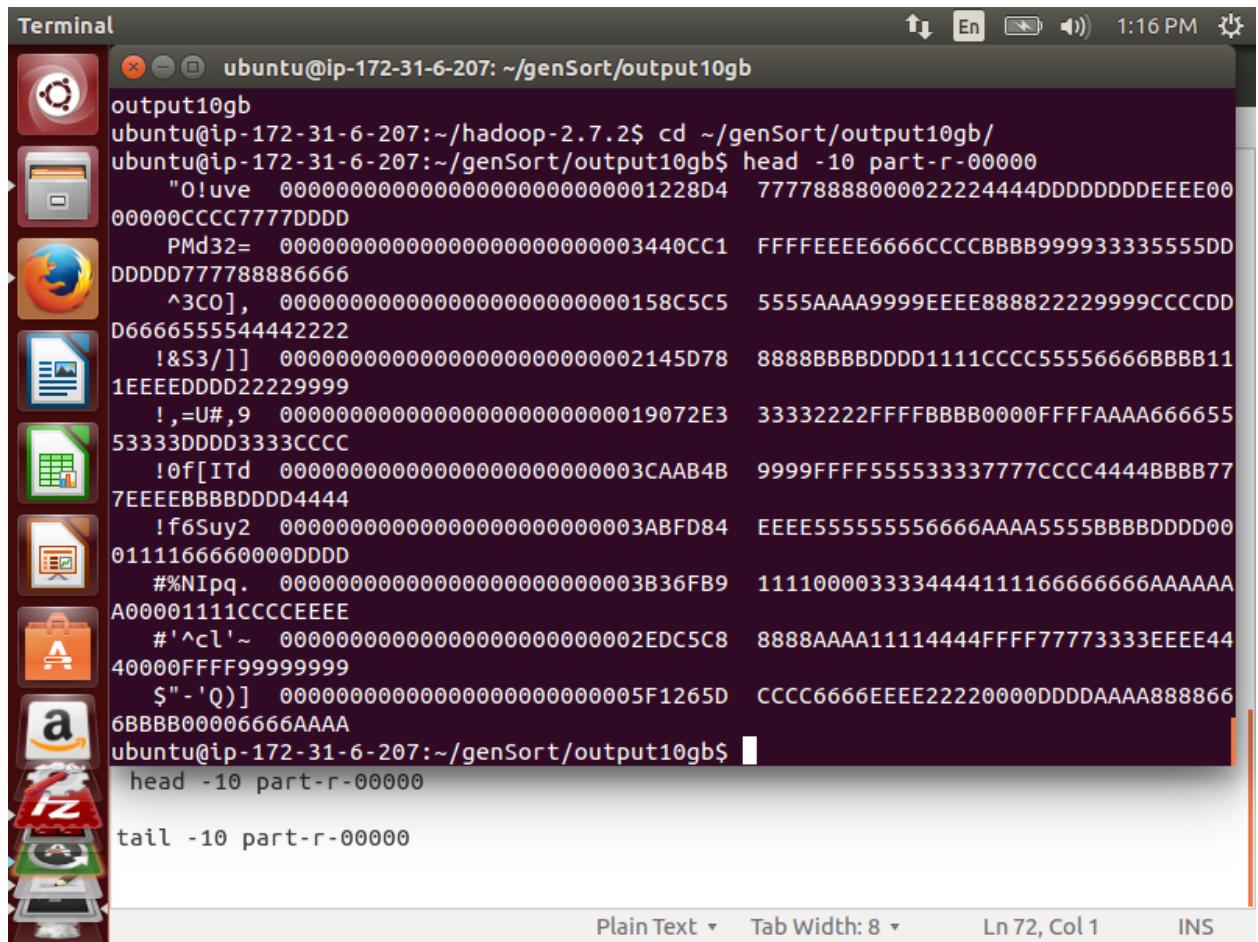
A single entry is listed in the table:

ID	User	Name	Application Type	Queue	Start Time	Finish Time	State	Final Status	Progress	Tracking UI	Blacklisted Nodes
application_1459100381153_0001	ubuntu	Hadoop_10gigaSort	MAPREDUCE	default	Sun Mar 27 12:43:47 -0500 2016	Sun Mar 27 13:04:12 -0500 2016	FINISHED	SUCCEEDED	N/A	History	N/A

At the bottom, it says "Showing 1 to 1 of 1 entries" and provides navigation links: First, Previous, 1, Next, Last.

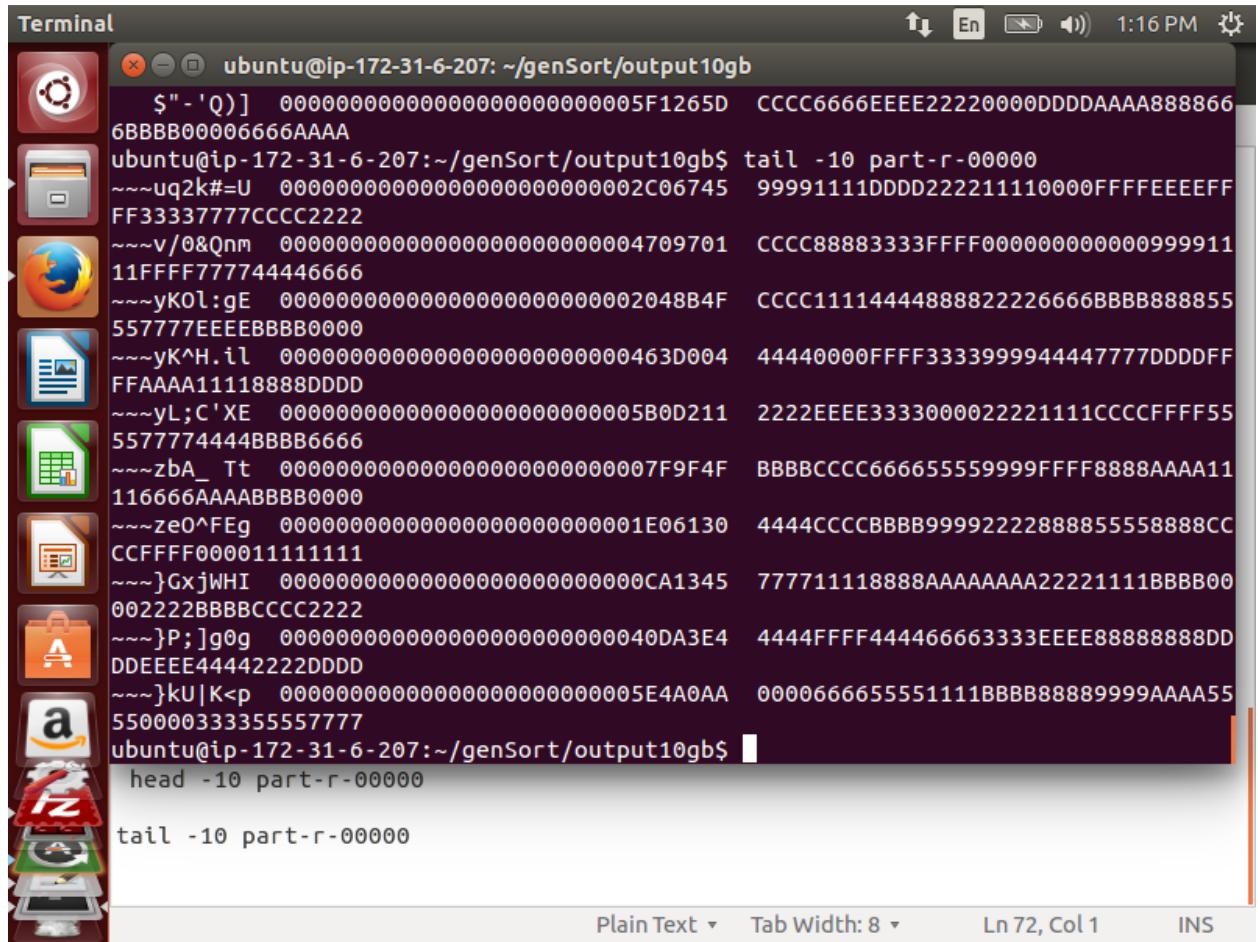
Here, time taken by Hadoop to sort 10 gb file is about 20 mins.

