Program Design Consideration

Java Version 1.8

Hadoop Version 2.7.2

Spark Version 1.6

OS ubuntu

Creating Virtual Cluster

For Creating virtual cluster we goto ec2 dashboard first and then launch an instance by clicking launch

Select the ami Ami Amazon Linux AMI 2016.03.0 (HVM), SSD Volume Type - ami-08111162

Select instance type as d2.xlarge or c3.large

Select instance price as spot or on demand (Selected spot instances for price 0.3)

Add Storage as required D2.xlarge comes with three storage sdb sdc sdd

C3 large has only local ebs blocks we can add storage under any device for storing more data

Add instance name then configure security group of instance

We Select All TCP ALL Icmp protocol to connect from anywhere to any host

When we launch instance we have to download pem file which is used later for connecting to it

After launching the instance we connect using

Ssh –i pem file ec2-user@hostname

Your instance will be seen in instances tab you can create image of instance by clicking on actions image create image

An ami image of the instance will be created which can be used for configuring 16 nodes

When you launch instance using your single node ami image by following above steps all instances will have files configured in it

1 Java Program Shared Memory

In Shared Memory we used threads to implement multiple data sorting we give threads in loop and read a file do quick sort and write intermediate data to temp file

After that we merge the file to create one big file

We use priority queue for merging the file

Priority queue maintain the Current String read by file and compare two file at a time and give the minimum value

To compile the code

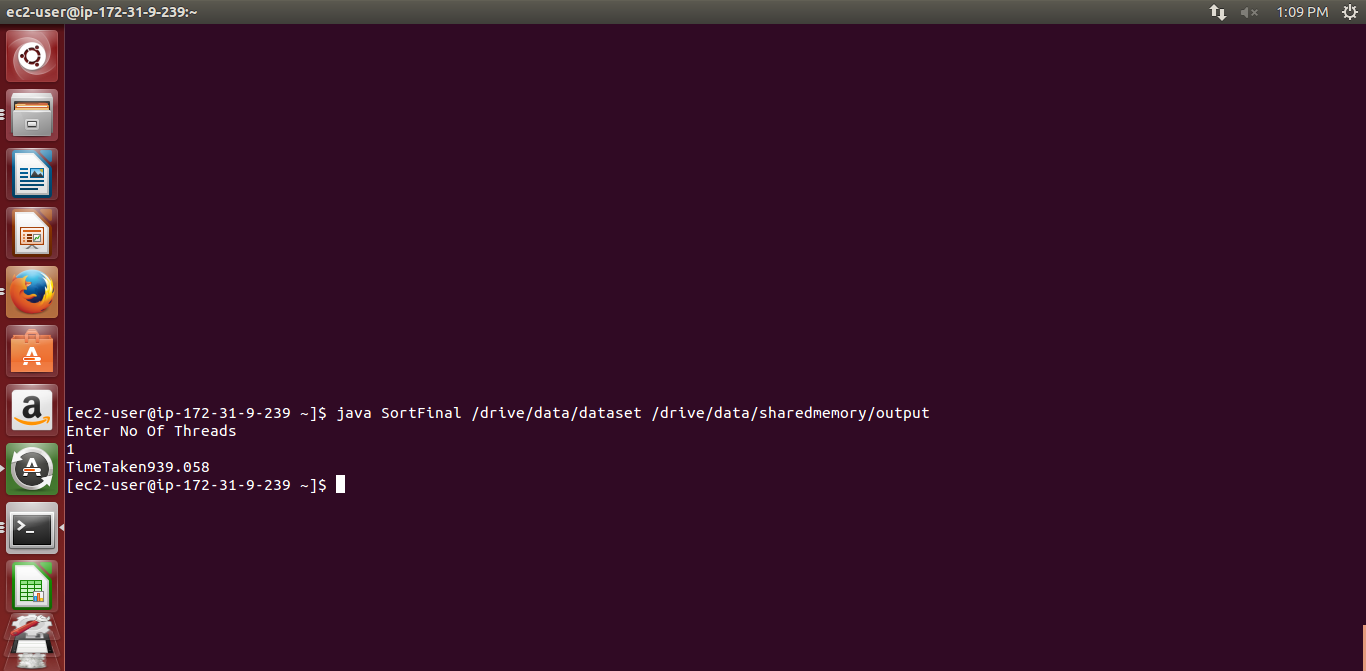
Javac MergeThread.java

Javac SortFinal.java

To Execute the program

Java SortFinal /dataset /output

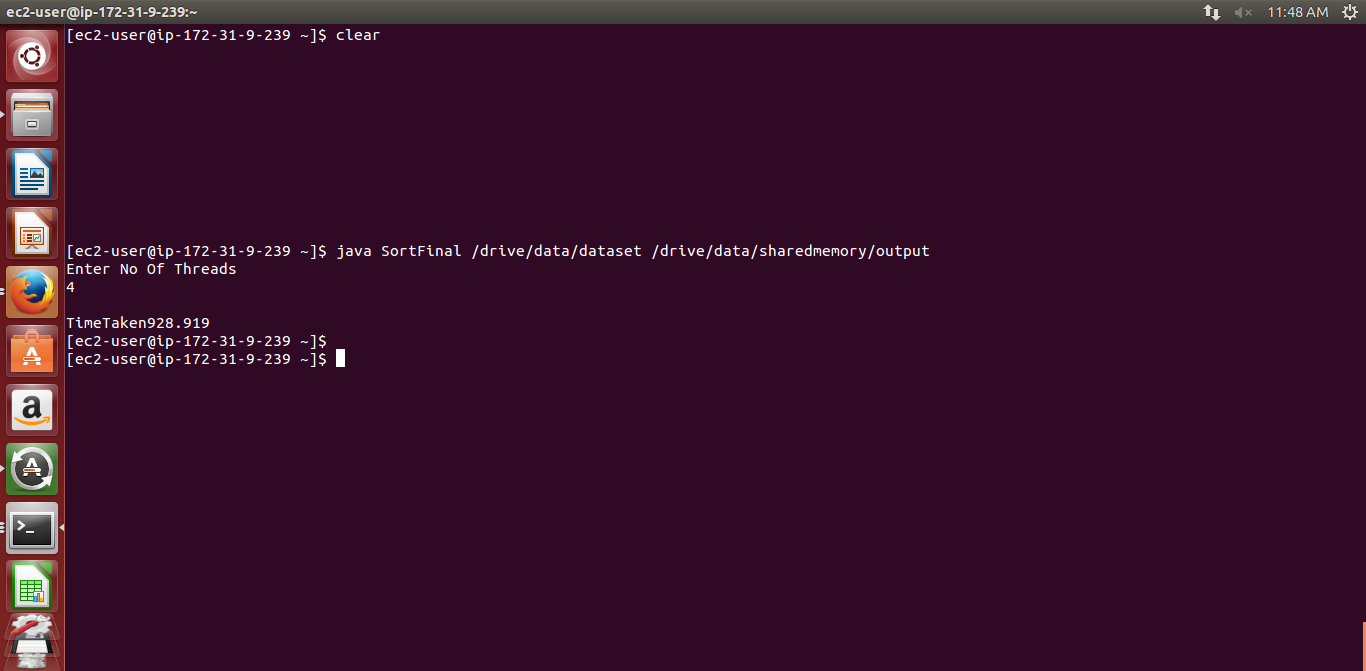
Shared Memory 1 node c3 large 10 gb 1 thread



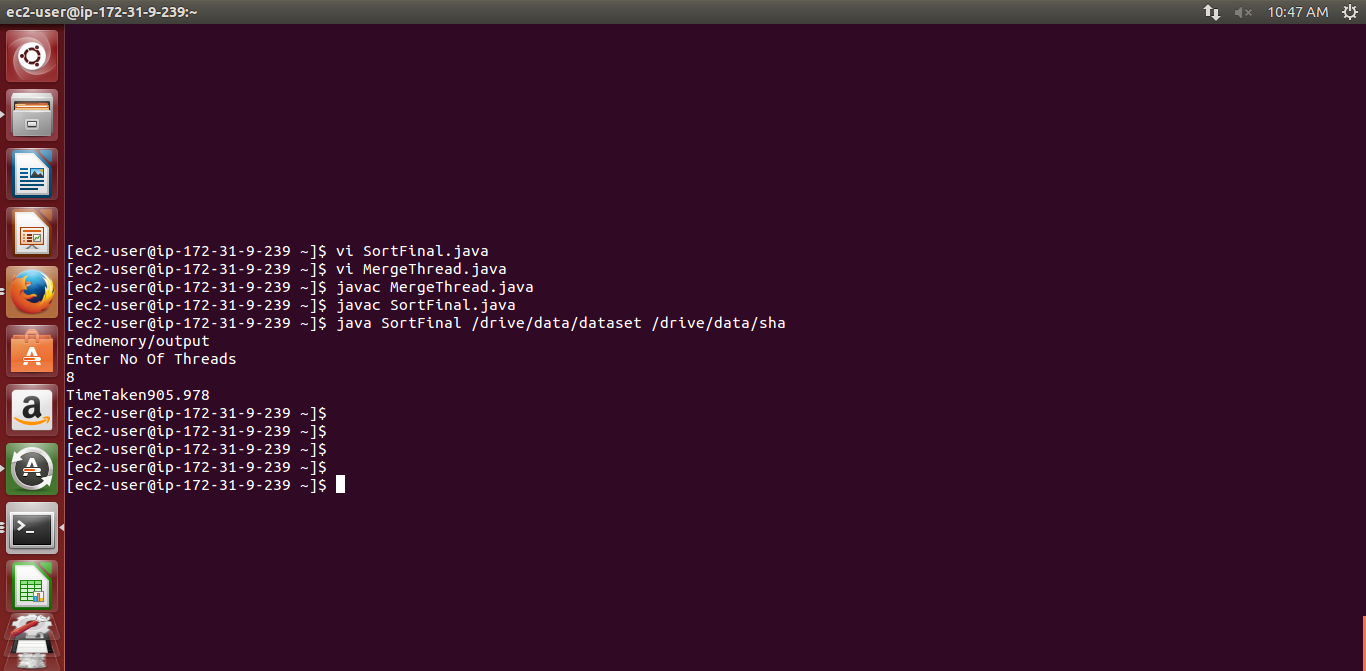
Shared Memory 1 node c3 large 10 gb 2 thread

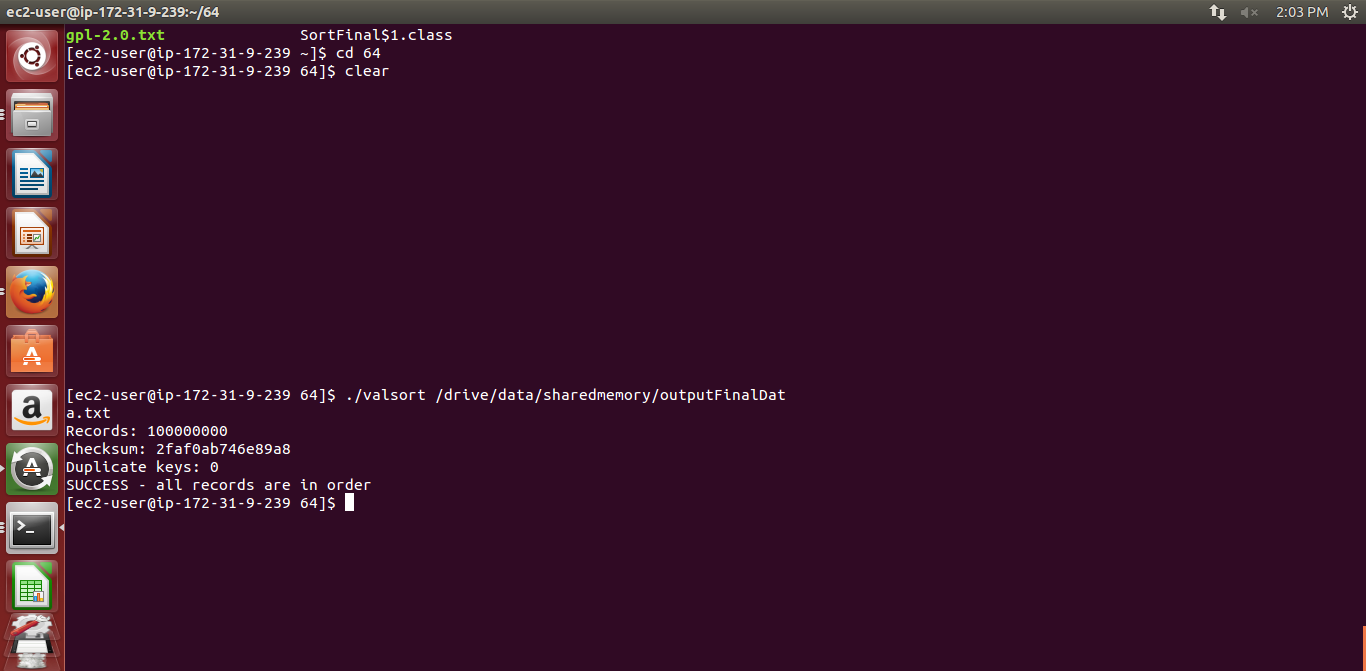


Shared Memory 1 node c3 large 10 gb 4 thread



Shared Memory 1 node c3 large 10 gb 8 thread





2 Hadoop

Code Explanation Design

Hadoop runs map reduce job

It takes an input string and coverts it into key value pair in map phase

In shuffle phase it sorts keys and stores them in intermediate data

In reduce phase it merges all key value pairs and store in final output file

Problem

In d2 x large instance we have to sort large amount of data 1 tb data b using mapreduce model

