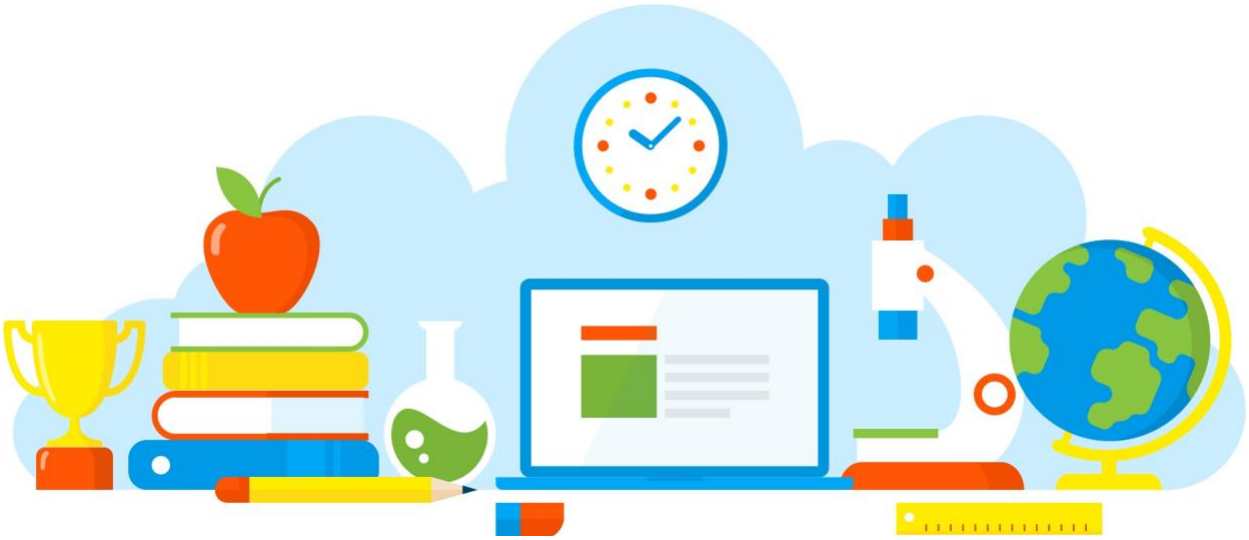


CS 553 - Cloud Computing

Illinois Institute of Technology

Team Members:

Sayed Ahmed - A20388365

Syed Hamdan Sher - A20378710

Hiral Ramani – A20370004

Introduction

The goal of this programming assignment is to enable you to gain experience programming with:

- Amazon Web Services, specifically the EC2 cloud (http://aws.amazon.com/ec2/)
- The Hadoop framework (http://hadoop.apache.org/)
- The Spark framework (http://spark.apache.org/)

Assignment

This programming assignment covers the TeraSort application implemented in 3 different ways: Java/C, Hadoop, and Spark. Our sorting application could read a large file and sort it in place. We will create 2 datasets, a small and a large dataset, which we will use to benchmark the 3 approaches to sorting: 128GB dataset and 1TB dataset. We generated our datasets using the file generator at [1] and Hadoop's TeraGen[2]; since storing 1TB dataset on Amazon S3 would be both expensive and slow, we are encouraged to generate the 1TB dataset on demand every we you want to benchmark our system.

Set Up.

Shared Memory :

Go to your AWS account and launch i3.large and i3.4xlarge.

Scp your shared memory programs along with gensort and other required files and script

Ssh in to the desired instance and install default-jdk and make :-

Sudo apt install default-jdk

Sudo apt install make

The number of threads is hard coded inside the file (default is 2) to run with different thread count please change nth= # thread count number in the main function and then run the java file.

Make sure you have to empty directory "input" and "output".

Mkdir input

Mkdir output

Generate input.txt using gensort file and store the input.txt in the same location of the java file.

./gensort -a <filesize> <filelocation>

Example for 128 GB : ./gensort -a 1374389534 input.txt

The input and output filename is hard cored in the code so make sure the input file name is input.txt when generating the file with the desired size using gensort.

Run make or compile the java file as javac filename.java and run by java filename

Note the output for reporting.

Hadoop :

Apache Hadoop

Apache Hadoop is the distributed software processing framework for a large dataset. Hadoop mapreduce essentially contains two main process which is Map and Reduce.

Map Phase:

This Phase of Hadoop takes the dataset as an input stored in the Hadoop File system (HDFS) and converts it into another dataset where, each data set is broken down into set to tuples represented as Key, Value Pair. Map phase main responsibility is to create the map of key- value pair.

Reduce Phase:

This phase takes an output from Map phase as an input and combines those set up tuples into a smaller dataset such that value with similar dataset will be in one set during reduce phase. The generated output is stored back to the HDFS.

The biggest advantage of Hadoop Map Reduce is that it’s easy to scale the processing of data from one node to multiple nodes.

Hadoop cluster with many nodes works in master-slave fashion, where one master who is responsible to distribute the jobs across Slaves (Workers).

What is Master?

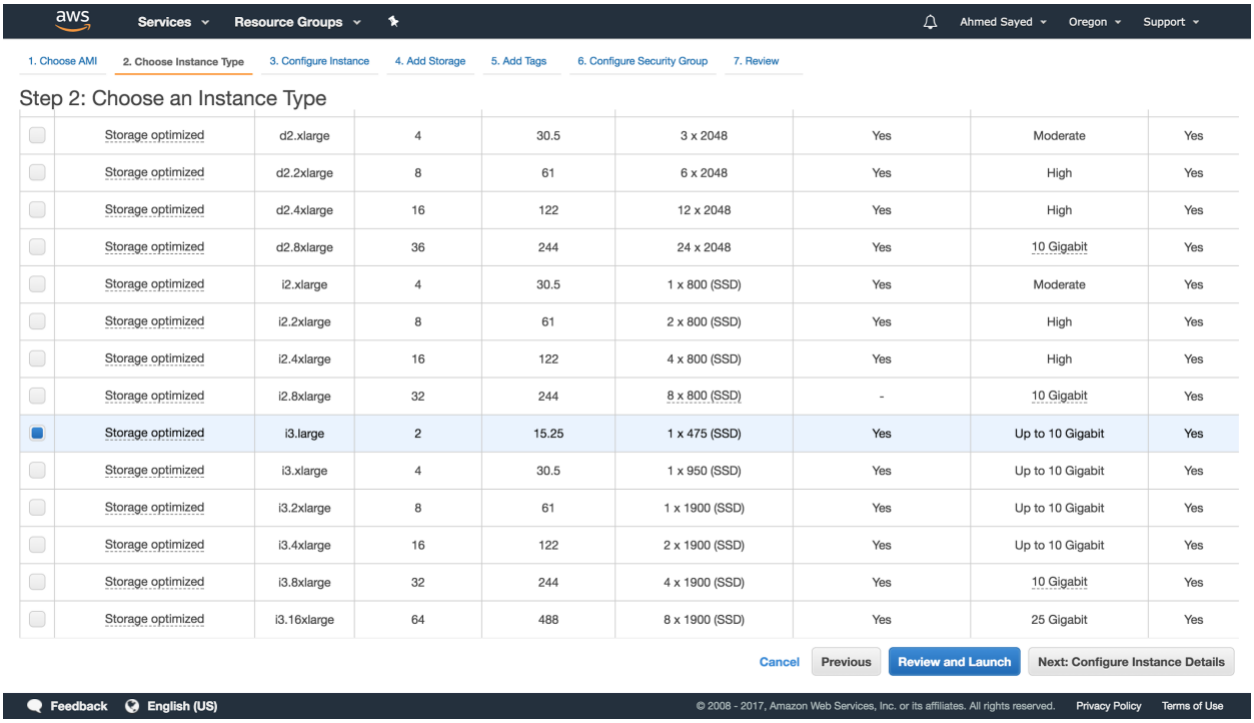
Hadoop's master node (Name Node) is responsible to manage the operations of file system namespace like opening, closing, renaming files and determining the mapping of blocks to Data Nodes.

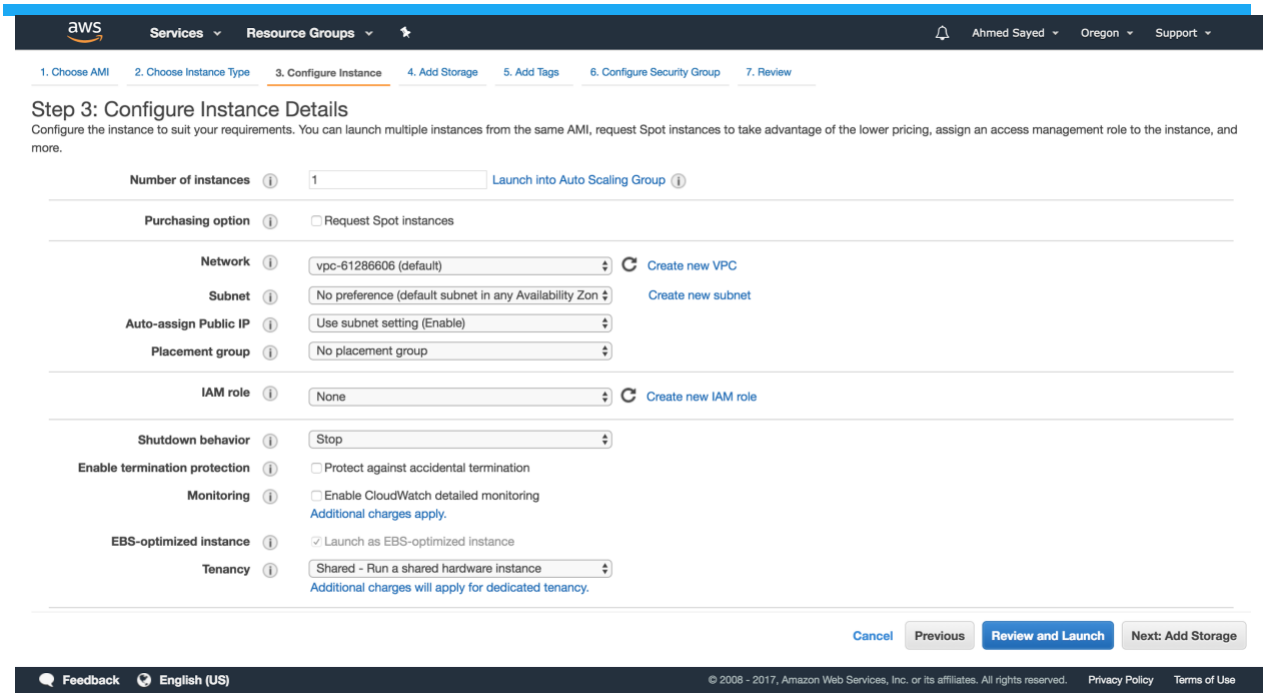
What is slave?

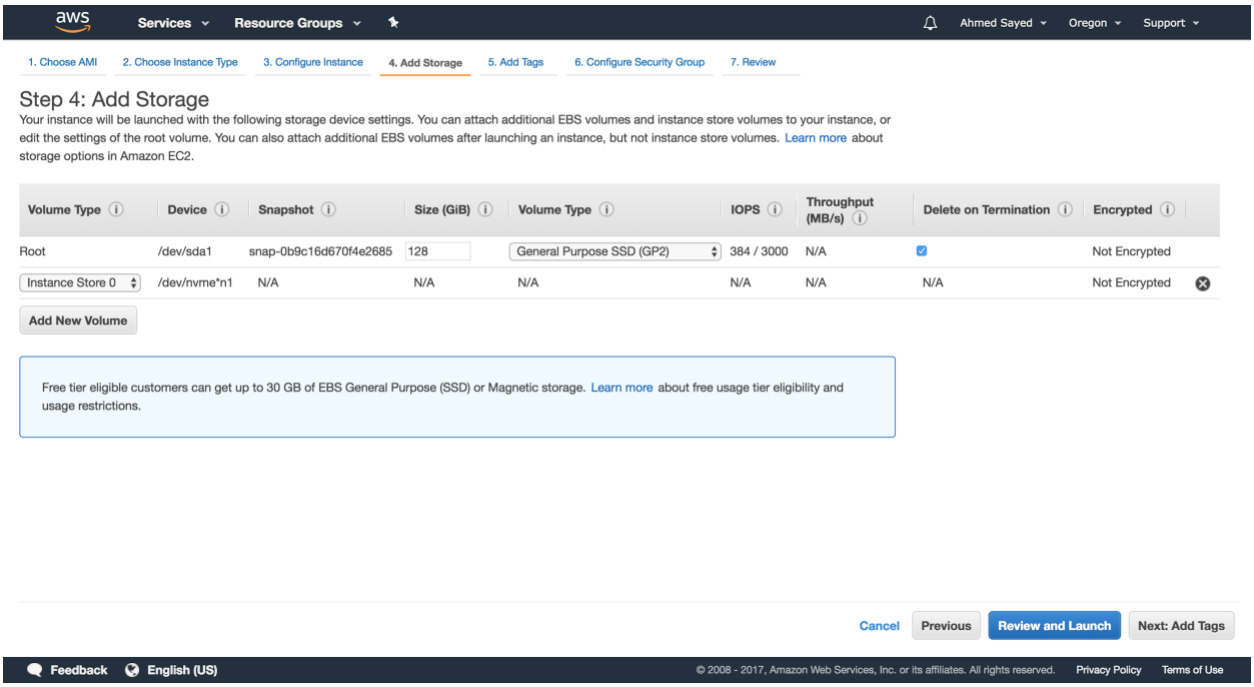
Hadoop's Slaves (Data Nodes) are responsible for serving read and write requests from the file system's clients along with perform block creation, deletion, and replication upon instruction from the Master.

