CS553 Cloud Computing

Programming Assignment #2

Report

Pranitha Nagavelli(A20345406)

The following are the important aspects in this assignment:

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

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

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

Sort in JAVA:

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

Sort in HADOOP:

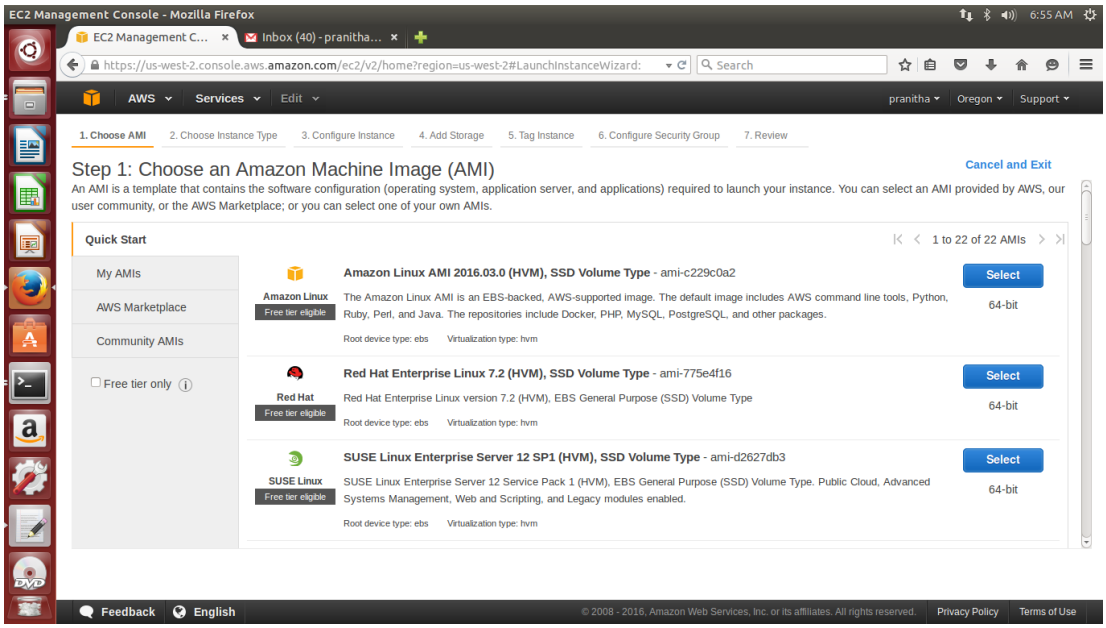
In Hadoop dataset produced from gensort of size 10GB and 100GB/1Tb has to be sorted which is achieved by implementing MapReduce code where the each line is divided into key and values. Each key is of first 10 bytes and remaining as values. Now this key value pairs are used in this sorting application.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

Sort in SPARK:

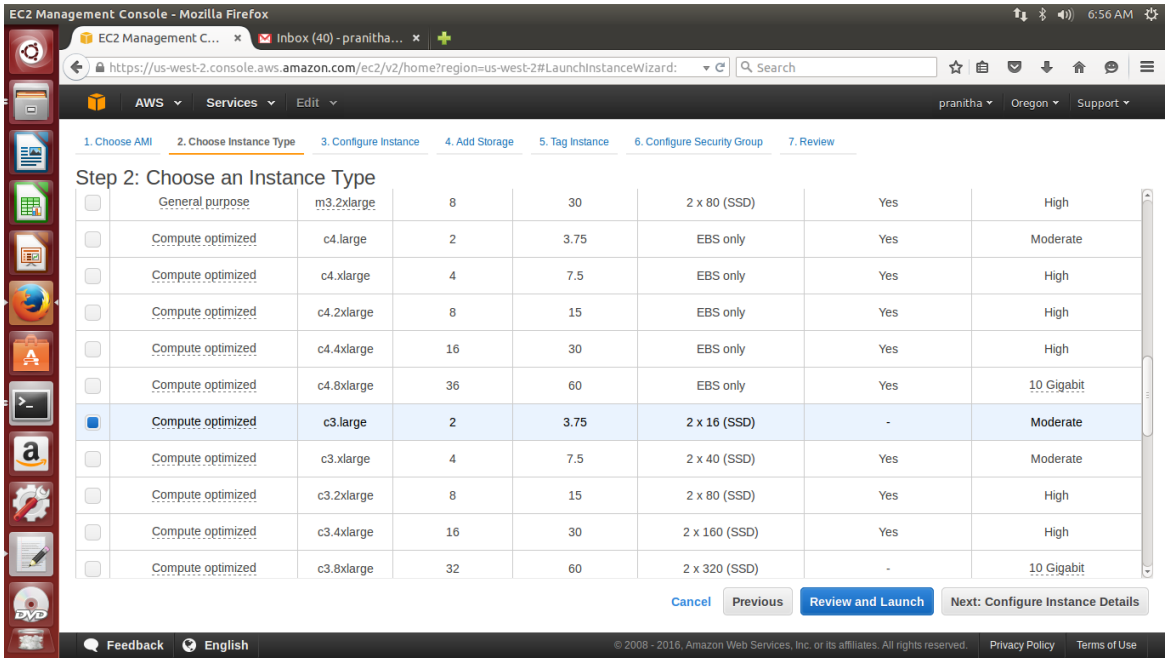
In spark dataset produced from gensort of size 10GB and 100GB/1Tb has to be sorted which is achieved by using some of the inbuilt functions line ‘sortByKey()’ where the each line is divided into key and values. Each key is of first 10 bytes and remaining as values. Now this key value pairs are used in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

Creating a AWS Instance:

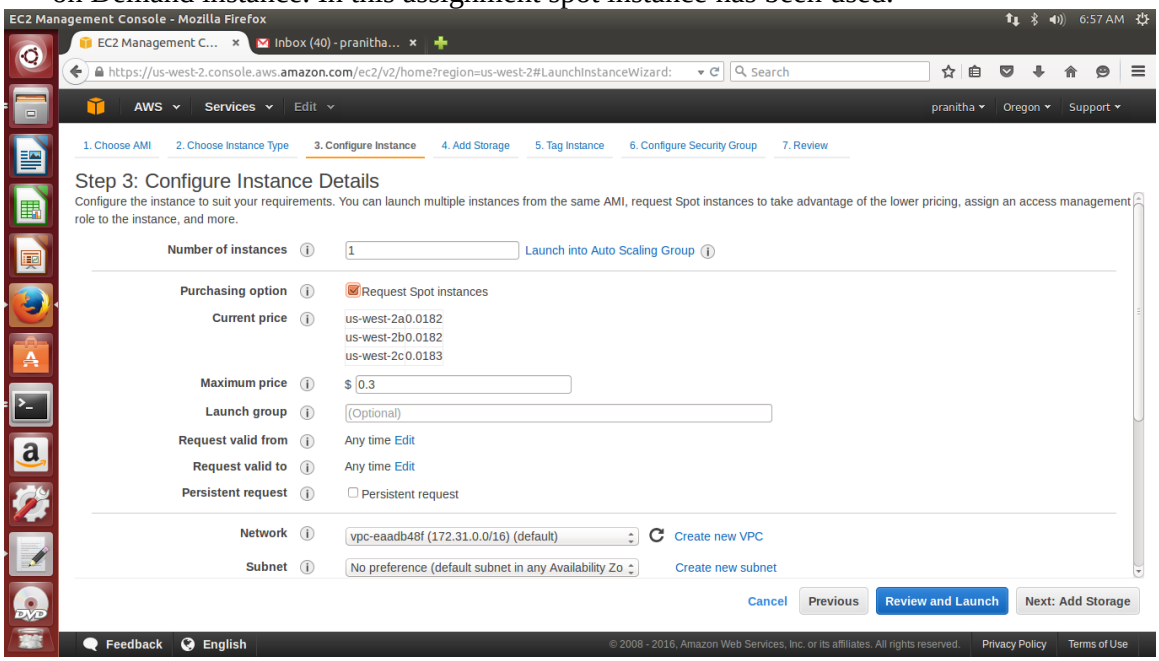
- Log in to Amazon AWS console, under services select EC2
- Now launch an instance, that best AMI that suits the requirement



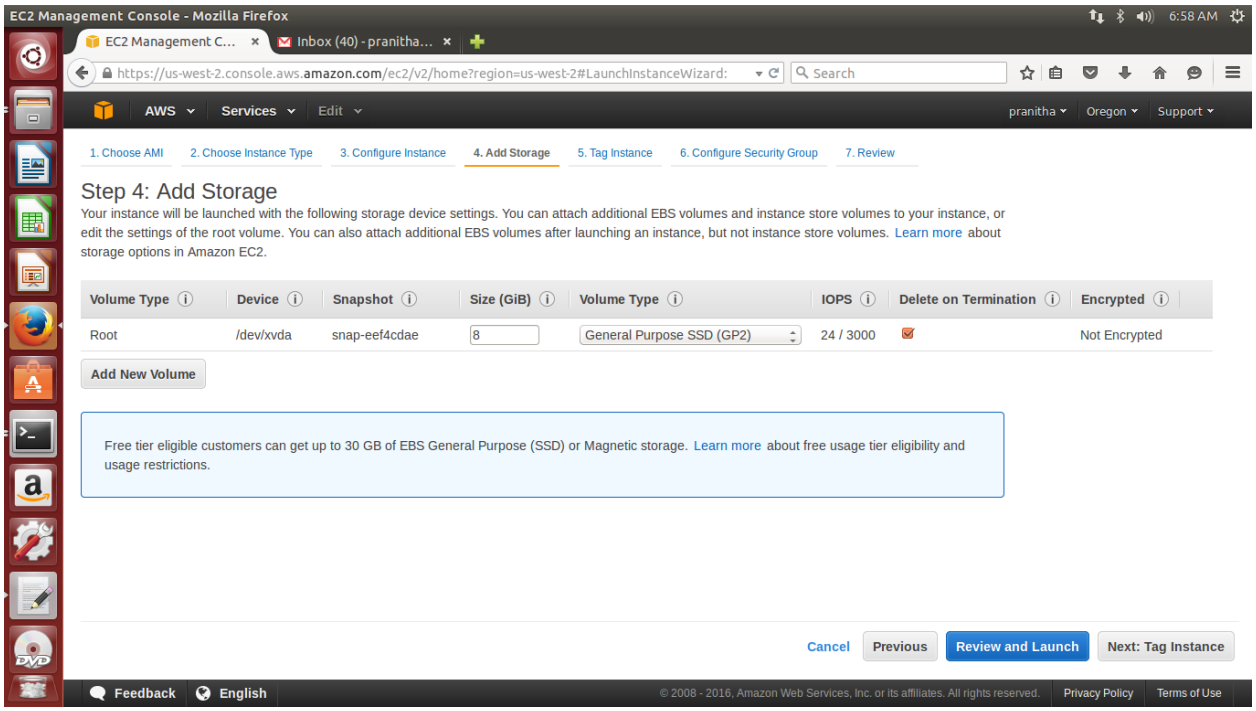
- Then choose the instance type which can be micro, small, medium, large Xlarge, 2Xlarge, 4Xlarge or 8Xlarge. For our assignment it is c3.large for 10GB data



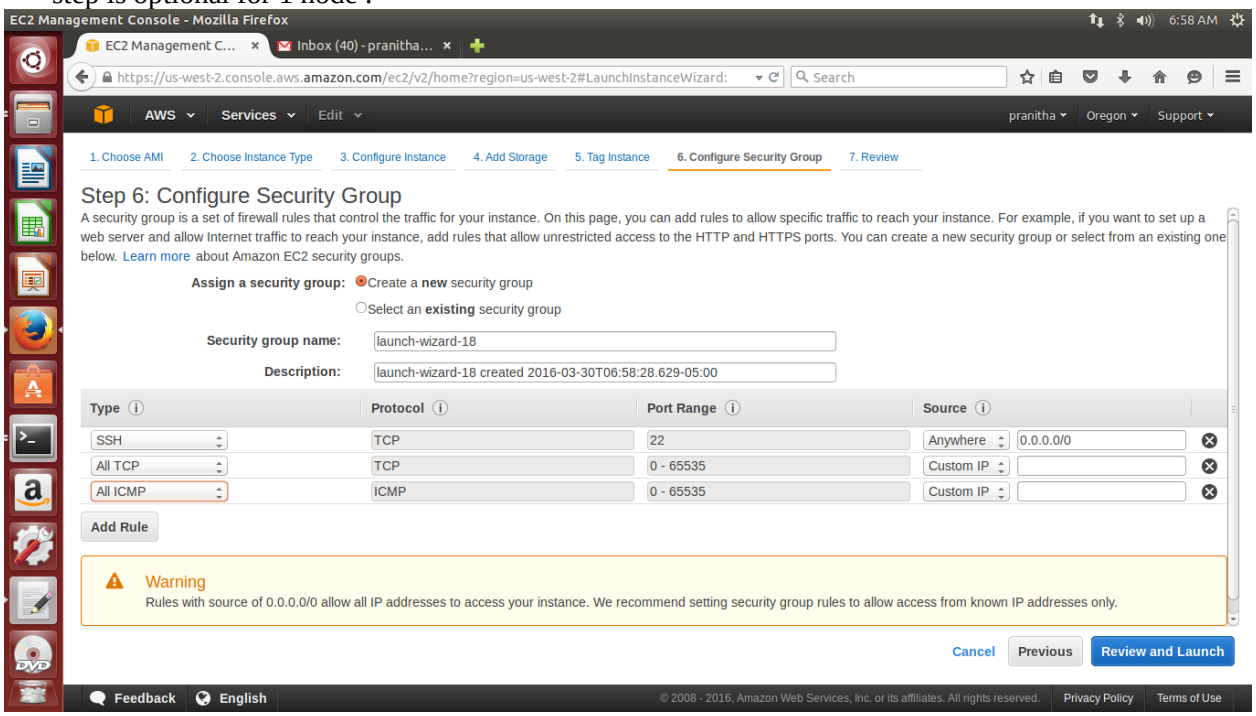
- Then we configure the instance i.e. we can choose the instance to be either spot instance or on Demand instance. In this assignment spot instance has been used.



- Add additional storage to selected instance.