We downloaded the Hadoop-2.7.2 package and sent it to Amazon instance

To install Hadoop on amazon ec2 first we create virtual cluster

We connect to aws instance with cluster by

ssh -‐I hadoop.pem ec2-user@hostname

we create keygen pasworldess ssh

Adding keyagent identity

Eval `ssh-agent –s`

Chmod 600 pemfile

ssh-add pemfile

we downloaded java using command

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

we set java version using command

export JAVA\_HOME=/usr/java/latest

export Hadoop\_Home=/home/ec2-user/Hadoop-2.7.2

to view the hdfs directory(Hadoop file system) we use

bin/Hadoop dfs –ls /

to get and put file in hdfs

bin/Hadoop dfs –put /localpath for file /hdfs path for file

bin/Hadoop dfs –get /hdfs path for file /local path for file

to run a jar in hadoop

bin/Hadoop jar jarname mainclassname filepath

bin/Hadoop jar PA2\_Hadoop.jar Driver\_Hadoop /dataset /output

we change following Configuration File for Hadoop

Core-site.xml

hdfs-site.xml

slaves

mapred-site.xml

Hadoop-env.sh

Change JAVA\_HOME path to /usr/java/latest java folder path downloaded

Configuration for Hadoop

core-site.xml

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://ec2-user@ec2-54-174-147-236.compute-1.amazonaws.com:8020</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/drive</value>

</property>

</configuration>

Hdfs-site.xml

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>/drive/data</value>

</property>

<property>

<name>dfs.name.dir</name>

<value>/drive/name</value>

</property>

<property>

<name>mapred.map.tasks</name>

<value>4</value>

</property>

<property>

<name>mapred.reduce.tasks</name>

<value>4</value>

</property>

</configuration>

Mapred-site.xml

<configuration>

<property>

<name>mapreduce.jobtracker.address</name>

<value>hdfs://ec2-54-174-147-236.compute-1.amazonaws.com:8021</value>

</property>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.map.memory.mb</name>

<value>4096</value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>8192</value>

</property>

<property>

<name>mapreduce.map.java.opts</name>

<value>-Xmx3072m</value>

</property>

<property>

<name>mapreduce.reduce.java.opts</name>

<value>-Xmx6144m</value>

</property>

</configuration>

Yarn-site.xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>ec2-54-174-147-236.compute-1.amazonaws.com:8030</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>ec2-54-174-147-236.compute-1.amazonaws.com:8032</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>ec2-54-174-147-236.compute-1.amazonaws.com:8088</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>ec2-54-174-147-236.compute-1.amazonaws.com:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>ec2-54-174-147-236.compute-1.amazonaws.com:8033</value>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

</configuration>

yarn-site.xml

Execution Process

We format HDFS name node using

bin/ hadoop namenode –format

we start master slave nodes for Hadoop using sbin/start-all.sh

To check all the nodes are running name node and data node are running we use jps command

For Multi node cluster we change the slaves file in Hadoop-2.7.2/etc/Hadoop directory

In this file we add all the slave nodes ip address

**Hadoop Configuration**

NameNode: It monitors the storage of data .It is central repository that manages the sharing of data throughout the cluster It stores the metadata of hdfs file system

DataNode: It is the node where data is stored in hdfs file system

Node Manager: Yarn Agent manages each slave in a cluster

Resource Manager: This Manager is used for getting cluster information, scheduling of jobs across slaves

Secondarynamenode : it is backup for namenode

Problems

For Hadoop faced the following problem

1. For 1 tb

First map reduce used to stop in 600 maps or so

Removed it by setting Hadoop.tmp.dir as 1 tb drive /drive in core-site.xml

<property>

<name>hadoop.tmp.dir</name>

<value>/drive</value>

</property>

1. For 1 tb

Reducer used to stop at 33.33 % as it started shuffle phase

Resolved it by increasing the memory limit of mapreduce in mapred-site.xml

<property>

<name>mapreduce.map.memory.mb</name>

<value>4096</value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>8192</value>