E. First 10 lines of sorted data:

A screenshot of an Ubuntu desktop environment. In the foreground, a terminal window titled "Terminal" is open, showing the command-line interface. The terminal window has a dark background with white text. At the top of the window, there is a title bar with the text "ubuntu@ip-172-31-6-207: ~/genSort/output10gb". Below the title bar, there is a toolbar with several icons, including a file browser, a terminal icon, and a power icon. The main area of the terminal window contains the output of a command. The command "head -10 part-r-00000" was run to display the first 10 lines of a file named "part-r-00000". The output consists of 10 lines of binary data, each line containing approximately 100 characters of hex digits. The terminal window is positioned in the center of the screen, with other desktop icons visible in the background.

```
ubuntu@ip-172-31-6-207:~/genSort/output10gb
output10gb
ubuntu@ip-172-31-6-207:~/hadoop-2.7.2$ cd ~/genSort/output10gb/
ubuntu@ip-172-31-6-207:~/genSort/output10gb$ head -10 part-r-00000
    "0!uve 000000000000000000000000000000001228D4 7777888800022224444DDDDDDDEEEE00
00000CCCC7777DDDD
    PMD32= 00000000000000000000000000000003440CC1 FFFFEEEE6666CCCCBBBB99993335555DD
DDDDDD777788886666
    ^3CO], 0000000000000000000000000000000158C5C5 5555AAA9999EEE888822229999CCCCDD
D666655554442222
    !&S3/] 00000000000000000000000000000002145D78 8888BBBBDDDD1111CCCC55556666BBBB11
1EEEEEDDDD22229999
    !,=U#,9 000000000000000000000000000000019072E3 33332222FFFFBBBB0000FFFFAAAA666655
53333DDDD3333CCCC
    !0f[ITd 00000000000000000000000000000003CAAB4B 9999FFFF55553337777CCCC4444BBBB77
7EEEEBBBBDDDD4444
    !f6Suy2 00000000000000000000000000000003ABFD84 EEEE55555556666AAA5555BBBBDDDD00
0111166660000DDDD
    #%NIpq. 00000000000000000000000000000003B36FB9 111100003333444411116666666AAAAAA
A00001111CCCIEEEE
    #'^cl'~ 00000000000000000000000000000002EDC5C8 8888AAAA11114444FFFF7777333EEE44
40000FFFF99999999
    $"-'Q)] 00000000000000000000000000000005F1265D CCCC6666EEEE22220000DDDAAAA888866
6BBBBB00006666AAAA
ubuntu@ip-172-31-6-207:~/genSort/output10gb$ head -10 part-r-00000
head -10 part-r-00000
tail -10 part-r-00000
```

F. Last 10 lines of sorted data file

A screenshot of an Ubuntu desktop environment. In the top right corner, there is a system tray with icons for battery, signal strength, and volume. The main focus is a terminal window titled "Terminal" located in the bottom left corner. The terminal window has a dark background and contains the following text:

```
ubuntu@ip-172-31-6-207: ~/genSort/output10gb
$"-Q)] 00000000000000000000000000000005F1265D CCCC6666EEEE22220000DDDDAAAA888866
6BBBBB00006666AAAA
ubuntu@ip-172-31-6-207:~/genSort/output10gb$ tail -10 part-r-00000
~~~uq2k#=U 00000000000000000000000000000002C06745 99991111DDDD222211110000FFFFEEEEFF
FF33337777CCCC2222
~~~v/0&Qnm 00000000000000000000000000000004709701 CCCC88883333FFFF000000000000999911
11FFFF77774446666
~~~yK0l:gE 00000000000000000000000000000002048B4F CCCC11114444888822226666BBBB888855
557777EEE BBBB0000
~~~yK^H.il 0000000000000000000000000000000463D004 44440000FFFF3333999944447777DDDDFF
FFAAAAA11118888DDDD
~~~yL;C'XE 00000000000000000000000000000005B0D211 2222EEEE3333000022221111CCCCFFFF55
5577774444BBBB6666
~~~zbA_ Tt 00000000000000000000000000000007F9F4F BBBBCCCC666655559999FFFF8888AAAA11
116666AAAABBBB0000
~~~ze0^FEg 00000000000000000000000000000001E06130 4444CCCCBBBB99992222888855558888CC
CCFFFF000011111111
~~~}GxjWH! 00000000000000000000000000000000CA1345 777711118888AAAAAAA22221111BBBB00
002222BBBBCCCC2222
~~~}P;]g0g 000000000000000000000000000000040DA3E4 4444FFFF444466663333EEEE88888888DD
DDEEEE44442222DDDD
~~~}kU|K<p 00000000000000000000000000000005E4A0AA 0000666655551111BBBB88889999AAAA55
550000333355557777
ubuntu@ip-172-31-6-207:~/genSort/output10gb$ head -10 part-r-00000
tail -10 part-r-00000
```

The terminal window also shows status indicators at the bottom: "Plain Text", "Tab Width: 8", "Ln 72, Col 1", and "INS".

G. Result of valsrt on 10gb data file

```
ubuntu@ip-172-31-6-207: ~/genSort/64
$"- 'Q)] 000000000000000000000000000000005F1265D CCCC6666EEEE22220000DDDAAAA8888666
6BBBBB00006666AAAA
ubuntu@ip-172-31-6-207:~/genSort/output10gb$ tail -10 part-r-00000
~~~uq2k#=U 00000000000000000000000000000002C06745 99991111DDDD222211110000FFFFEEEEFF
FF33337777CCCC2222
~~~v/0&Qnm 000000000000000000000000000004709701 CCCC88883333FFFF000000000000999911
11FFFF77774446666
~~~yK0l:gE 000000000000000000000000000002048B4F CCCC11114444888822226666BBBB888855
557777EEEEBBBB0000
~~~yK^H.il 00000000000000000000000000000463D004 44440000FFFF3333999944447777DDDDFF
FFAAAAA11118888DDDD
~~~yL;C'XE 000000000000000000000000000005B0D211 2222EEEE3333000022221111CCCCFFFF55
5577774444BBBB6666
~~~zbA_ Tt 00000000000000000000000000000007F9F4F BBBBCCCC666655559999FFFF8888AAAA11
116666AAAABBBA0000
~~~ze0^FEg 000000000000000000000000000001E06130 4444CCCCBBBB99992222888855558888CC
CCFFF00001111111
~~~}GxjWHI 00000000000000000000000000000000CA1345 777711118888AAAAAAA22221111BBBB00
002222BBBBCCCC2222
~~~}P;]g0g 0000000000000000000000000000040DA3E4 4444FFFF444466663333EEEE88888888DD
DDEEEE44442222DDDD
~~~}kU|K<p 000000000000000000000000000005E4A0AA 0000666655551111BBBB88889999AAAA55
550000333355557777
ubuntu@ip-172-31-6-207:~/genSort/output10gb$ cd ..
ubuntu@ip-172-31-6-207:~/genSort$ cd 64
ubuntu@ip-172-31-6-207:~/genSort/64$ ./valsrt ~/genSort/output10gb/part-r-00000
Records: 100000000
Checksum: 2faf0ab746e89a8
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-6-207:~/genSort/64$
```

H. Namenode started for 100 GB of data file with 16 nodes.

Nameno Screenshot

En 3:34 AM Anuradha Chaudhary

Inbox (490) ... Hadoop Fil... EC2 Manag... Namenode i... x Spark Master ... +

91-38-132.compute-1.amazonaws.com:50070 Google

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Take Screenshot

Overview 'ec2-52-91-38-132.compute-1.amazonaws.com:9000' (active)

Grab the whole screen

Started: Tue Mar 29 08:23:15 UTC 2016

Version: 2.7.2, rb165c4fe8a74265c792ce23f546c64604acf0e41

Compiled: 2016-01-26T00:08Z by Jenkins from (detached from b165c4f)