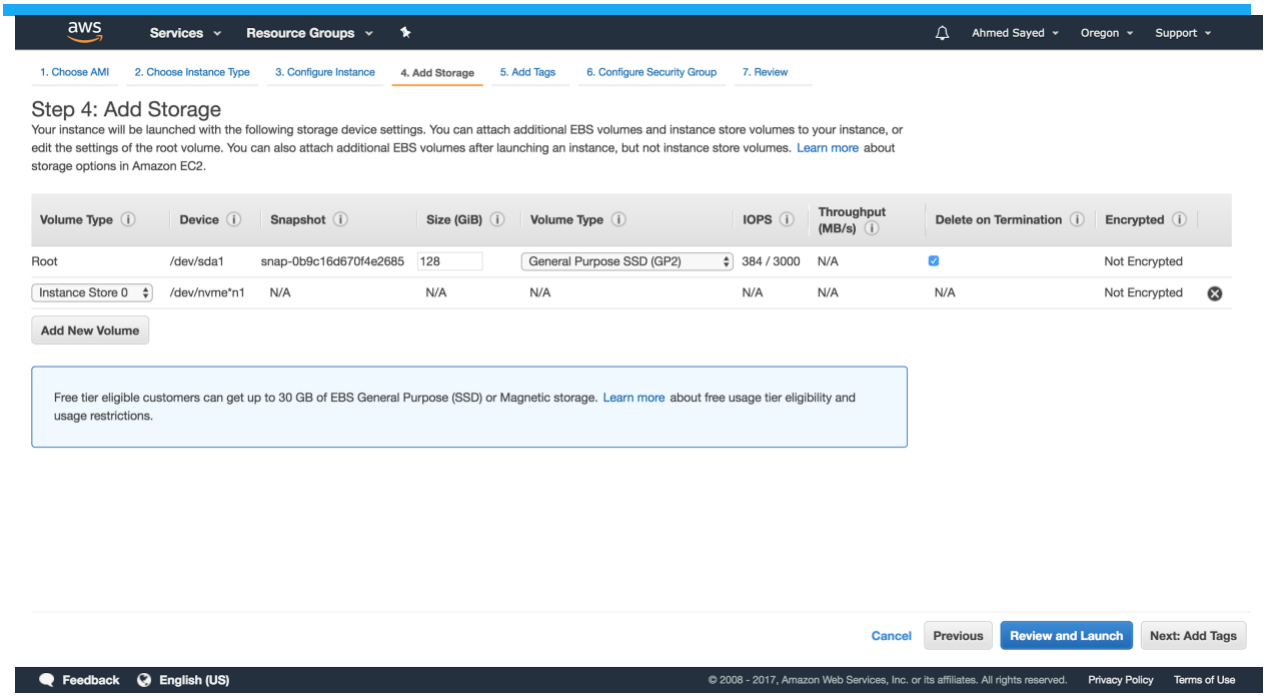
Configuration on Single Node [6,7]:

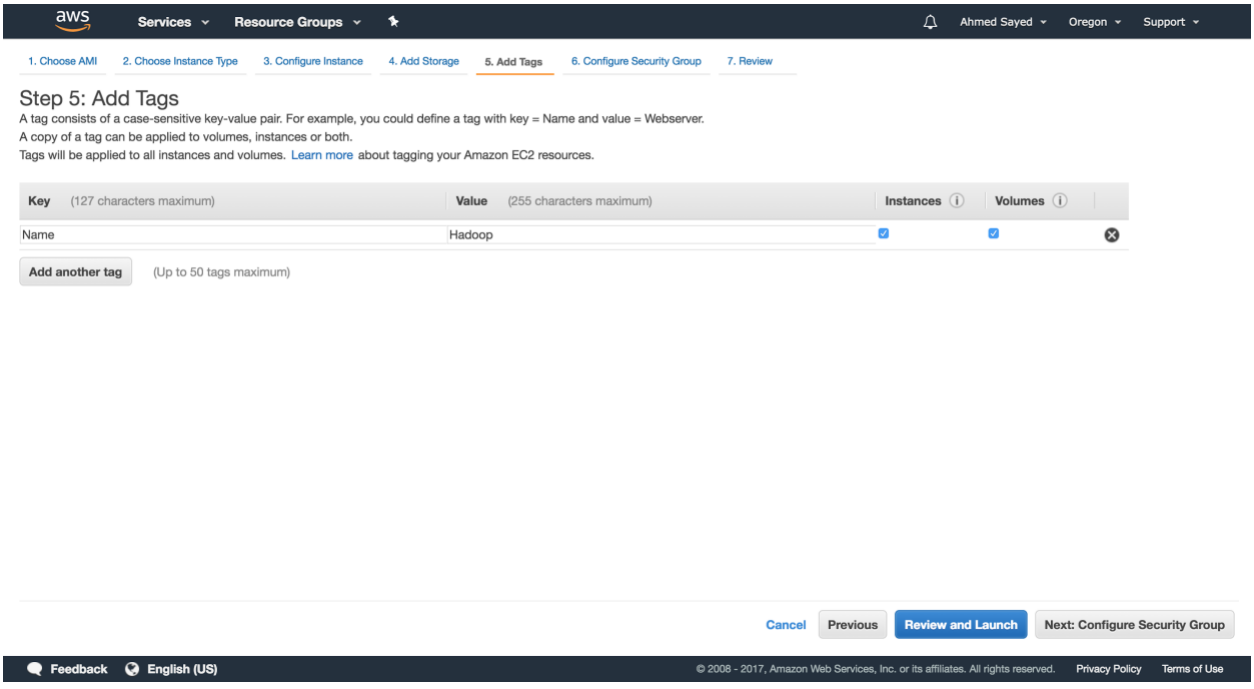
Follow the following steps in order to install Hadoop on an Instance

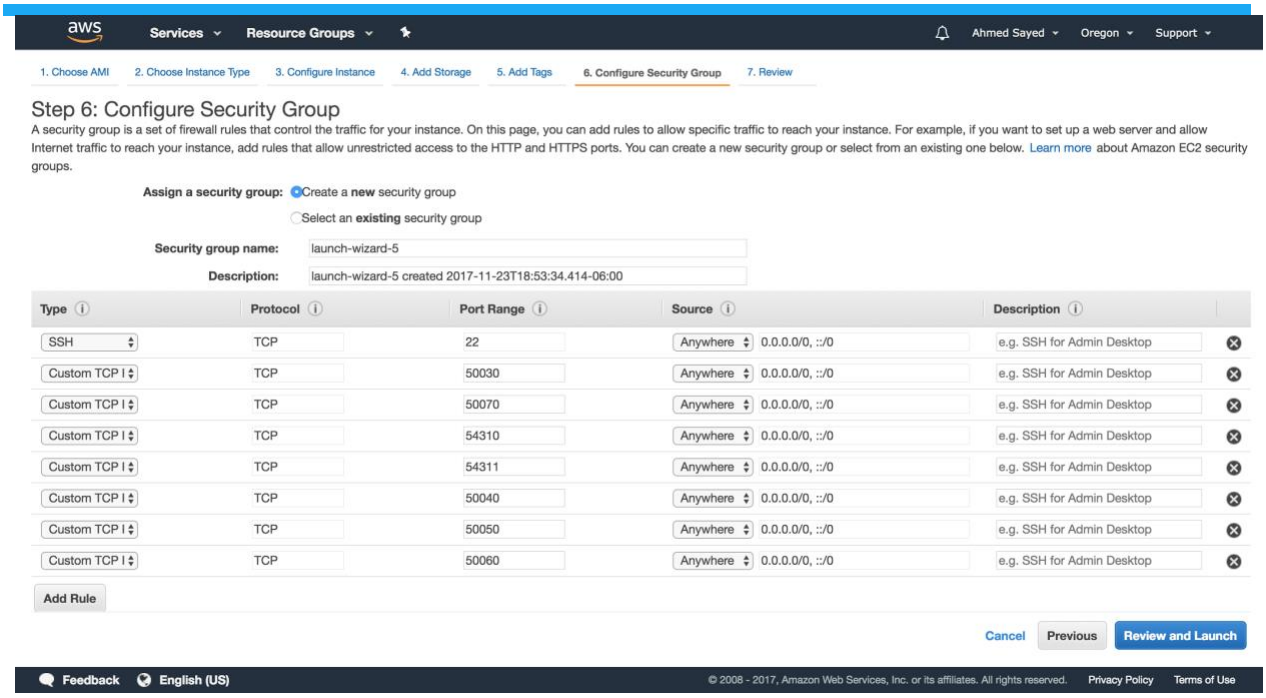


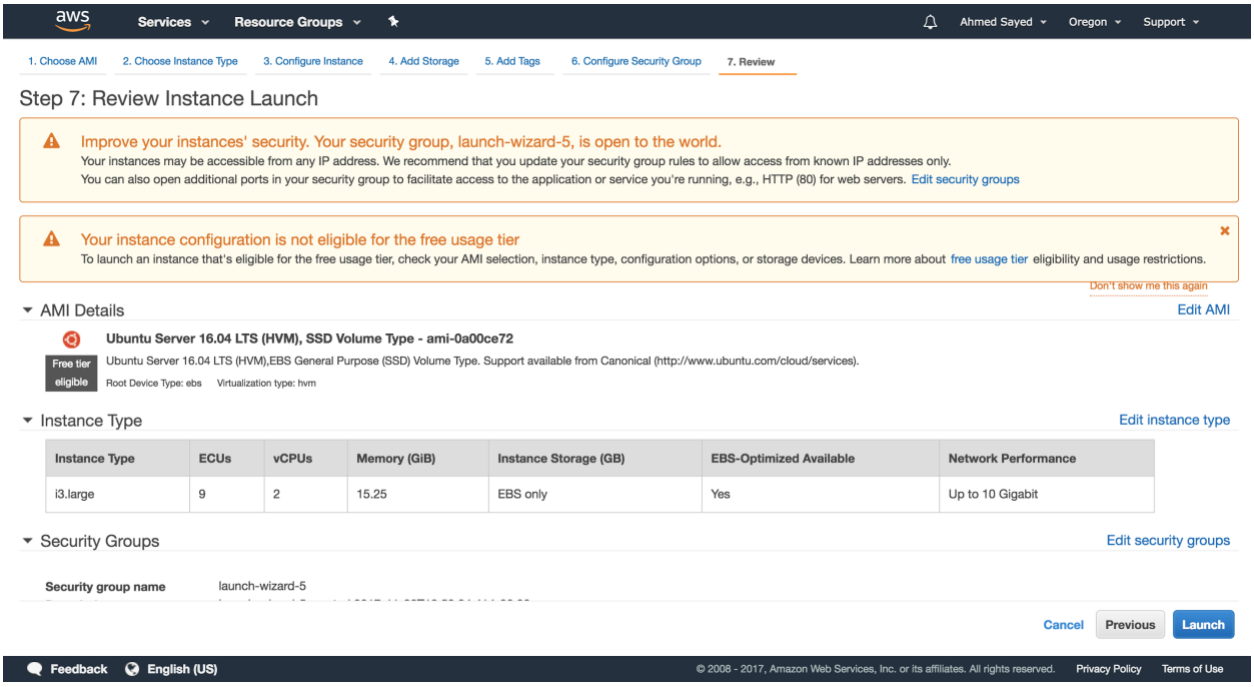

aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances
1
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Network
vpc-61286606 (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zon
Create new subnet
Auto-assign Public IP
Use subnet setting (Enable)
Placement group
No placement group
IAM role
None
Create new IAM role
Shutdown behavior
Stop
Enable termination protection
Protect against accidental termination
Monitoring
Enable CloudWatch detailed monitoring
Additional charges apply.
EBS-optimized instance
Launch as EBS-optimized instance
Tenancy
Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.
Cancel
Previous
Review and Launch
Next: Add Storage
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

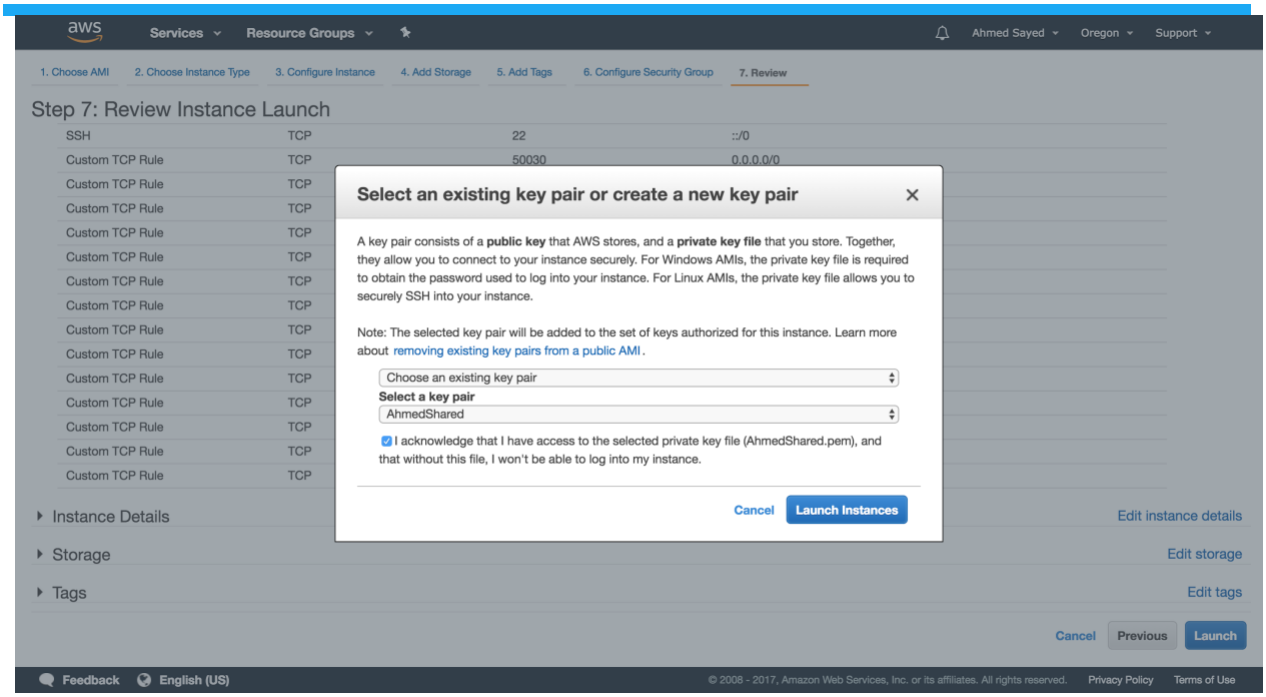

aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 4: Add Storage
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.
Volume Type
Device
Snapshot
Size (GiB)
Volume Type
IOPS
Throughput (MB/s)
Delete on Termination
Encrypted
Root
/dev/sda1
snap-0b9c16d670f4e2685
128
General Purpose SSD (GP2)
384 / 3000
N/A
Not Encrypted
Instance Store 0
/dev/nvme*n1
N/A
N/A
N/A
N/A
N/A
N/A
Not Encrypted
Add New Volume
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.
Cancel
Previous
Review and Launch
Next: Add Tags
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