</property>

<property>

<name>mapreduce.map.java.opts</name>

<value>-Xmx3072m</value>

</property>

<property>

<name>mapreduce.reduce.java.opts</name>

<value>-Xmx6144m</value>

</property>

And increasing reduce tasks to 4 in hdfs-site.xml

<property>

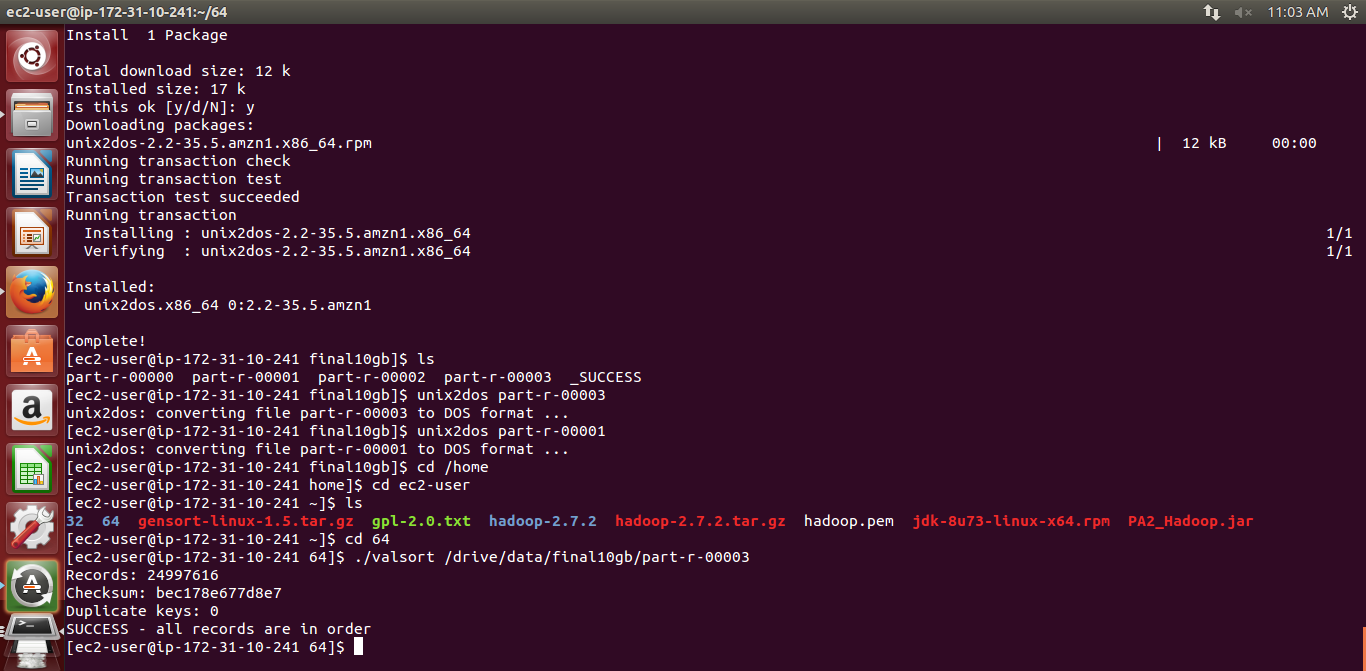
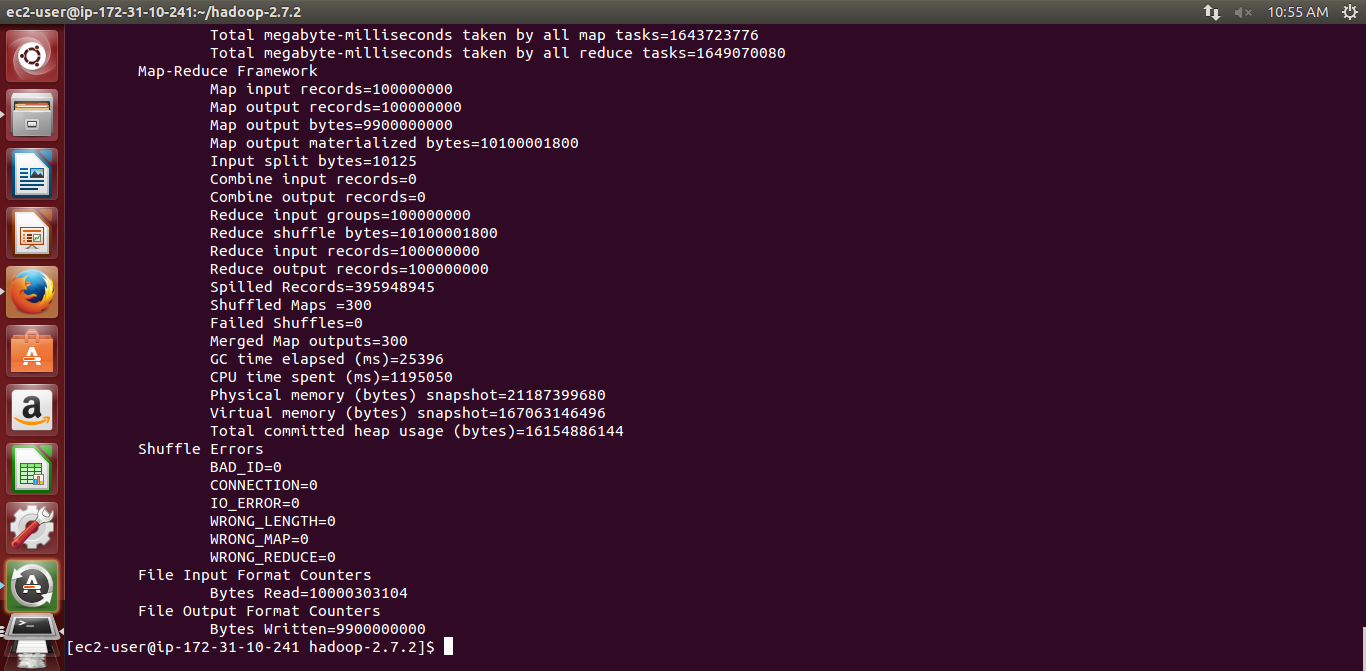
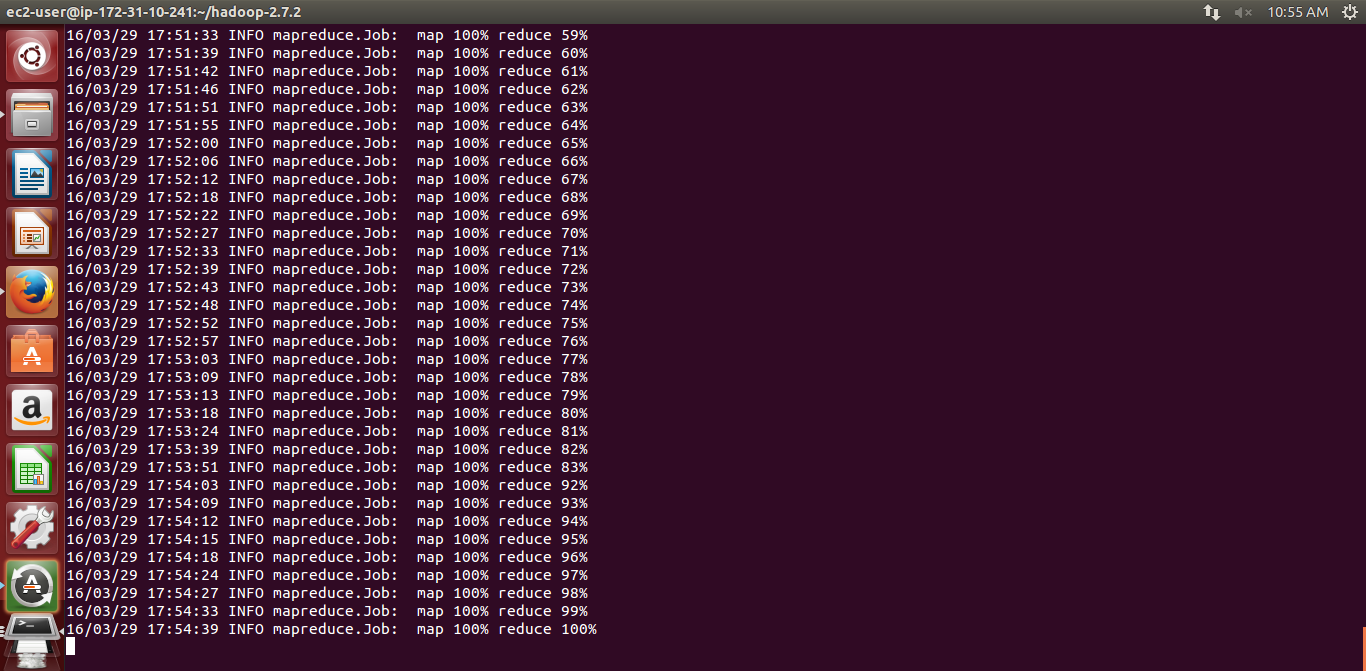
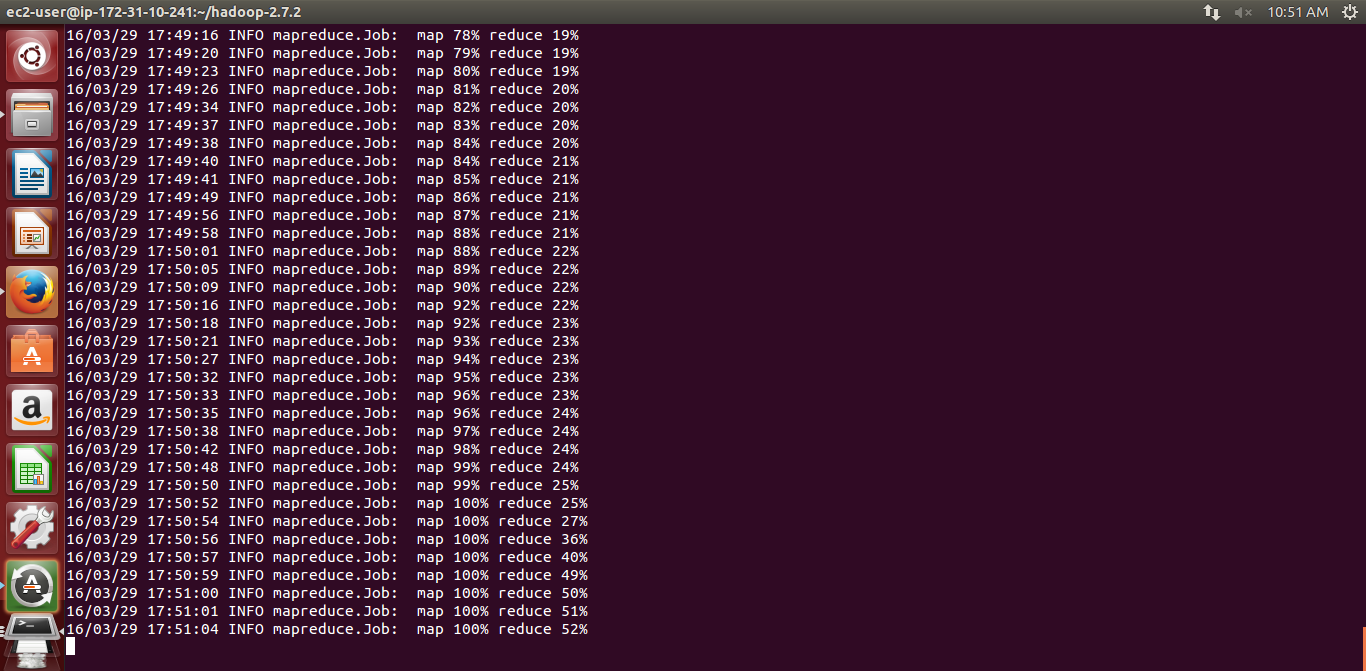
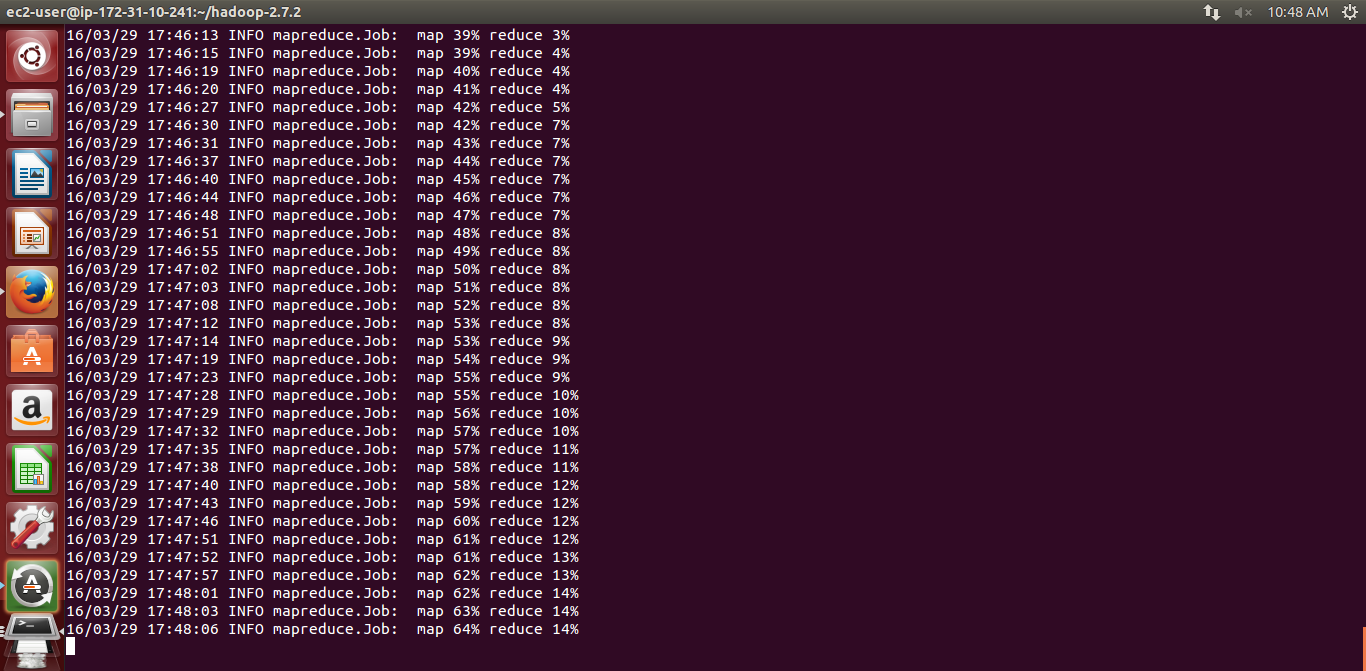
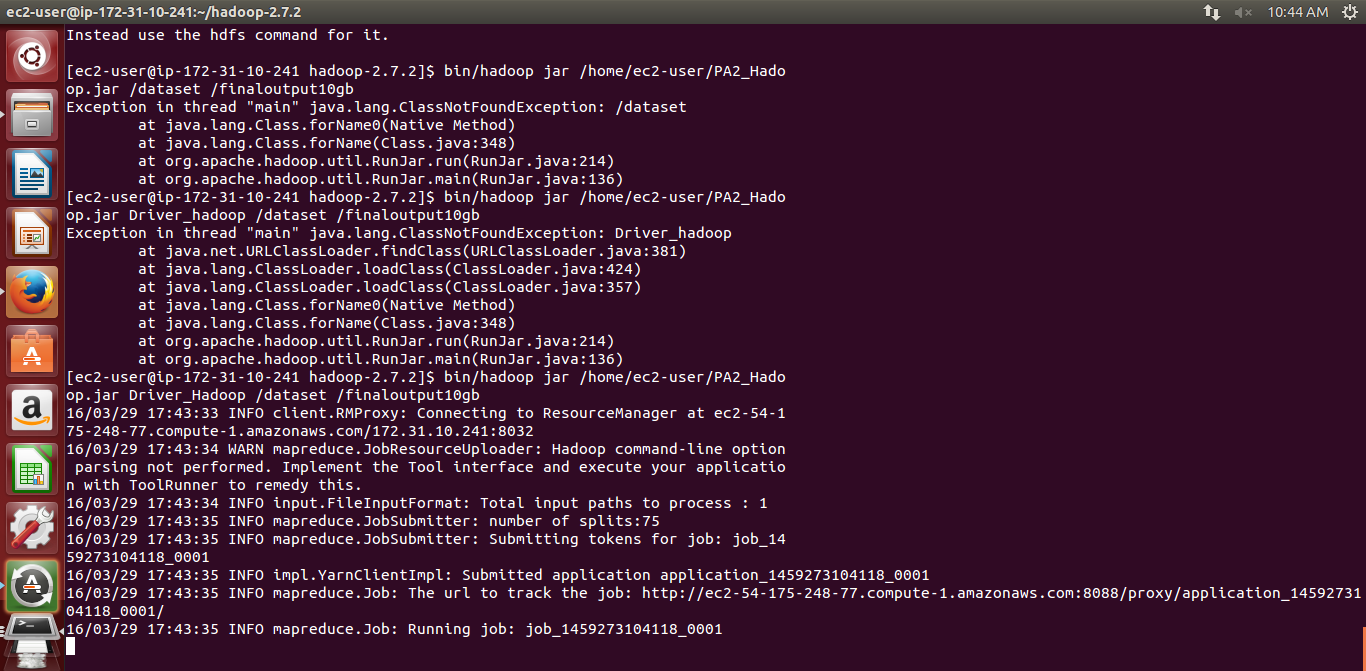
<name>mapred.reduce.tasks</name>