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



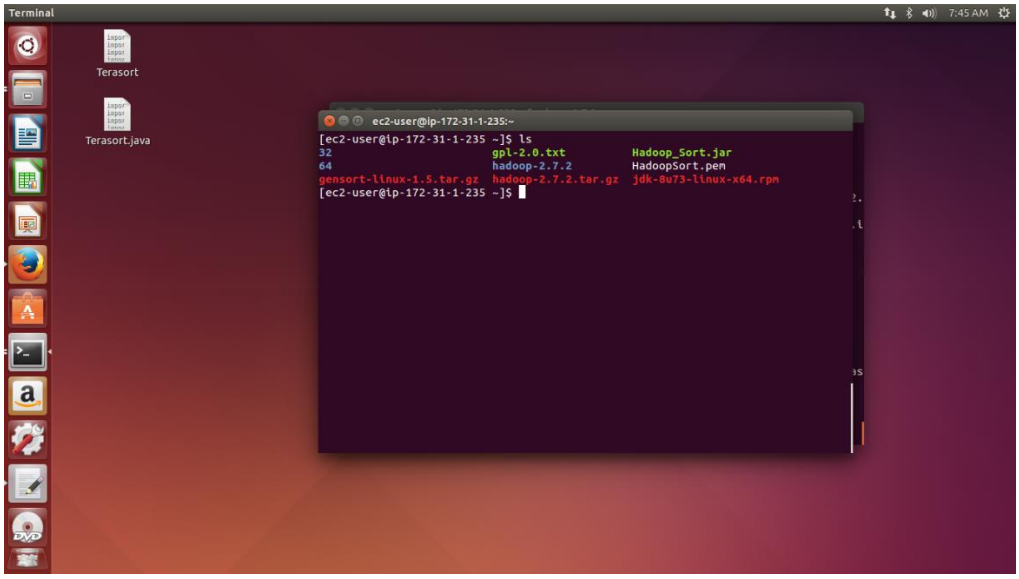
- Launch the spot instance once the request is accepted the instance will start

To connect to the node:

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

After the connection is established:

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



1.SharedMemory:

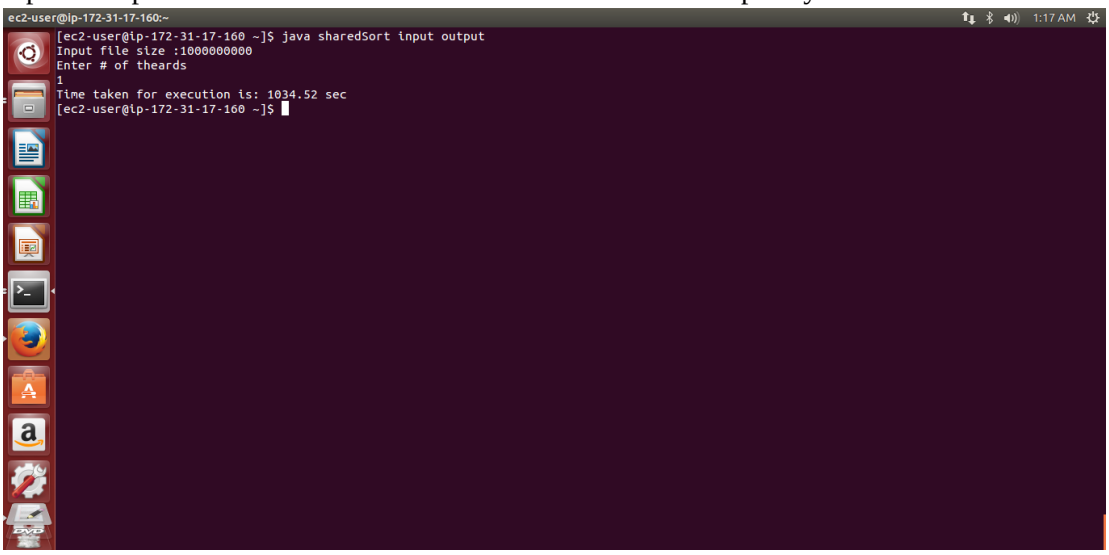
System Configuration:

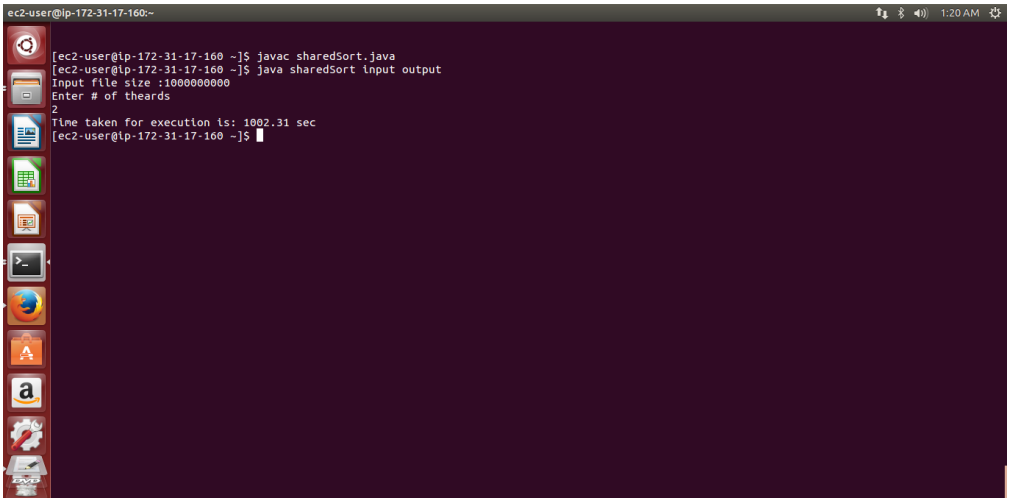
JAVA VERSION:1.8

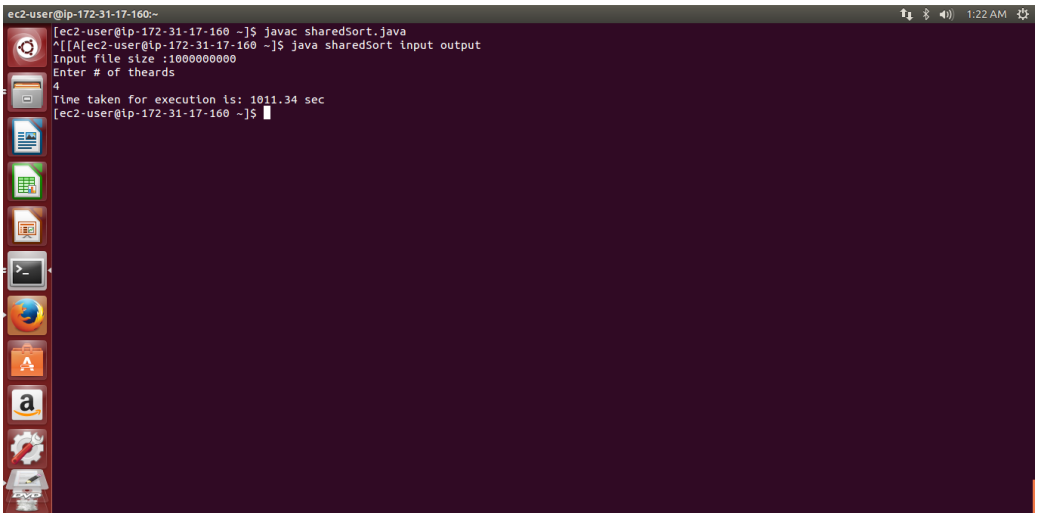
INSTANCE TYPE: c3. Large (10GB) UBUNTU

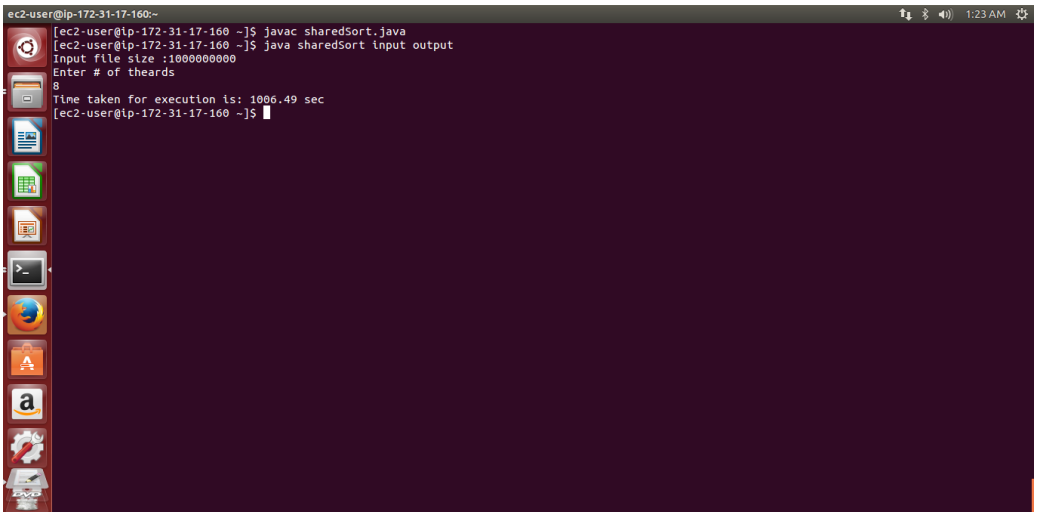
Implementation:

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





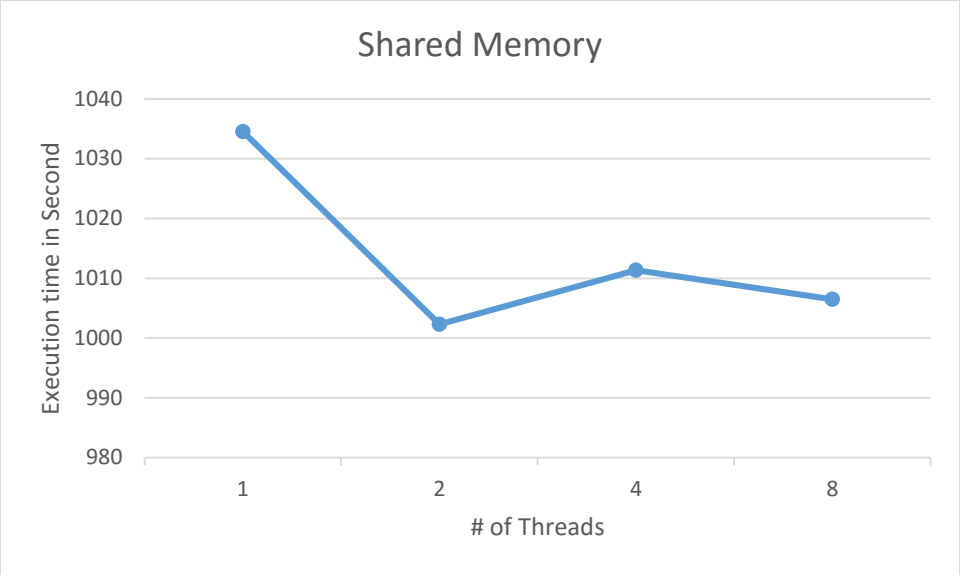




Performance:

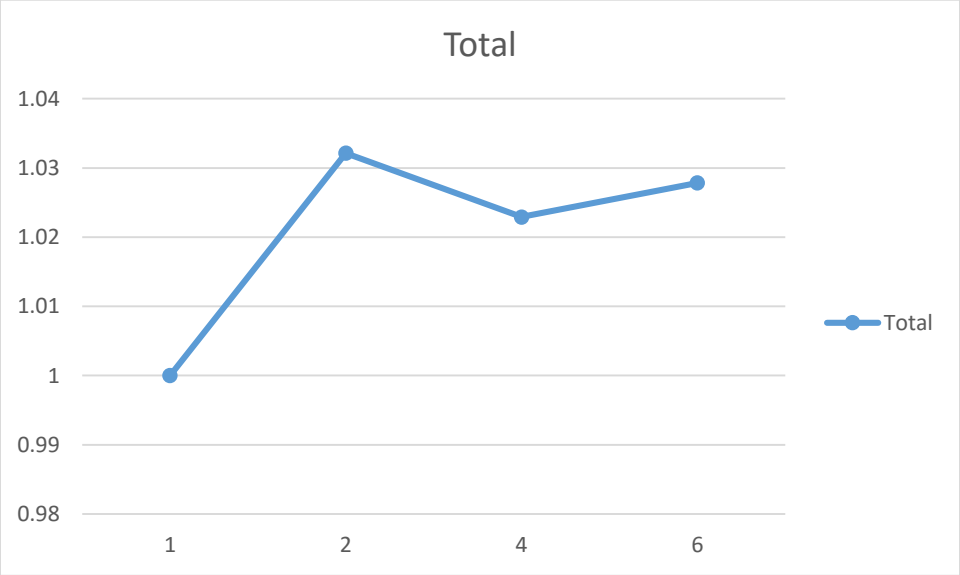
10GB dataset

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



Shared Memory Speedup using 1 thread value

# of thread	Execution time	2 Thread value	Speed-up factor
1	**1034.52**	**1034.52**	**1**
2	**1002.31**	**1034.52**	**1.03213577**
4	**1011.34**	**1034.52**	**1.02292009**
8	**1006.49**	**1034.52**	**1.02784926**



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

2.Hadoop:

System Configuration:

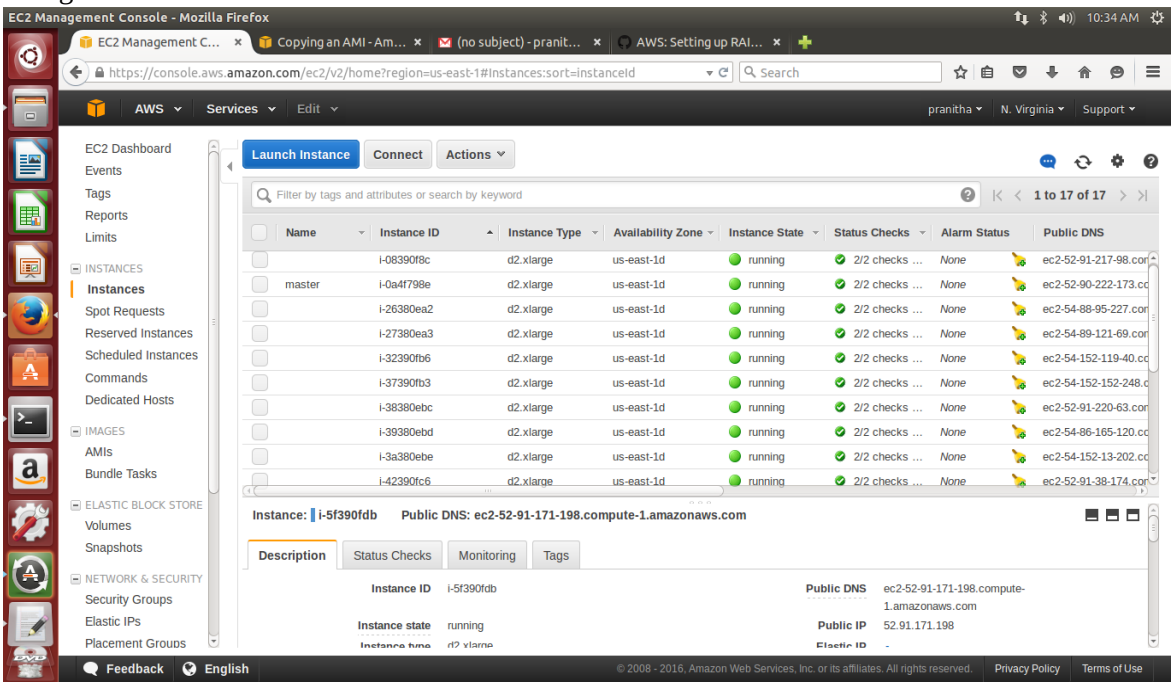
VERSION: Hadoop 2.7.2

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

Hadoop and Aws Configuration:

For 16 Node AWS cluster:

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



Configuration of Hadoop:

Files that have to be modified to complete the Hadoop setup

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

1. ~/.bashrc:

```
#HADOOP VARIABLES START
```

export JAVA_HOME=/usr/java/latest

export HADOOP_INSTALL=/usr/ubuntu/installs/hadoop

export PATH=$PATH:$HADOOP_INSTALL/bin

export PATH=$PATH:$HADOOP_INSTALL/sbin

export HADOOP_MAPRED_HOME=$HADOOP_INSTALL

export HADOOP_COMMON_HOME=$HADOOP_INSTALL

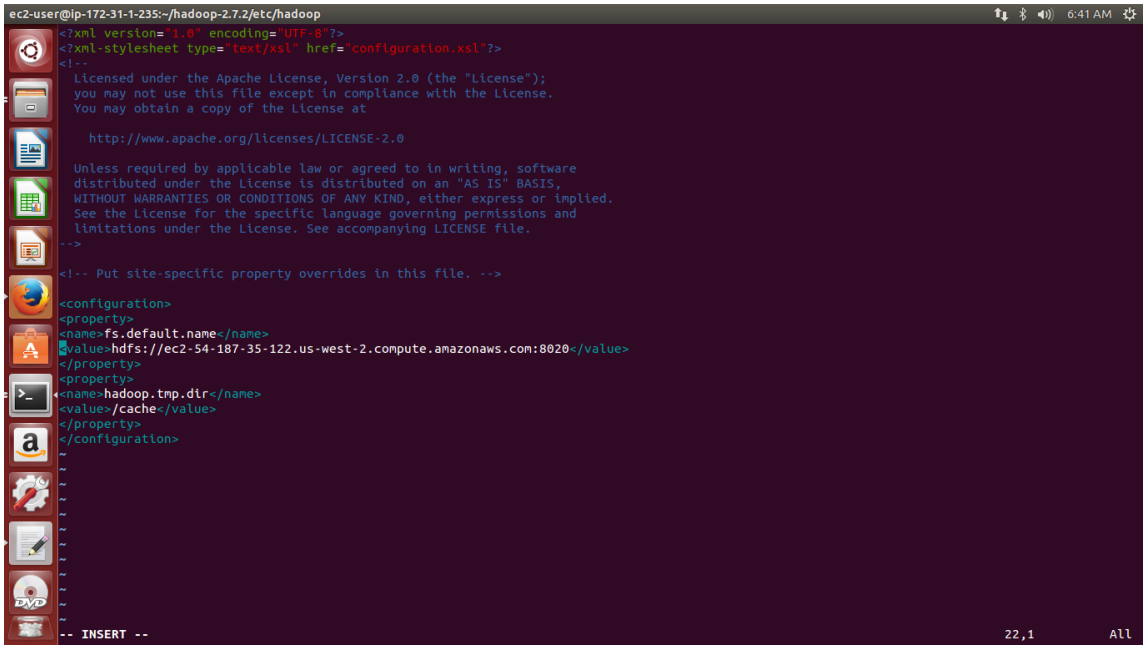
export HADOOP_HDFS_HOME=$HADOOP_INSTALL

export YARN_HOME=$HADOOP_INSTALL

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native

export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"