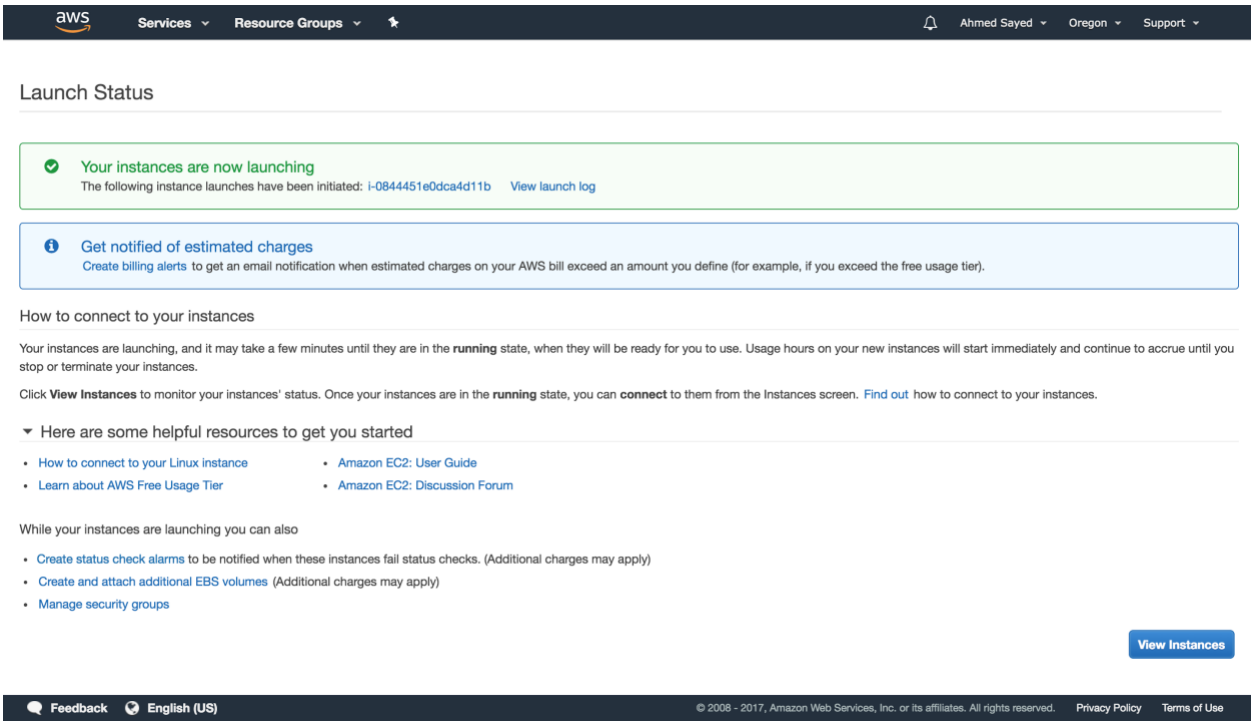

aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 4: Add Storage
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.
Volume Type
Device
Snapshot
Size (GiB)
Volume Type
IOPS
Throughput (MB/s)
Delete on Termination
Encrypted
Root
/dev/sda1
snap-0b9c16d670f4e2685
128
General Purpose SSD (GP2)
384 / 3000
N/A
Not Encrypted
Instance Store 0
/dev/nvme*n1
N/A
N/A
N/A
N/A
N/A
N/A
Not Encrypted
Add New Volume
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.
Cancel
Previous
Review and Launch
Next: Add Tags
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

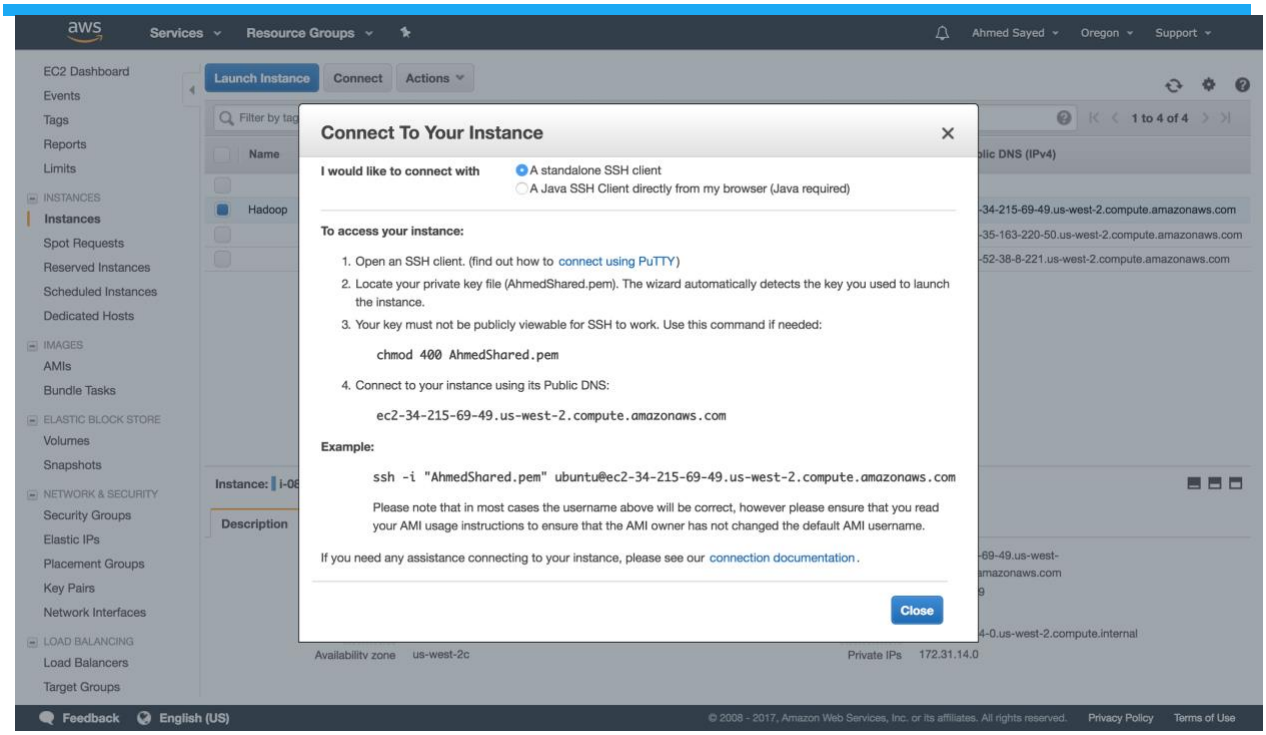

aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 5: Add Tags
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.
Key (127 characters maximum)
Value (255 characters maximum)
Instances
Volumes
Name
Hadoop
Add another tag
(Up to 50 tags maximum)
Cancel
Previous
Review and Launch
Next: Configure Security Group
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 6: Configure Security Group
A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.
Assign a security group: Create a new security group
Select an existing security group
Security group name: launch-wizard-5
Description: launch-wizard-5 created 2017-11-23T18:53:34.414-06:00
Type
Protocol
Port Range
Source
Description
SSH TCP 22 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Custom TCP I TCP 50030 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Custom TCP I TCP 50070 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Custom TCP I TCP 54310 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Custom TCP I TCP 54311 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Custom TCP I TCP 50040 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Custom TCP I TCP 50050 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Custom TCP I TCP 50060 Anywhere 0.0.0.0/0, ::/0 e.g. SSH for Admin Desktop
Add Rule
Cancel
Previous
Review and Launch
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 7: Review Instance Launch
Improve your instances' security. Your security group, launch-wizard-5, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups
Your instance configuration is not eligible for the free usage tier
To launch an instance that's eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. Learn more about free usage tier eligibility and usage restrictions.
Don't show me this again
AMI Details
Edit AMI
Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-0a00ce72
Free tier eligible
Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root Device Type: ebs
Virtualization type: hvm
Instance Type
Edit instance type
Instance Type
ECUs
vCPUs
Memory (GiB)
Instance Storage (GB)
EBS-Optimized Available
Network Performance
i3.large
9
2
15.25
EBS only
Yes
Up to 10 Gigabit
Security Groups
Edit security groups
Security group name
launch-wizard-5
Cancel
Previous
Launch
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 7: Review Instance Launch
SSH TCP 22 ::/0
Custom TCP Rule TCP 50030 0.0.0.0/0
Select an existing key pair or create a new key pair
A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.
Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.
Choose an existing key pair
Select a key pair
AhmedShared
I acknowledge that I have access to the selected private key file (AhmedShared.pem), and that without this file, I won't be able to log into my instance.
Cancel
Launch Instances
Instance Details
Edit instance details
Storage
Edit storage
Tags
Edit tags
Cancel
Previous
Launch
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


aws
Services
Resource Groups
Ahmed Sayed
Oregon
Support
Launch Status
Your instances are now launching
The following instance launches have been initiated: i-0844451e0dca4d11b View launch log
Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).
How to connect to your instances
Your instances are launching, and it may take a few minutes until they are in the running state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.
Click View Instances to monitor your instances' status. Once your instances are in the running state, you can connect to them from the Instances screen. Find out how to connect to your instances.
Here are some helpful resources to get you started
How to connect to your Linux instance
Learn about AWS Free Usage Tier
Amazon EC2: User Guide
Amazon EC2: Discussion Forum
While your instances are launching you can also
Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
Create and attach additional EBS volumes (Additional charges may apply)
Manage security groups
View Instances
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use



Once the instance is ready, Copy the above command in Terminal and SSH in to it

```
[Ahmeds-MacBook-Pro:PA2 ahmedsayed$ ssh -i "AhmedShared.pem" ubuntu@ec2-34-212-113-254.us-west-2.compute.amazonaws.com
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1041-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

13 packages can be updated.
8 updates are security updates.


Last login: Sun Dec  3 12:22:09 2017 from 207.237.204.157
ubuntu@ip-172-31-9-44:~$
```

Installing Hadoop on Instance:

echo "Update apt-get"

sudo apt-get update

echo "Download Java"

sudo apt-get install default-jre //Install Default JRE version

sudo apt-get install default-jdk //Install Default JDK version

sudo apt-get install vim //Install VIM editor

wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz

//Download jdk for Linux

tar -xvzf jdk-8u131-linux-x64.tar.gz //Unzip it

ln -s jdk1.8.0_131 jdk //Link it

echo "Download Hadoop"

wget https://dist.apache.org/repos/dist/release/hadoop/common/hadoop-2.7.4/hadoop-2.7.4.tar.gz

//Download Hadoop

tar xfz hadoop-2.7.4.tar.gz //Unzip the Folder

ln -s hadoop-2.7.4 hadoop //Link the hadoop

echo "ssh localhost to connect it whenever one need it"

ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

echo "Path Set"

echo "export JAVA_HOME=/home/ubuntu/jdk

export HADOOP_INSTALL=/home/ubuntu/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" >> ~/.bashrc

```
echo "Configuring Hadoop for Single Node"

echo $(wget -qO- http://instance-data/latest/meta-data/public-ipv4) > hadoop/etc/hadoop/masters
                                                                    //Adding Node IP as Master's IP

echo $(wget -qO- http://instance-data/latest/meta-data/public-ipv4) > hadoop/etc/hadoop/slaves

                                                                    //Adding Node IP as Slaves's IP


echo "Configure core-site.xml"

echo "<property>

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

   <value>/home/ubuntu/tmp</value>

 </property>

 <property>

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

   <value>hdfs://$(wget -qO- http://instance-data/latest/meta-data/local-ipv4)</value>

 </property>" > coreNew

sed  '/<configuration>/r coreNew hadoop/etc/hadoop/core-site.xml > coreOld

mv coreOld hadoop/etc/hadoop/core-site.xml


echo "Configure hdfs-site.xml"

echo " <property>

  <name>dfs.replication</name>

   <value>1</value>

 </property>" > hdfsNew

sed  '/<configuration>/r hdfsNew hadoop/etc/hadoop/hdfs-site.xml > hdfsOld

mv hdfsOld  hadoop/etc/hadoop/hdfs-site.xml
```

```
cp hadoop/etc/hadoop/mapred-site.xml.template hadoop/etc/hadoop/mapred-site.xml
```

echo “Configure mapred-site.xml”

```
echo " <property>
   <name>mapred.job.tracker</name>
   <value>$(wget -qO- http://instance-data/latest/meta-data/public-ipv4):54311</value>
 </property>" > mapredNew
sed  '/<configuration>/r mapredNew hadoop/etc/hadoop/mapred-site.xml > mapredOld
mv mapredOld hadoop/etc/hadoop/mapred-site.xml
```

echo “Configure yarn-site.xml”

```
echo " <property>
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>
 </property>
 <property>
   <name>yarn.resourcemanager.address</name>
   <value>localhost:50040</value>
 </property>
 <property>
   <name>yarn.nodemanager.address</name>
   <value>localhost:50050</value>
 </property>
 <property>
   <name>yarn.nodemanager.localizer.address</name>
```

<value>localhost:50060</value>

</property>" > yarnNew

sed '/<configuration>/r yarnNew' hadoop/etc/hadoop/yarn-site.xml > yarnOld

mv yarnOld hadoop/etc/hadoop/yarn-site.xml

rm yarnNew

echo "Configure hadoop-env.sh"

sed -i 's/${JAVA_HOME}/\/home\/ubuntu\/jdk/g' hadoop/etc/hadoop/hadoop-env.sh

echo "Format DFS file system"

hdfs namenode -format

echo "Start DFS"

Start-dfs.sh

echo "Start Yarn"

Start-yarn.sh

echo "To Test Hadoop is Running properly"

JPS //Following output indicates all nodes are up

```
[ubuntu@ip-172-31-9-44:~$ jps
14178 SecondaryNameNode
14452 NodeManager
14728 Jps
14329 ResourceManager
13850 NameNode
13996 DataNode
```

//Also one can track using browser

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Overview 'ip-172-31-9-44.us-west-2.compute.internal:8020' (active)

Started:	Sun Dec 03 06:36:37 UTC 2017
Version:	2.7.4, rcd915e1e8d9d0131462a0b7301586c175728a282
Compiled:	2017-08-01T00:29Z by kshvachk from branch-2.7.4
Cluster ID:	CID-8041f28f-e914-483f-a247-3ba201ed0fa0
Block Pool ID:	BP-129193080-172.31.9.44-1512282979190

Summary

Security is off.

Safemode is off.

9 files and directories, 259 blocks = 268 total filesystem object(s).