Cluster ID: CID-230d348d-f72f4176-aa78-4764e6d8220c

Block Pool ID: BP-554143833-172.31.13.104-1459239675020

Include pointer

Summary

Security is off. Apply Effect: None

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

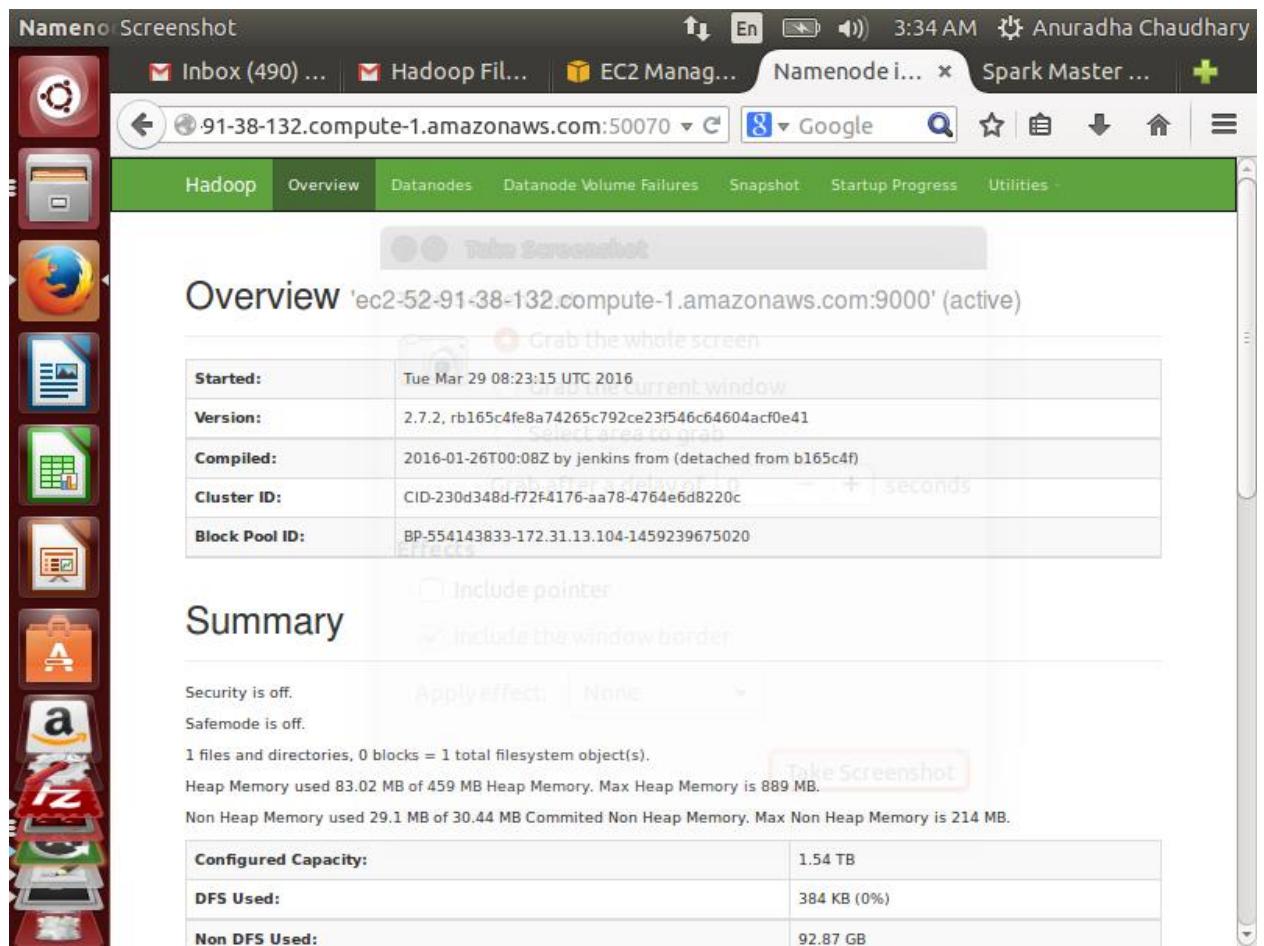
Heap Memory used 83.02 MB of 459 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 29.1 MB of 30.44 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity: 1.54 TB

DFS Used: 384 KB (0%)

Non DFS Used: 92.87 GB



- I. Namenode started for 100 GB of data file with 16 nodes.

The screenshot shows a Mozilla Firefox browser window with the title "Namenode Screenshot - Mozilla Firefox". The address bar displays "91-38-132.compute-1.amazonaws.com:50070". The main content area is divided into several sections:

- HDFS Metrics:**

DFS Remaining:	1.45 TB (94.09%)
Block Pool Used:	384 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	16 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	3/29/2016, 3:23:15 AM
- NameNode Journal Status:**

Current transaction ID: 17

Journal Manager	State
FileJournalManager(root=/achaud/hadoop/dfs/name)	EditLogFileOutputStream(/achaud/hadoop/dfs/name/current/edits_inprogress_00000000000000000017)
- NameNode Storage:**

J. Namenode started for 100 GB of data file with 16 nodes.

Nameno Screenshot

91-38-132.compute-1.amazonaws.com:50070

Dead Nodes: 0 (Decommissioned: 0)
Decommissioning Nodes: 0
Total Datanode Volume Failures: 0 (0 B)
Number of Under-Replicated Blocks: 0
Number of Blocks Pending Deletion: 0
Block Deletion Start Time: 3/29/2016, 3:23:15 AM

NameNode Journal Status

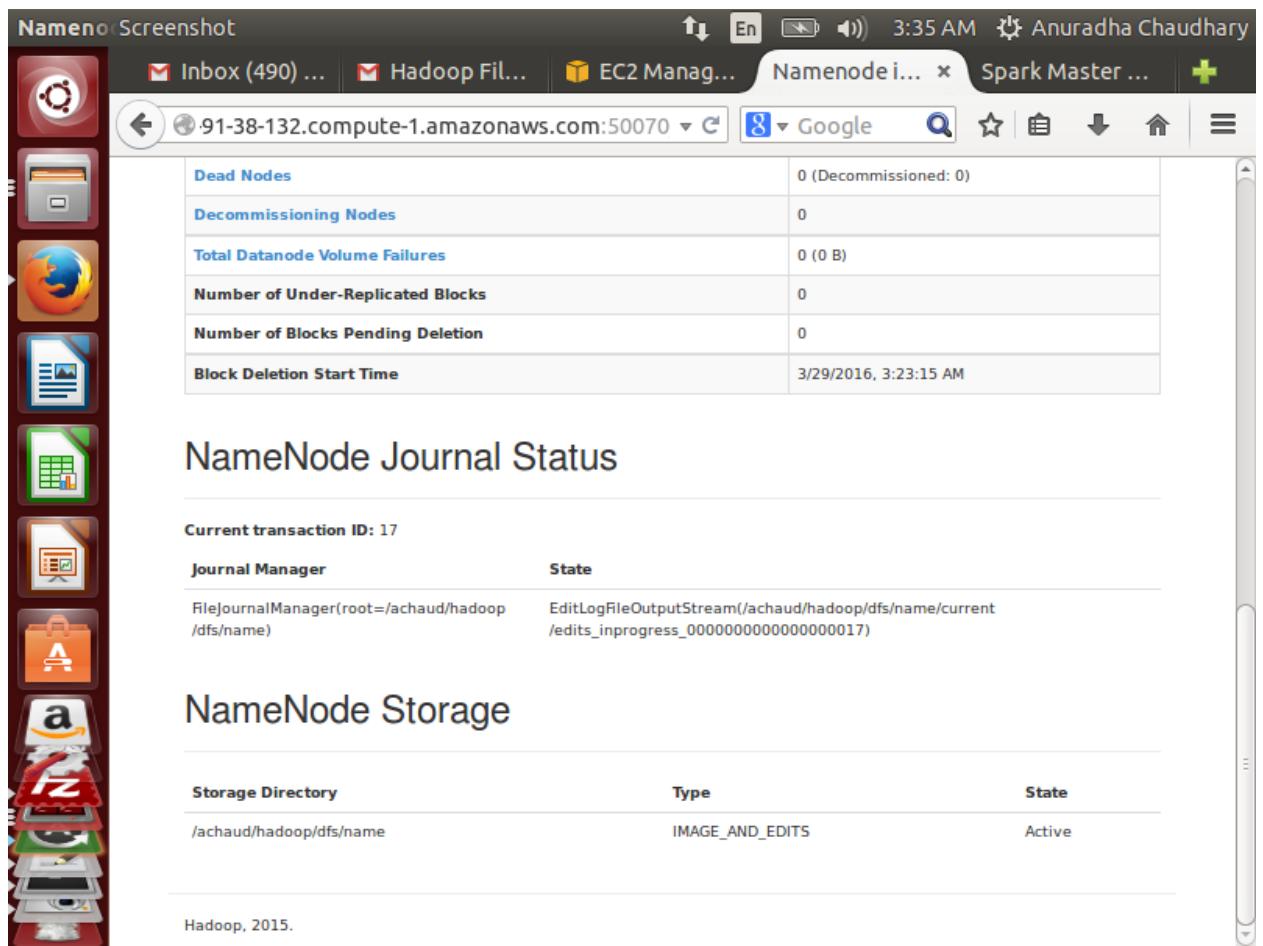
Current transaction ID: 17

Journal Manager	State
FileJournalManager(root=/achaud/hadoop/dfs/name)	EditLogFileOutputStream(/achaud/hadoop/dfs/name/current /edits_inprogress_00000000000000000017)

NameNode Storage

Storage Directory	Type	State
/achaud/hadoop/dfs/name	IMAGE_AND_EDITS	Active

Hadoop, 2015.