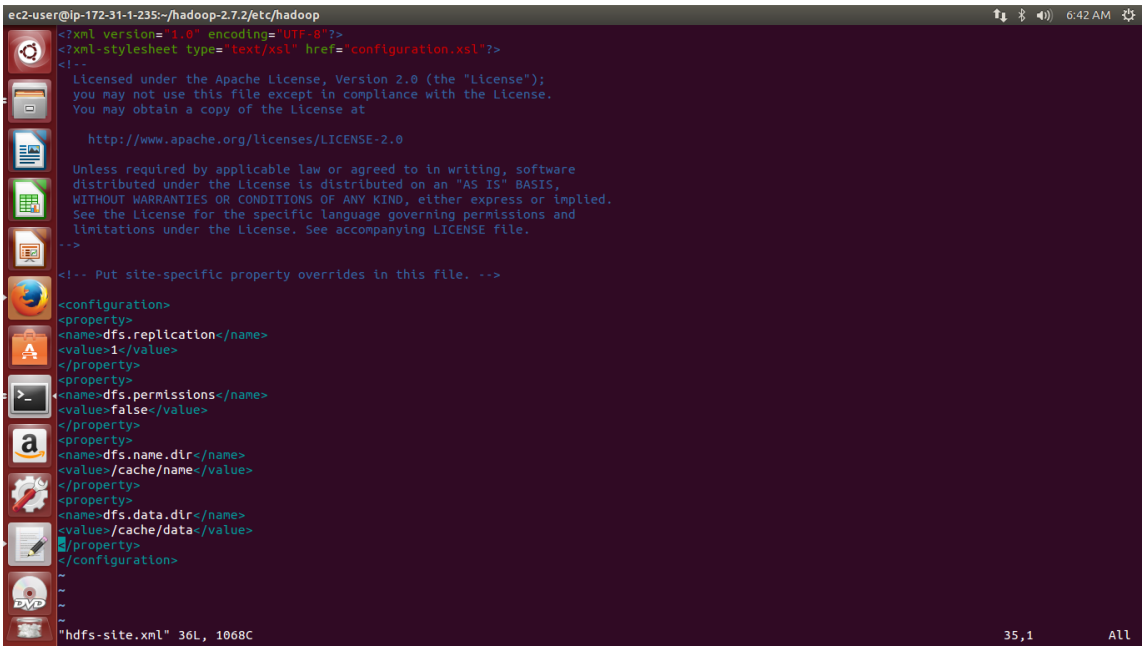
2.core-site.xml



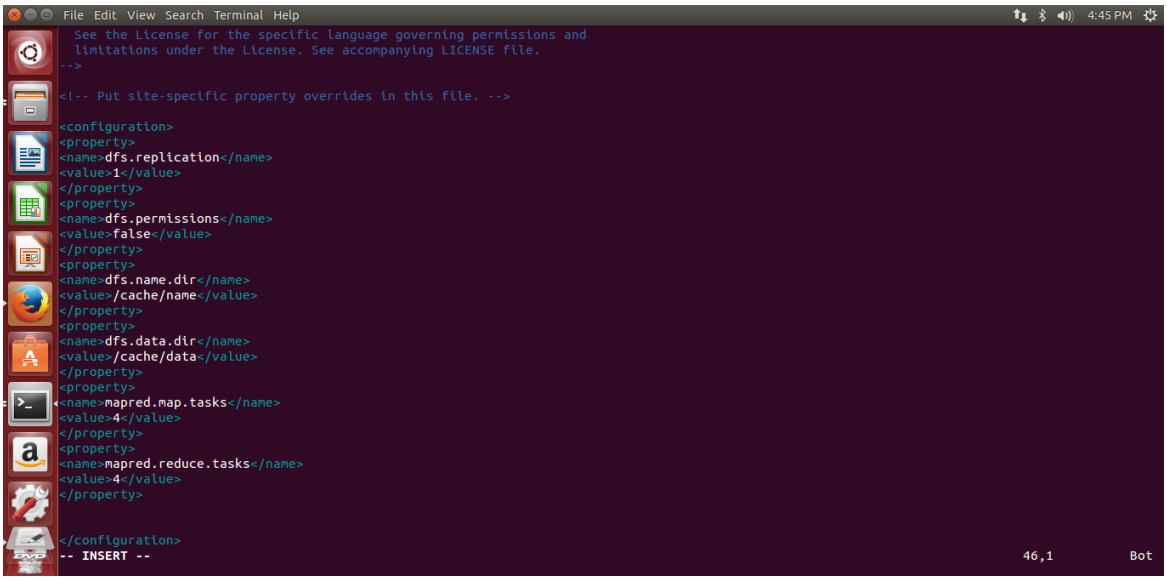
- Same configuration for both 1 node and 16 node cluster.
- In 16 node cluster all the 16 slaves will have master configuration only

3.hdfs-site.xml

For 1 node



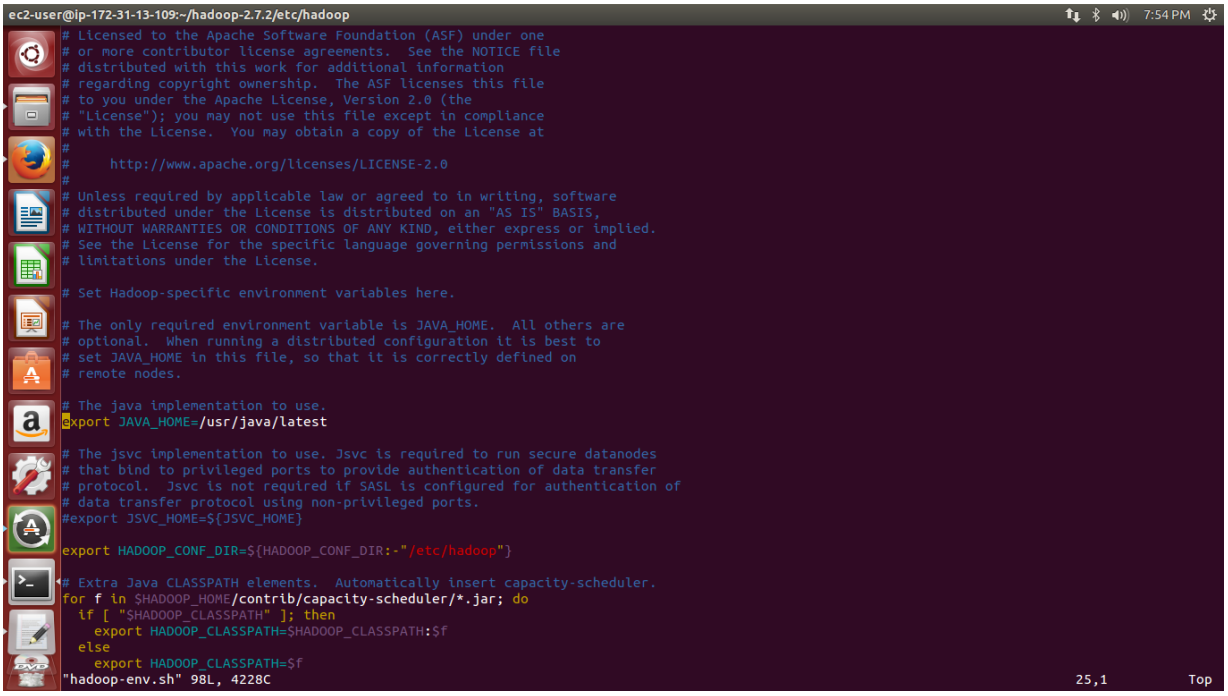
For 16 node



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

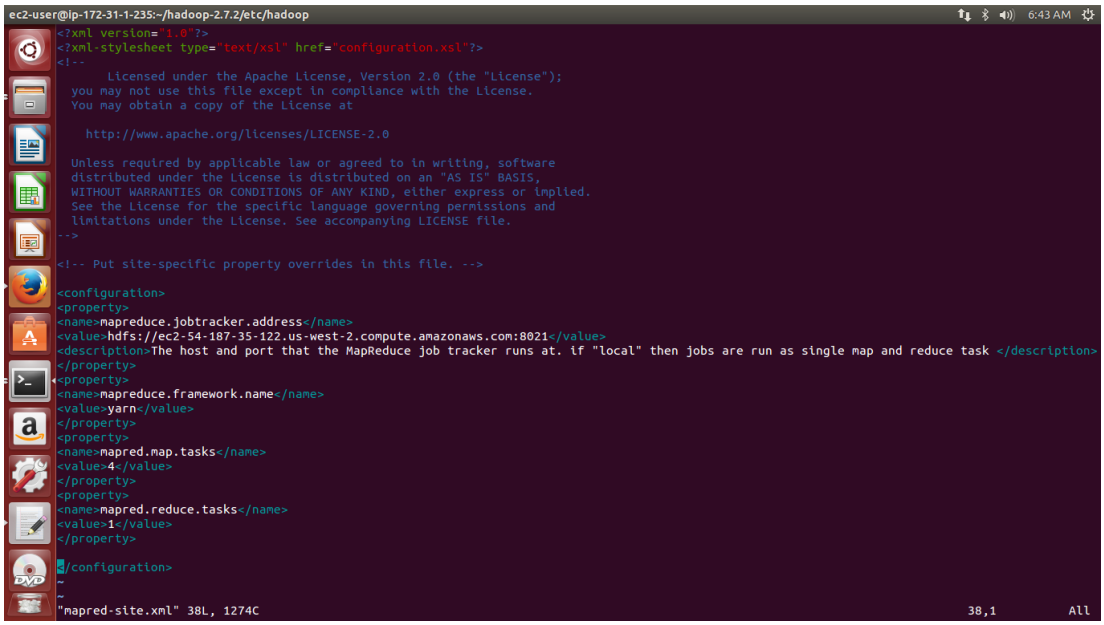
16 Node: sudo mdadm --create --verbose /dev/md0 --level=stripe --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd
Followed by :
sudo mkfs.ext4 /dev/md0
sudo mount -t ext4 -o noatime,nodiratime,rw /dev/md0 /cache