Heap Memory used 38.22 MB of 434 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 50.72 MB of 51.6 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	290.76 GB
DFS Used:	32.44 GB (11.16%)
Non DFS Used:	132.07 GB
DFS Remaining:	126.23 GB (43.41%)
Block Pool Used:	32.44 GB (11.16%)
DataNodes usages% (Min/Median/Max/stdDev):	11.16% / 11.16% / 11.16% / 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	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	03/12/2017, 00:36:37

NameNode Journal Status

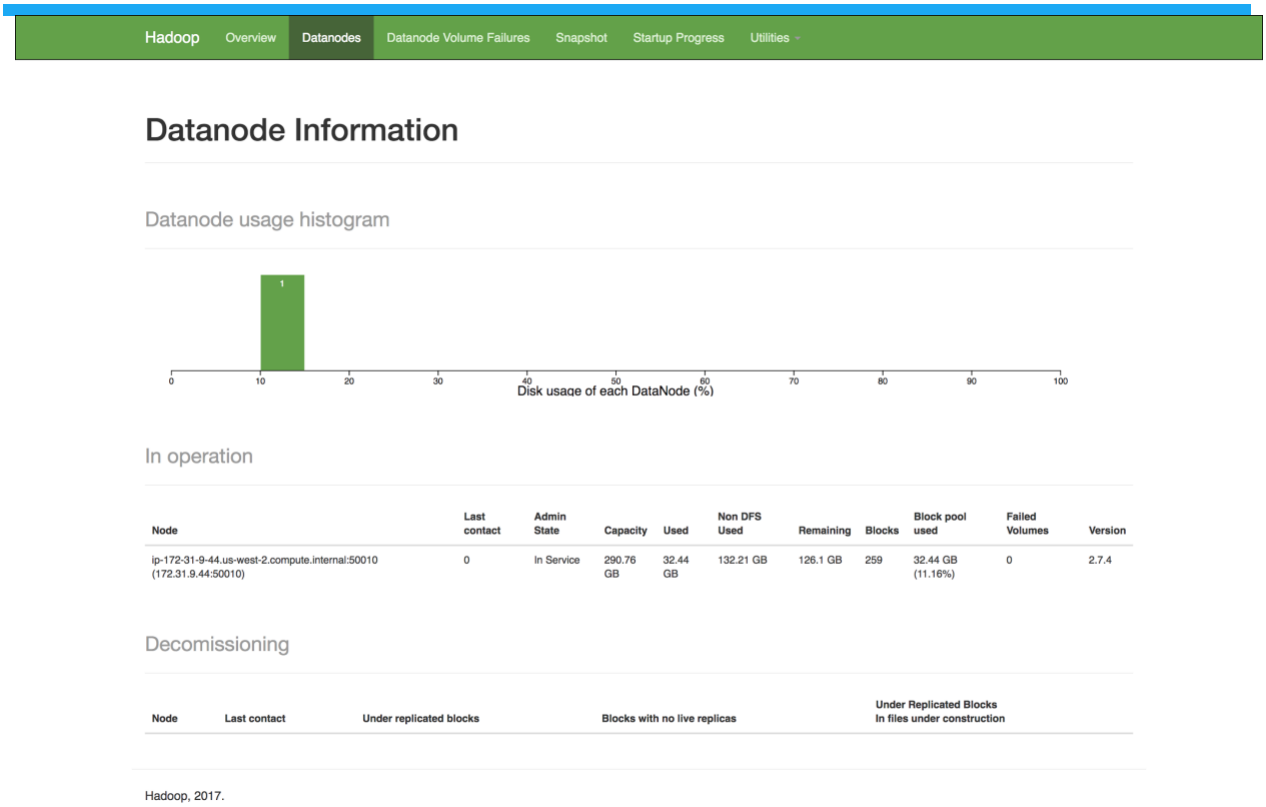
Current transaction ID: 1487

Journal Manager	State
FileJournalManager(root=/home/ubuntu/tmp/dfs/name)	EditLogFileOutputStream(/home/ubuntu/tmp/dfs/name/current/edits_inprogress_0000000000000001483)

NameNode Storage

Storage Directory	Type	State
/home/ubuntu/tmp/dfs/name	IMAGE_AND_EDITS	Active

Hadoop, 2017.

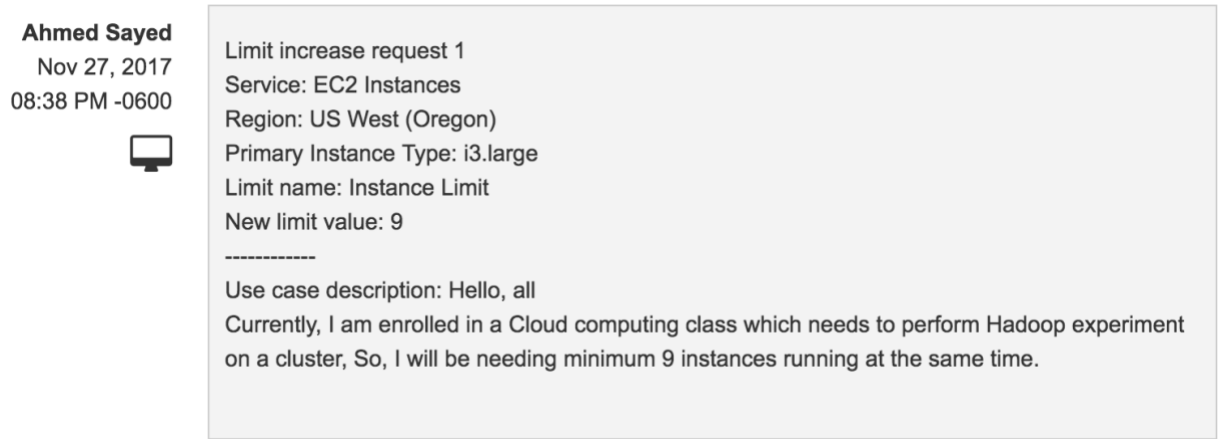


Hadoop, 2017.

Configuring Hadoop for Multi Node Cluster:

- **Create an 8 Node Instance**

Note: We have taken a special permission from Amazon to increase the instance limit, the proof for the above is in below screenshots



Amazon Web Services
Nov 28, 2017
02:42 AM -0600

Hello Ahmed,

I trust you are having a lovely day! This is Rodney from AWS billing and accounts.

I'm currently working on your i3.large Instances limit increase request for 9 in the US West (Oregon) region. In this particular case, I have to collaborate with my Service Team to get approval.

I'm going to hold on to your case and the second I get an update from my Service Team I will let you know. I understand this is a really important resource for you so I will do my best to expedite this request.

Thank you very much for your patience while we work on this.

In the meantime, if there is anything you need, please feel free to reply to this case.

Have a lovely day!

Best regards,

Rodney W
Amazon Web Services

Check out the AWS Support Knowledge Center, a knowledge base of articles and videos that answer customer questions about AWS services:
https://aws.amazon.com/premiumsupport/knowledge-center/?icmpid=support_email_category

We value your feedback. Please rate my response using the link below

Case Details

Basic Support Plan Change

Subject	Limit Increase: EC2 Instances		
Case ID	4680901651	**Status**	Pending Customer Action
Created	Nov 27, 2017 08:38 PM -0600	**Severity**	General guidance
Case type	Service Limits	**Category**	Service Limit Increase, EC2 Instances
By	asayed2@hawk.iit.edu	**CCd emails**	

Correspondence

Reply Close Case

Amazon Web Services
Nov 28, 2017
08:14 AM -0600

Hello,

Thank you for your patience while we reviewed your request.

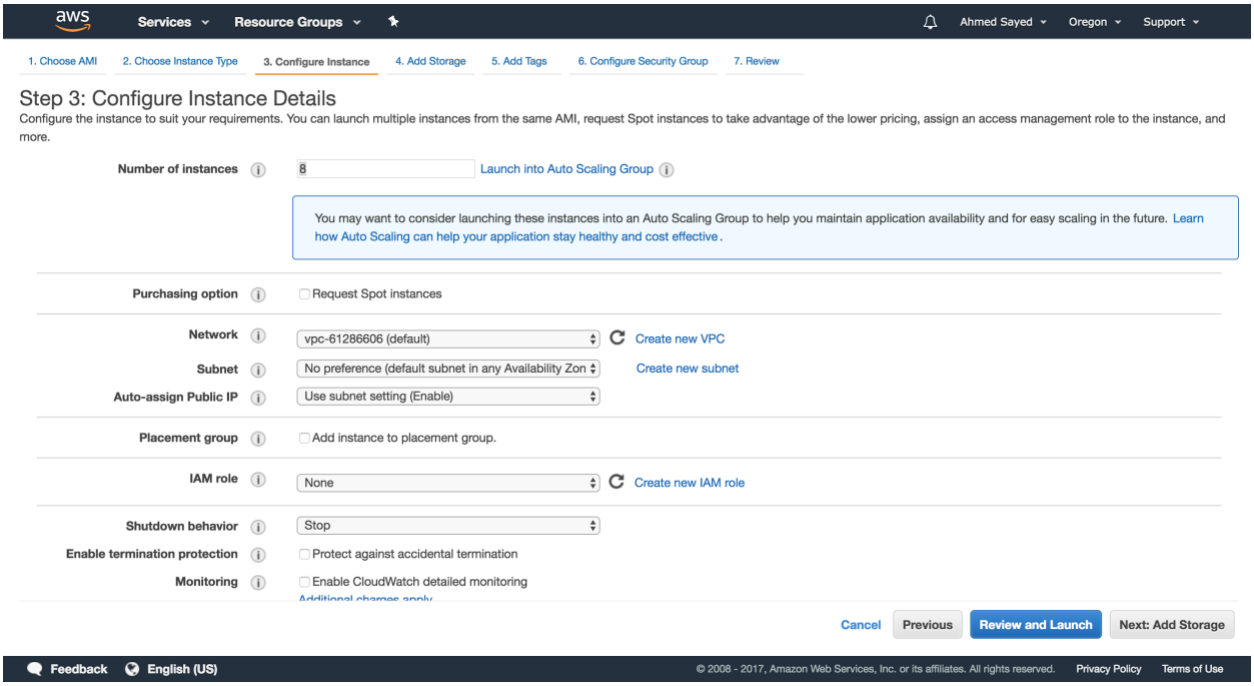
I'm happy to inform you that we've approved and processed your i3.large instance limit increase request for the US West (Oregon) region, and your new limit is 9. Please keep in mind that it can sometimes take up to 15 minutes for the new limit to propagate and become available for use.

I hope this helps, but please let us know if you require further assistance.

Have a wonderful day!

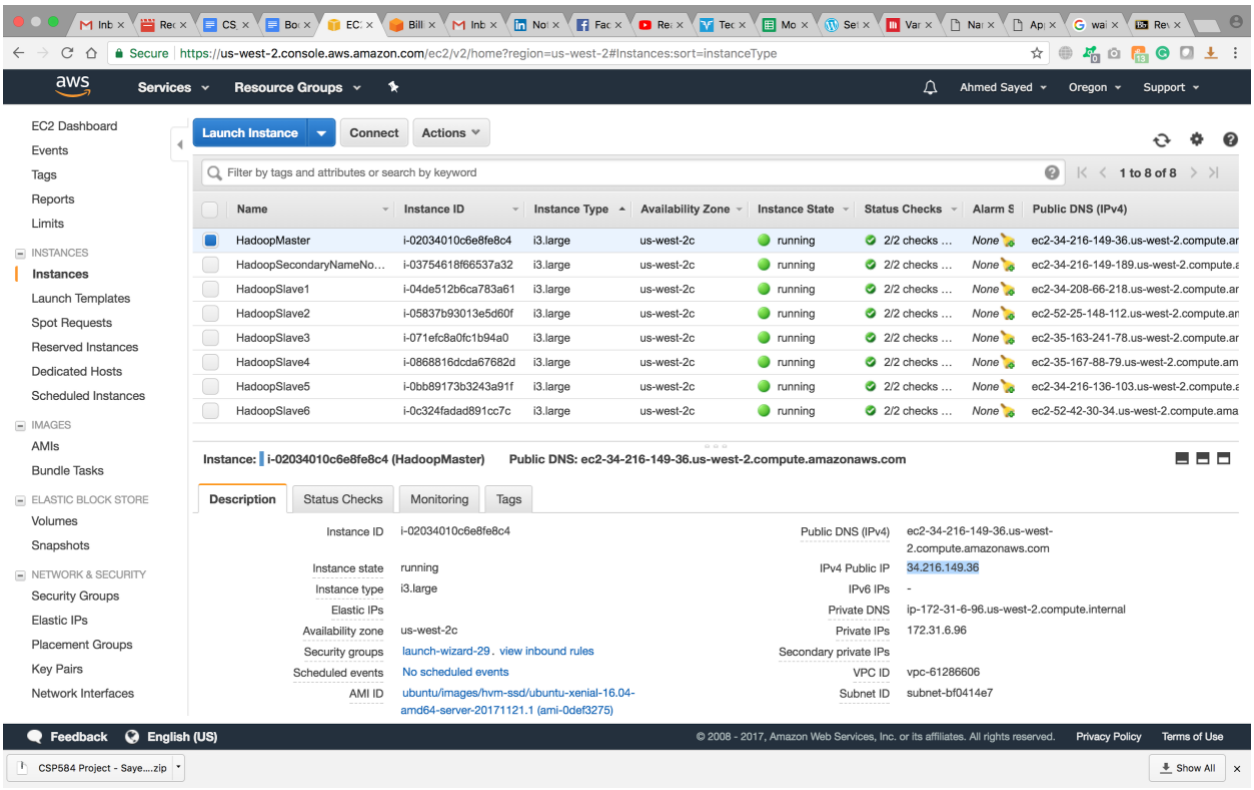
Best regards,

Rodney W
Amazon Web Services



//Add 8 in number of instances

- **Rename all the nodes, with one as a Master, one as a Secondary Node and remaining 6 as a Slaves**



- **Repeat the same process of installing Hadoop as we did above before Single Node Configuration**
- **Change all the hostname to there DNS for simplicity for example:**

  sudo hostname ec2-34-216-149-36.us-west-2.compute.amazonaws.com
- **Configure core-site.xml - add Master Address**

```
<configuration>
<property>
        <name>fs.default.name</name>
        <value>hdfs://ec2-34-215-219-28.us-west-2.compute.amazonaws.com:9000</value>
</property>
<property>
        <name>hadoop.tmp.dir</name>
         <value>/home/ubuntu/hdfstmp</value>
        <description>base location for other hdfs directories.</description>
</property>
</configuration>
```