<value>4</value>

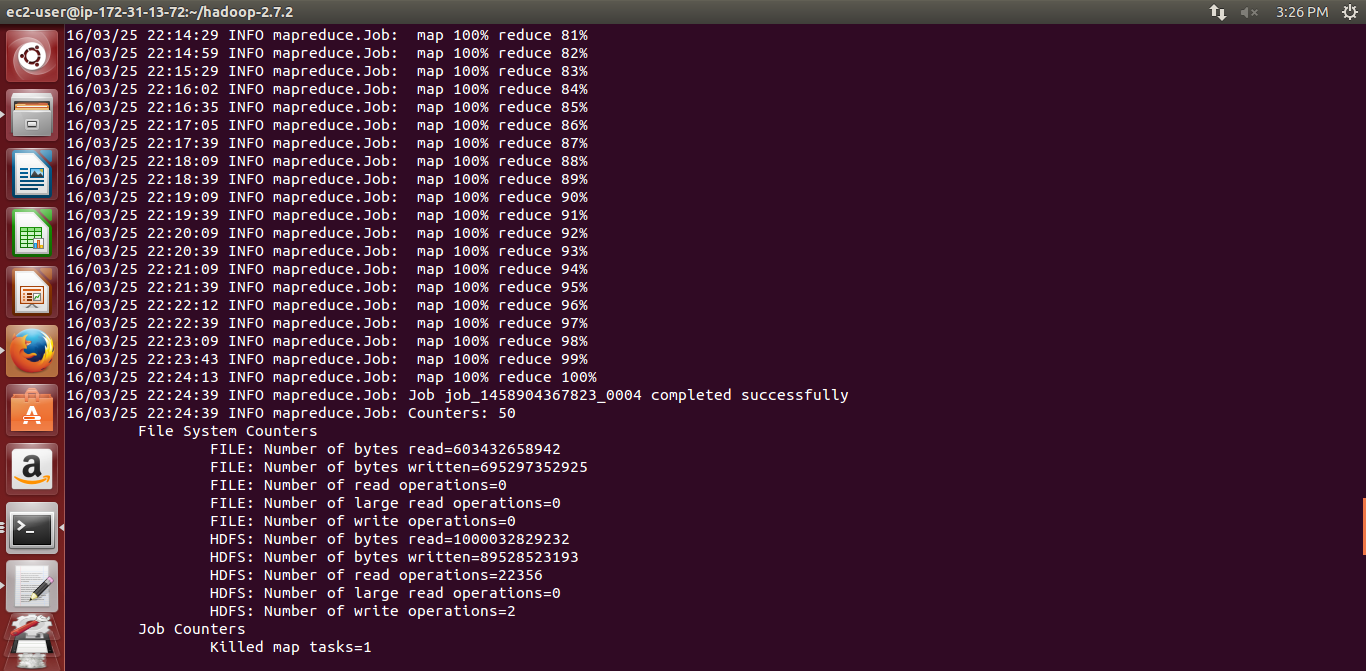
</property>

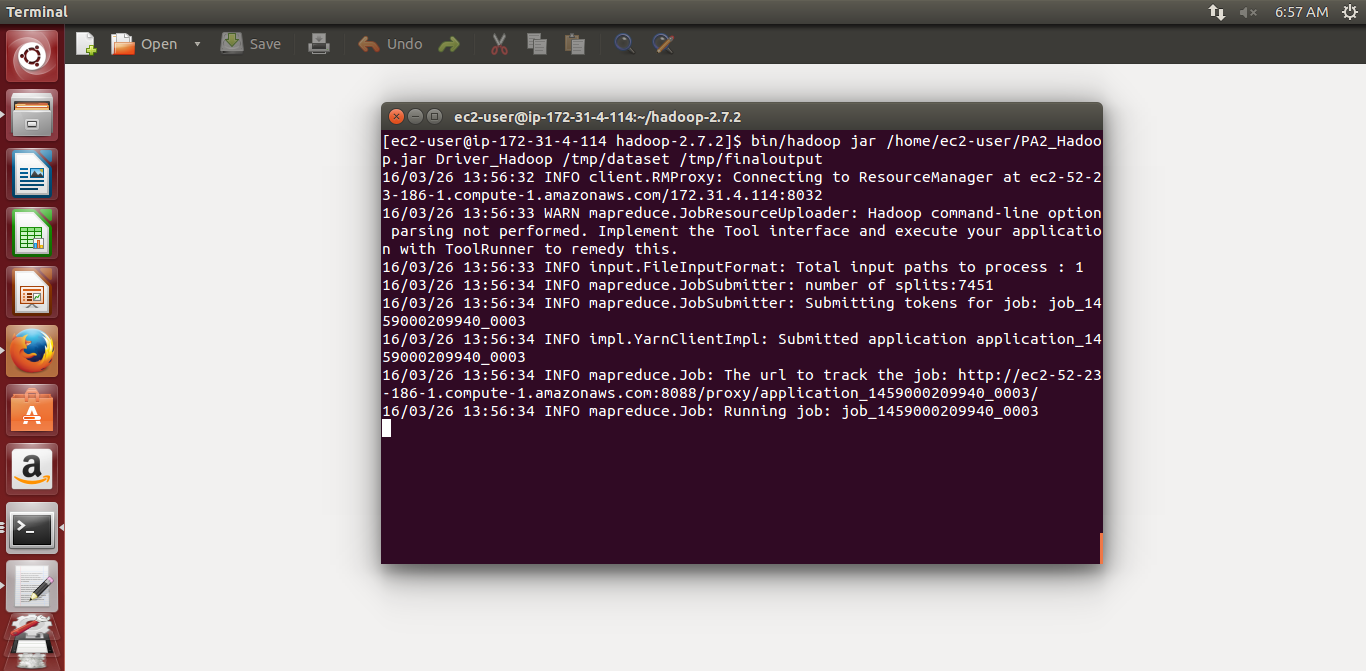
Hadoop Screenshots

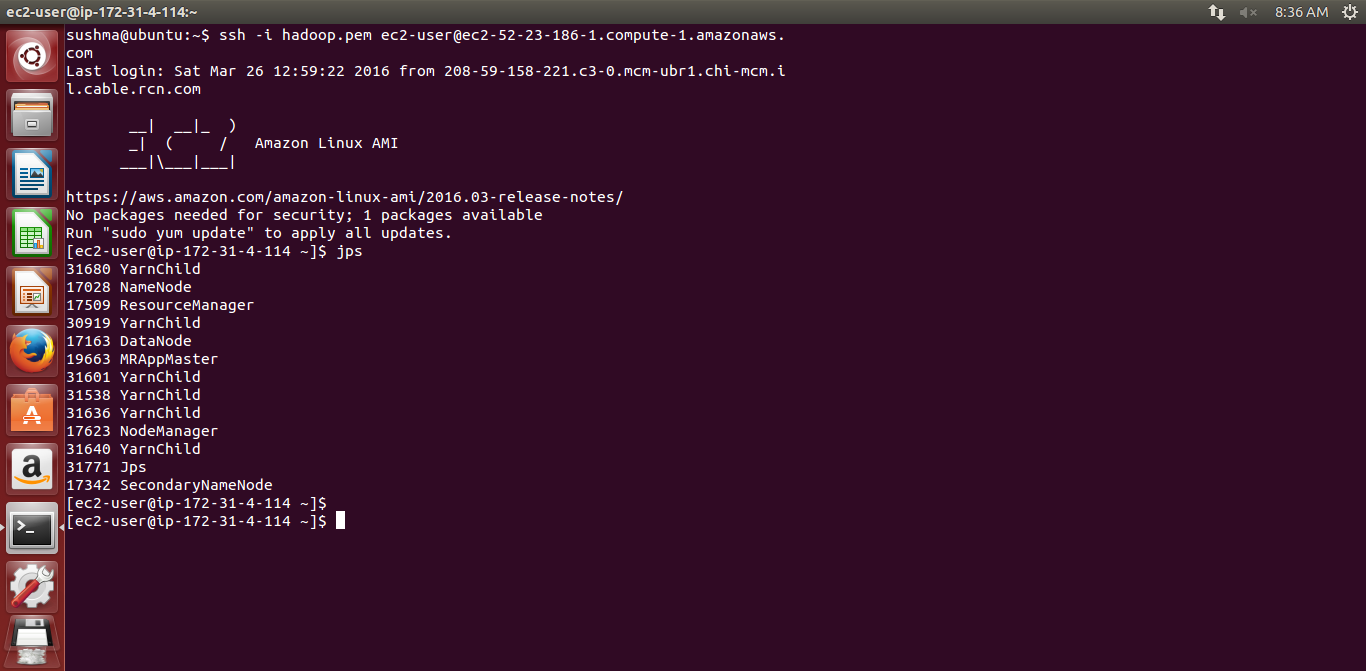
10 gb c3 xlarge

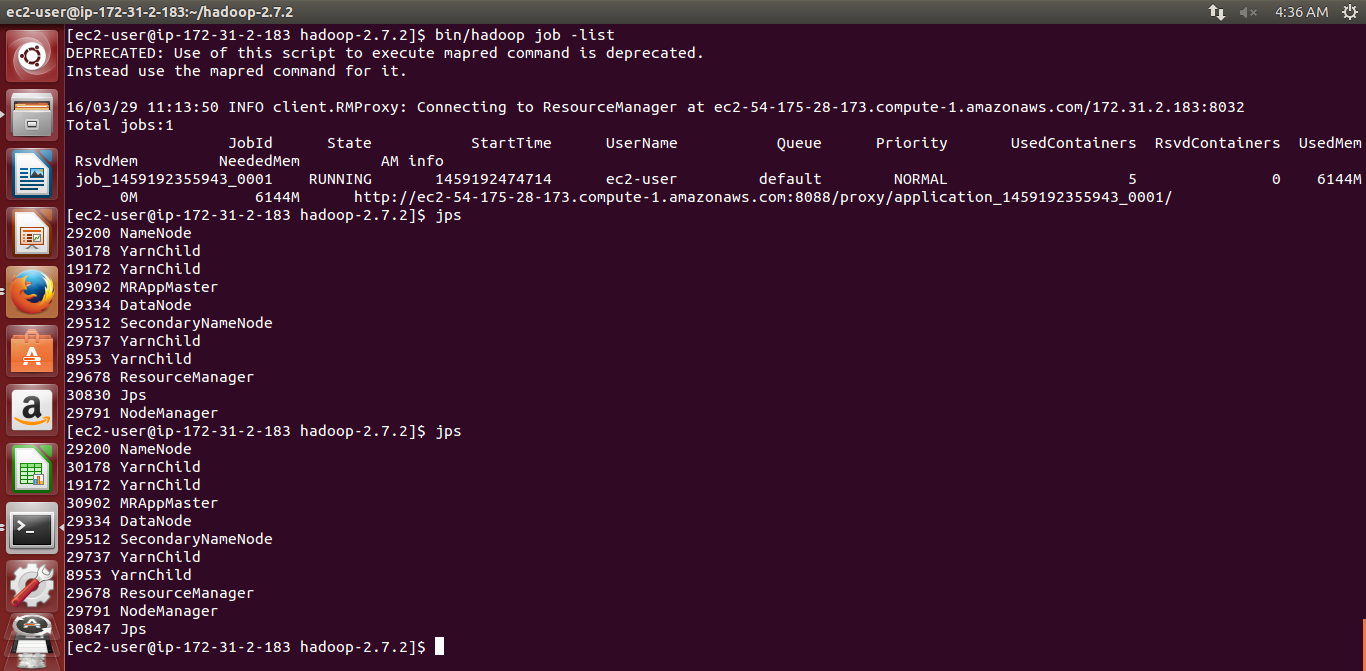


1 node 1 tb d2 xlarge

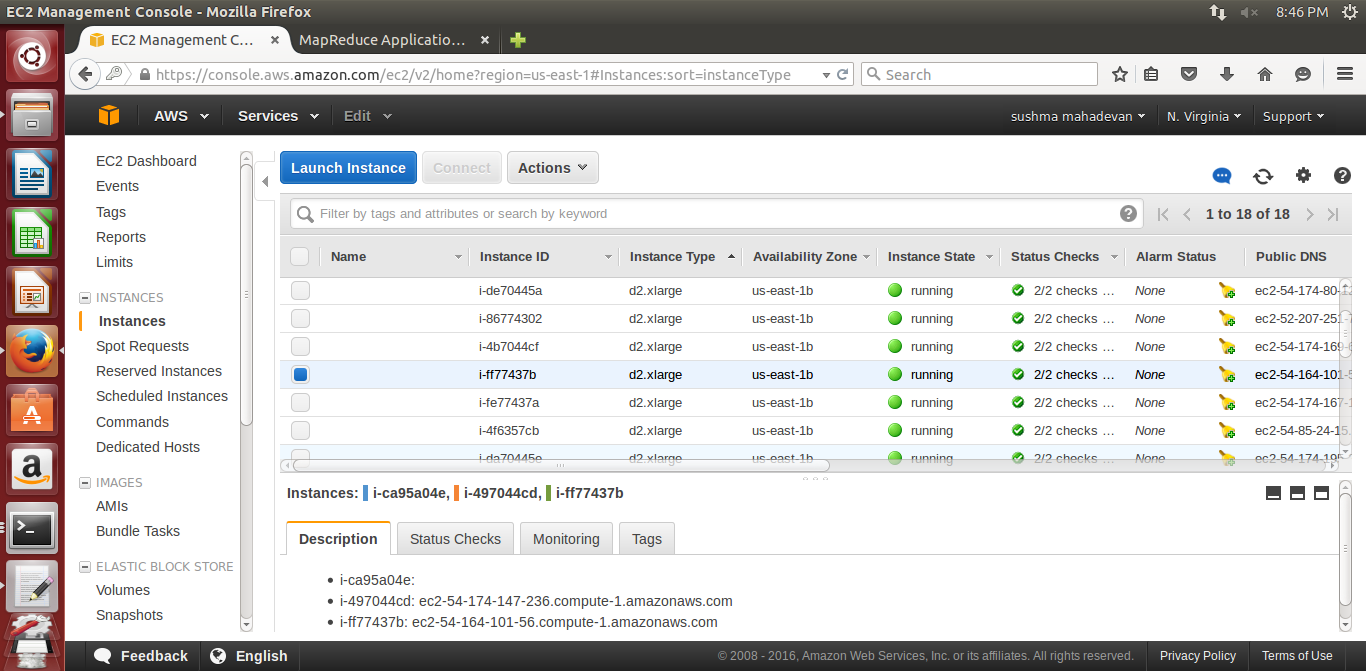


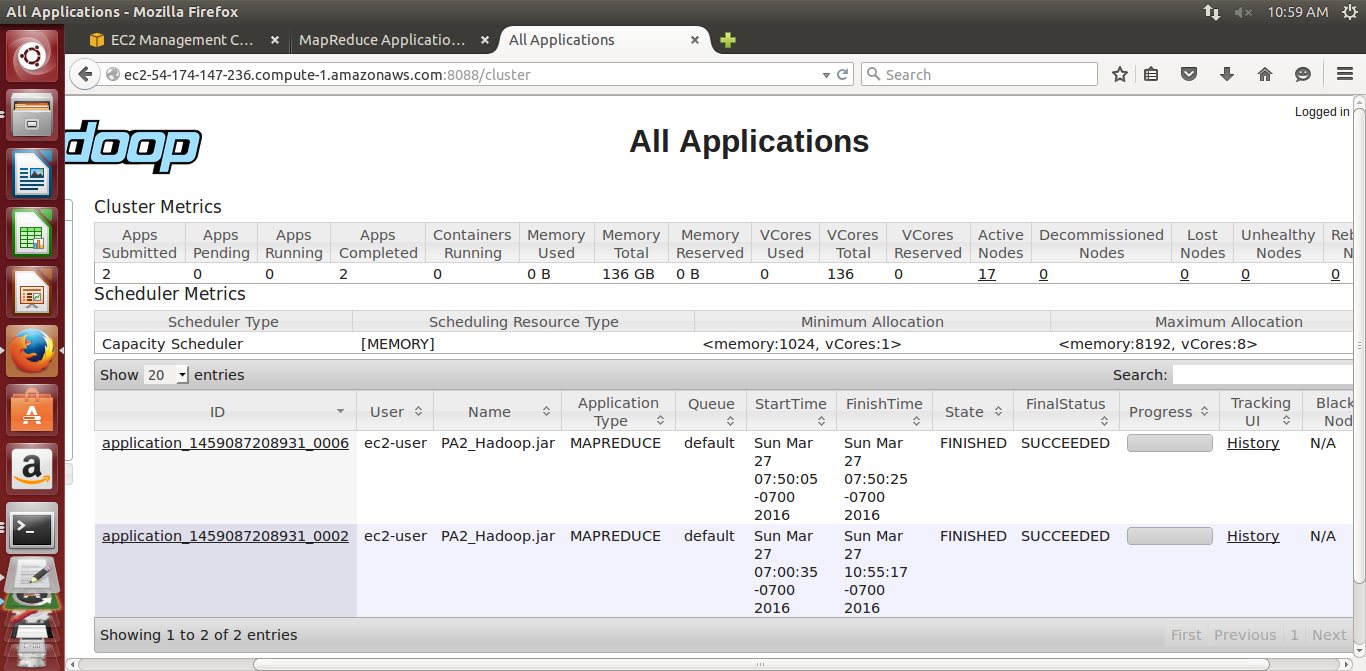




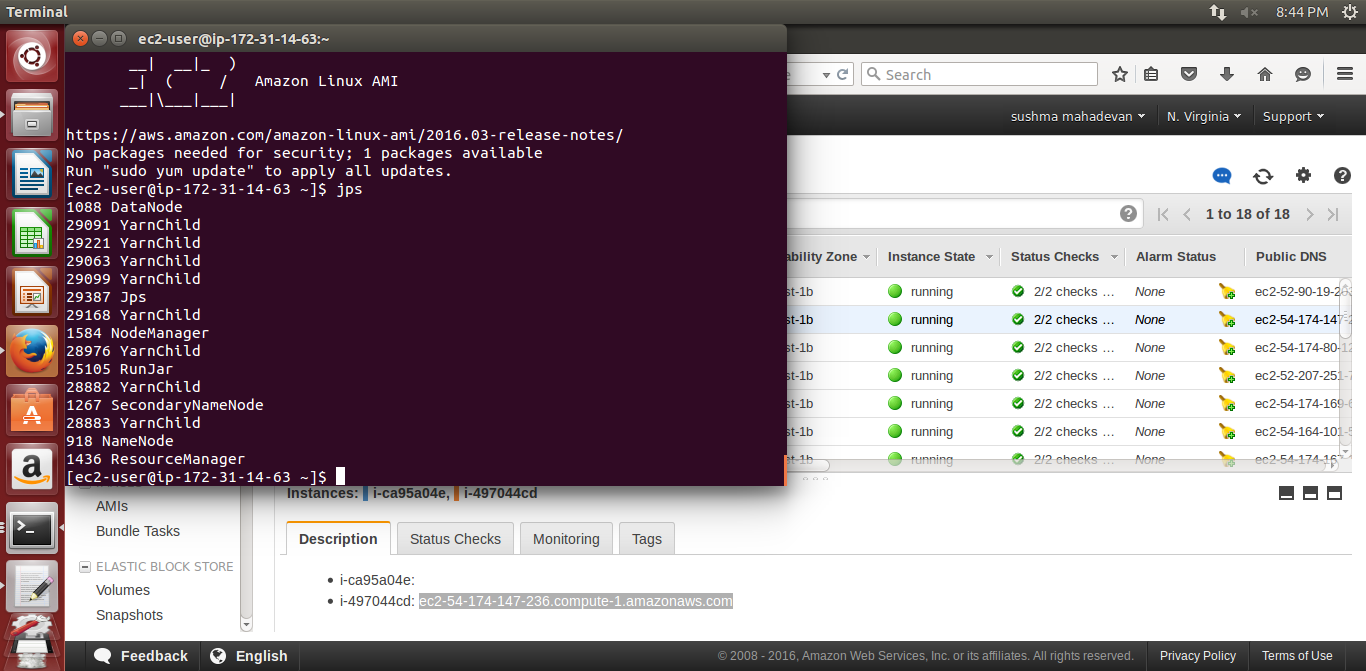


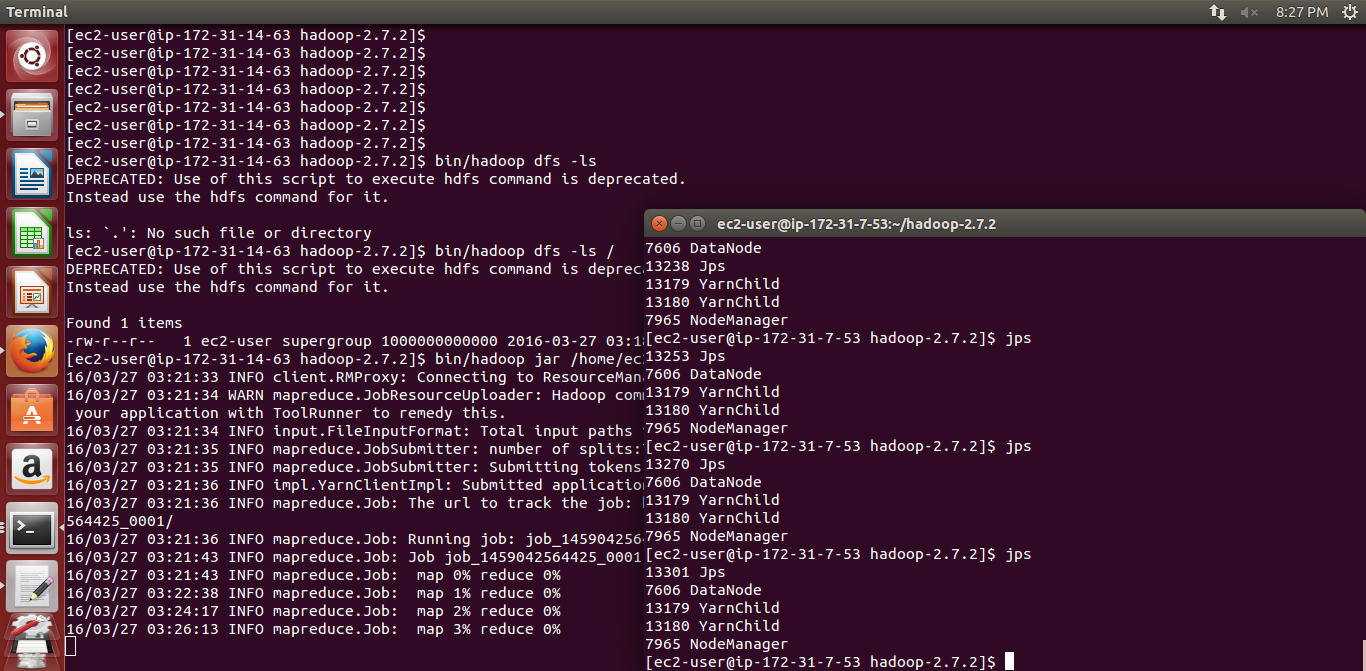
16 node running

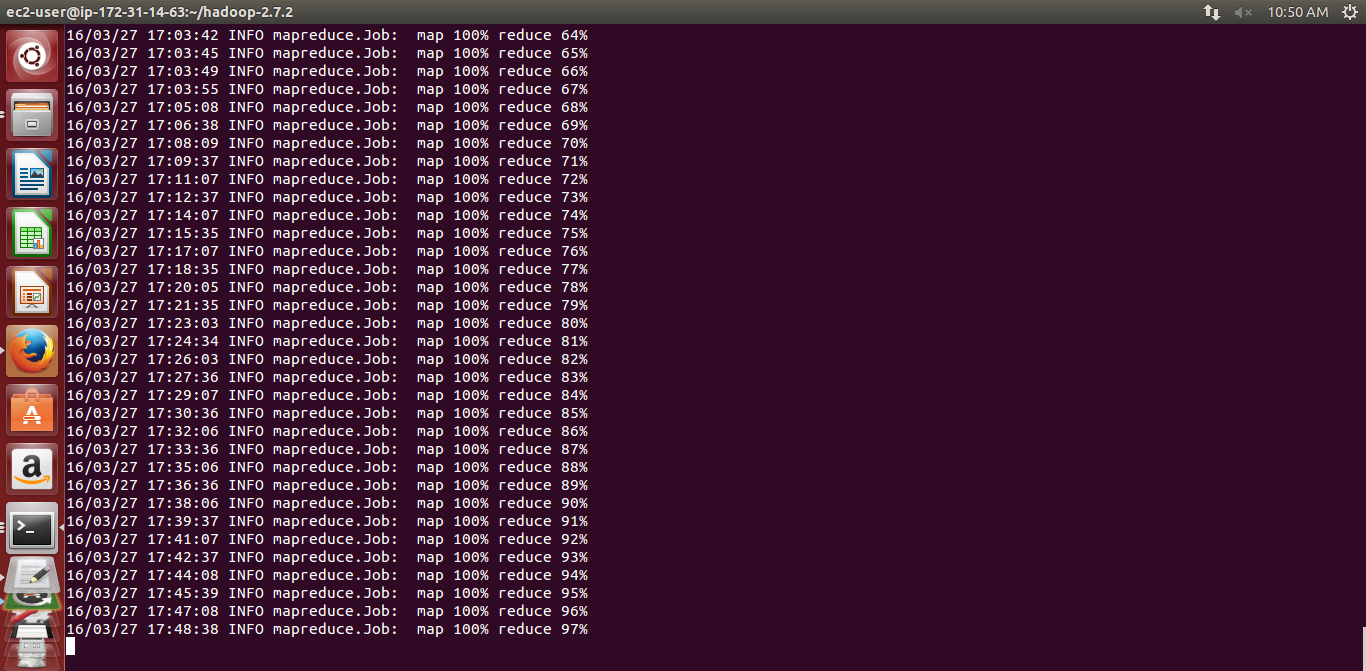


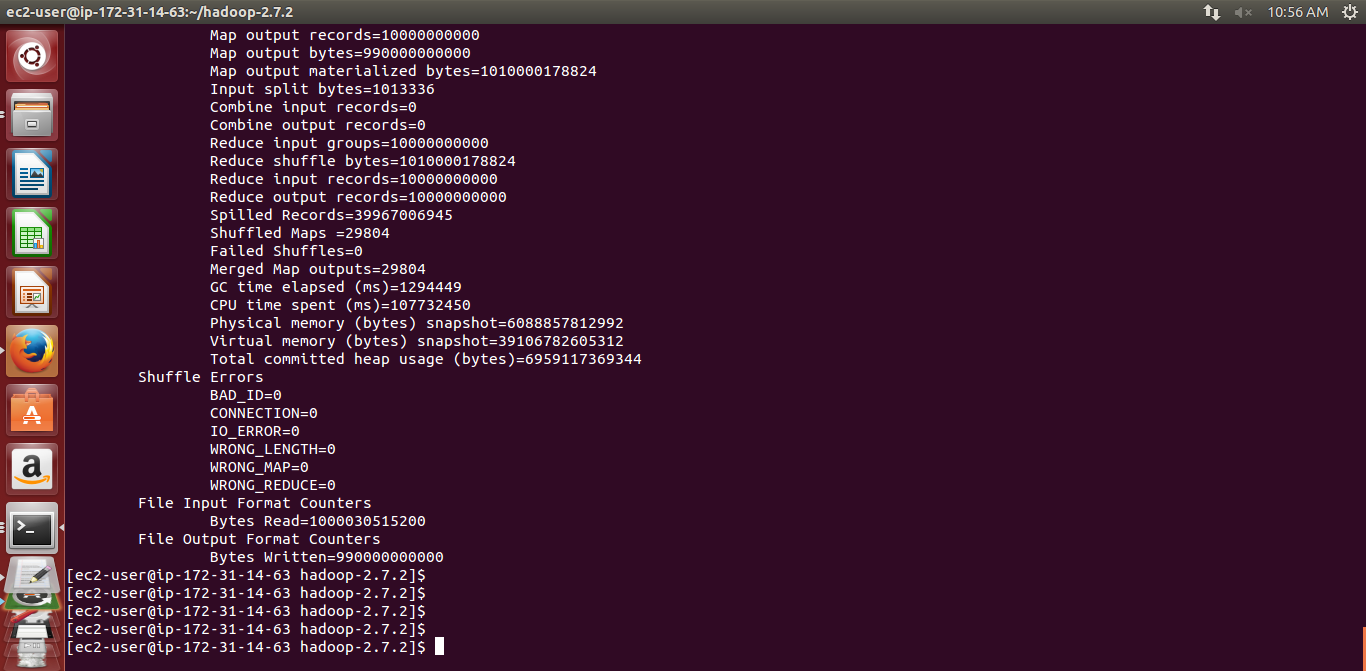


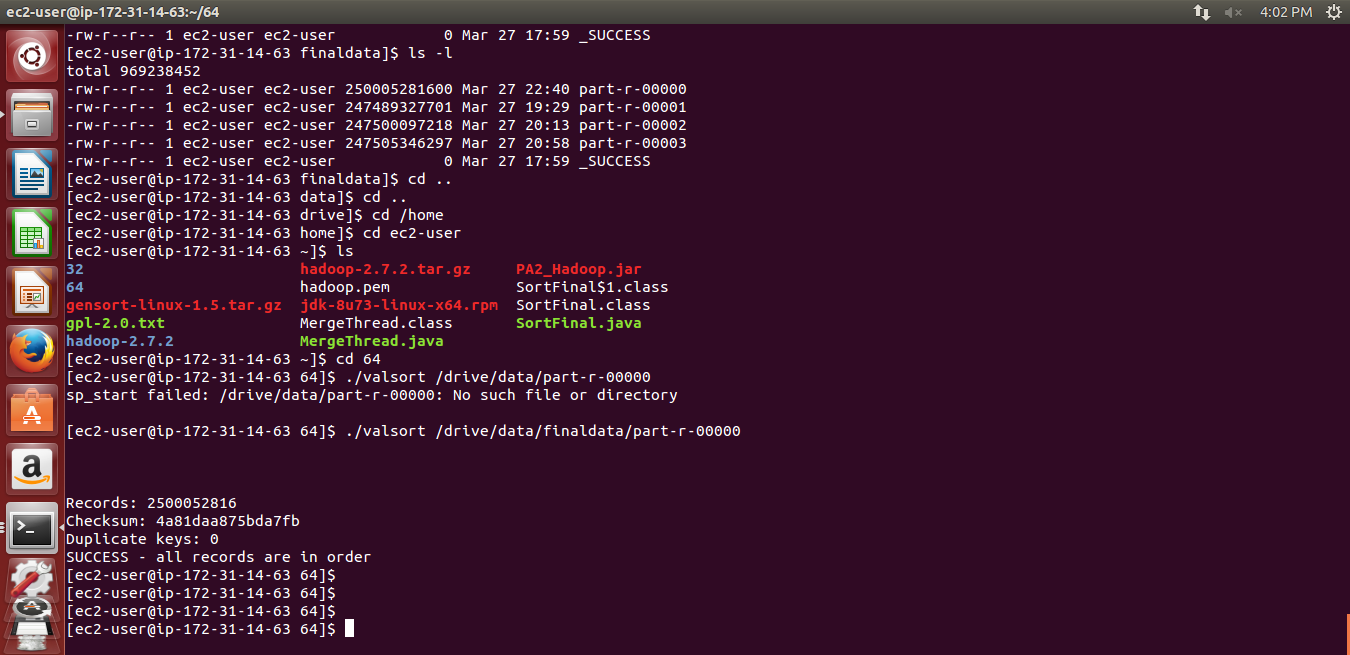












Hadoop for 10 gb execution time is

12.4 min= 745 seconds

Hadoop for 100 gb execution time is

25.8 min=1548 seconds

Hadoop for 1Tb Execution Time is

3 hr 55 min = 12780 seconds

Script for Automation

For Automation we used Ami image of application and