4.Hadoop-env.sh

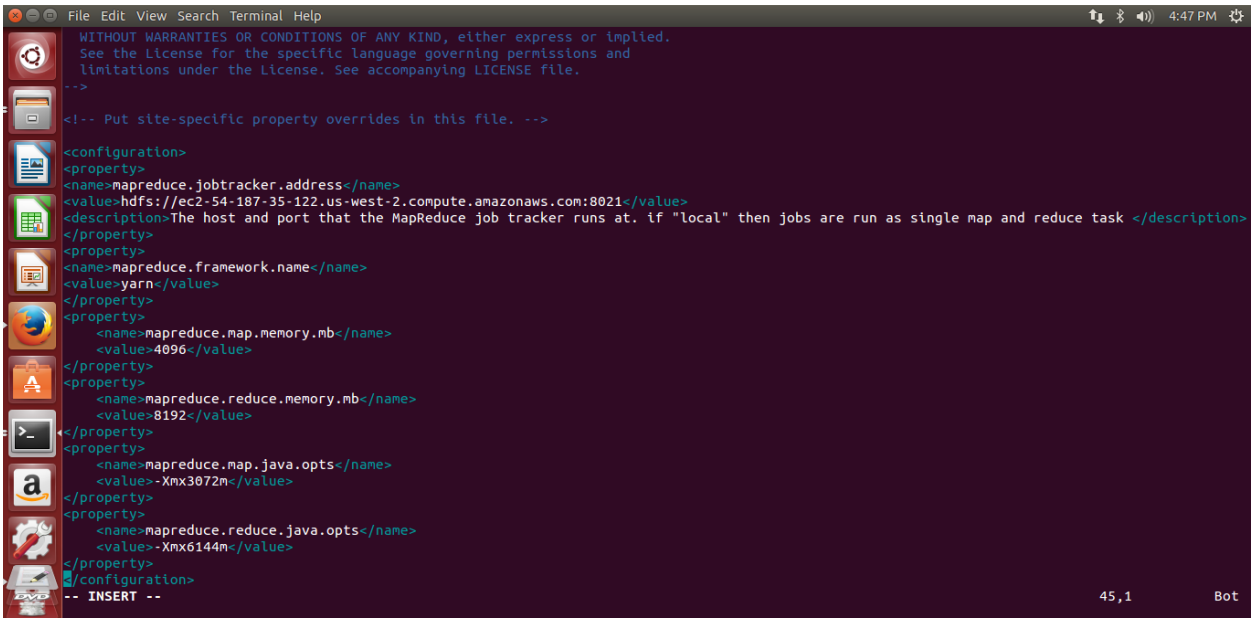


5.mapred-site

For 1 node

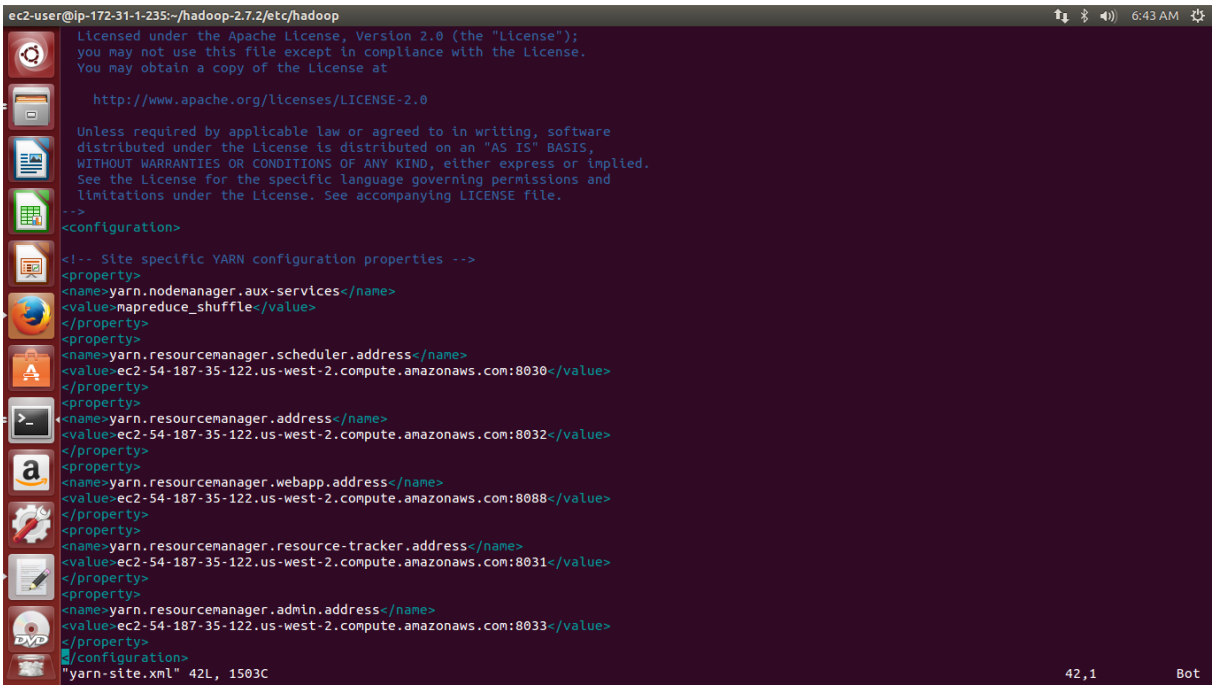


For 16 node

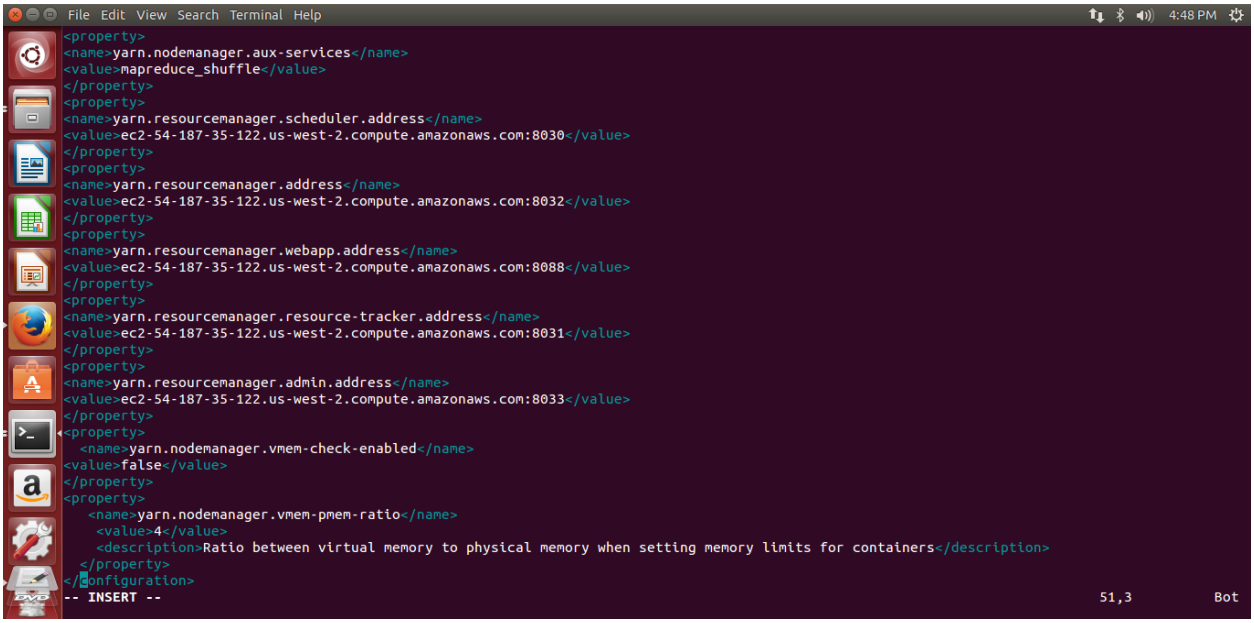


6.yarn-site.xml

For 1 node

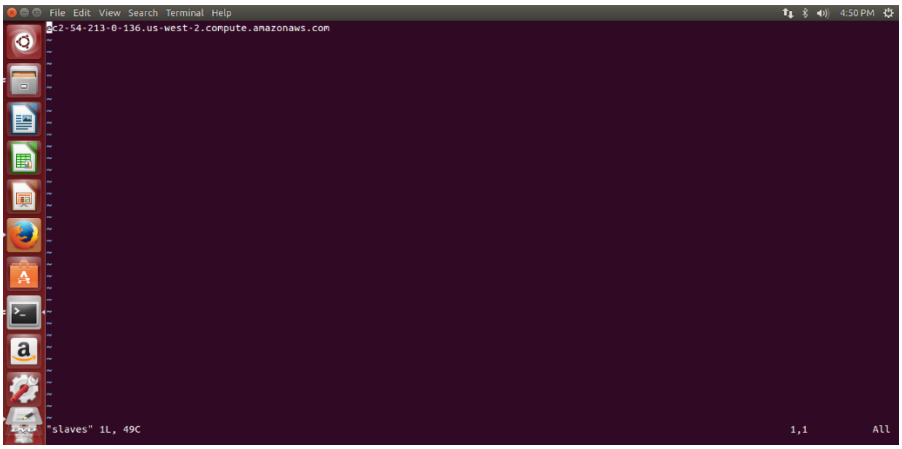


for 16 node

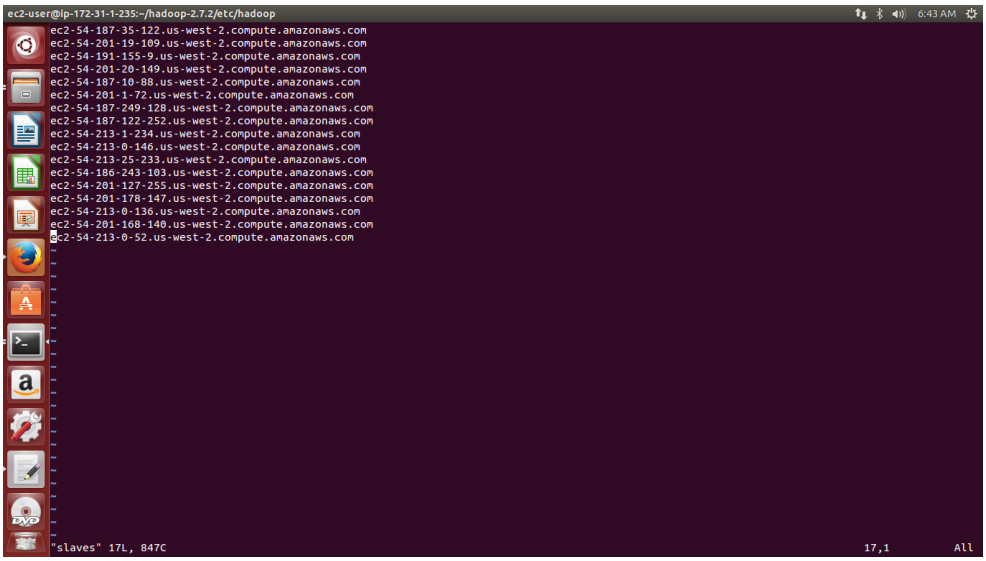


7.slaves

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



Master node of 16 node cluster



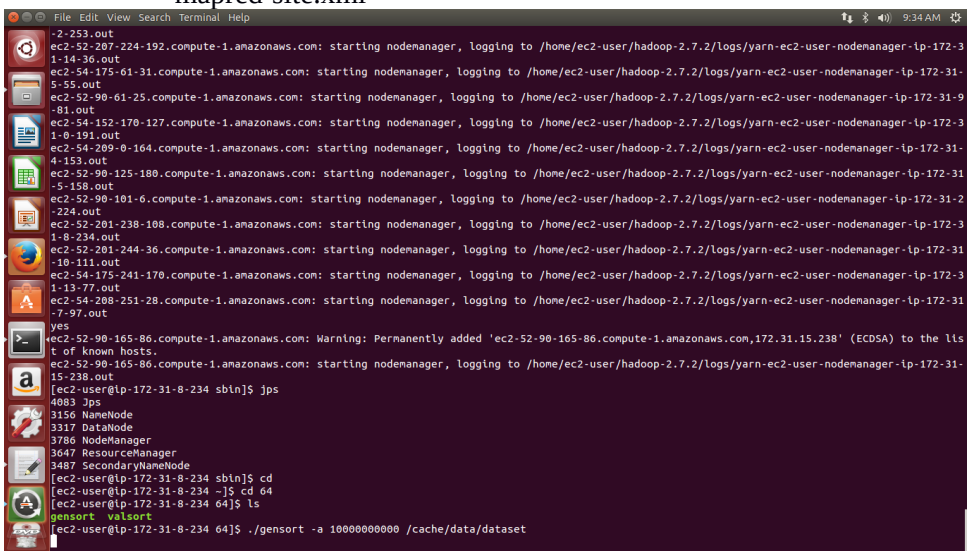
Implementation:

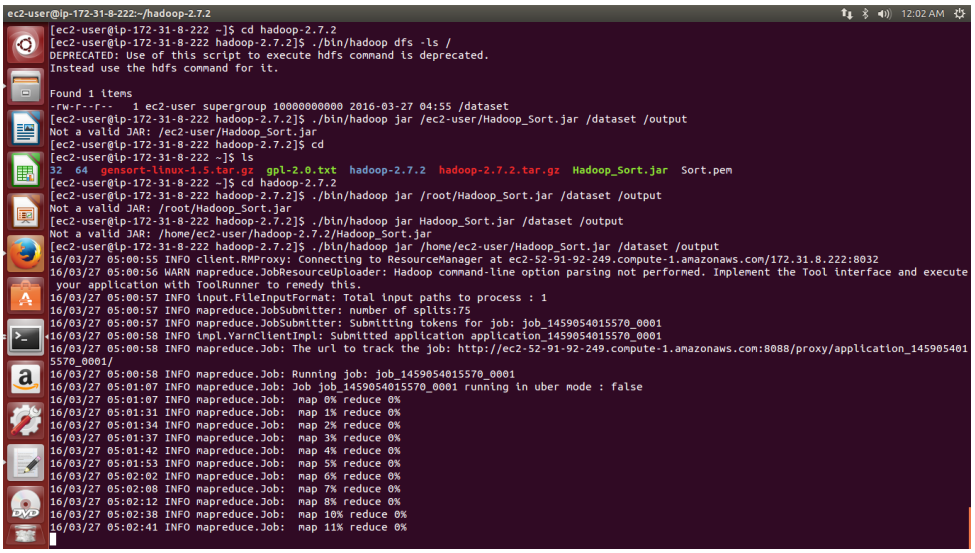
1. Hadoop MapReduce has been implemented in java.
2. Launch the instance and do all necessary implementation stated above
3. Once the implementations are done, add storage depending on file size to be sorted
4. Get the jar file that has been implemented using SCP command from local system to instance.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

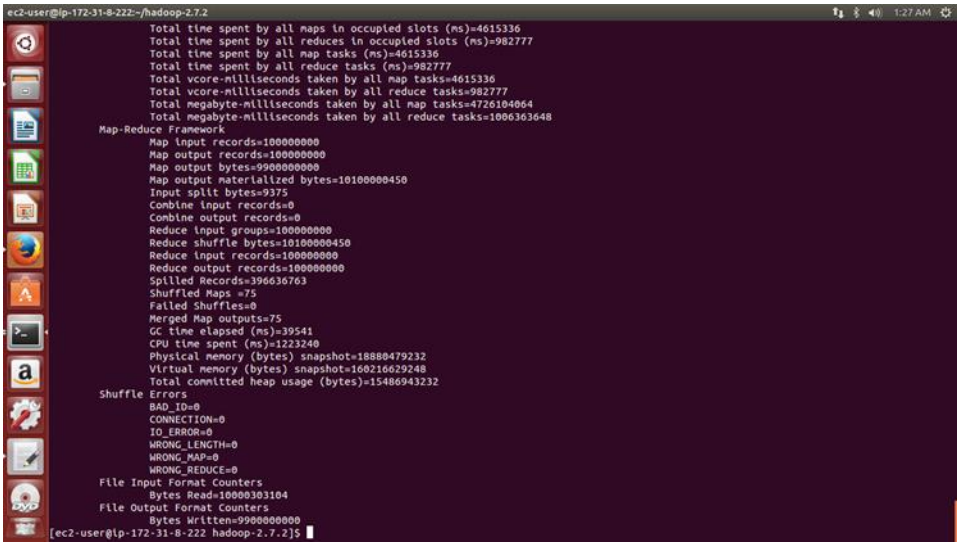
10. In between this mapper and reducer phase there is a shuffle/sort phase where the data is sorted depending on the key values and is sent in to the mapper.
11. The output is sent into the HDFS which can be accessed using get command
12. This output value is checked using ./valsort in gensort

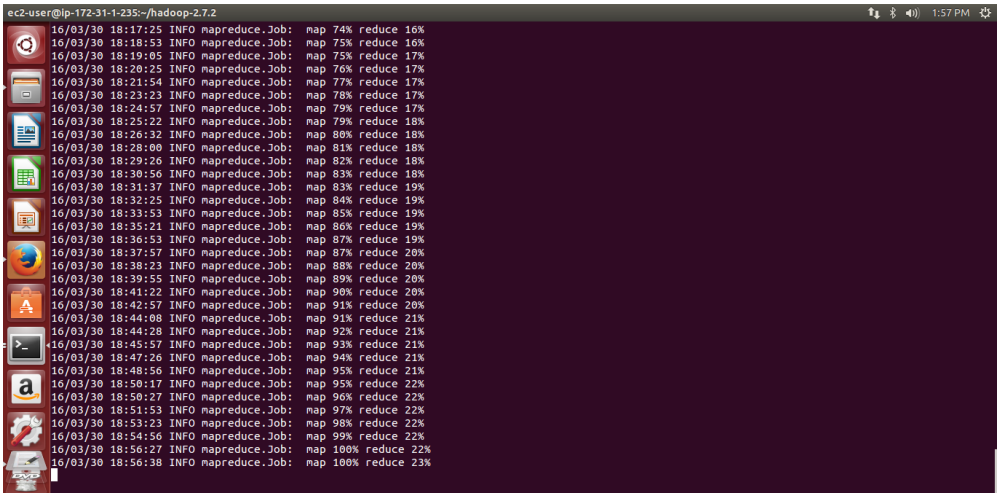
1) What is a Master node? What is a Slaves node?
    - The master node is the one which stores a lot of data and runs parallel computations on data, it controls all the slaves.It keeps track of all the all slaves it controls namenode and datanode,it divides the work between the slaves.
    - The slave node has only information about itself it follows the commands of the master takes up the assigned task by the master,it does the work of storing data and running the computations
2) Why do we need to set unique available ports to those configuration files on a shared environment? What errors or side-effects will show if we use same port number for each user?
    - If ports are same it would interfere with the operations or computations of each other, that to in shared environment ,multiple processes will be running if they share same port it lead insecure operations along with it the operations are interfered and also will consume more time.

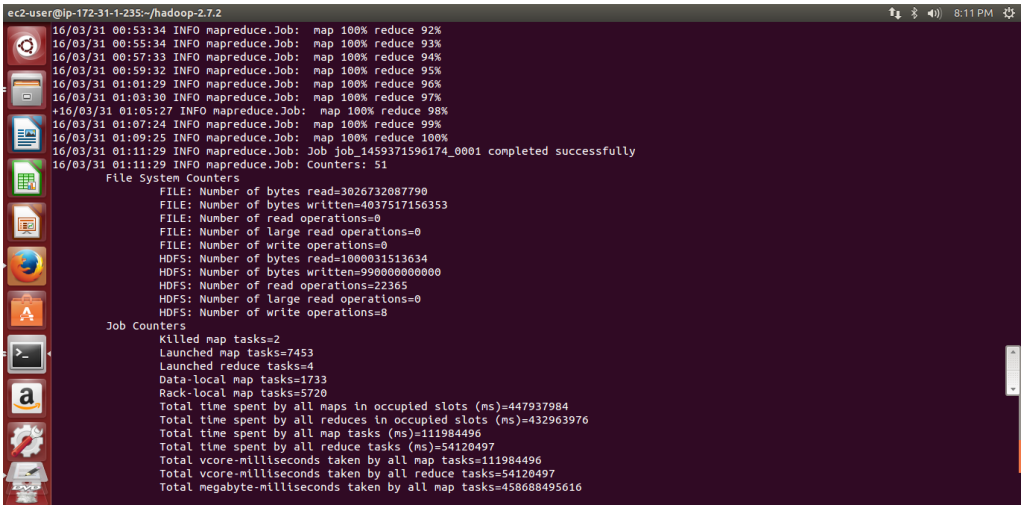
3) How can we change the number of mappers and reducers from the configuration file?
    - It can be done by adding two properties mapred.map.tasks amd mapred.reduce.task in mapred-site.xml