- **Configure hdfs-site..xml - add Master Address**

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>                                        //No Replication
        </property>
        <property>
                <name>dfs.permissions</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.namenode.secondary.http-address</name>
               <value>ec2-54-187-222-213.us-west-2.compute.amazonaws.com:50090</value>
        </property>
</configuration>
```

- **Configure mapred-site..xml - add Master Address**

```
<configuration>
        <property>
        <name>mapreduce.job.tracker</name>
        <value>hdfs://ec2-34-215-219-28.us-west-2.compute.amazonaws.com:9001</value>
        </property>
        <property>
                 <name> mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
        <property>
```

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

- **Configure yarn-site.xml**

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value></property>
  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>ec2-34-215-219-28.us-west-2.compute.amazonaws.com:9070</value>
  </property>
 <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>ec2-34-215-219-28.us-west-2.compute.amazonaws.com:9074</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>ec2-34-215-219-28.us-west-2.compute.amazonaws.com:9076</value>
  </property>
```

```
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>ec2-34-215-219-28.us-west-2.compute.amazonaws.com:9078</value>
</property>
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>ec2-34-215-219-28.us-west-2.compute.amazonaws.com:9082</value>
</property>
<property>
<name>yarn.nodemanager.resource.memory-mb</name>
<value>3072</value>
</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>
</configuration>
```

- **Create a Master file - This will be in Master and Secondary Node /hadoop/etc/hadoop**

```
vi masters                                          //Name of the file masters
ec2-34-215-219-28.us-west-2.compute.amazonaws.com   //Master Node
ec2-34-216-117-68.us-west-2.compute.amazonaws.com   //SecondaryNamde Node
```

- **Create a Slave file - This will be in Master and Secondary Node /hadoop/etc/hadoop**

```
ec2-34-216-138-46.us-west-2.compute.amazonaws.com   //Slave 0
```

ec2-34-216-125-157.us-west-2.compute.amazonaws.com //Slave 1

ec2-35-164-171-31.us-west-2.compute.amazonaws.com //Slave 2

ec2-34-216-143-121.us-west-2.compute.amazonaws.com //Slave 3

ec2-52-41-29-80.us-west-2.compute.amazonaws.com //Slave 4

ec2-34-215-230-137.us-west-2.compute.amazonaws.com //Slave 5

- **Add Corresponding Slaves address in each slaves**

  eg. For Slave 0

  vii slaves

  ec2-34-216-138-46.us-west-2.compute.amazonaws.com

- **Add a cloud.key in all the instance and will ssh from Master to try connecting it**

  eval `ssh-agent -s`

  ssh-add AhmedShared.pem

  Eg. Try ssh with Slaves address

```
[ubuntu@ip-172-31-6-96:~$ ssh -i "AhmedShared.pem" ubuntu@ec2-34-216-149-189.us-west-2.compu
te.amazonaws.com
The authenticity of host 'ec2-34-216-149-189.us-west-2.compute.amazonaws.com (172.31.6.255)
' can't be established.
ECDSA key fingerprint is SHA256:bJRnQSxgNWEzIxSFRw7kYuNkyIv1SrALvO+VqUiWThE.
[Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-34-216-149-189.us-west-2.compute.amazonaws.com' (ECDSA) to
the list of known hosts.
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1041-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

5 packages can be updated.
0 updates are security updates.


Last login: Sun Dec  3 01:29:44 2017 from 207.237.204.157
```

  Now exit to Logout

```
[ubuntu@ip-172-31-6-255:~$ exit
logout
Connection to ec2-34-216-149-189.us-west-2.compute.amazonaws.com closed.
```

  Repeat the above step for all Slaves to see if Master is reachable to all slaves.

- **Format the HDFS in Masters**

hdfs namenode -format

- **Start all services in Masters, which in turn try to start all Slaves**

  Start-all.sh

- **Type JPS to check the nodes**

  For Master we can see

  2497 Jps

  2228 ResourceManager

  2071 SecondaryNameNode

  1851 NameNode

  For SecondarNode we can see

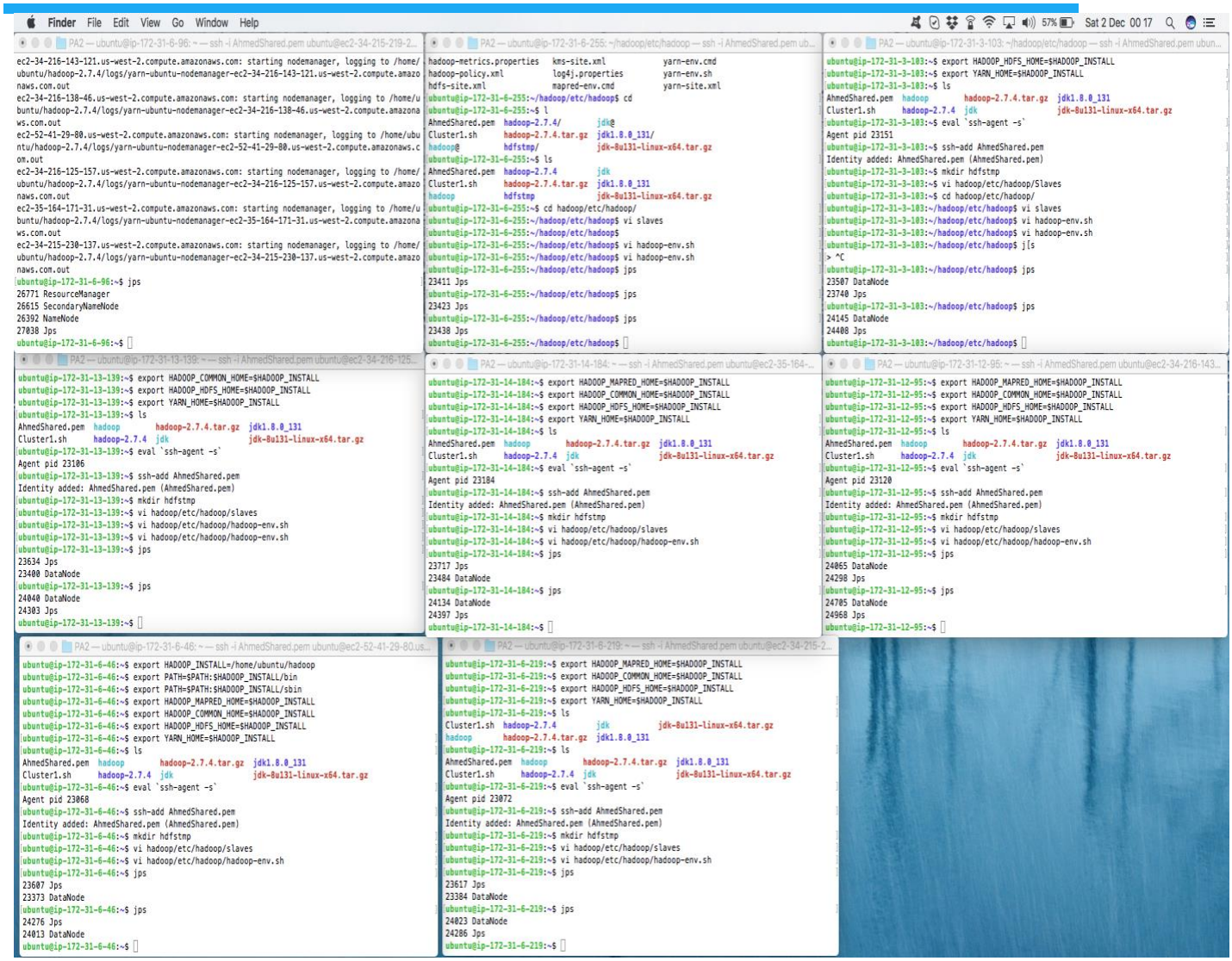
  2497 Jps

  2071 SecondaryNameNode

  For Slaves we can see

  1826 Jps

  1588 DataNode

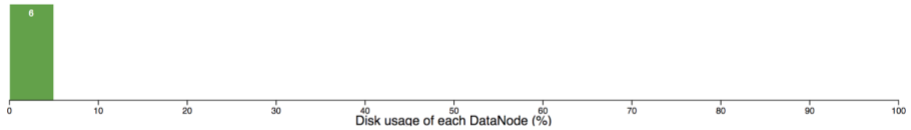


- **We can also check the status in browser**

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Datanode Information

Datanode usage histogram



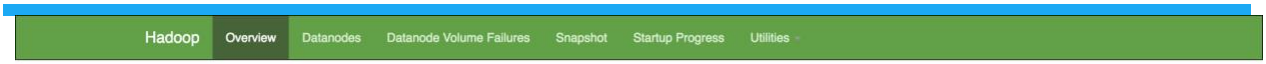
In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ec2-34-216-138-46.us-west-2.compute.amazonaws.com:50010 (172.31.3.103:50010)	1	In Service	290.76 GB	24 KB	2.78 GB	287.97 GB	0	24 KB (0%)	0	2.7.4
ec2-34-216-143-121.us-west-2.compute.amazonaws.com:50010 (172.31.12.95:50010)	1	In Service	290.76 GB	24 KB	2.78 GB	287.97 GB	0	24 KB (0%)	0	2.7.4
ec2-34-216-117-68.us-west-2.compute.amazonaws.com:50010 (172.31.6.255:50010)	1	In Service	290.76 GB	24 KB	2.78 GB	287.97 GB	0	24 KB (0%)	0	2.7.4
ec2-34-216-125-157.us-west-2.compute.amazonaws.com:50010 (172.31.14.184:50010)	1	In Service	290.76 GB	24 KB	2.78 GB	287.97 GB	0	24 KB (0%)	0	2.7.4
ec2-52-41-29-80.us-west-2.compute.amazonaws.com:50010 (172.31.6.46:50010)	1	In Service	290.76 GB	24 KB	2.78 GB	287.97 GB	0	24 KB (0%)	0	2.7.4
ec2-34-215-230-137.us-west-2.compute.amazonaws.com:50010 (172.31.6.219:50010)	1	In Service	290.76 GB	24 KB	2.78 GB	287.97 GB	0	24 KB (0%)	0	2.7.4

Decomissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction

Hadoop, 2017.



Overview 'ec2-34-215-219-28.us-west-2.compute.amazonaws.com:8020' (active)

Started:	Sat Dec 02 06:16:35 UTC 2017
Version:	2.7.4, rcd915e1e8d9d0131462a0b7301586c175728a282
Compiled:	2017-08-01T00:29Z by kshvachk from branch-2.7.4
Cluster ID:	CID-e605589a-f44d-4f34-bec8-c3e808d43ece
Block Pool ID:	BP-854754809-172.31.6.96-1512194588408

Summary

Security is off.

Safemode is off.

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

Heap Memory used 80.32 MB of 217.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 40.1 MB of 40.88 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	1.7 TB
DFS Used:	144 KB (0%)
Non DFS Used:	16.67 GB
DFS Remaining:	1.69 TB (99.04%)
Block Pool Used:	144 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	6 (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	02/12/2017, 00:16:35

NameNode Journal Status

Current transaction ID: 6

Journal Manager	State
FileJournalManager(root=/home/ubuntu/hdfstmp/dfs/name)	EditLogFileOutputStream(/home/ubuntu/hdfstmp/dfs/name/current/edits_inprogress_0000000000000000006)

NameNode Storage

Storage Directory	Type	State
/home/ubuntu/hdfstmp/dfs/name	IMAGE_AND_EDITS	Active

Hadoop, 2017.

- **To start the job we can start at master**

```
[ubuntu@ip-172-31-6-96:~$ hadoop jar hadoop-2.7.4/share/hadoop/mapreduce/hadoop-mapreduce-ex]
amples-2.7.4.jar teragen 85800345 /teraInput8GB
17/12/03 01:41:48 INFO client.RMProxy: Connecting to ResourceManager at ec2-34-216-149-36.u
s-west-2.compute.amazonaws.com/172.31.6.96:9076
17/12/03 01:41:49 INFO terasort.TeraSort: Generating 85800345 using 2
17/12/03 01:41:50 INFO mapreduce.JobSubmitter: number of splits:2
17/12/03 01:41:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1512265063012
_0001
17/12/03 01:41:50 INFO impl.YarnClientImpl: Submitted application application_1512265063012
_0001
17/12/03 01:41:51 INFO mapreduce.Job: The url to track the job: http://ec2-34-216-149-36.us
-west-2.compute.amazonaws.com:9078/proxy/application_1512265063012_0001/
17/12/03 01:41:51 INFO mapreduce.Job: Running job: job_1512265063012_0001
```

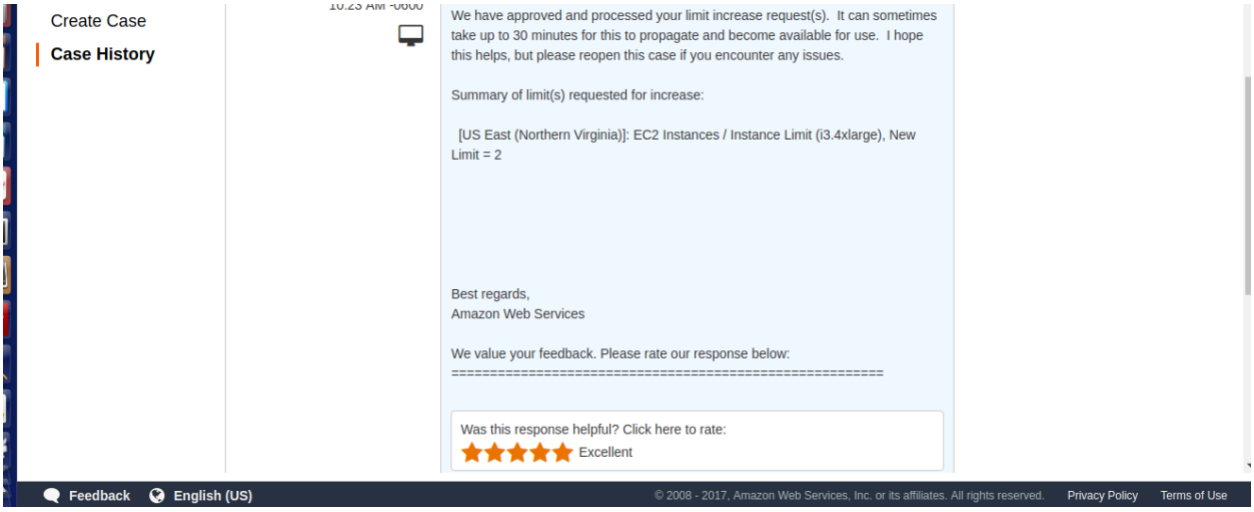
```
PA2 — ubuntu@ec2-34-215-219-28: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-215-2...
s-west-2.compute.amazonaws.com/172.31.6.96:9076
17/12/02 07:36:25 INFO impl.YarnClientImpl: Killed application application_1512196130380_00
01
Killed job job_1512196130380_0001
[ubuntu@ip-172-31-6-96:~$ hadoop job -list                                              ]
DEPRECATED: Use of this script to execute mapred command is deprecated.
Instead use the mapred command for it.

17/12/02 07:36:30 INFO client.RMProxy: Connecting to ResourceManager at ec2-34-215-219-28.u
s-west-2.compute.amazonaws.com/172.31.6.96:9076
Total jobs:0
                  JobId      State            StartTime      UserName           Queue      P
riority  UsedContainers  RsvdContainers  UsedMem         RsvdMem         NeededMem
AM info
```