Scp to send files to each slave node

Scripts for running configuration in 16 nodes

scp -i pemfile configurationfile ec2-user@hostnameofslave

Adding keyagent identity

we add key to ssh so that each slave can be accessed by master using passphraseles ssh

Eval `ssh-agent –s`

Chmod 600 pemfile

ssh-add pemfile

Configuration in each slave node

Creating raid

sudo mdadm --create --verbose /dev/md0 --level=stripe --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd

sudo mkfs.ext4 /dev/md0

sudo mkdir /drive

sudo mount -t ext4 -o noatime,nodiratime,rw /dev/md0 /drive

cd /drive

sudo chown ec2-user /drive

mkdir data

mkdir name

mount

3 Spark

Configuration File for Spark

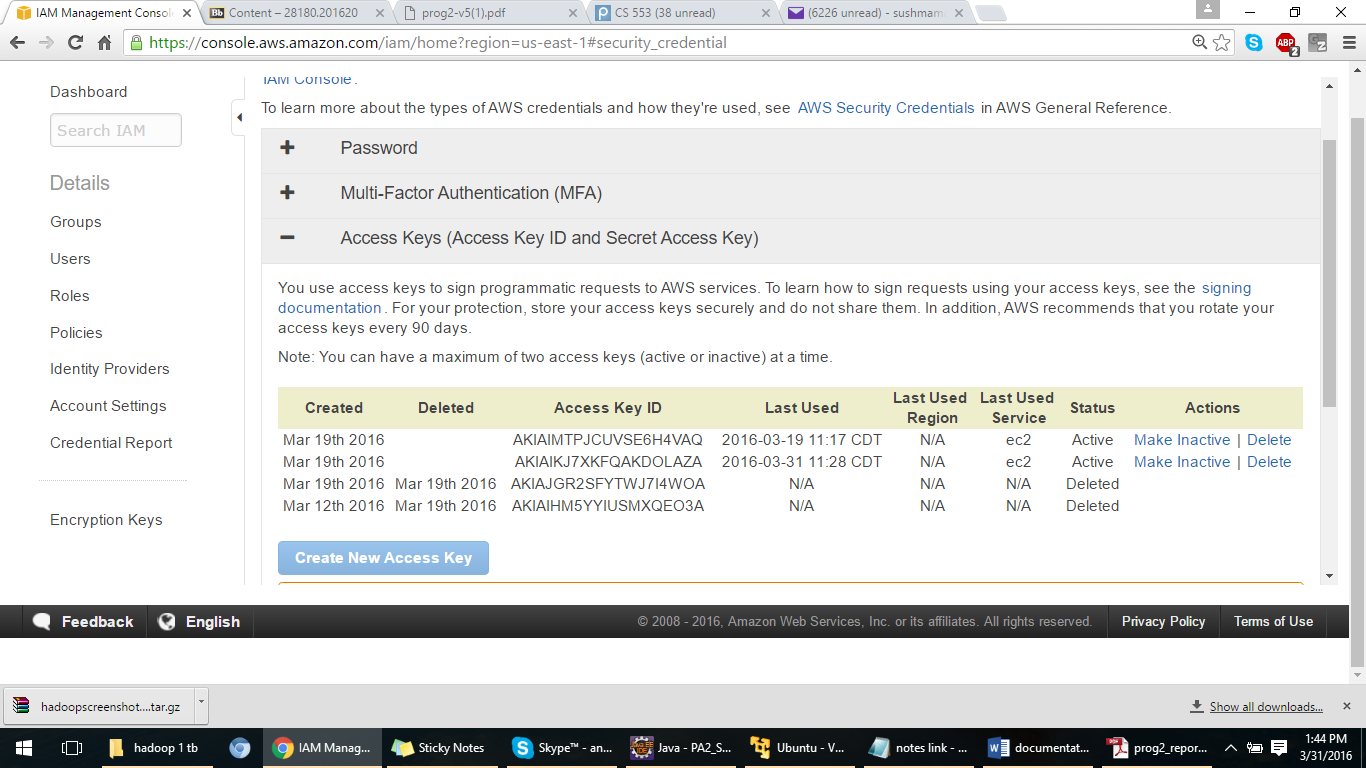
For Spark we check whether all workers are running using jps command

All slave nodes will have worker and data node

Master node will have a master resource manager and name node

Spark Run commands

We first download security credentials file from ec2 dashboard from the security credentials tab



We first export awssecretkey and awskeyid using

export AWS\_ACCESS\_KEY\_ID=keyed

export AWS\_SECRET\_ACCESS\_KEY=liQQanhZzwZu5NZZjATHdm4SYfotiR+118Ulgcdt

Inside spark-1.6.0-bin-hadoop-2.6/ec2 folder

We Start Spark cluster instances using command

. /spark-ec2 –k keypairname –I keypair with path –s noofslaves –t instancetype –r region –spot-price=price launch clustername

. /spark-ec2 -k spark -i /home/sushma/spark.pem -s 1 -t d2.xlarge -r us-east-1 --spot-price=0.2 launch spark

Goto spark folder and execute command

./bin/spark-submit --class (classname)TeraSort\_Spark --master spark://masterhostname:7077

(Jarname)PA2\_Spark.jar /dataset(inputfilename) /output(outputfilename)