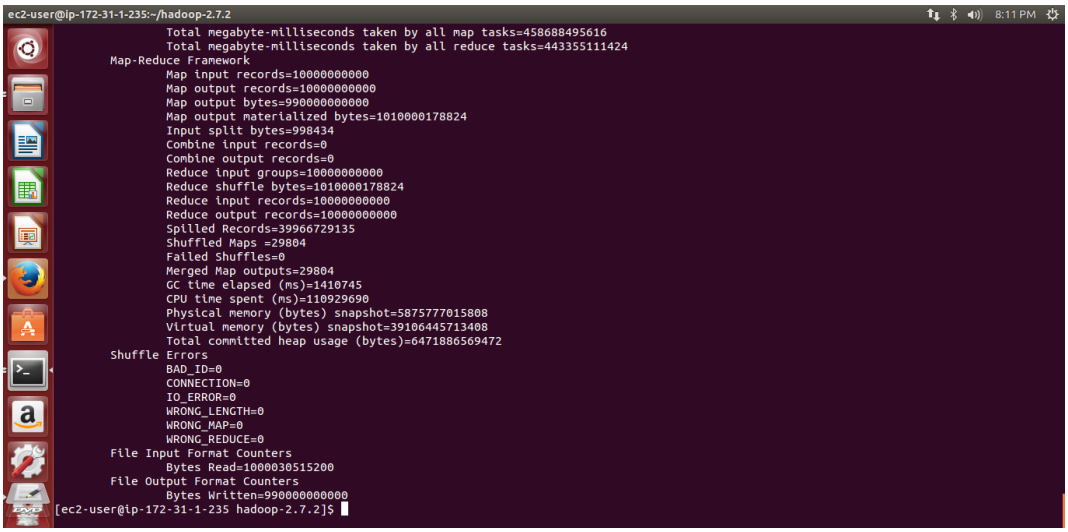


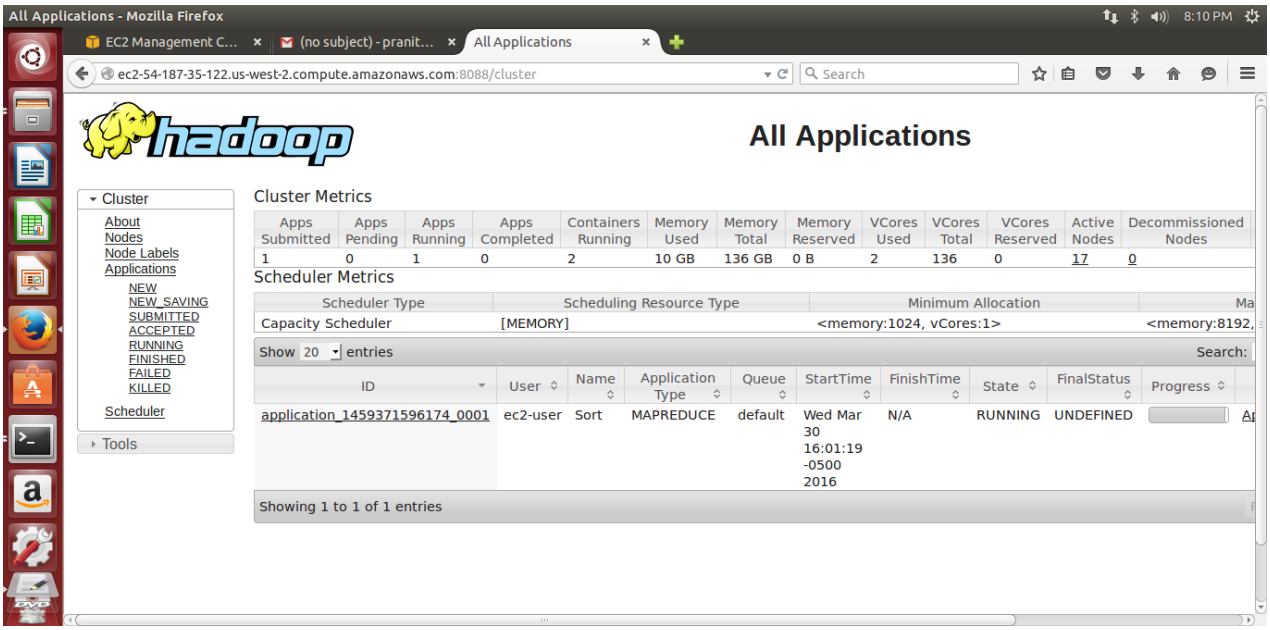

ec2-user@ip-172-31-8-222:~/hadoop-2.7.2
12:02 AM
[ec2-user@ip-172-31-8-222 ~]$ cd hadoop-2.7.2
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop dfs -ls /
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Found 1 items
-rw-r--r-- 1 ec2-user supergroup 10000000000 2016-03-27 04:55 /dataset
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar /ec2-user/Hadoop_Sort.jar /dataset /output
Not a valid JAR: /ec2-user/Hadoop_Sort.jar
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ cd
[ec2-user@ip-172-31-8-222 ~]$ ls
32 64 gensort-linux-1.5.tar.gz gpl-2.0.txt hadoop-2.7.2 hadoop-2.7.2.tar.gz Hadoop_Sort.jar Sort.pem
[ec2-user@ip-172-31-8-222 ~]$ cd hadoop-2.7.2
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar /root/Hadoop_Sort.jar /dataset /output
Not a valid JAR: /root/Hadoop_Sort.jar
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar Hadoop_Sort.jar /dataset /output
Not a valid JAR: /home/ec2-user/hadoop-2.7.2/Hadoop_Sort.jar
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$ ./bin/hadoop jar /home/ec2-user/Hadoop_Sort.jar /dataset /output
16/03/27 05:00:55 INFO client.RMProxy: Connecting to ResourceManager at ec2-52-91-92-249.compute-1.amazonaws.com/172.31.8.222:8032
16/03/27 05:00:56 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
16/03/27 05:00:57 INFO input.FileInputFormat: Total input paths to process : 1
16/03/27 05:00:57 INFO mapreduce.JobSubmitter: number of splits:75
16/03/27 05:00:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459054015570_0001
16/03/27 05:00:58 INFO impl.YarnClientImpl: Submitted application application_1459054015570_0001
16/03/27 05:00:58 INFO mapreduce.Job: The url to track the job: http://ec2-52-91-92-249.compute-1.amazonaws.com:8088/proxy/application_1459054015570_0001/
16/03/27 05:00:58 INFO mapreduce.Job: Running job: job_1459054015570_0001
16/03/27 05:01:07 INFO mapreduce.Job: Job job_1459054015570_0001 running in uber mode : false
16/03/27 05:01:07 INFO mapreduce.Job: map 0% reduce 0%
16/03/27 05:01:31 INFO mapreduce.Job: map 1% reduce 0%
16/03/27 05:01:34 INFO mapreduce.Job: map 2% reduce 0%
16/03/27 05:01:37 INFO mapreduce.Job: map 3% reduce 0%
16/03/27 05:01:42 INFO mapreduce.Job: map 4% reduce 0%
16/03/27 05:01:53 INFO mapreduce.Job: map 5% reduce 0%
16/03/27 05:02:02 INFO mapreduce.Job: map 6% reduce 0%
16/03/27 05:02:08 INFO mapreduce.Job: map 7% reduce 0%
16/03/27 05:02:12 INFO mapreduce.Job: map 8% reduce 0%
16/03/27 05:02:38 INFO mapreduce.Job: map 10% reduce 0%
16/03/27 05:02:41 INFO mapreduce.Job: map 11% reduce 0%

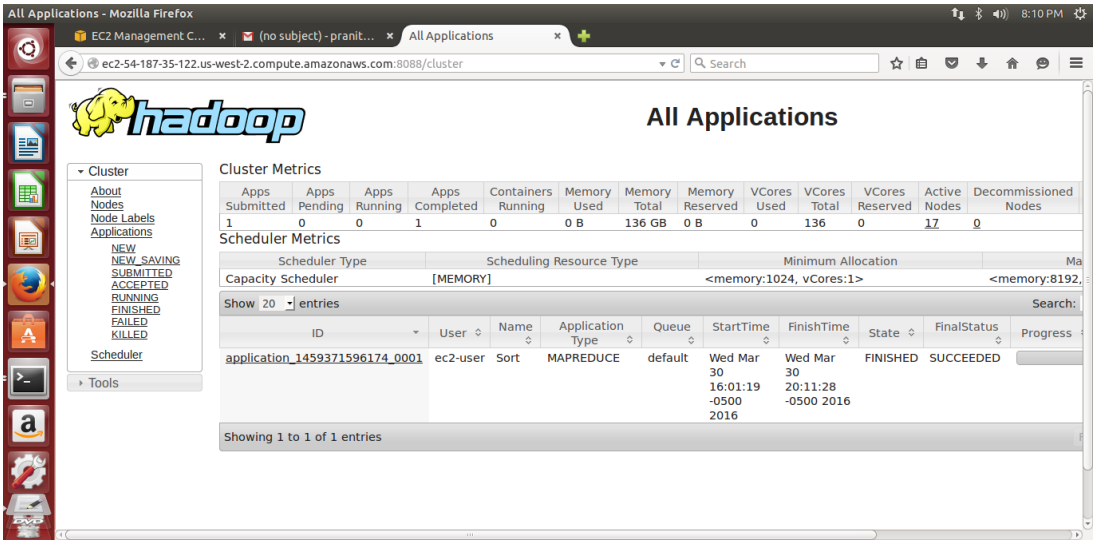

ec2-user@ip-172-31-8-222:~/hadoop-2.7.2
1:27 AM
Total time spent by all maps in occupied slots (ms)=4615336
Total time spent by all reduces in occupied slots (ms)=982777
Total time spent by all map tasks (ms)=4615336
Total time spent by all reduce tasks (ms)=982777
Total vcore-milliseconds taken by all map tasks=4615336
Total vcore-milliseconds taken by all reduce tasks=982777
Total megabyte-milliseconds taken by all map tasks=4726104064
Total megabyte-milliseconds taken by all reduce tasks=1006363648
Map-Reduce Framework
Map input records=100000000
Map output records=100000000
Map output bytes=9900000000
Map output materialized bytes=10100000450
Input split bytes=9375
Combine input records=0
Combine output records=0
Reduce input groups=100000000
Reduce shuffle bytes=10100000450
Reduce input records=100000000
Reduce output records=100000000
Spilled Records=396636763
Shuffled Maps =75
Failed Shuffles=0
Merged Map outputs=75
GC time elapsed (ms)=39541
CPU time spent (ms)=1223240
Physical memory (bytes) snapshot=18880479232
Virtual memory (bytes) snapshot=160216629248
Total committed heap usage (bytes)=15486943232
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=10000303104
File Output Format Counters
Bytes Written=9900000000
[ec2-user@ip-172-31-8-222 hadoop-2.7.2]$


ec2-user@ip-172-31-1-235:~/hadoop-2.7.2
1:57 PM
16/03/30 18:17:25 INFO mapreduce.Job: map 74% reduce 16%
16/03/30 18:18:53 INFO mapreduce.Job: map 75% reduce 16%
16/03/30 18:19:05 INFO mapreduce.Job: map 75% reduce 17%
16/03/30 18:20:25 INFO mapreduce.Job: map 76% reduce 17%
16/03/30 18:21:54 INFO mapreduce.Job: map 77% reduce 17%
16/03/30 18:23:23 INFO mapreduce.Job: map 78% reduce 17%
16/03/30 18:24:57 INFO mapreduce.Job: map 79% reduce 17%
16/03/30 18:25:22 INFO mapreduce.Job: map 79% reduce 18%
16/03/30 18:26:32 INFO mapreduce.Job: map 80% reduce 18%
16/03/30 18:28:00 INFO mapreduce.Job: map 81% reduce 18%
16/03/30 18:29:26 INFO mapreduce.Job: map 82% reduce 18%
16/03/30 18:30:56 INFO mapreduce.Job: map 83% reduce 18%
16/03/30 18:31:37 INFO mapreduce.Job: map 83% reduce 19%
16/03/30 18:32:25 INFO mapreduce.Job: map 84% reduce 19%
16/03/30 18:33:53 INFO mapreduce.Job: map 85% reduce 19%
16/03/30 18:35:21 INFO mapreduce.Job: map 86% reduce 19%
16/03/30 18:36:53 INFO mapreduce.Job: map 87% reduce 19%
16/03/30 18:37:57 INFO mapreduce.Job: map 87% reduce 20%
16/03/30 18:38:23 INFO mapreduce.Job: map 88% reduce 20%
16/03/30 18:39:55 INFO mapreduce.Job: map 89% reduce 20%
16/03/30 18:41:22 INFO mapreduce.Job: map 90% reduce 20%
16/03/30 18:42:57 INFO mapreduce.Job: map 91% reduce 20%
16/03/30 18:44:08 INFO mapreduce.Job: map 91% reduce 21%
16/03/30 18:44:28 INFO mapreduce.Job: map 92% reduce 21%
16/03/30 18:45:57 INFO mapreduce.Job: map 93% reduce 21%
16/03/30 18:47:26 INFO mapreduce.Job: map 94% reduce 21%
16/03/30 18:48:56 INFO mapreduce.Job: map 95% reduce 21%
16/03/30 18:50:17 INFO mapreduce.Job: map 95% reduce 22%
16/03/30 18:50:27 INFO mapreduce.Job: map 96% reduce 22%
16/03/30 18:51:53 INFO mapreduce.Job: map 97% reduce 22%
16/03/30 18:53:23 INFO mapreduce.Job: map 98% reduce 22%
16/03/30 18:54:56 INFO mapreduce.Job: map 99% reduce 22%
16/03/30 18:56:27 INFO mapreduce.Job: map 100% reduce 22%
16/03/30 18:56:38 INFO mapreduce.Job: map 100% reduce 23%


ec2-user@ip-172-31-1-235:~/hadoop-2.7.2
8:11 PM
16/03/31 00:53:34 INFO mapreduce.Job: map 100% reduce 92%
16/03/31 00:55:34 INFO mapreduce.Job: map 100% reduce 93%
16/03/31 00:57:33 INFO mapreduce.Job: map 100% reduce 94%
16/03/31 00:59:32 INFO mapreduce.Job: map 100% reduce 95%
16/03/31 01:01:29 INFO mapreduce.Job: map 100% reduce 96%
16/03/31 01:03:30 INFO mapreduce.Job: map 100% reduce 97%
+16/03/31 01:05:27 INFO mapreduce.Job: map 100% reduce 98%
16/03/31 01:07:24 INFO mapreduce.Job: map 100% reduce 99%
16/03/31 01:09:25 INFO mapreduce.Job: map 100% reduce 100%
16/03/31 01:11:29 INFO mapreduce.Job: Job job_1459371596174_0001 completed successfully
16/03/31 01:11:29 INFO mapreduce.Job: Counters: 51
File System Counters
FILE: Number of bytes read=3026732087790
FILE: Number of bytes written=4037517156353
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=1000031513634
HDFS: Number of bytes written=990000000000
HDFS: Number of read operations=22365
HDFS: Number of large read operations=0
HDFS: Number of write operations=8
Job Counters
Killed map tasks=2
Launched map tasks=7453
Launched reduce tasks=4
Data-local map tasks=1733
Rack-local map tasks=5720
Total time spent by all maps in occupied slots (ms)=447937984
Total time spent by all reduces in occupied slots (ms)=432963976
Total time spent by all map tasks (ms)=111984496
Total time spent by all reduce tasks (ms)=54120497
Total vcore-milliseconds taken by all map tasks=111984496
Total vcore-milliseconds taken by all reduce tasks=54120497
Total megabyte-milliseconds taken by all map tasks=458688495616


ec2-user@ip-172-31-1-235:~/hadoop-2.7.2
8:11 PM
Total megabyte-milliseconds taken by all map tasks=458688495616
Total megabyte-milliseconds taken by all reduce tasks=443355111424
Map-Reduce Framework
Map input records=10000000000
Map output records=10000000000
Map output bytes=990000000000
Map output materialized bytes=1010000178824
Input split bytes=998434
Combine input records=0
Combine output records=0
Reduce input groups=10000000000
Reduce shuffle bytes=1010000178824
Reduce input records=10000000000
Reduce output records=10000000000
Spilled Records=39966729135
Shuffled Maps =29804
Failed Shuffles=0
Merged Map outputs=29804
GC time elapsed (ms)=1410745
CPU time spent (ms)=110929690
Physical memory (bytes) snapshot=5875777015808
Virtual memory (bytes) snapshot=39106445713408
Total committed heap usage (bytes)=6471886569472
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=1000030515200
File Output Format Counters
Bytes Written=990000000000
[ec2-user@ip-172-31-1-235 hadoop-2.7.2]$





Performance:

Time taken to sort 10GB :736.8 sec

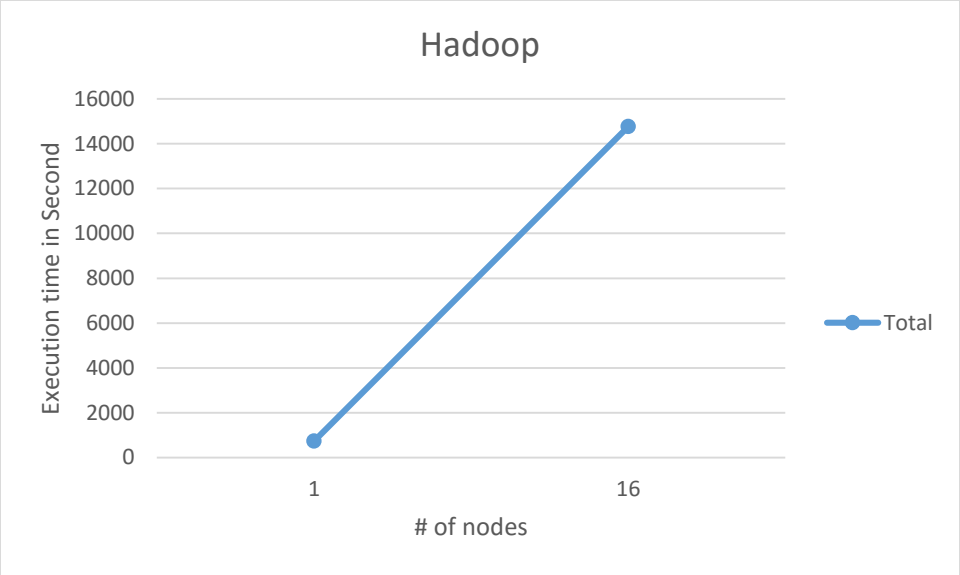
100GB: 1459.23 sec

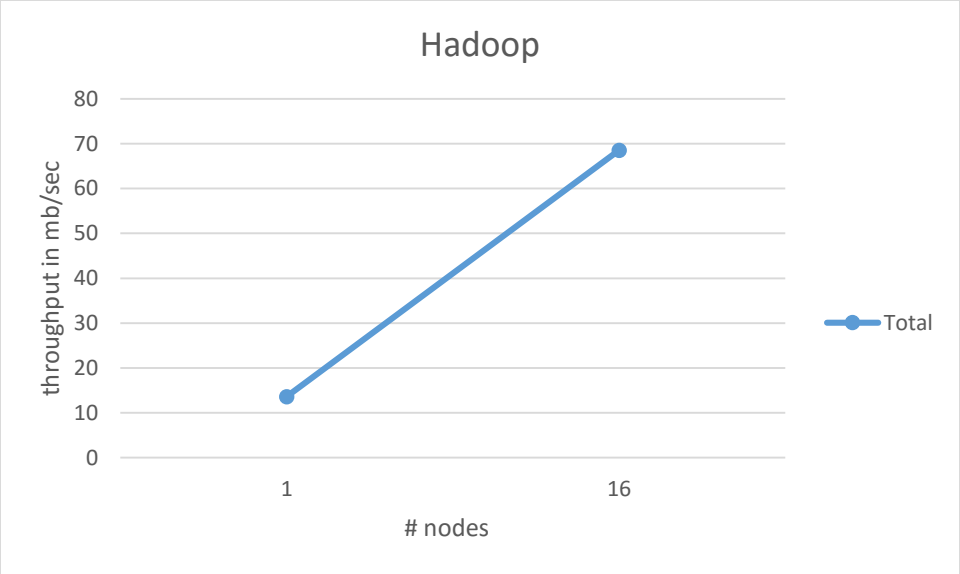
1TB: 14769 sec

Time taken to sort 10GB :13.5722041 mb/sec

100GB: 68.5292928 mb/sec

1TB: 677.093913 mb/sec


Hadoop
16000
14000
12000
10000
8000
6000
4000
2000
0
Execution time in Second
1
16
# of nodes
Total


Hadoop
80
70
60
50
40
30
20
10
0
throughput in mb/sec
1
16
nodes
Total

3.Spark:

System Configuration:

VERSION: spark-1.6.0-bin-hadoop2.6

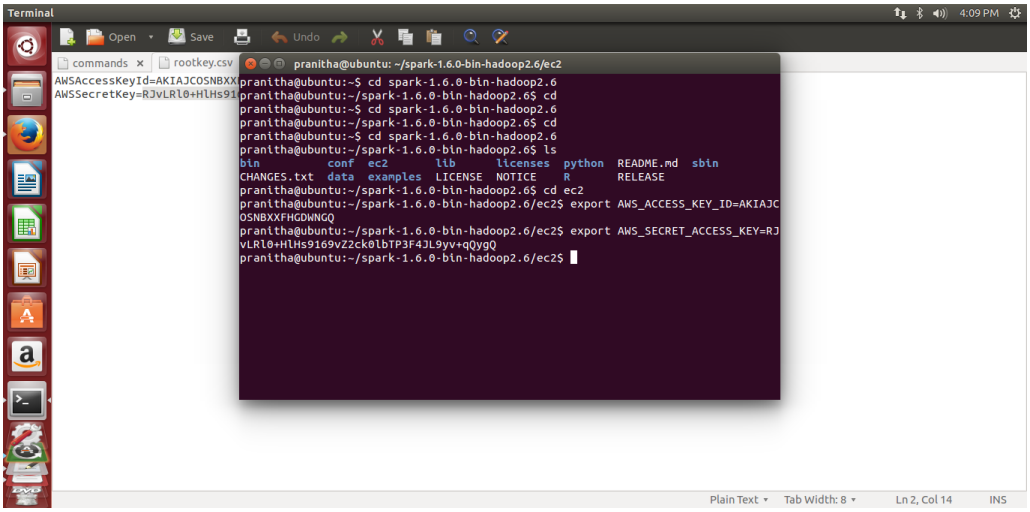
INSTANCE TYPE: c3. Large (10GB&100GB) & d2.Xlarge(1TB) UBUNTU

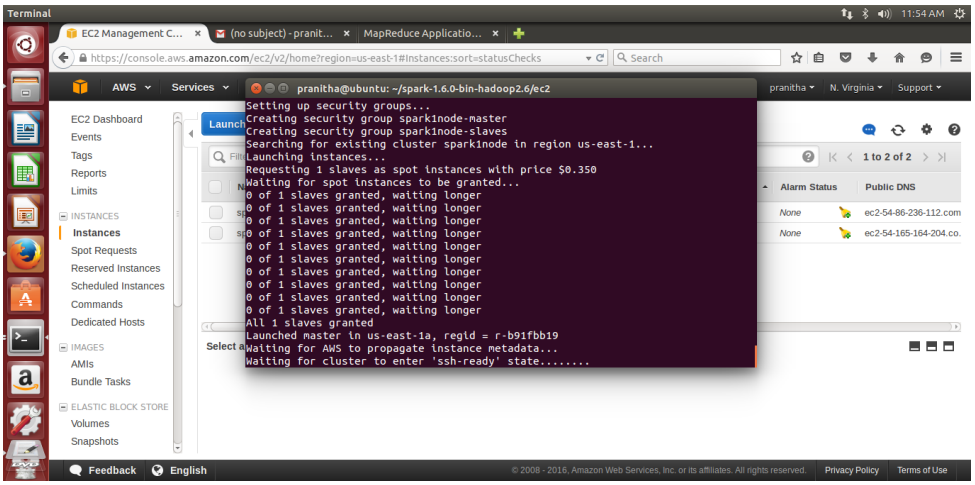
JDK:1.8

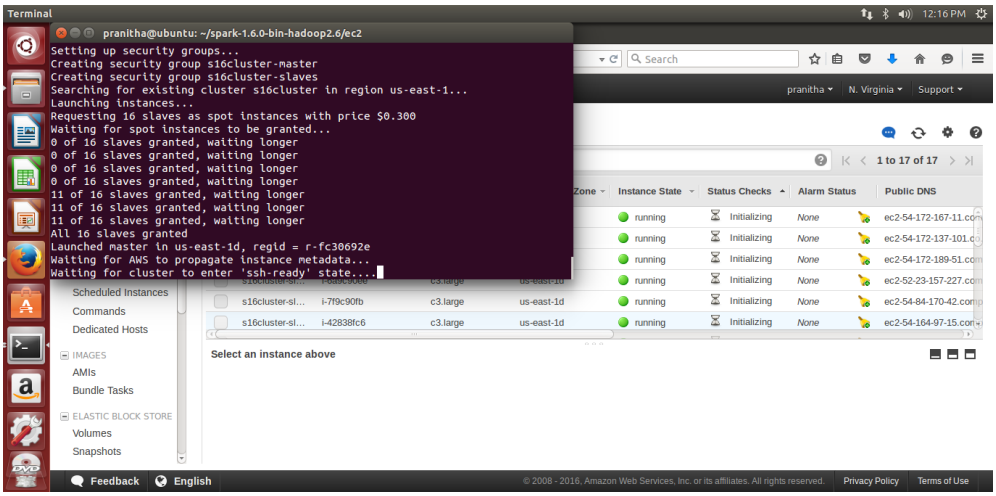
Spark Configuration:

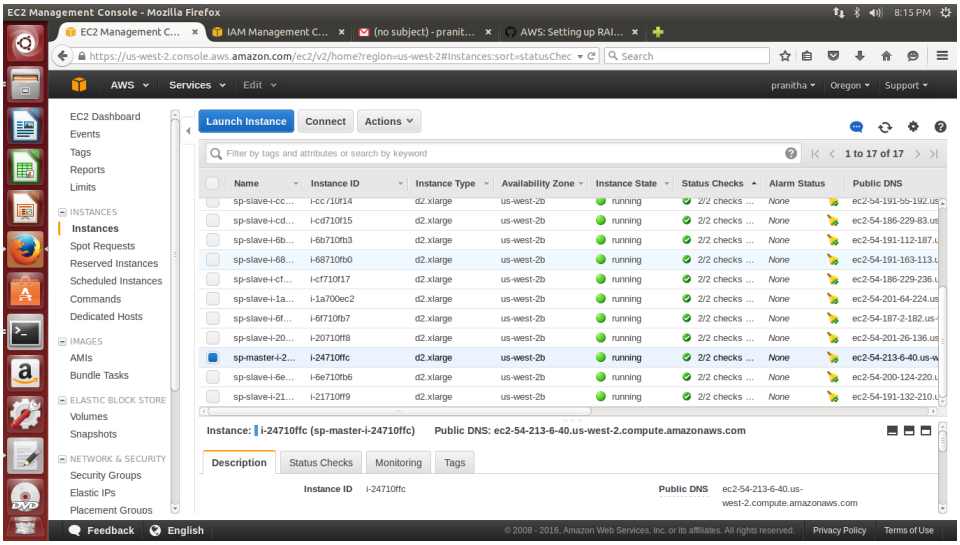
To implement spark in aws instance there are a set of commands that create appropriate setup for us to directly implement on it

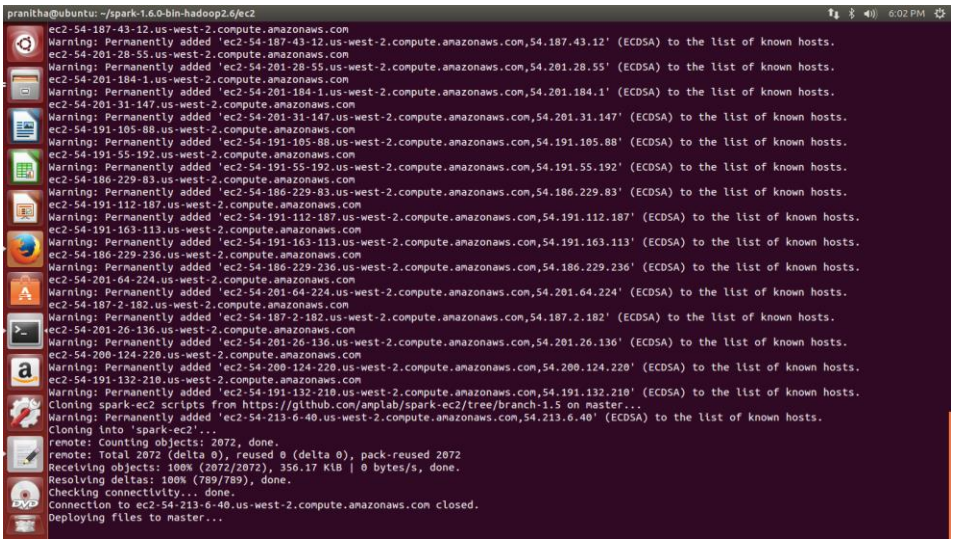
- export AWS_ACCESS_KEY_ID=<Access key id>
- export AWS_SECRET_ACCESS_KEY=<secret acces key values>
- ./spark-ec2 -k <key pair name> -i <keypair.pem> -s <no of slaves> -t <instance-type> -r us-west-2 --spot-price=0.2 launch <cluster-name>
- ./spark-ec2 -k <keypair> -i <keypair.pem> login <cluster-name>

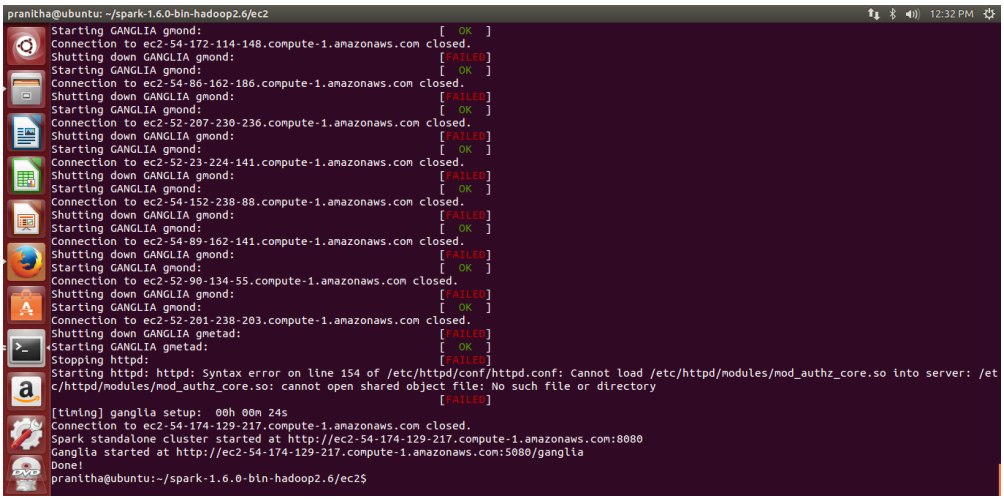





Terminal
pranitha@ubuntu: ~/spark-1.6.0-bin-hadoop2.6/ec2
Setting up security groups...
Creating security group s16cluster-master
Creating security group s16cluster-slaves
Searching for existing cluster s16cluster in region us-east-1...
Launching instances...
Requesting 16 slaves as spot instances with price $0.300
Waiting for spot instances to be granted...
0 of 16 slaves granted, waiting longer
0 of 16 slaves granted, waiting longer
0 of 16 slaves granted, waiting longer
0 of 16 slaves granted, waiting longer
11 of 16 slaves granted, waiting longer
11 of 16 slaves granted, waiting longer
11 of 16 slaves granted, waiting longer
All 16 slaves granted
Launched master in us-east-1d, regid = r-fc30692e
Waiting for AWS to propagate instance metadata...
Waiting for cluster to enter 'ssh-ready' state...
Instance State
Status Checks
Alarm Status
Public DNS
Select an instance above
Feedback
English


EC2 Management Console - Mozilla Firefox
Launch Instance
Connect
Actions
Instance: i-24710ffc (sp-master-i-24710ffc)
Public DNS: ec2-54-213-6-40.us-west-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Feedback
English