- We can also track with URL as stated in the above snapshot

Spark :

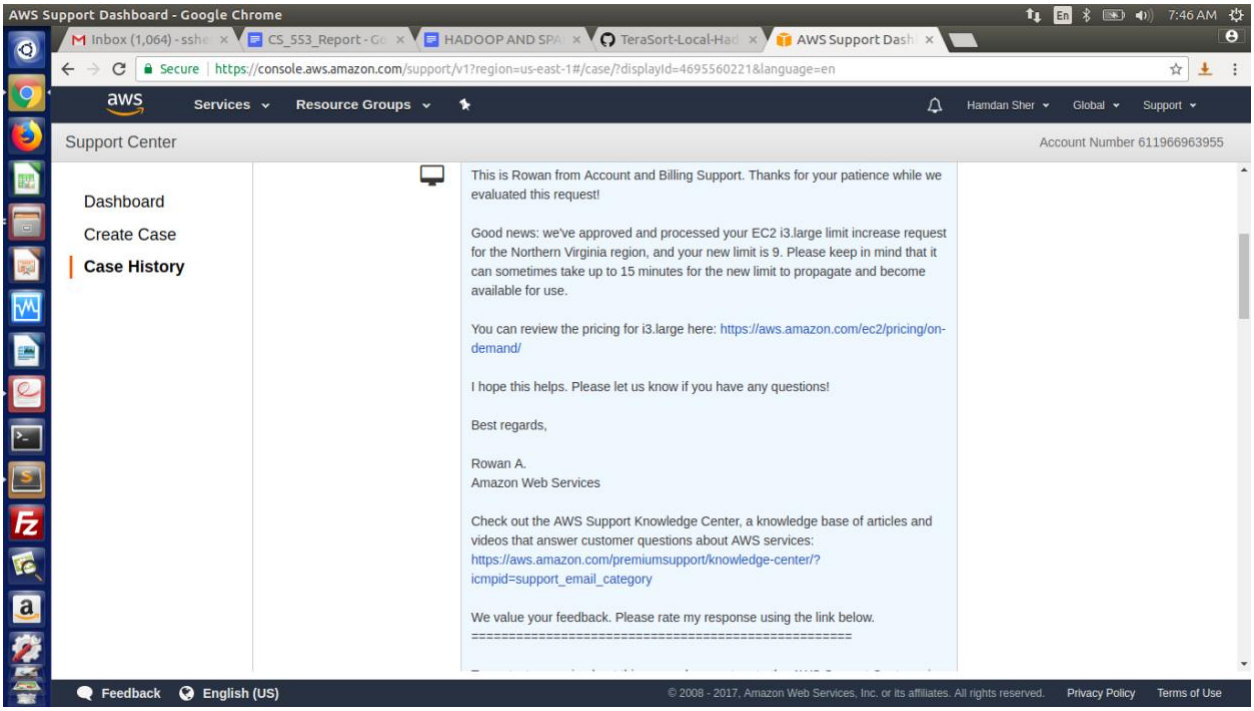
Instances set-up for single node Spark implementation.

AWS REQUEST and i3.large, i3.4xlarge limit increase. We requested Amazon to increase the limit for the instance required. Below are the screen-shots of AWS responses, it takes at the max of 24 hours for AWS to process your request.



I3.4xlarge

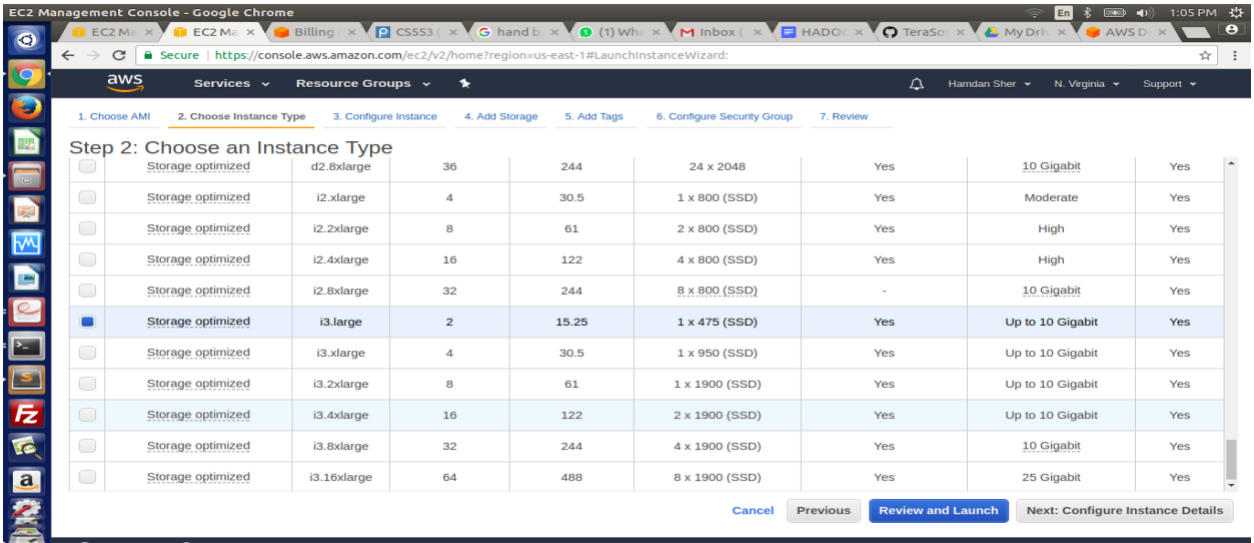
and

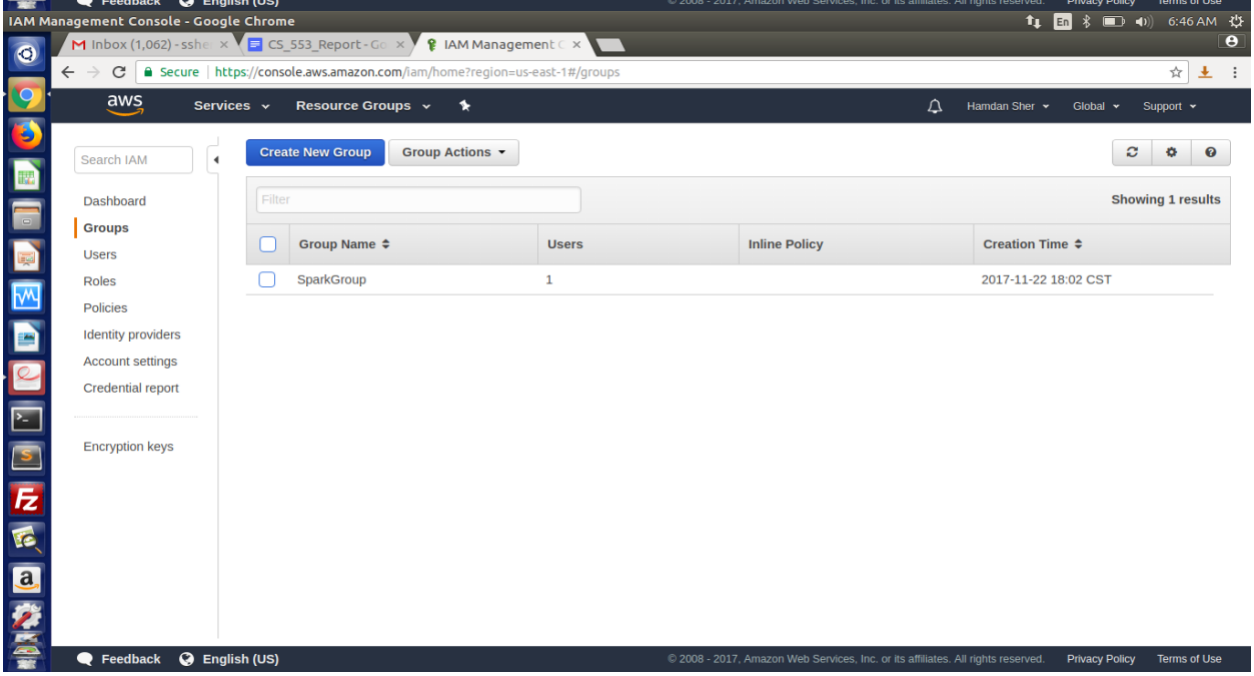


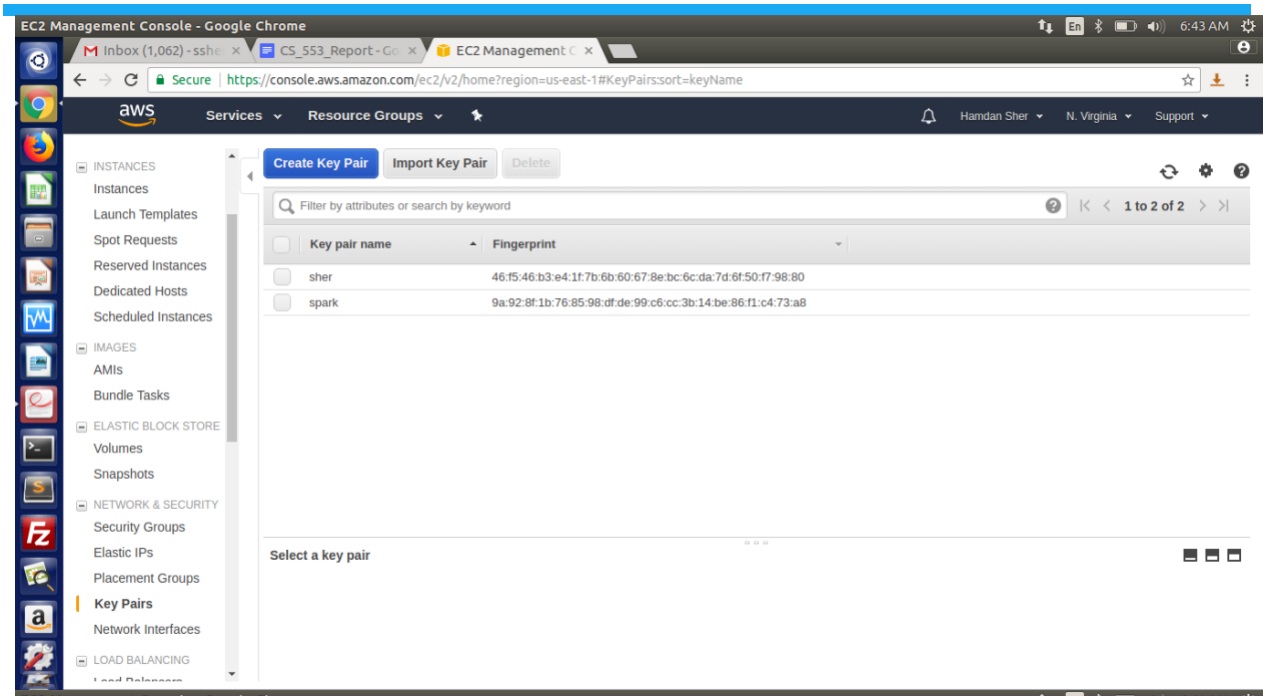
I3.large for cluster generation of 8 nodes

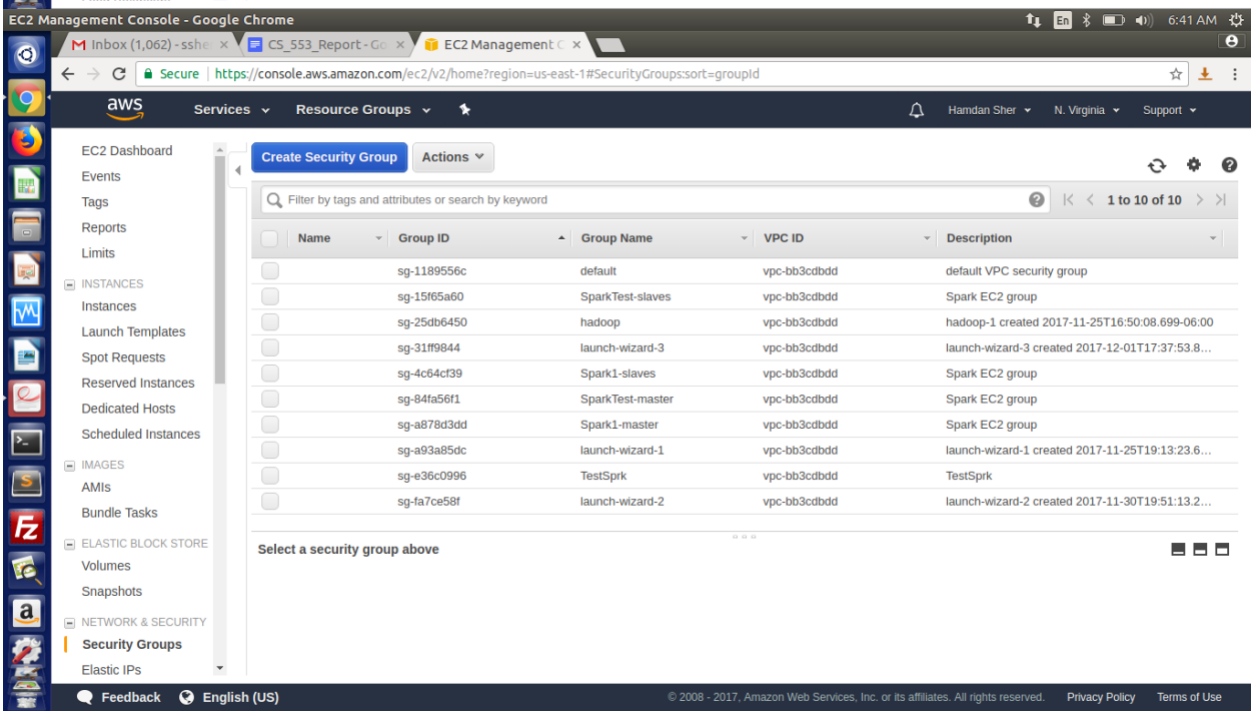
Select the instance type and launch the instance with high EBS storage for both i3.large and i3.4xlarge configuration. Above pictures shows the creation of the instances, security group and key pair required for the assignment.