K. Datanode started for 100 GB of data file with 16 nodes.

Namenode information - Mozilla Firefox

3:59 AM Anuradha Chaudhary

Inbox (490 ...) Fwd: Hado... EC2 Manag... Namenode i... Spark Master ... +

ec2-52-91-38-132.compute-1.amazonaws.com Google

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ec2-54-84-252-253.compute-1.amazonaws.com:50010 (172.31.6.128:50010)	2	In Service	98.3 GB	6.3 GB	6.43 GB	85.57 GB	52	6.3 GB (6.41%)	0	2.7.2
ec2-52-87-254-99.compute-1.amazonaws.com:50010 (172.31.1.30:50010)	2	In Service	98.3 GB	6.68 GB	6.3 GB	85.32 GB	56	6.68 GB (6.79%)	0	2.7.2
ec2-52-91-112-7.compute-1.amazonaws.com:50010 (172.31.3.113:50010)	0	In Service	98.3 GB	6.3 GB	6.2 GB	85.8 GB	51	6.3 GB (6.41%)	0	2.7.2
ec2-52-201-214-144.compute-1.amazonaws.com:50010 (172.31.13.47:50010)	2	In Service	98.3 GB	6.93 GB	6.2 GB	85.17 GB	57	6.93 GB (7.05%)	0	2.7.2
ec2-52-91-43-195.compute-1.amazonaws.com:50010 (172.31.8.70:50010)	2	In Service	98.3 GB	7.06 GB	6.36 GB	84.88 GB	59	7.06 GB (7.18%)	0	2.7.2
ec2-54-84-223-125.compute-1.amazonaws.com:50010 (172.31.5.186:50010)	0	In Service	98.3 GB	5.29 GB	6.2 GB	86.81 GB	44	5.29 GB (5.38%)	0	2.7.2
ec2-54-84-186-47.compute-1.amazonaws.com:50010	1	In Service	98.3 GB	6.05 GB	6.2 GB	86.05 GB	49	6.05 GB (6.15%)	0	2.7.2

L. Datanode started for 100 GB of data file with 16 nodes.

Namenode information - Mozilla Firefox

ec2-52-91-38-132.compute-1.amazonaws.com (172.31.13.151:50010)

	Service	GB	Used GB	Percent	Nodes
ec2-52-207-252-105.compute-1.amazonaws.com:50010 (172.31.4.69:50010)	In Service	98.3 GB	5.29 GB	5.38%	44
ec2-54-172-28-19.compute-1.amazonaws.com:50010 (172.31.7.216:50010)	In Service	98.3 GB	5.54 GB	5.64%	46
ec2-54-152-240-118.compute-1.amazonaws.com:50010 (172.31.0.249:50010)	In Service	98.3 GB	5.17 GB	5.25%	41
ec2-54-84-210-242.compute-1.amazonaws.com:50010 (172.31.5.102:50010)	In Service	98.3 GB	4.79 GB	4.87%	41
ec2-52-90-101-18.compute-1.amazonaws.com:50010 (172.31.5.252:50010)	In Service	98.3 GB	6.3 GB	6.41%	52
ec2-52-90-109-33.compute-1.amazonaws.com:50010 (172.31.9.108:50010)	In Service	98.3 GB	5.17 GB	5.25%	43
ec2-54-86-246-7.compute-1.amazonaws.com:50010 (172.31.9.23:50010)	In Service	98.3 GB	7.19 GB	7.31%	60
ec2-54-164-92-128.compute-1.amazonaws.com:50010 (172.31.11.52:50010)	In Service	98.3 GB	5.04 GB	5.13%	42
ec2-54-164-118-158.compute-1.amazonaws.com:50010 (172.31.6.219:50010)	In Service	98.3 GB	4.79 GB	4.87%	42

Decommissioning

M. Application status on completion of task for 100 GB of data file with 16 nodes.

The screenshot shows the 'All Applications' page of the Hadoop Job History UI. The URL is 2-91-38-132.compute-1.amazonaws.com:9064. The main content area displays the 'All Applications' section with the title 'hadoop'. Below it, there are two tables: 'Cluster Metrics' and 'Scheduler Metrics'. The 'Cluster Metrics' table shows 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	0	1	0	0 B	51 GB	0 B	0	136	0	17	0	0	0	0	0

The 'Scheduler Metrics' table shows the following data:

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:3072, vCores:8>

Below these tables, a detailed view of a single application is shown:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1459239812100_0001	ubuntu	Hadoop_Sort	MAPREDUCE	default	Tue Mar 29 03:57:57 -0500 2016	Tue Mar 29 04:13:32 -0500 2016	FINISHED	SUCCEEDED	N/A	History	N/A

Showing 1 to 1 of 1 entries

- AS it can be seen from the screen shot above, time taken for sorting 100 GB of data on 16 nodes in around 15 mins which lesser than the time taken for sorting 10 GB of data on 1 node.
- This shows that with more numer of slaves we get better and faster results on Hadoop.

N. First 10 lines of sorted file

```
ubuntu@ip-172-31-13-104: /achaud/output          ↑↓ En 🔍 4:45 AM ⚡ Anuradha Chaudhary
 _partition.lst  part-r-00008  part-r-00017  part-r-00026  part-r-00035
 part-r-00000  part-r-00009  part-r-00018  part-r-00027  part-r-00036
 part-r-00001  part-r-00010  part-r-00019  part-r-00028  part-r-00037
 part-r-00002  part-r-00011  part-r-00020  part-r-00029  part-r-00038
 part-r-00003  part-r-00012  part-r-00021  part-r-00030  part-r-00039
 part-r-00004  part-r-00013  part-r-00022  part-r-00031  _SUCCESS
 part-r-00005  part-r-00014  part-r-00023  part-r-00032
 part-r-00006  part-r-00015  part-r-00024  part-r-00033
 part-r-00007  part-r-00016  part-r-00025  part-r-00034
ubuntu@ip-172-31-13-104:/achaud/output$ head -10 part-r-00000
!4+ABv 0000000000000000000000000000000017F7E829 EEEE3333444411112222888833334444666
633332222DDDEEEE
"0!uve 000000000000000000000000000000001228D4 77778888000022224444DDDDDDDEEEE000
00000CCCC7777DDDD
%!$sU( 000000000000000000000000000000002E6C821C 2222333377774444555511119999CCCC444
4EEEEFFFFF11115555
&5rX|X 00000000000000000000000000000000399BC288 5555CCCCBBBB99999999DDDD11110000111
1EEEE7777DDDD9999
'ic%So 0000000000000000000000000000000031F06B7D EEEEBBBBAAA8888DDDDDDDD77772222444
4111166664444AAAA
*0G1Io 000000000000000000000000000000003B5E85A1 1111AAAA9999CCCCBBBB111199991111333
39999111AAAA6666
,(GhT_ 000000000000000000000000000000002D0172DC 1111CCCC1111DDDCCCCEE9999CCCC888
8CCCCFFFF55555555
0*vYm3 0000000000000000000000000000000026D61578 DDDD7777AAAAEEEEEE6666AAAA2222CCC
C5555555522229999
2C>)8d 0000000000000000000000000000000026C79E66 444400001111CCCC6666BBBB55557777666
6CCCC2222AAAABBBB
PMd32= 000000000000000000000000000000003440CC1 FFFFEEEE6666CCCCBBBB999933335555DDD
DDDDD777788886666
ubuntu@ip-172-31-13-104:/achaud/output$
```

O. Last 10 lines of sorted file

```
ubuntu@ip-172-31-13-104: /achaud/output
399991111AAAA6666
,(GhT_ 000000000000000000000000000000002D0172DC 1111CCCC1111DDDCCCEEE9999CCCC888
8CCCCFFFF55555555
0*vYm3 0000000000000000000000000000000026D61578 DDDD7777AAAAEEEEEE6666AAAA2222CCC
C5555555522229999
2C>)8d 0000000000000000000000000000000026C79E66 444400001111CCCC6666BBBB55557777666
6CCCC2222AAAABBBB
PMd32= 000000000000000000000000000000003440CC1 FFFFEEEE6666CCCCBBBB99993335555DDD
DDDD777788886666
ubuntu@ip-172-31-13-104:/achaud/output$ tail -10 part-r-00039
~~~#iay1X 0000000000000000000000000000000025D35EDF 6666AAAA555599997777000022223333888
8FFFFF999922220000
~~~-+@p){@ 000000000000000000000000000000008542F4 7777333555511111110000CCCC5555999
9AAAA7777DDDDDDDD
~~~,R^_?n 000000000000000000000000000000001034E347 11111119999000011118888AAAA5555444
4EEEEEE999933338888
~~~.Ey`^) 0000000000000000000000000000000016F0E66B CCCC6666DDDD2222DDDD111188889999EEE
EEEEEEEEBBBB4444
~~~-4!kA7x 000000000000000000000000000000001F1A1E26 EEEE777711117777BBBB1111EEEE8888444
4DDDDDDDDDEEEEBBBB
~~~-8Ii/!@ 000000000000000000000000000000001F05932F 11119999BBBB44447777000011114444CCC
CAAA6666DDDD0000
~~~-<I'5>F 0000000000000000000000000000000008CB2293 88883333BBBB11116666999988885555888
8888822228888CCCC
~~~-G- )m^) 0000000000000000000000000000000013397F73 DDDDFFFBBBBCCCCFFFF44446666AAAA111
133333333AAAACCCC
~~~-c+I&cP 00000000000000000000000000000000074BDF64 8888000055550000DDDD22227777AAAA000
033332222AAAADDDD
~~~-hb&5X* 0000000000000000000000000000000032C0E06B 7777BBBBBBBB9999EEEEAAAAAAA0000CCC
CDDDD4444BBBB4444
ubuntu@ip-172-31-13-104:/achaud/output$
```