Configuration in each slave for sparks

Mount 100 gb storage directory

Set the dfs replication to 1

Set the datanode directory to the mounted drive for 100 gb /vol0

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.block.size</name>

<value>134217728</value>

</property>

<property>

<name>dfs.blocksize</name>

<value>134217728</value>

</property>

<property>

<name>dfs.data.dir</name>

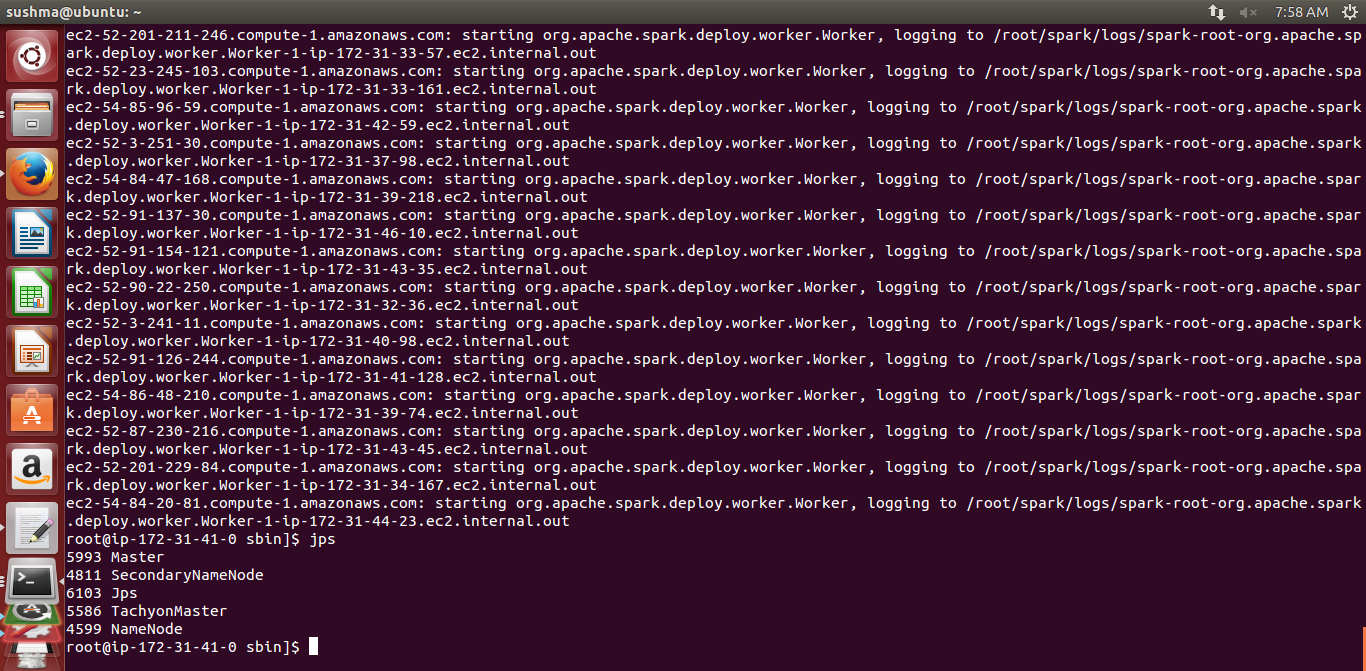
<value>/vol0/data,/mnt/ephemeral-hdfs/data,/mnt2/ephemeral-hdfs/data</value>