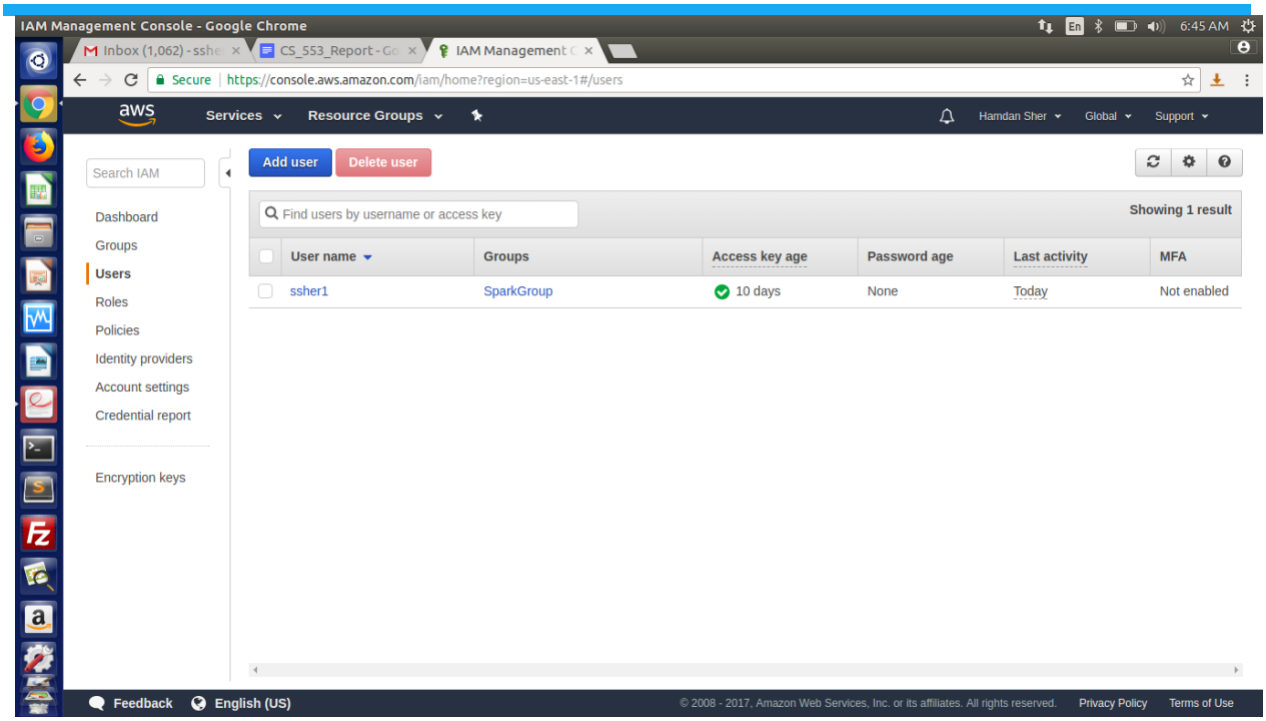
Single node i3.large and i3.4xlarge:



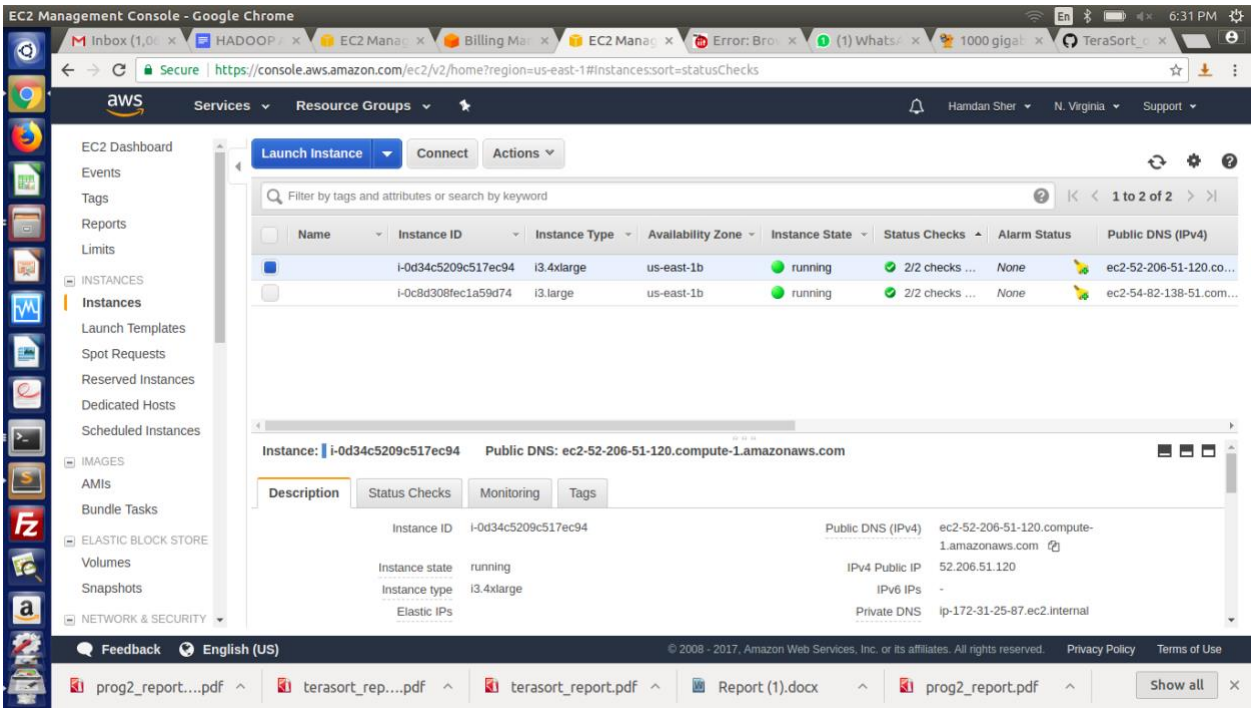



EC2 Management Console - Google Chrome
6:43 AM
Inbox (1,062) - sshe
CS_553_Report - Go
EC2 Management C
Secure
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#KeyPairs:sort=keyName
aws
Services
Resource Groups
Hamdan Sher
N. Virginia
Support
INSTANCES
Instances
Launch Templates
Spot Requests
Reserved Instances
Dedicated Hosts
Scheduled Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
LOAD BALANCING
Create Key Pair
Import Key Pair
Delete
Filter by attributes or search by keyword
1 to 2 of 2
Key pair name
Fingerprint
sher
46:f5:46:b3:e4:1f:7b:6b:60:67:8e:bc:6c:da:7d:6f:50:f7:98:80
spark
9a:92:8f:1b:76:85:98:df:de:99:c6:cc:3b:14:be:86:f1:c4:73:a8
Select a key pair


EC2 Management Console - Google Chrome
6:41 AM
Inbox (1,062) - sshe
CS_553_Report - Go
EC2 Management C
Secure
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#SecurityGroups:sort=groupId
aws
Services
Resource Groups
Hamdan Sher
N. Virginia
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Spot Requests
Reserved Instances
Dedicated Hosts
Scheduled Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Create Security Group
Actions
Filter by tags and attributes or search by keyword
1 to 10 of 10
Name
Group ID
Group Name
VPC ID
Description
sg-1189556c default vpc-bb3cdbdd default VPC security group
sg-15f65a60 SparkTest-slaves vpc-bb3cdbdd Spark EC2 group
sg-25db6450 hadoop vpc-bb3cdbdd hadoop-1 created 2017-11-25T16:50:08.699-06:00
sg-31ff9844 launch-wizard-3 vpc-bb3cdbdd launch-wizard-3 created 2017-12-01T17:37:53.8…
sg-4c64cf39 Spark1-slaves vpc-bb3cdbdd Spark EC2 group
sg-84fa56f1 SparkTest-master vpc-bb3cdbdd Spark EC2 group
sg-a878d3dd Spark1-master vpc-bb3cdbdd Spark EC2 group
sg-a93a85dc launch-wizard-1 vpc-bb3cdbdd launch-wizard-1 created 2017-11-25T19:13:23.6…
sg-e36c0996 TestSprk vpc-bb3cdbdd TestSprk
sg-fa7ce58f launch-wizard-2 vpc-bb3cdbdd launch-wizard-2 created 2017-11-30T19:51:13.2…
Select a security group above
Feedback
English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use



Single node instance running with large EBS storage for generating 1 TB and 128 GB gensort data. [Instances have limit of 2 TB storage (2047 GB exact) for additional storage create new storage option I selected EBS storage.]



SSH in to the instance and install the requirements. You may manually do it or create a script for it. Below is the installation processes :-

```
clear

echo -e "*** Cloud setup ***\n"


echo -e "Downloading setup files...\n"


echo -e "1. JAVA Jdk\n"

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.tar.gz

echo -e "Done !\n"


echo -e "2. Hadoop\n"

wget http://mirror.reverse.net/pub/apache/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz

echo -e "Done !\n"


echo -e "3. Spark\n"

wget http://d3kbcqa49mib13.cloudfront.net/spark-1.6.0-bin-hadoop2.6.tgz

echo -e "Done !\n"


echo -e "4. Scala\n"

wget http://downloads.lightbend.com/scala/2.11.8/scala-2.11.8.tgz

echo -e "Done !\n"


echo -e "5. Gensort\n"
```

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

echo -e "Done !\n"


echo -e "Downloads complete...\n"


echo -e "Installing Java... \n"

sudo tar -xvzf jdk-8u73-linux-x64.tar.gz

ln -s jdk1.8.0_73 jdk


echo -e "Installing Hadoop... \n"

sudo tar -xvzf hadoop-2.7.2.tar.gz

ln -s hadoop-2.7.2 hadoop


echo -e "Insatlling Spark... \n"

sudo tar -xvzf spark-1.6.0-bin-hadoop2.6.tgz

ln -s spark-1.6.0-bin-hadoop2.6 spark


echo -e "Insatlling Scala... \n"

sudo tar -xvzf scala-2.11.8.tgz

ln -s scala-2.11.8 scala


echo -e "Extracting Gensort... \n"

sudo tar -xvzf gensort-linux-1.5.tar.gz
```

```
mkdir Gensort

mv ~/64 ~/32 ~/

sudo chown -R ubuntu ~/*

sudo chgrp -R ubuntu ~/*



bashName="~/.bashrc"

sudo chmod 777 "$bashName"


echo 'export PATH=~/hadoop/bin:~/hadoop/sbin:~/jdk/bin:~/scala/bin:~/spark/bin:$PATH' >> ~/.bashrc