P. Result of valsrt for 100Gb sorted file

```
ubuntu@ip-172-31-13-104: ~/genSort/64
^C
ubuntu@ip-172-31-13-104:~/genSort/64$ cd ..
ubuntu@ip-172-31-13-104:~/genSort$ cd ..
ubuntu@ip-172-31-13-104:~$ cd /achaud/output/
ubuntu@ip-172-31-13-104:/achaud/output$ ls
part-r-00000 part-r-00008 part-r-00016 part-r-00024 part-r-00032
part-r-00001 part-r-00009 part-r-00017 part-r-00025 part-r-00033
part-r-00002 part-r-00010 part-r-00018 part-r-00026 part-r-00034
part-r-00003 part-r-00011 part-r-00019 part-r-00027 part-r-00035
part-r-00004 part-r-00012 part-r-00020 part-r-00028 part-r-00036
part-r-00005 part-r-00013 part-r-00021 part-r-00029 part-r-00037
part-r-00006 part-r-00014 part-r-00022 part-r-00030 part-r-00038
part-r-00007 part-r-00015 part-r-00023 part-r-00031 part-r-00039
ubuntu@ip-172-31-13-104:/achaud/output$ cat * > /achaud/output
-bash: /achaud/output: Is a directory
ubuntu@ip-172-31-13-104:/achaud/output$ cat * > /achaud/output/outputf
ubuntu@ip-172-31-13-104:/achaud/output$ ./valsrt /achaud/output/outputf
-bash: ./valsrt: No such file or directory
ubuntu@ip-172-31-13-104:/achaud/output$ cd ..
ubuntu@ip-172-31-13-104:/achaud$ cd ..
ubuntu@ip-172-31-13-104:/$ cd home/ubuntu/genSort/64
ubuntu@ip-172-31-13-104:~/genSort/64$ ./valsrt /achaud/output/outputf

Records: 10000000000
Checksum: 1dc615efb9dfe11
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-13-104:~/genSort/64$ 
ubuntu@ip-172-31-13-104:~/genSort/64$ 
ubuntu@ip-172-31-13-104:~/genSort/64$ 
```

3. SPARK:

A. Job completion status for sorting 10 GB data file. As it can be seen time taken for sorting 10 GB data by Spark is 14 mins.

The screenshot shows a Linux desktop interface with several application icons in a vertical dock on the left. A web browser window is open, displaying the Spark Master UI at 87-222-214.compute-1.amazonaws.com:8080. The title bar of the browser says "Spark Master at spark://ec2-52-87-222-214.compute-1.amazonaws.com:7077". The main content area displays the following information:

Spark 1.6.1 **Spark Master at spark://ec2-52-87-222-214.compute-1.amazonaws.com:7077**

URL: spark://ec2-52-87-222-214.compute-1.amazonaws.com:7077
REST URL: spark://ec2-52-87-222-214.compute-1.amazonaws.com:6066 (cluster mode)

Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 2.7 GB Total, 0.0 B Used
Applications: 0 Running, 2 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20160327235158-172.31.46.185-52545	172.31.46.185:52545	ALIVE	2 (0 Used)	2.7 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160328002110-0001	Spark shell	2	2.4 GB	2016/03/28 00:21:10	root	FINISHED	14 min
app-20160328001821-0000	Spark shell	2	2.4 GB	2016/03/28 00:18:21	root	FINISHED	59 s

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160328002110-0001	Spark shell	2	2.4 GB	2016/03/28 00:21:10	root	FINISHED	14 min
app-20160328001821-0000	Spark shell	2	2.4 GB	2016/03/28 00:18:21	root	FINISHED	59 s

B. Terminal status on completion of sorting 10 GB data file. As it can be seen here as it shows time taken is 14 mins

```
anuradha@anuradha-VirtualBox: ~/Desktop/Spark/spark-1  7:39 PM Anuradha Chaudhary
16/03/28 00:33:54 INFO scheduler.TaskSetManager: Finished task 68.0 in stage 2.0
(TID 218) in 12854 ms on ip-172-31-46-185.ec2.internal (69/75)
16/03/28 00:34:06 INFO scheduler.TaskSetManager: Starting task 71.0 in stage 2.0
(TID 221, ip-172-31-46-185.ec2.internal, partition 71,NODE_LOCAL, 1894 bytes)
16/03/28 00:34:06 INFO scheduler.TaskSetManager: Finished task 69.0 in stage 2.0
(TID 219) in 12443 ms on ip-172-31-46-185.ec2.internal (70/75)
16/03/28 00:34:09 INFO scheduler.TaskSetManager: Starting task 72.0 in stage 2.0
(TID 222, ip-172-31-46-185.ec2.internal, partition 72,NODE_LOCAL, 1894 bytes)
16/03/28 00:34:09 INFO scheduler.TaskSetManager: Finished task 70.0 in stage 2.0
(TID 220) in 15281 ms on ip-172-31-46-185.ec2.internal (71/75)
16/03/28 00:34:21 INFO scheduler.TaskSetManager: Starting task 73.0 in stage 2.0
(TID 223, ip-172-31-46-185.ec2.internal, partition 73,NODE_LOCAL, 1894 bytes)
16/03/28 00:34:21 INFO scheduler.TaskSetManager: Finished task 71.0 in stage 2.0
(TID 221) in 15156 ms on ip-172-31-46-185.ec2.internal (72/75)
16/03/28 00:34:21 INFO scheduler.TaskSetManager: Starting task 74.0 in stage 2.0
(TID 224, ip-172-31-46-185.ec2.internal, partition 74,NODE_LOCAL, 1894 bytes)
16/03/28 00:34:21 INFO scheduler.TaskSetManager: Finished task 72.0 in stage 2.0
(TID 222) in 12560 ms on ip-172-31-46-185.ec2.internal (73/75)
16/03/28 00:34:32 INFO scheduler.TaskSetManager: Finished task 73.0 in stage 2.0
(TID 223) in 11240 ms on ip-172-31-46-185.ec2.internal (74/75)
16/03/28 00:34:32 INFO scheduler.TaskSetManager: Finished task 74.0 in stage 2.0
(TID 224) in 10891 ms on ip-172-31-46-185.ec2.internal (75/75)
16/03/28 00:34:32 INFO scheduler.DAGScheduler: ResultStage 2 (saveAsTextFile at
<console>:30) finished in 467.651 s
16/03/28 00:34:32 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 2.0, whose t
asks have all completed, from pool
16/03/28 00:34:32 INFO scheduler.DAGScheduler: Job 1 finished: saveAsTextFile at
<console>:30, took 727.979745 s
t1: Long = 3195442933268
Time taken for 1 GB sorting: 790773      ms
```

C. Result of valsort for 10Gb sorted file

```
anuradha@anuradha-VirtualBox: ~/Desktop/Spark/spark-1.6.1 | En | 8:16 PM | Anuradha Chaudhary
99CCCC000044448888
}u%d!@0Nbd 00000000000000000000000000000000492FFE1 FFFF3333777700009999EEEE7777666688
88222211116666666
}u%d\Nyo~ 000000000000000000000000000000003D1CFD 11115555FFFF444411111117777EEE99
9922228888CCCCAAA
}u%ndnpMF| 00000000000000000000000000000000F4D843 3333CCCCAAAA5555000022228888999922
2288885555DDDDCCCC
}u%dw8j&ey 0000000000000000000000000000000017BAB4B AAAAEEEEEE7777CCCC0000DDDD333366
66EEEEBBBBBBBB4444
}u%e1}rA'D 0000000000000000000000000000000040DD128 8888BBBBAAAA000044448888FFFF1111AA
AABBAB2222BBBB9999
}u%fKNz<C* 0000000000000000000000000000000025F9B1A 2222CCCCAAAA66667777CCCCBBBB8888AA
AA33336666AAAA7777
root@ip-172-31-35-4 out1]$ ^C
root@ip-172-31-35-4 out1]$ cd
root@ip-172-31-35-4 ~]$ cd genSort/64/
root@ip-172-31-35-4 64]$ chmod 777 gensort
root@ip-172-31-35-4 64]$ chmod 777 valsor
root@ip-172-31-35-4 64]$ cd
root@ip-172-31-35-4 ~]$ cd /achaud/output/out1/
root@ip-172-31-35-4 out1]$ cat * > terasort_10GB_output.txt
root@ip-172-31-35-4 out1]$ cd
root@ip-172-31-35-4 ~]$ cd genSort/64/
^[[C^[[D^C-35-4 64]$ ./valsor /achaud/output/out1/terasort_10GB_output.txt
^[[C^[[D^C-35-4 64]$ ./valsor /achaud/output/out1/terasort_10GB_output.txt
root@ip-172-31-35-4 64]$ ./valsor /achaud/output/out1/terasort_10GB_output.txt
Records: 100000000
Checksum: 2fae59101920038
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-35-4 64]$
```