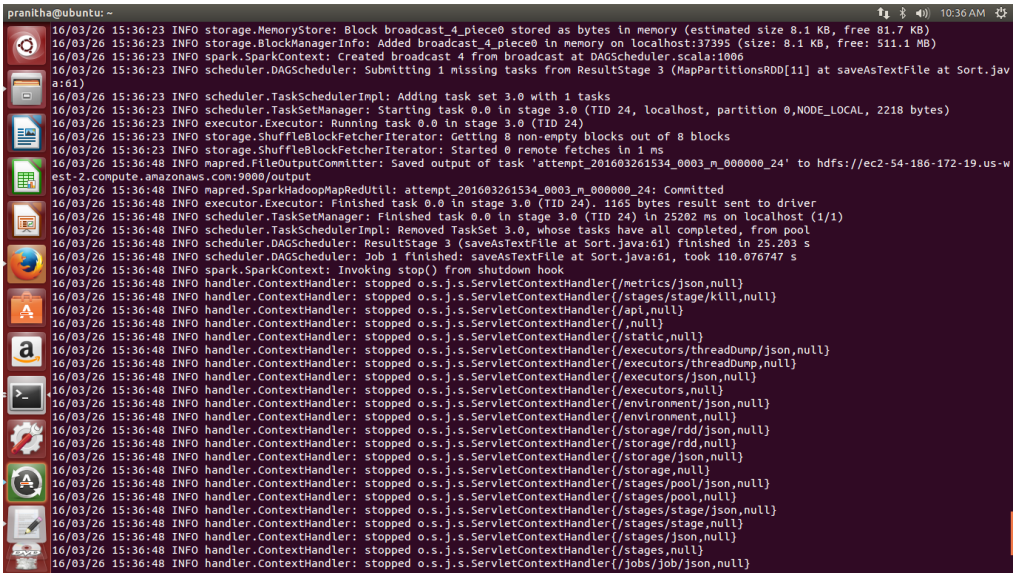

pranitha@ubuntu: ~/spark-1.6.0-bin-hadoop2.6/ec2
Warning: Permanently added 'ec2-54-187-43-12.us-west-2.compute.amazonaws.com,54.187.43.12' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-201-28-55.us-west-2.compute.amazonaws.com,54.201.28.55' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-201-184-1.us-west-2.compute.amazonaws.com,54.201.184.1' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-201-31-147.us-west-2.compute.amazonaws.com,54.201.31.147' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-191-105-88.us-west-2.compute.amazonaws.com,54.191.105.88' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-191-55-192.us-west-2.compute.amazonaws.com,54.191.55.192' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-186-229-83.us-west-2.compute.amazonaws.com,54.186.229.83' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-191-112-187.us-west-2.compute.amazonaws.com,54.191.112.187' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-191-163-113.us-west-2.compute.amazonaws.com,54.191.163.113' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-186-229-236.us-west-2.compute.amazonaws.com,54.186.229.236' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-201-64-224.us-west-2.compute.amazonaws.com,54.201.64.224' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-187-2-182.us-west-2.compute.amazonaws.com,54.187.2.182' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-201-26-136.us-west-2.compute.amazonaws.com,54.201.26.136' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-200-124-220.us-west-2.compute.amazonaws.com,54.200.124.220' (ECDSA) to the list of known hosts.
Warning: Permanently added 'ec2-54-191-132-210.us-west-2.compute.amazonaws.com,54.191.132.210' (ECDSA) to the list of known hosts.
Cloning spark-ec2 scripts from https://github.com/amplab/spark-ec2/tree/branch-1.5 on master...
Warning: Permanently added 'ec2-54-213-6-40.us-west-2.compute.amazonaws.com,54.213.6.40' (ECDSA) to the list of known hosts.
Cloning into 'spark-ec2'...
remote: Counting objects: 2072, done.
remote: Total 2072 (delta 0), reused 0 (delta 0), pack-reused 2072
Receiving objects: 100% (2072/2072), 356.17 KiB | 0 bytes/s, done.
Resolving deltas: 100% (789/789), done.
Checking connectivity... done.
Connection to ec2-54-213-6-40.us-west-2.compute.amazonaws.com closed.
Deploying files to master...

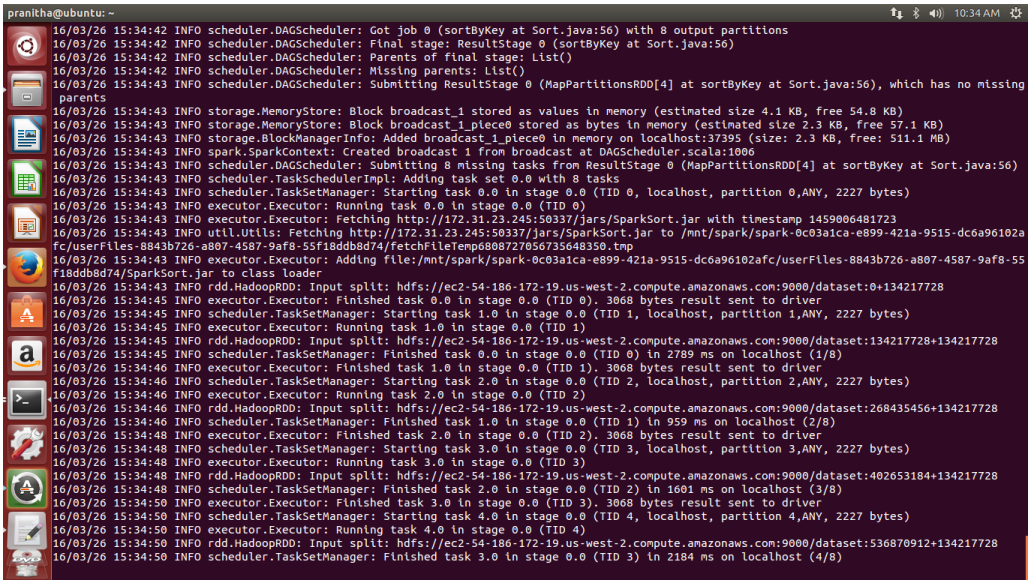


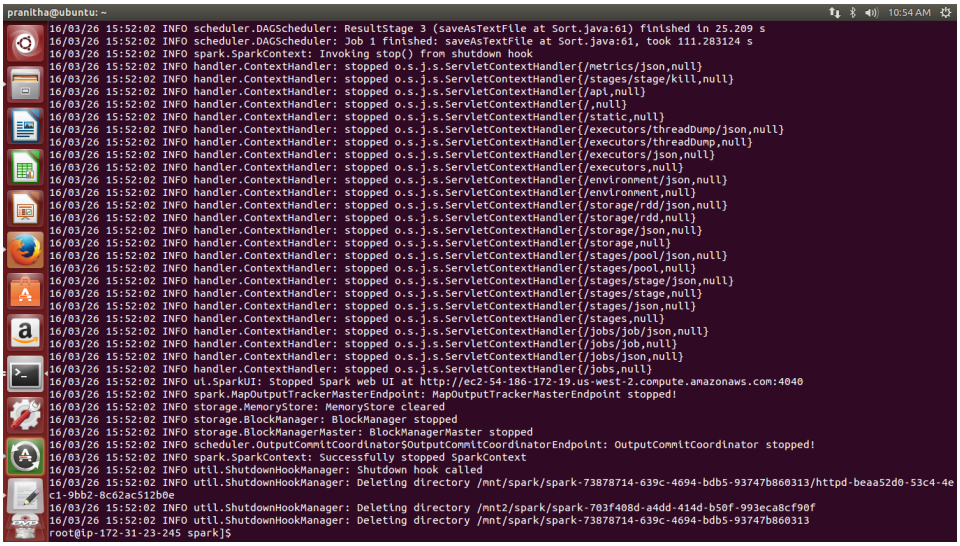
- Once master and slaves are all up running, open master instance from where all slave nodes can be operated
- To transfer files like jar, gensort etc. use SCP command and sent it to master.
- Check java version if its below 1.8 update using following commands
    1. wget --no-check-certificate --no-cookies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

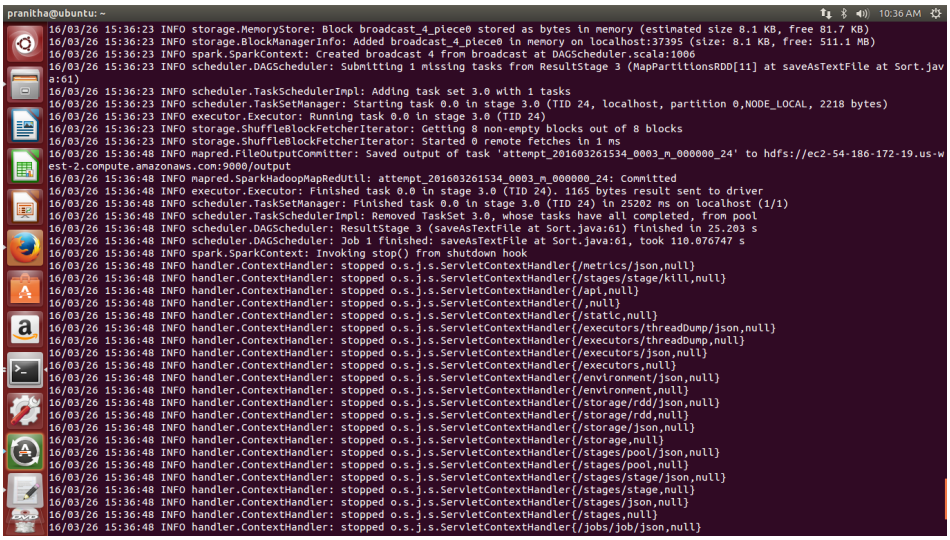
Implementation:

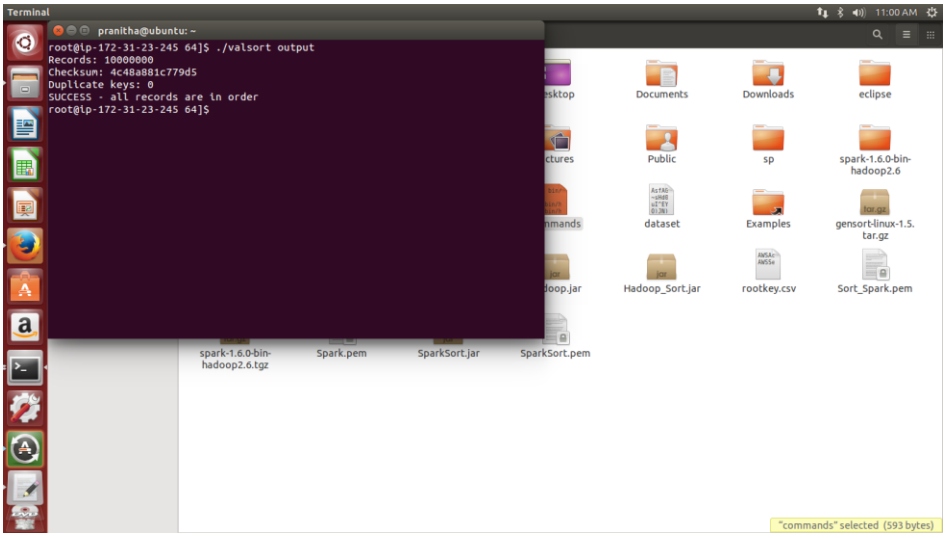
1. Spark sort has been implemented in java.
2. Launch the instance and do all necessary implementation stated above
3. Once the implementations are done, add storage depending on file size to be sorted
4. Get the jar file that has been implemented using SCP command from local system to instance.
5. Create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


pranitha@ubuntu: ~
10:36 AM
16/03/26 15:36:23 INFO storage.MemoryStore: Block broadcast_4_piece0 stored as bytes in memory (estimated size 8.1 KB, free 81.7 KB)
16/03/26 15:36:23 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on localhost:37395 (size: 8.1 KB, free: 511.1 MB)
16/03/26 15:36:23 INFO spark.SparkContext: Created broadcast 4 from broadcast at DAGScheduler.scala:1006
16/03/26 15:36:23 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 3 (MapPartitionsRDD[11] at saveAsTextFile at Sort.java:61)
16/03/26 15:36:23 INFO scheduler.TaskSchedulerImpl: Adding task set 3.0 with 1 tasks
16/03/26 15:36:23 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 3.0 (TID 24, localhost, partition 0,NODE_LOCAL, 2218 bytes)
16/03/26 15:36:23 INFO executor.Executor: Running task 0.0 in stage 3.0 (TID 24)
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Getting 8 non-empty blocks out of 8 blocks
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
16/03/26 15:36:48 INFO mapred.FileOutputCommitter: Saved output of task 'attempt_201603261534_0003_m_000000_24' to hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/output
16/03/26 15:36:48 INFO mapred.SparkHadoopMapRedUtil: attempt_201603261534_0003_m_000000_24: Committed
16/03/26 15:36:48 INFO executor.Executor: Finished task 0.0 in stage 3.0 (TID 24). 1165 bytes result sent to driver
16/03/26 15:36:48 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 3.0 (TID 24) in 25202 ms on localhost (1/1)
16/03/26 15:36:48 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
16/03/26 15:36:48 INFO scheduler.DAGScheduler: ResultStage 3 (saveAsTextFile at Sort.java:61) finished in 25.203 s
16/03/26 15:36:48 INFO scheduler.DAGScheduler: Job 1 finished: saveAsTextFile at Sort.java:61, took 110.076747 s
16/03/26 15:36:48 INFO spark.SparkContext: Invoking stop() from shutdown hook
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs/job/json,null}


pranitha@ubuntu: ~
10:34 AM
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Got job 0 (sortByKey at Sort.java:56) with 8 output partitions
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Final stage: ResultStage 0 (sortByKey at Sort.java:56)
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Parents of final stage: List()
16/03/26 15:34:42 INFO scheduler.DAGScheduler: Missing parents: List()
16/03/26 15:34:43 INFO scheduler.DAGScheduler: Submitting ResultStage 0 (MapPartitionsRDD[4] at sortByKey at Sort.java:56), which has no missing parents
16/03/26 15:34:43 INFO storage.MemoryStore: Block broadcast_1 stored as values in memory (estimated size 4.1 KB, free 54.8 KB)
16/03/26 15:34:43 INFO storage.MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 2.3 KB, free 57.1 KB)
16/03/26 15:34:43 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on localhost:37395 (size: 2.3 KB, free: 511.1 MB)
16/03/26 15:34:43 INFO spark.SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1006
16/03/26 15:34:43 INFO scheduler.DAGScheduler: Submitting 8 missing tasks from ResultStage 0 (MapPartitionsRDD[4] at sortByKey at Sort.java:56)
16/03/26 15:34:43 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0 with 8 tasks
16/03/26 15:34:43 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, partition 0,ANY, 2227 bytes)
16/03/26 15:34:43 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)
16/03/26 15:34:43 INFO executor.Executor: Fetching http://172.31.23.245:50337/jars/SparkSort.jar with timestamp 1459006481723
16/03/26 15:34:43 INFO util.Utils: Fetching http://172.31.23.245:50337/jars/SparkSort.jar to /mnt/spark/spark-0c03a1ca-e899-421a-9515-dc6a96102afc/userFiles-8843b726-a807-4587-9af8-55f18ddb8d74/fetchFileTemp6808727056735648350.tmp
16/03/26 15:34:43 INFO executor.Executor: Adding file:/mnt/spark/spark-0c03a1ca-e899-421a-9515-dc6a96102afc/userFiles-8843b726-a807-4587-9af8-55f18ddb8d74/SparkSort.jar to class loader
16/03/26 15:34:43 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:0+134217728
16/03/26 15:34:45 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 3068 bytes result sent to driver
16/03/26 15:34:45 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, localhost, partition 1,ANY, 2227 bytes)
16/03/26 15:34:45 INFO executor.Executor: Running task 1.0 in stage 0.0 (TID 1)
16/03/26 15:34:45 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:134217728+134217728
16/03/26 15:34:45 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 2789 ms on localhost (1/8)
16/03/26 15:34:46 INFO executor.Executor: Finished task 1.0 in stage 0.0 (TID 1). 3068 bytes result sent to driver
16/03/26 15:34:46 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 0.0 (TID 2, localhost, partition 2,ANY, 2227 bytes)
16/03/26 15:34:46 INFO executor.Executor: Running task 2.0 in stage 0.0 (TID 2)
16/03/26 15:34:46 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:268435456+134217728
16/03/26 15:34:46 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 959 ms on localhost (2/8)
16/03/26 15:34:48 INFO executor.Executor: Finished task 2.0 in stage 0.0 (TID 2). 3068 bytes result sent to driver
16/03/26 15:34:48 INFO scheduler.TaskSetManager: Starting task 3.0 in stage 0.0 (TID 3, localhost, partition 3,ANY, 2227 bytes)
16/03/26 15:34:48 INFO executor.Executor: Running task 3.0 in stage 0.0 (TID 3)
16/03/26 15:34:48 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:402653184+134217728
16/03/26 15:34:48 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 1601 ms on localhost (3/8)
16/03/26 15:34:50 INFO executor.Executor: Finished task 3.0 in stage 0.0 (TID 3). 3068 bytes result sent to driver
16/03/26 15:34:50 INFO scheduler.TaskSetManager: Starting task 4.0 in stage 0.0 (TID 4, localhost, partition 4,ANY, 2227 bytes)
16/03/26 15:34:50 INFO executor.Executor: Running task 4.0 in stage 0.0 (TID 4)
16/03/26 15:34:50 INFO rdd.HadoopRDD: Input split: hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/dataset:536870912+134217728
16/03/26 15:34:50 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 0.0 (TID 3) in 2184 ms on localhost (4/8)