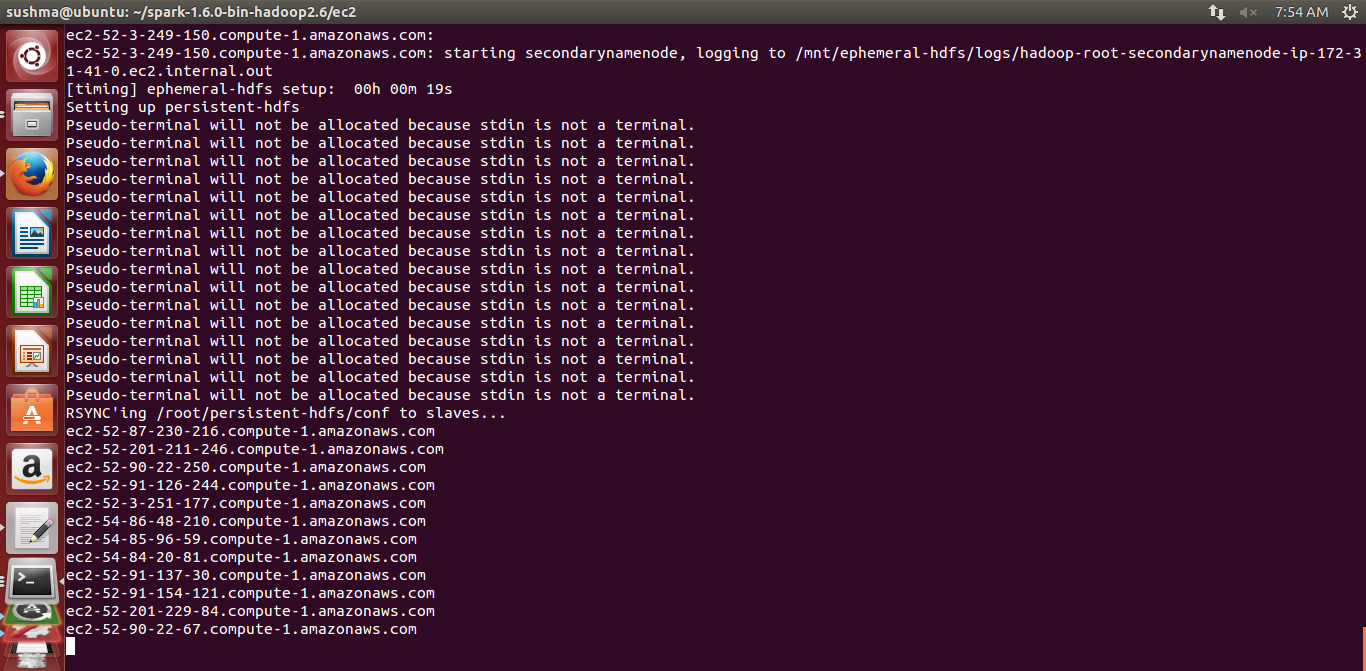
</property>

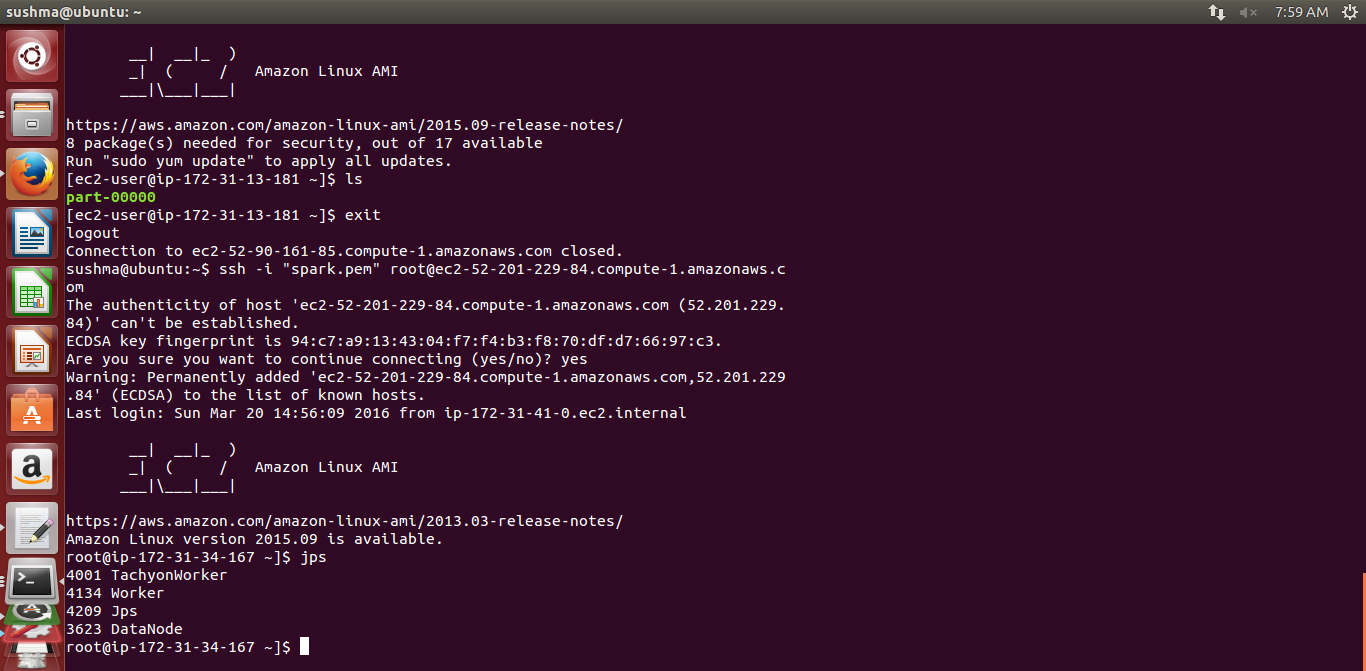
Changed the java version to 1.8 using

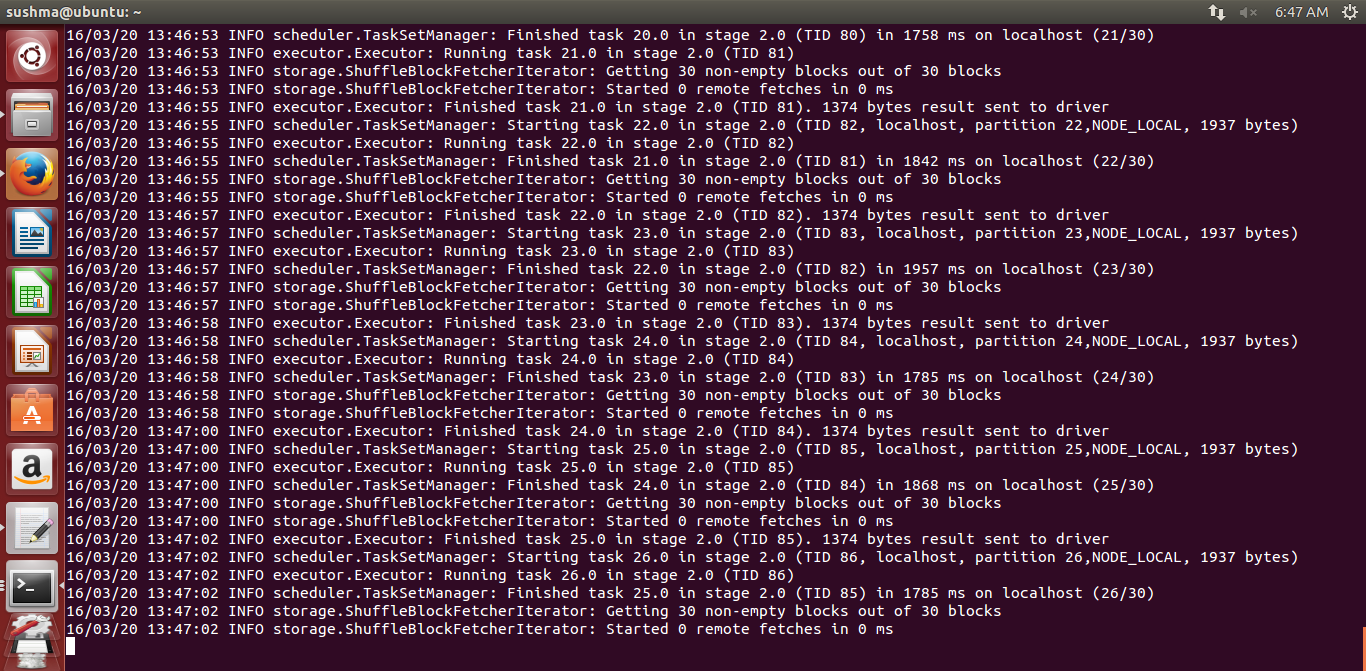
Export JAVA\_HOME=/usr/java/latest

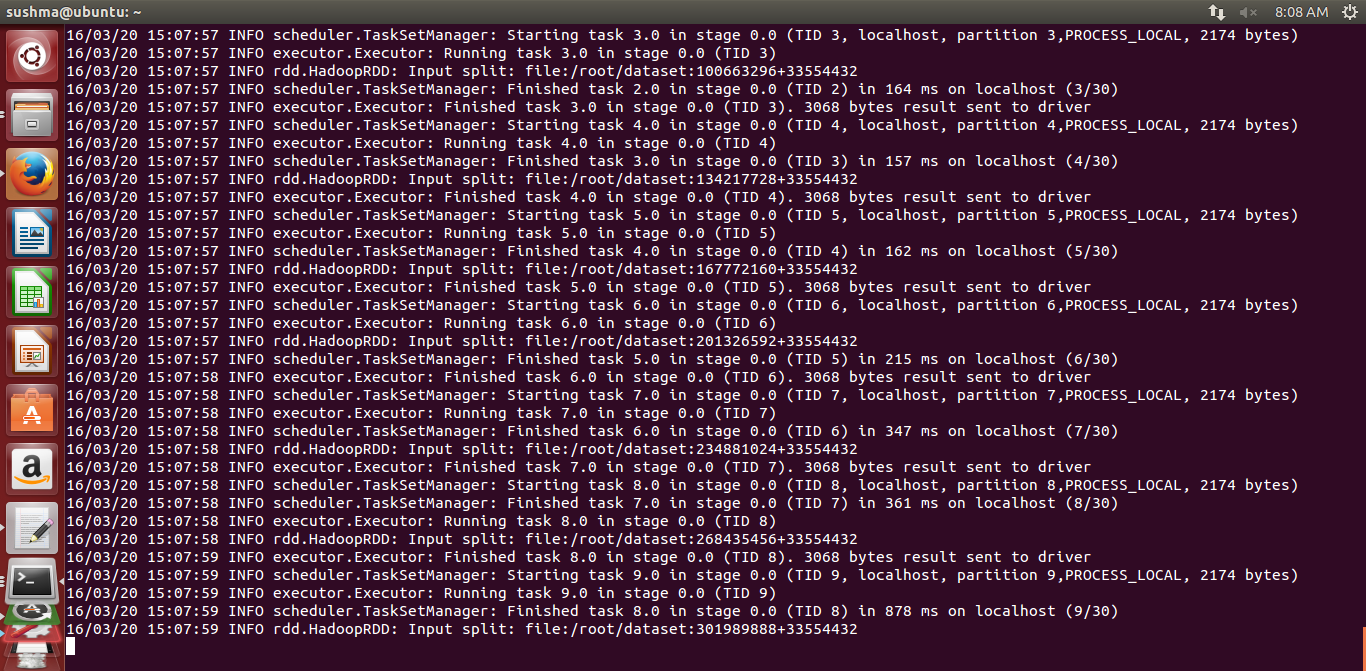
Spark Screenshots

****

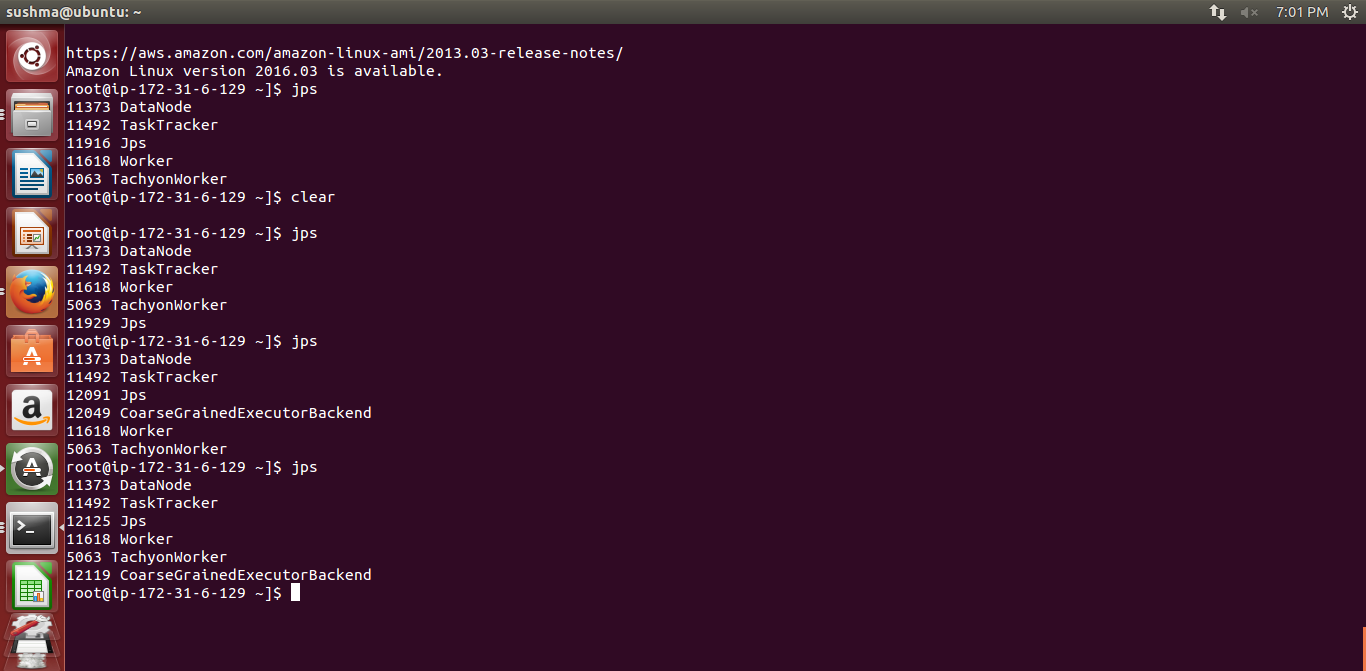
****

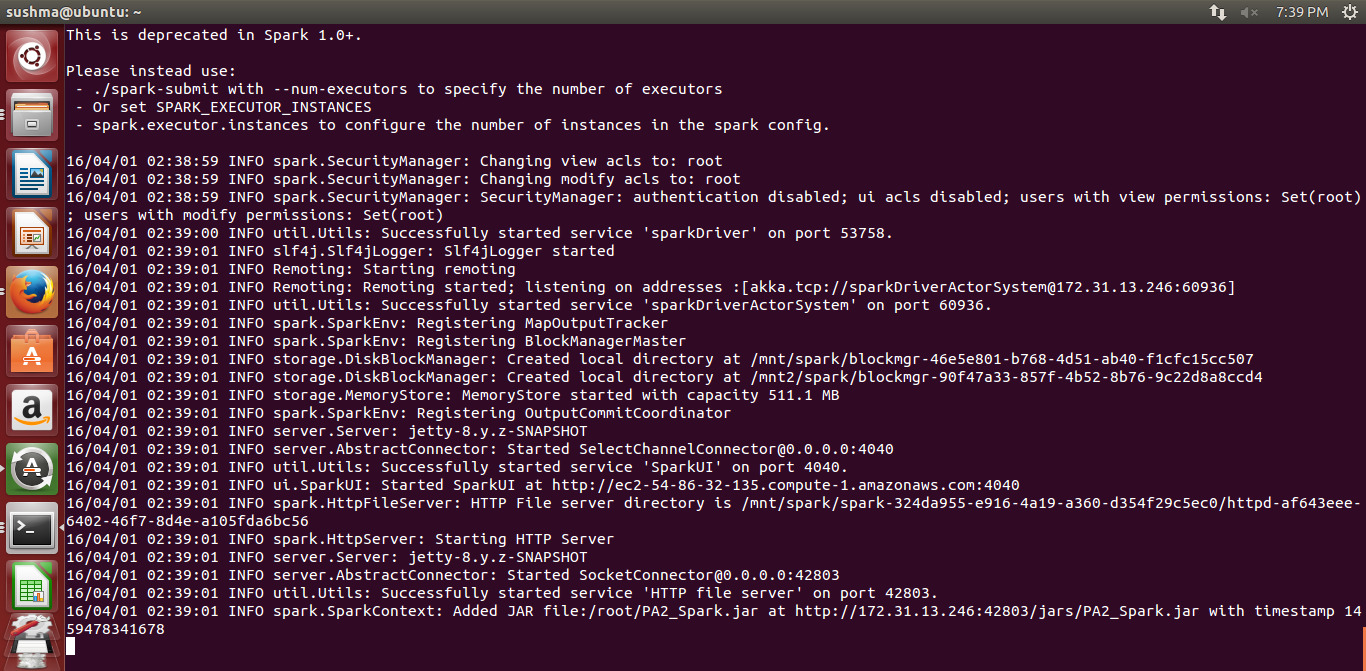






Spark Program Running in Worker





NameNode: It monitors the storage of data .It is central repository that manages the sharing of data throughout the cluster It stores the metadata of hdfs file system .Only master node has namenode to control the system

DataNode: It is the node where data is stored in hdfs file system slave which process the data contain datanode

Node Manager: Yarn Agent manages each slave in a cluster

Resource Manager: This Manager is used for getting cluster information, scheduling of jobs across slaves

SecondaryNameNode: it is backup for namenode

Master: Master Node

Worker: They are slave nodes that perform task assigned

Spark Execution Time

Spark Program 10 Gb Execution Time: 14.5 min =870 seconds

Spark Program 100 Gb Execution Time: 23.6 min =1416 seconds

4 Questions

1 What is a Master node? What is a Slaves node?

Ans Master node is node that starts the application process which controls all the processing of application in different node

It contains Hadoop name node and resource manager the name node controls the application

Slave nodes are nodes where application runs and process map reduce on data

It contains node manager and data node the data node performs all the tasks of mapreduce

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?

Ans Shared environment we need to set each port no If port are same then the application wont run as each process (resource manager ,namenode ,nodemanager,datanode )will listen to same port then they wont start as there would be conflict in their operation.For each process to run we need to set unique port .Error Side effects shown as job process wont start hence program wont run

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

No of mappers and reducers can be changes by adding

mapred.map.tasks and mapred.reduce.tasks in hdfs-site.xml

or mapreduce.job.maps and mapreduce.job.reduces in mapred-site.xml file or

<property>

<name>mapred.map.tasks</name>

<value>4</value>

</property>

<property>

<name>mapred.reduce.tasks</name>

<value>4</value>

</property>

We can also set the map and reduce in mapred-site.xml

Graphs

Performance [40 points] Compare the performance of the three versions of TeraSort (Shared-Memory, Hadoop, and Spark) on 1 node scale and explain your observations; compare the Shared-Memory performance of 1 node to Hadoop and Spark TeraSort at 16 node scales and explain your observations. You should be doing strong scaling experiments as you scale up from 1 node to 16 nodes.

Draw an execution line chart and a speed up line chart for 1 node and 16 node cases, for Java, Hadoop, and Spark TeraSort

Draw an execution line chart and a speed up line chart for 1 node and 16 node cases, for Java, Hadoop, and Spark TeraSort. For Spark and Hadoop, compute two different speedups, using different base cases; one speedup (speedup-shared-memory) should be relative to the Shared-memory TeraSort performance (note that you might get a speedup less than 1); the second speedup (speedup-spark & speedup-hadoop) should be relative to the Spark or Hadoop performance at 1 node scale respectively (should be a number greater than 1).

Hadoop

Hadoop for 10 gb execution time is

12.4 min= 745 seconds

Hadoop for 100 gb execution time is

25.8 min=1548 seconds

Throughput for Hadoop

For 10 gb=10000 mb

Throughput for 1 node=size/timetaken=10000/745=13.42

Throughput for 16 node=size/timetaken=10gb/=100000/1548=64.59

Throughput for 1 tb Hadoop=1000000/12780=782.47

As no of nodes increase the time increases but execution scales up faster as we increase the nodes

Hadoop does not store the intermediate shuffle data hence it takes time for map reduce

16 node Hadoop and spark comparison

As no of nodes increase the time increases but execution scales up faster as we increase the nodes

As spark does in memory computation and stores intermediate values

Spark Program 10 Gb Execution Time: 14.5 min =870 seconds

Spark Program 100 Gb Execution Time: 23.6 min =1416 seconds

Throughput for 1 node=size/timetaken=10000/870 =11.49

Throughput for 16 node=size/timetaken=100000/1416=70.62

Shared Memory

Shared Memory Graph

As we increase no of threads time taken to read and write decreases upto a certain limit and increase again as threads take more time waiting and synchronizing to read sort and write file back and merge files. Hence best performance is achieved at 2 threads

Best Performance Is Achieved in thread 2 at 905 seconds

|  |  |  |  |
| --- | --- | --- | --- |
| Threads | Execution Time | Constant Value of Thread Best Performance Thread | Spped up  Performance |
| Thread1 | 939.05 | 905.45 | 0.96 |
| Thread2 | 905.45 | 905.45 | 1 |
| Thread3 | 928.91 | 905.45 | 0.974 |
| Thread4 | 905.97 | 905.45 | 0.99 |

Throughput for shared memory =10000/905.45=11 mb/sec

Execution Time v/s Speed up for Shared Memory

All Three Comparision for 1 node

Performance Calculation

One node Hadoop takes 745 seconds while spark takes 870 seconds and java shared memory program take 905 seconds

For 16 nodes 100 gb Hadoop takes around 26 minutes while spark takes around 24 minutes and shared memory takes around 5200 seconds

So Spark is better overall for both 1 node and 16 nodes

What conclusions can you draw? Which seems to be best at 1 node scale? How about 16 nodes? Can you predict which would be best at 100 node scale? How about 1000 node scales? Compare your results with those from the Sort Benchmark [9], specifically the winners in 2013 and 2014 who used Hadoop and Spark. Also, what can you learn from the CloudSort benchmark, a report can be found at [10]

Ans

For 100 nodes the performance will scale up more as we distribute and sort data across more instances.

For 16 nodes the performance was around 3 hrs whereas 1 node it was approx. 10 hrs

So we achieved thrice the performance benefit by spreading and sorting data across multiple nodes

Hence on 100 nodes we may get more scale up performance of sorting in seconds and for 1000 nodes we may get performance in milliseconds