D. Namenode status on starting sort of 100 GB data on 16 nodes. As it can be seen that 16 nodes are live here.

The screenshot shows a web browser window with the following details:

- Address Bar:** ec2-54-175-123-23.compute-1.amazonaws.co
- Title Bar:** NameNode 'ip-172-31-10-135.ec2.internal:9000'
- Content Area:**
 - Cluster Summary:** 7 files and directories, 746 blocks = 753 total. Heap Size is 289.5 MB / 889 MB (32%)

Configured Capacity	:	480.5 GB
DFS Used	:	93.96 GB
Non DFS Used	:	45.62 GB
DFS Remaining	:	340.91 GB
DFS Used%	:	19.55 %
DFS Remaining%	:	70.95 %
Live Nodes	:	16
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0
 - NameNode Storage:**

Storage Directory	Type	State
/mnt/ephemeral-hdfs/dfs/name	IMAGE_AND_EDITS	Active
 - File Browser:** [Browse the filesystem](#)
 - Logs:** [Namenode Logs](#)

E. On Job completion status of Spark for 16 nodes.

Spark Master at spark://ec2-54-175-123-23.compute-1.amazonaws.com:7077

URL: spark://ec2-54-175-123-23.compute-1.amazonaws.com:7077
REST URL: spark://ec2-54-175-123-23.compute-1.amazonaws.com:6066 (cluster mode)

Alive Workers: 16

Cores in use: 32 Total, 32 Used

Memory in use: 42.7 GB Total, 38.3 GB Used

Applications: 1 Running, 2 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20160328222253-172.31.0.89-57331	172.31.0.89:57331	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.1.132-41772	172.31.1.132:41772	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.11.152-56325	172.31.11.152:56325	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.12.222-33186	172.31.12.222:33186	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.12.241-57923	172.31.12.241:57923	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.13.235-41408	172.31.13.235:41408	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.14.169-39520	172.31.14.169:39520	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.14.201-39237	172.31.14.201:39237	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.2.133-52434	172.31.2.133:52434	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)
worker-20160328222253-172.31.2.239-54079	172.31.2.239:54079	ALIVE	2 (2 Used)	2.7 GB (2.4 GB Used)

F. On Job completion status of Spark for 16 nodes.

Spark Master at spark://ec2-54-175-123-23.compute-1.amazonaws.com:7077

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160329010353-0002	(kill) Spark shell	32	2.4 GB	2016/03/29 01:03:53	root	RUNNING	1.3 min

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160329010235-0001	Spark shell	32	2.4 GB	2016/03/29 01:02:35	root	FINISHED	1.1 min
app-20160329010011-0000	Spark shell	32	2.4 GB	2016/03/29 01:00:11	root	FINISHED	2.2 min

G. Result of valsort for 100Gb sorted file

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal" and the command being run is "root@ip-172-31-10-135 64] \$./valsort /vol0/Output.txt". The output of the command is displayed in the terminal window:

```
Records: 10000000000
Checksum: 1dcfd7de9014883a9
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-10-135 64] $
```

Below the terminal window, a file manager interface is visible. It shows a sidebar with icons for Home, Desktop, Applications, Places, and Dash. The main area displays a list of files and folders. At the bottom, there is a status bar with the message "Selected 1 directory. 2 files. Total size: 275.7 KB".

Server/Local file	Directio	Remote file	Size	Priority	Status
Queued files	Failed transfers	Successful transfers (20)			

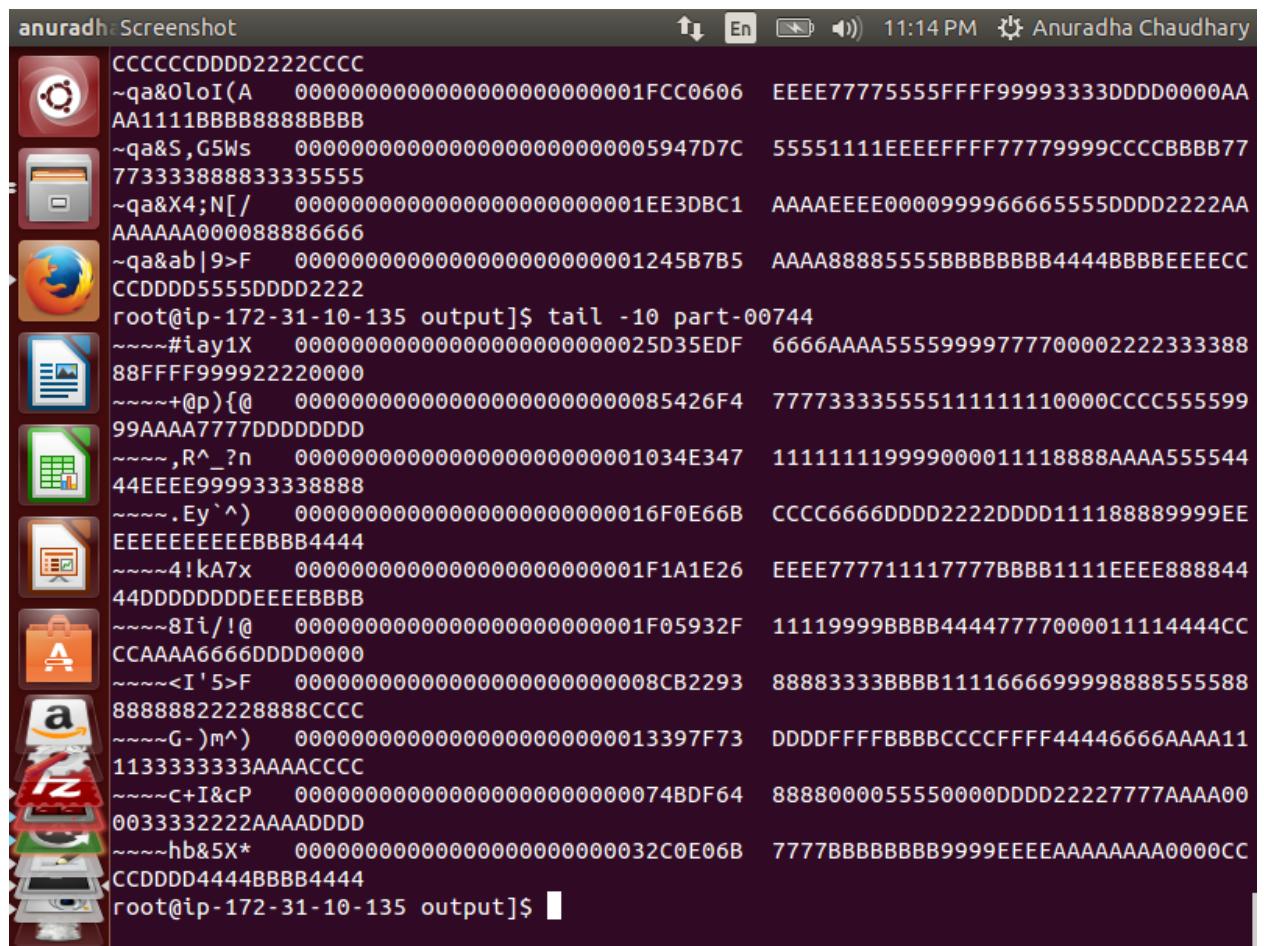
At the bottom right, there is a message "Queue: empty" and some small icons.