pranitha@ubuntu: ~
10:54 AM
16/03/26 15:52:02 INFO scheduler.DAGScheduler: ResultStage 3 (saveAsTextFile at Sort.java:61) finished in 25.209 s
16/03/26 15:52:02 INFO scheduler.DAGScheduler: Job 1 finished: saveAsTextFile at Sort.java:61, took 111.283124 s
16/03/26 15:52:02 INFO spark.SparkContext: Invoking stop() from shutdown hook
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs/job/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs/job,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs/json,null}
16/03/26 15:52:02 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs,null}
16/03/26 15:52:02 INFO ui.SparkUI: Stopped Spark web UI at http://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:4040
16/03/26 15:52:02 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/03/26 15:52:02 INFO storage.MemoryStore: MemoryStore cleared
16/03/26 15:52:02 INFO storage.BlockManager: BlockManager stopped
16/03/26 15:52:02 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
16/03/26 15:52:02 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/03/26 15:52:02 INFO spark.SparkContext: Successfully stopped SparkContext
16/03/26 15:52:02 INFO util.ShutdownHookManager: Shutdown hook called
16/03/26 15:52:02 INFO util.ShutdownHookManager: Deleting directory /mnt/spark/spark-73878714-639c-4694-bdb5-93747b860313/httpd-beaa52d0-53c4-4ec1-9bb2-8c62ac512b0e
16/03/26 15:52:02 INFO util.ShutdownHookManager: Deleting directory /mnt2/spark/spark-703f408d-a4dd-414d-b50f-993eca8cf90f
16/03/26 15:52:02 INFO util.ShutdownHookManager: Deleting directory /mnt/spark/spark-73878714-639c-4694-bdb5-93747b860313
root@ip-172-31-23-245 spark]$


pranitha@ubuntu: ~
10:36 AM
16/03/26 15:36:23 INFO storage.MemoryStore: Block broadcast_4_piece0 stored as bytes in memory (estimated size 8.1 KB, free 81.7 KB)
16/03/26 15:36:23 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on localhost:37395 (size: 8.1 KB, free: 511.1 MB)
16/03/26 15:36:23 INFO spark.SparkContext: Created broadcast 4 from broadcast at DAGScheduler.scala:1006
16/03/26 15:36:23 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 3 (MapPartitionsRDD[11] at saveAsTextFile at Sort.java:61)
16/03/26 15:36:23 INFO scheduler.TaskSchedulerImpl: Adding task set 3.0 with 1 tasks
16/03/26 15:36:23 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 3.0 (TID 24, localhost, partition 0,NODE_LOCAL, 2218 bytes)
16/03/26 15:36:23 INFO executor.Executor: Running task 0.0 in stage 3.0 (TID 24)
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Getting 8 non-empty blocks out of 8 blocks
16/03/26 15:36:23 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
16/03/26 15:36:48 INFO mapred.FileOutputCommitter: Saved output of task 'attempt_201603261534_0003_m_000000_24' to hdfs://ec2-54-186-172-19.us-west-2.compute.amazonaws.com:9000/output
16/03/26 15:36:48 INFO mapred.SparkHadoopMapRedUtil: attempt_201603261534_0003_m_000000_24: Committed
16/03/26 15:36:48 INFO executor.Executor: Finished task 0.0 in stage 3.0 (TID 24). 1165 bytes result sent to driver
16/03/26 15:36:48 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 3.0 (TID 24) in 25202 ms on localhost (1/1)
16/03/26 15:36:48 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
16/03/26 15:36:48 INFO scheduler.DAGScheduler: ResultStage 3 (saveAsTextFile at Sort.java:61) finished in 25.203 s
16/03/26 15:36:48 INFO scheduler.DAGScheduler: Job 1 finished: saveAsTextFile at Sort.java:61, took 110.076747 s
16/03/26 15:36:48 INFO spark.SparkContext: Invoking stop() from shutdown hook
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/json,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages,null}
16/03/26 15:36:48 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/jobs/job/json,null}


Terminal
11:00 AM
pranitha@ubuntu: ~
root@ip-172-31-23-245 64]$ ./valsort output
Records: 10000000
Checksum: 4c48a881c779d5
Duplicate keys: 0
SUCCESS - all records are in order
root@ip-172-31-23-245 64]$
Documents
Downloads
eclipse
Public
sp
spark-1.6.0-bin-hadoop2.6
dataset
Examples
gensort-linux-1.5.tar.gz
Hadoop_Sort.jar
rootkey.csv
Sort_Spark.pem
spark-1.6.0-bin-hadoop2.6.tgz
Spark.pem
SparkSort.jar
SparkSort.pem
"commands" selected (593 bytes)

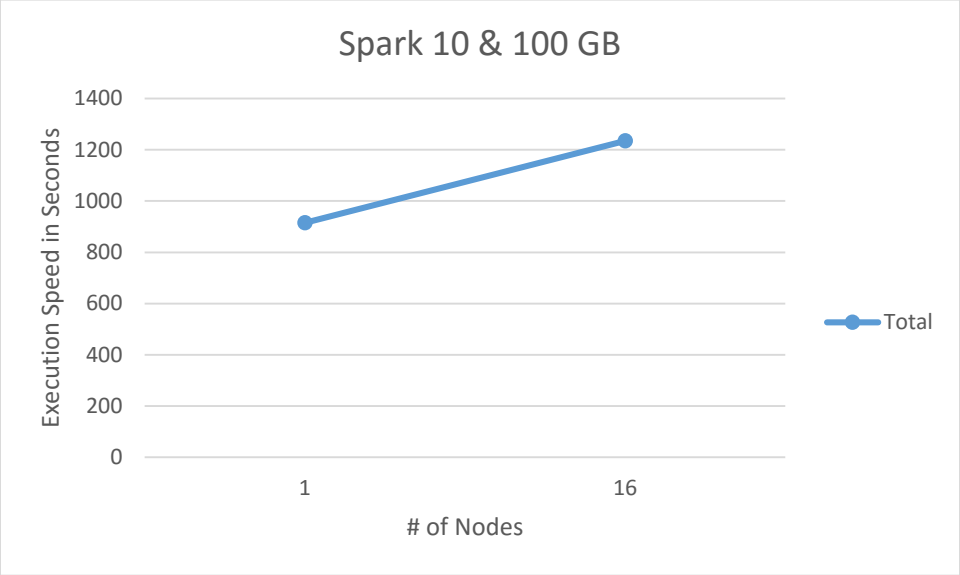
Performance:

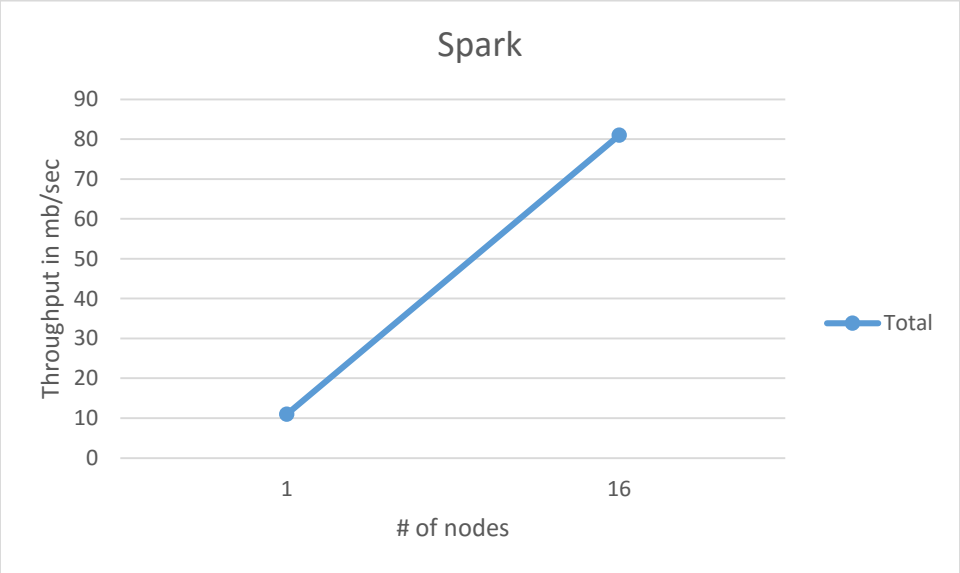
Time taken to sort 10GB:915.6 sec

100GB:1235.34 sec

Throughput 10GB:10.9217999 mb/sec

100GB:80.9493743 mb/sec

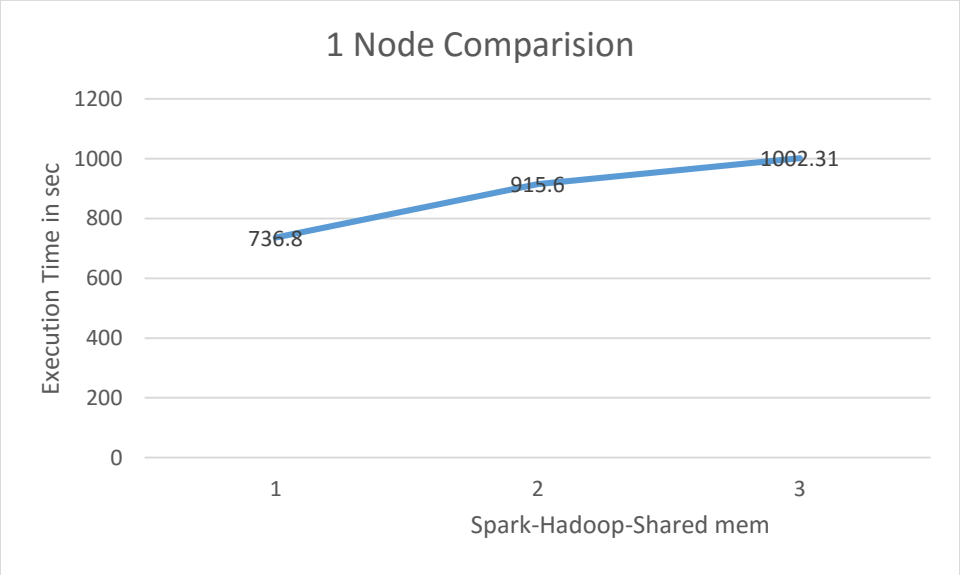




Comparison of performances:

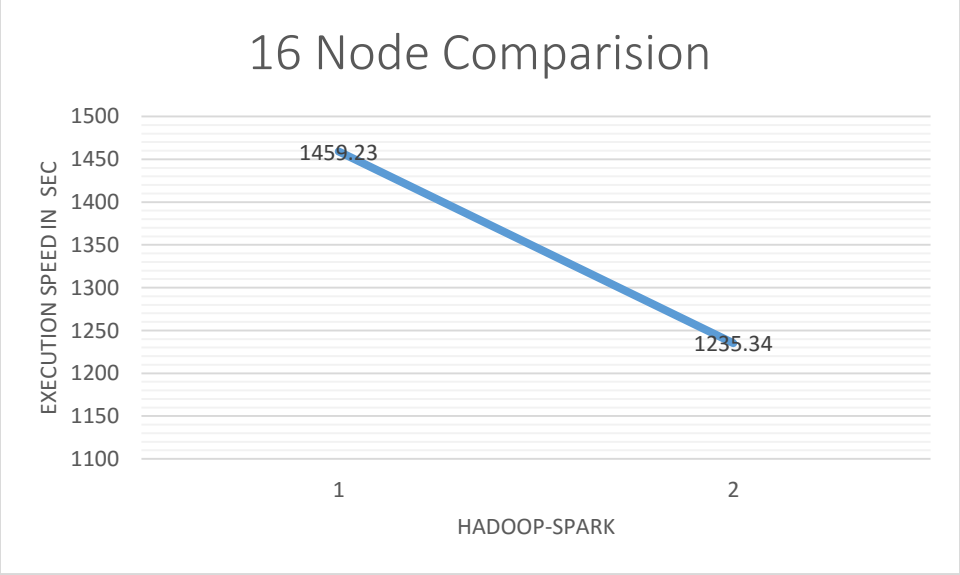
For 10GB

	Execution Time
Shared-Memory	1002**.31**
Hadoop	736.8
Spark	915.6



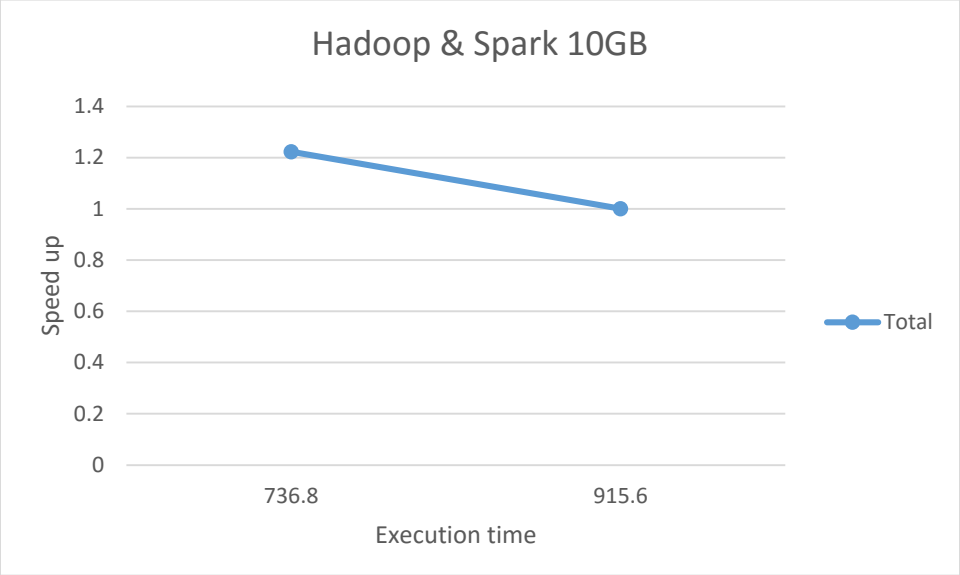
For 100GB

	Execution Time
Hadoop	1459.23
Spark	1235.34



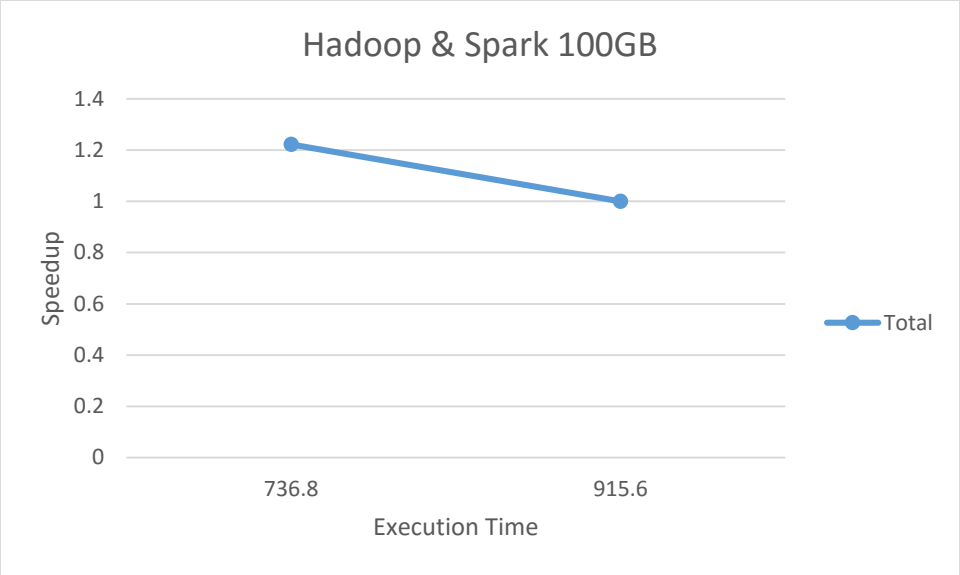
Hadoop and spark speedup for 10 GB

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



Hadoop and spark speedup for 100GB

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



What Conclusion are drawn?

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

What seems to be best at 1node scale?

- Hadoop

What seems to be best at 100 node scale?

- Spark