H. First 10 lines of sorted file

```
anuradha@anuradha-VirtualBox: ~/Desktop/Spark/spark-1 ↑ En ⟲ ⟳ 11:13 PM ⚡ Anuradha Chaudhary
part-00118  part-00243  part-00368  part-00493  part-00618  part-00743
part-00119  part-00244  part-00369  part-00494  part-00619  part-00744
part-00120  part-00245  part-00370  part-00495  part-00620  _SUCCESS
part-00121  part-00246  part-00371  part-00496  part-00621
part-00122  part-00247  part-00372  part-00497  part-00622
part-00123  part-00248  part-00373  part-00498  part-00623
part-00124  part-00249  part-00374  part-00499  part-00624
root@ip-172-31-10-135 output]$ head -10 part00000
head: cannot open `part00000' for reading: No such file or directory
root@ip-172-31-10-135 output]$ head -10 part-00000
    !4+ABv      00000000000000000000000000000017F7E829  EEEE33344441112228888333444466
6633332222DDDEEEE
    "0!uve      0000000000000000000000000000001228D4  7777888800002224444DDDDDDDEEE00
000000CCCC7777DDDD
    %!$sU(      0000000000000000000000000000002E6C821C  222233377744455551119999CCCC44
44EEEEFFFF11115555
    &5rX|X      0000000000000000000000000000399BC288  5555CCCCBBBB99999999DDDD111000011
11EEEE7777DDDD9999
    'ic%So      000000000000000000000000000031F06B7D  EEEEBBBAAA8888DDDDDD777722244
44111166664444AAAA
    *0G1Io      0000000000000000000000000003B5E85A1  1111AAAA9999CCCCBBB111999911133
3399991111AAAA6666
    ,(GhT_      00000000000000000000000000002D0172DC  1111CCCC1111DDDCCCCEE9999CCCC88
88CCCCFFFF55555555
    0*vYm3      000000000000000000000000000026D61578  DDDD7777AAAAEEEEEE6666AAAA222CC
CC5555555522229999
    2C>)8d      000000000000000000000000000026C79E66  444400001111CCCC6666BBBB5555777766
66CCCC2222AAAABBBB
    PMd32=      00000000000000000000000000003440CC1  FFFFEEEE6666CCCCBBBB999933335555DD
DDDDDD777788886666
root@ip-172-31-10-135 output]$
```

I. Last 10 lines of sorted file

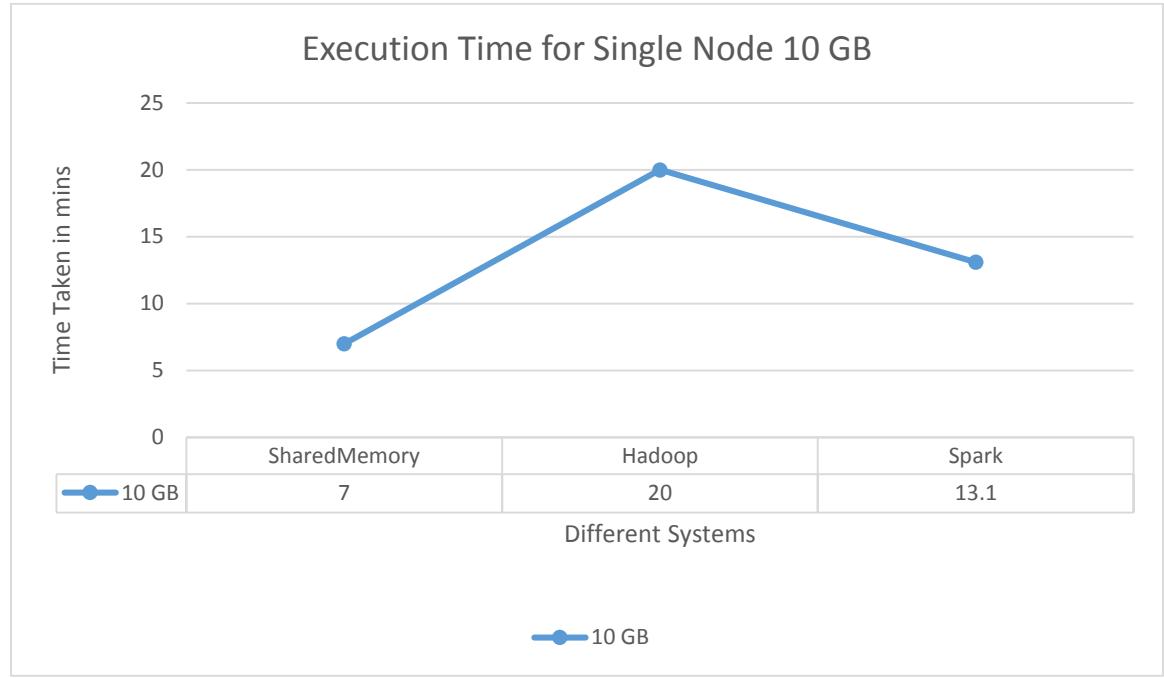
anuradhaScreenshot 11:14 PM Anuradha Chaudhary



```
CCCCCDDDD2222CCCC
~qa&0loI(A 000000000000000000000000000000001FCC0606 EEEE77775555FFFF99993333DDDD0000AA
AA111BBBB8888BBBB
~qa&S,G5Ws 000000000000000000000000000000005947D7C 55551111EEEEFFFF77779999CCCCBBBB77
7733388883333555
~qa&X4;N[/_ 000000000000000000000000000000001EE3DBC1 AAAAEEEE0000999966665555DDDD2222AA
AAAAAA000088886666
~qa&ab|9>F 000000000000000000000000000000001245B7B5 AAAA88885555BBBBBBBB4444BBBBEEEECC
CCDDDD5555DDDD2222
root@ip-172-31-10-135 output]$ tail -10 part-00744
~~~~#iay1X 00000000000000000000000000000025D35EDF 6666AAAA55559999777700002222333388
88FFFF999922220000
~~~~+@p){@ 0000000000000000000000000000000085426F4 7777333355511111110000CCCC555599
99AAAA7777DDDDDD
~~~~,R^_?n 0000000000000000000000000000001034E347 11111119999000011118888AAAA555544
44EEEE999933338888
~~~~.Ey`^) 0000000000000000000000000000000016F0E66B CCCC6666DDDD2222DDDD111188889999EE
EEEEEEEEEEBBBB4444
~~~~4!kA7x 0000000000000000000000000000001F1A1E26 EEEE777711117777BBBB1111EEEE888844
44DDDDDDDEEEEBBBB
~~~~8Ii/!@ 0000000000000000000000000000001F05932F 11119999BBBB44447777000011114444CC
CCAAA6666DDDD0000
~~~~<I'5>F 000000000000000000000000000000008CB2293 88883333BBBB1111666699998888555588
88888822228888CCCC
~~~~G-)m^) 0000000000000000000000000000000013397F73 DDDDFFFFB BBBBCCCCFFFF44446666AAAA11
1133333333AAAACCCC
~~~~c+I&cP 0000000000000000000000000000000074BDF64 8888000055550000DDDD22227777AAAA00
0033332222AAAADD
~~~~hb&5X* 00000000000000000000000000000032C0E06B 7777BBBBBBBB9999EEEEAAAAAAA0000CC
CCDDDD4444BBBB4444
root@ip-172-31-10-135 output]$
```

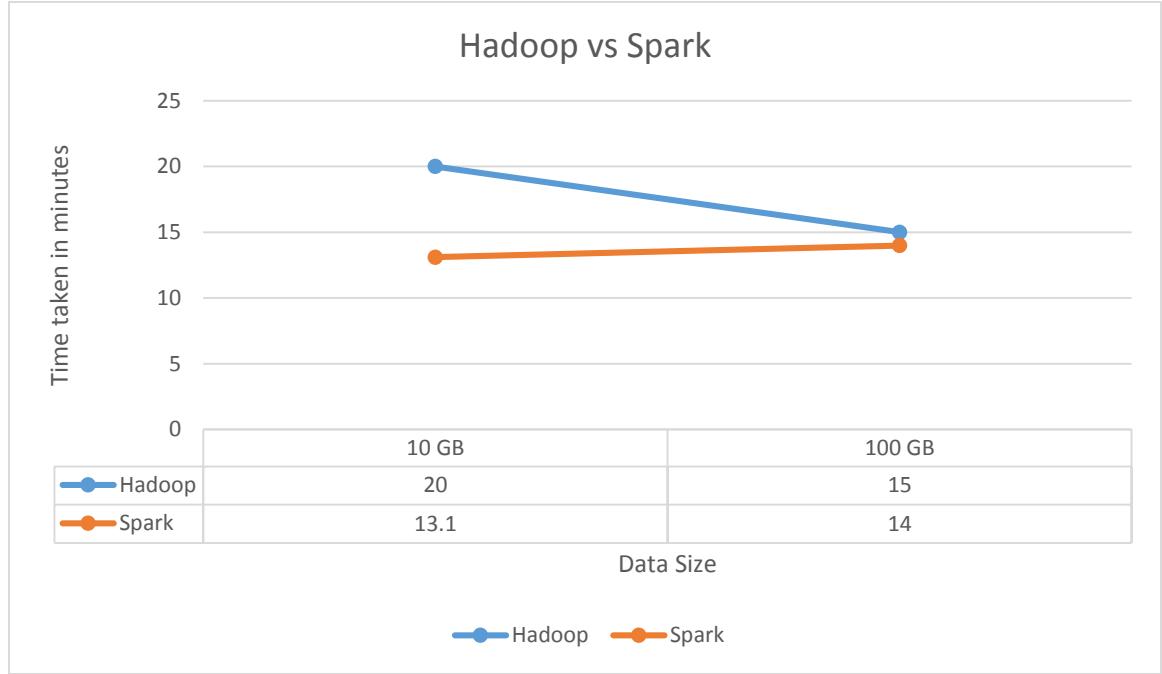
EXECUTION TIME DATA:

- **10 GB, Single Node:**



- Y-axis represents here time taken in mins for sorting.
- X-axis represents here different systems.
- As it can be seen from the graph that least time is taken by shared memory program i.e. 7 mins while Hadoop takes 20 mins and Spark takes 13.1 mins.
- This is because we know, shared memory executes simply on machine using multiple threads and does not require any configuration while Hadoop and Spark create over head for setting up the 1 Node virtual cluster and hence the delay.
- Thus, results make sense as shared memory takes the least time.

- **100 GB, 16 Nodes and 10 GB, 1 Node:**

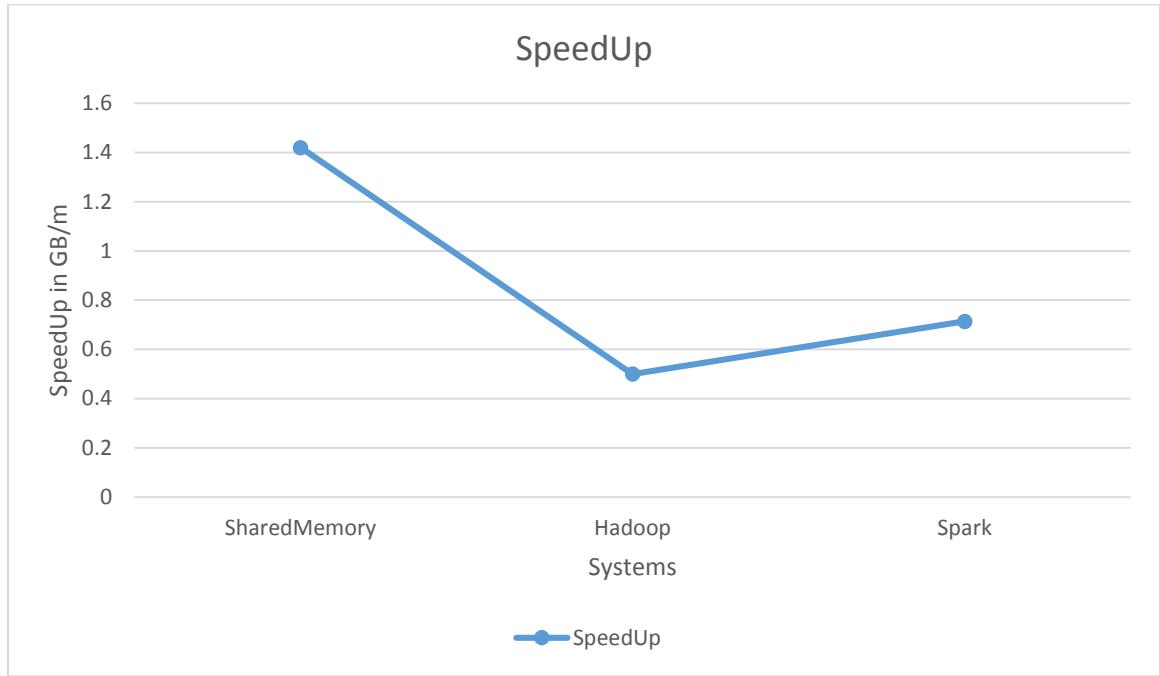


- Graph shows comparison of Hadoop and Spark for 10 GB sorting on 1 Node and 100 GB sorting on 17 Nodes.
- X-axis represents here datasize
- Y-axis represents here time taken in mins for sorting.
- As it can be seen from the graph, time taken by Hadoop for sorting 10 GB data on single is 20 mins while sorting 100 GB data on 17 nodes is 15 mins. We see a decrease in time taken. This is because Hadoop is easily scalable and as it has more number of workers on 17 nodes cluster, work is distributed and hence done efficiently resulting in decrease in time taken.
- Also it can be seen from graph, time taken by Spark for sorting 10GB data on single node is 13.1 mins while sorting 100 GB data on 17 nodes is 14 mins. Here we see time taken for 100 GB on 17 nodes and 10 GB on single node remains almost the same. This is because Spark reuses intermediate generated data and so even after X number of iterations the time taken will remain the same.
- On comparing Hadoop and Spark we know Spark is super lightweight analytic tool. Spark is faster than Hadoop as Spark was designed in order to overcome the drawbacks of Hadoop. This is because Spark uses RDDs underneath and keeps data in memory which fastens the sorts. There is no such notion of keeping data in memory in Hadoop.

- From graph above we can verify that Spark is faster than Hadoop.

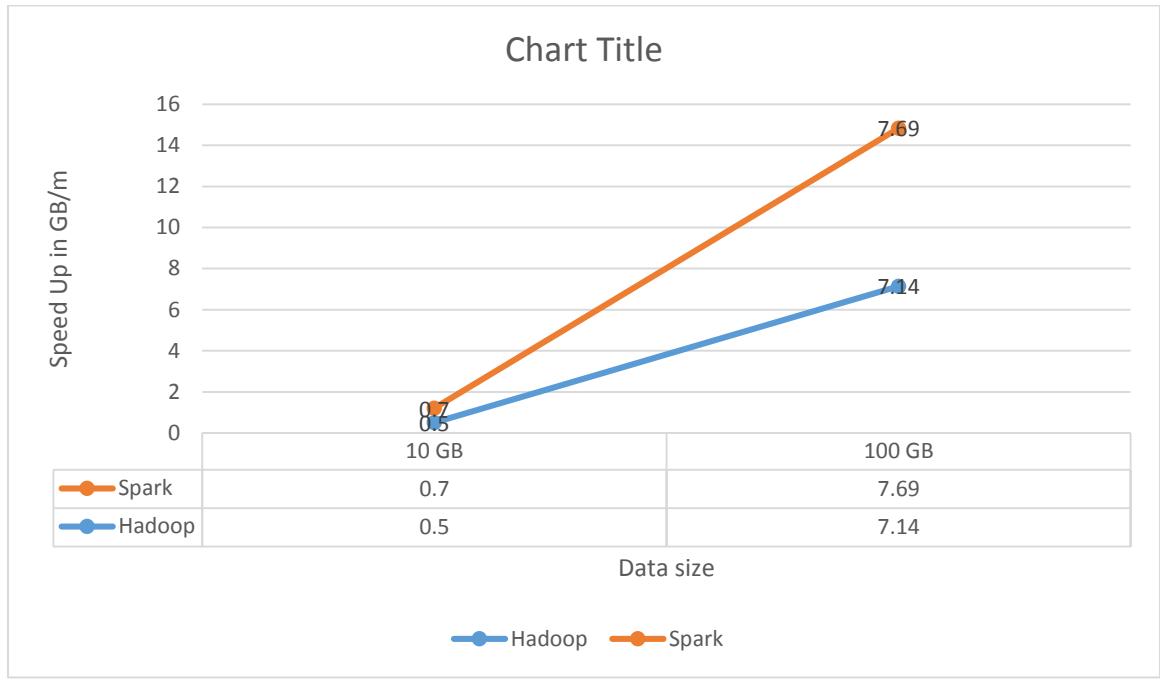
SPEEDUP:

- **10 GB 1 Node:**



- We know speed up is Data_Size/Time_Taken.
- Y-axis in graph shows Speed Up in GB/mins.
- X-axis in graph shows different systems used.
- As time taken by shared memory was least for 100 GB sorting amongst Hadoop, Spark and Shared Memory, Speed Up of Shared Memory is most i.e 1.42.
- Second comes spark which is faster than Hadoop but slower than shared memory for 1 Node i.e. 0.7.
- Last comes Hadoop with speedup of 0.5.
- Hence, the results shown here in graph makes sense.

- **10 GB 1 Node vs 100 GB 16 Nodes:**



- X-axis here represents data size
- Y-axis here represents speed up in GB/mins.
- We know, speed up is data size/time taken.
- As from graph it can be seen that speed Up of spark increases linearly and very fast compared to Hadoop.
- This is because the time taken by Hadoop increases with number of iterations while it remains nearly same for spark.
- Speed Up of Spark is more compared to that of Hadoop.

From these experiments following conclusions can be drawn:

- At 1 node scale, shared memory seems best as it the fastest amongst 3 for single node.
- At 16 nodes, either Hadoop or Spark can be used as time will be nearly same for both.
- At 100 nodes, as Spark is faster than Hadoop as it uses RDDs which keeps data in memory, spark should be used.
- At 1000 nodes scale too Spark should be used as it is faster than Hadoop and is very light weight.