echo 'export HADOOP_HOME=~/hadoop' >> ~/.bashrc

echo 'export JAVA_HOME=~/jdk' >> ~/.bashrc

echo 'export SCALA_HOME=~/scala' >> ~/.bashrc

echo 'export SPARK_HOME=~/spark' >> ~/.bashrc

echo


source ~/.bashrc


java -version

echo

hadoop version

echo

scala -version
```

echo

If you can see the version number of java hadoop and spark and scala then you have successfully install the required settings.

Login to i3.large/i3.4xlarge and go to /root/ubuntu

Generate file from gensort 128 GB and 1 TB one at a time [check the storage limit you have on your instance]

--> /root/ephemeral-hdfs/bin/hadoop fs -mkdir /user

--> /root/ephemeral-hdfs/bin/hadoop fs -mkdir /user/root

--> /root/ephemeral-hdfs/bin/hadoop fs -put input.txt input.txt

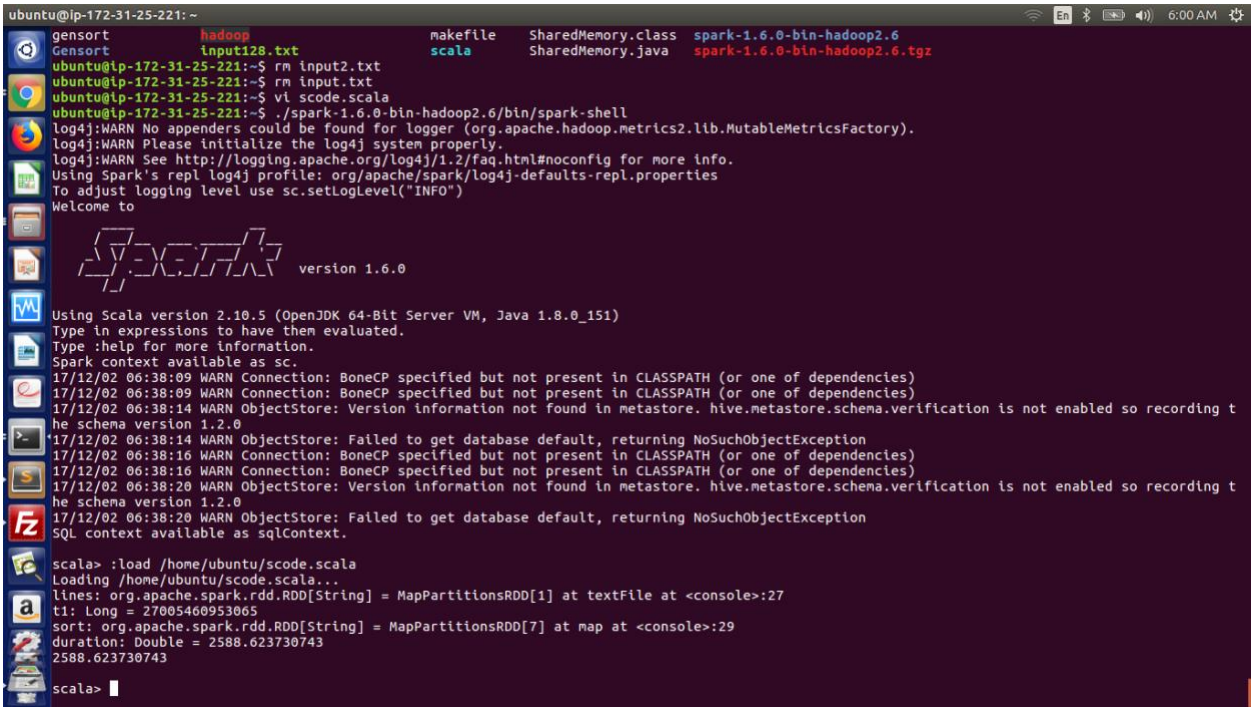
Or store it in your EBS storage [only possible in single node]

Go to the location where your spark-shell command is and launch spark-shell

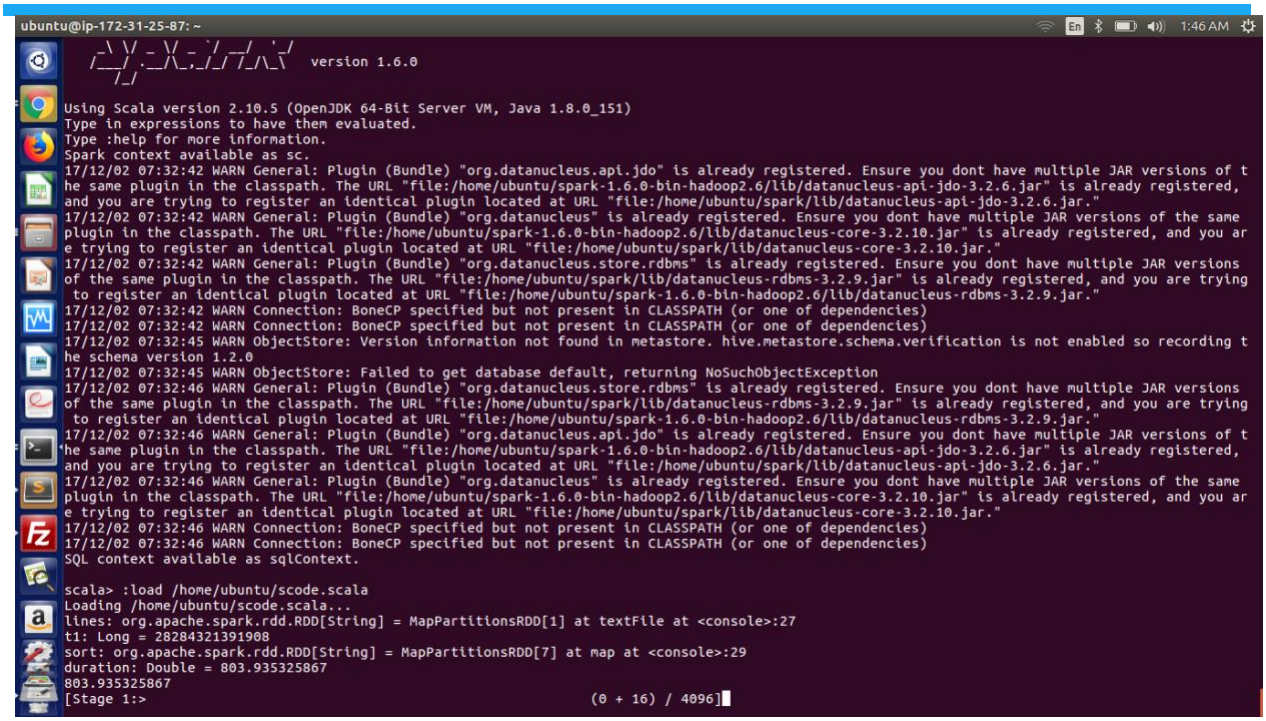
[Note: make sure you have set up hadoop as described above in the Hadoop set up section section.

After launching spark to run the sorting code written in Scala do :

scala> :load scode.scala



Time in seconds 2588.6 (splitting,sorting and splitting) to run 128 GB file (i3.large)



Time in seconds 803.9 (splitting,sorting and splitting) to run 128 GB file (13.4x large)

Time in seconds 6282.47 sec (splitting,sorting and splitting) to run 1 TB file (13.4x large)

Evaluation for Single node is discussed in the Performance evaluation section of this report.

[Note: (Only for Single node this is done)To make the spark implementation faster I have eliminated the hadoop part in it and transfer the 128 GB and 1 TB data directly from the large storage I generated. But it can be done using hadoop dfs as well]

MULTIPLE NODE CLUSTER i3.large

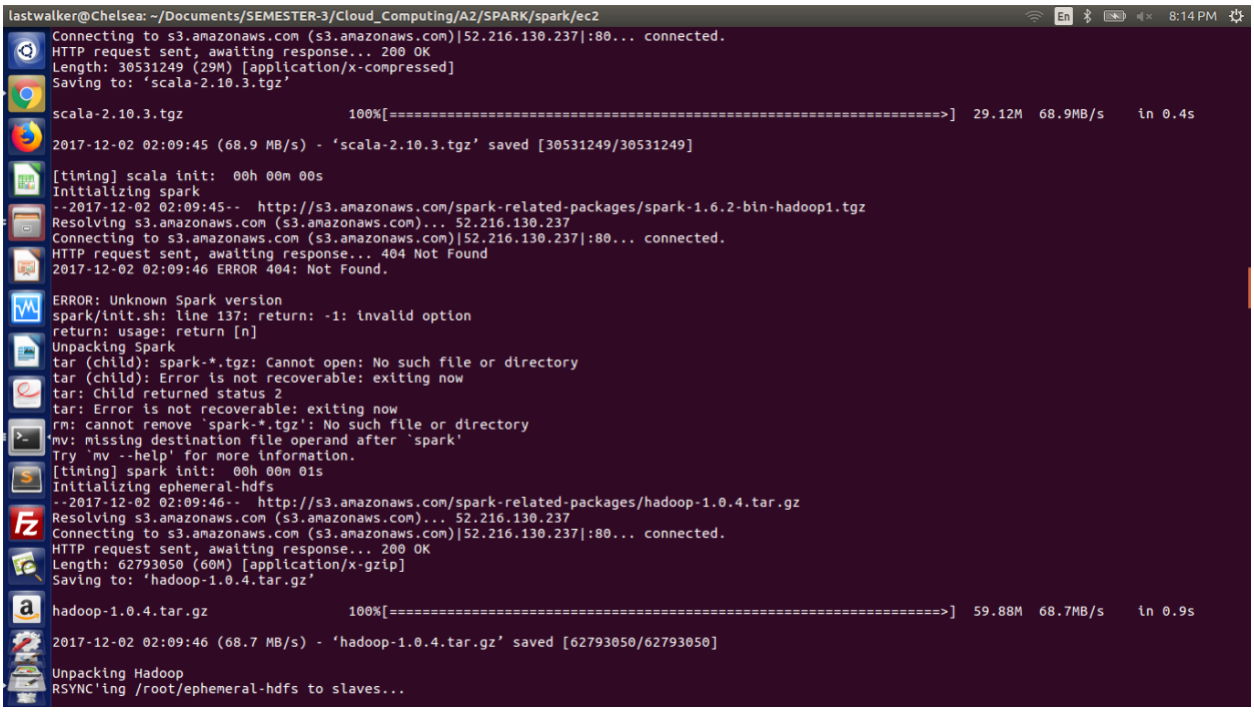
To run Spark program for 1TB GB on i3.large 8 node cluster we have to do following steps :
Request AWS to increase your limit to the something you want for a specific instance you will run your cluster with.
Download Spark-ec2 from [3] link provided in the reference section.

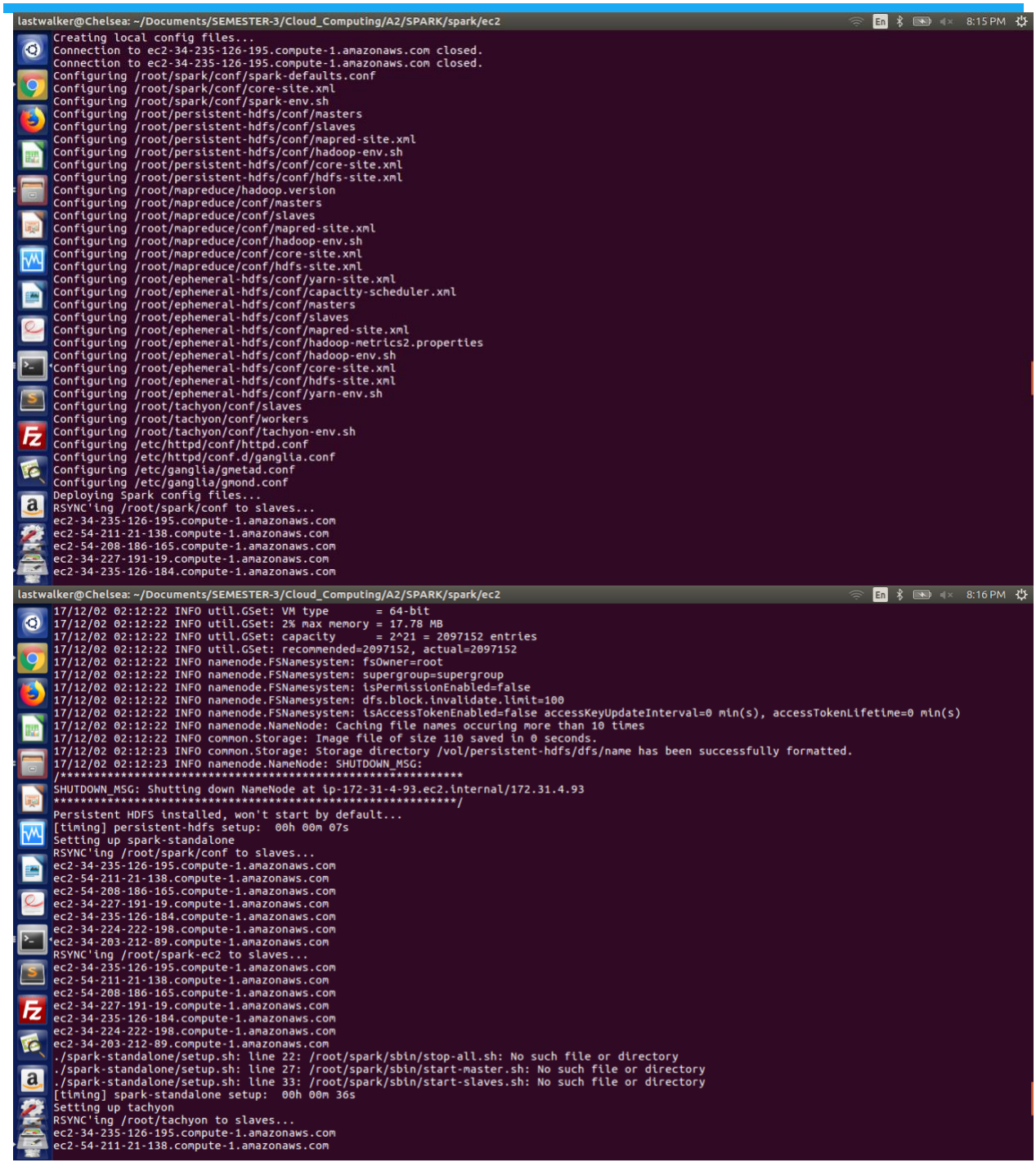
Locally export AWS_SECRET_ACCESS_KEY and export AWS_ACCESS_KEY_ID

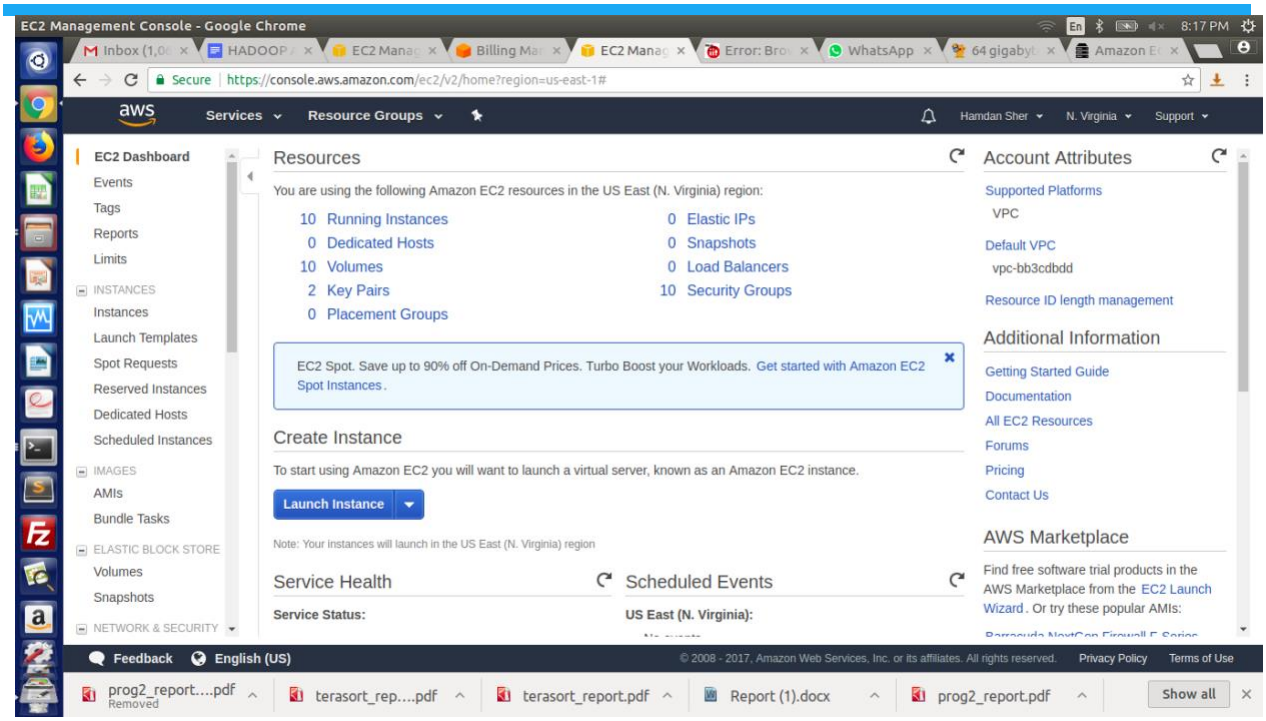
go to folder spark/ec2/ and below command to start Spark cluster

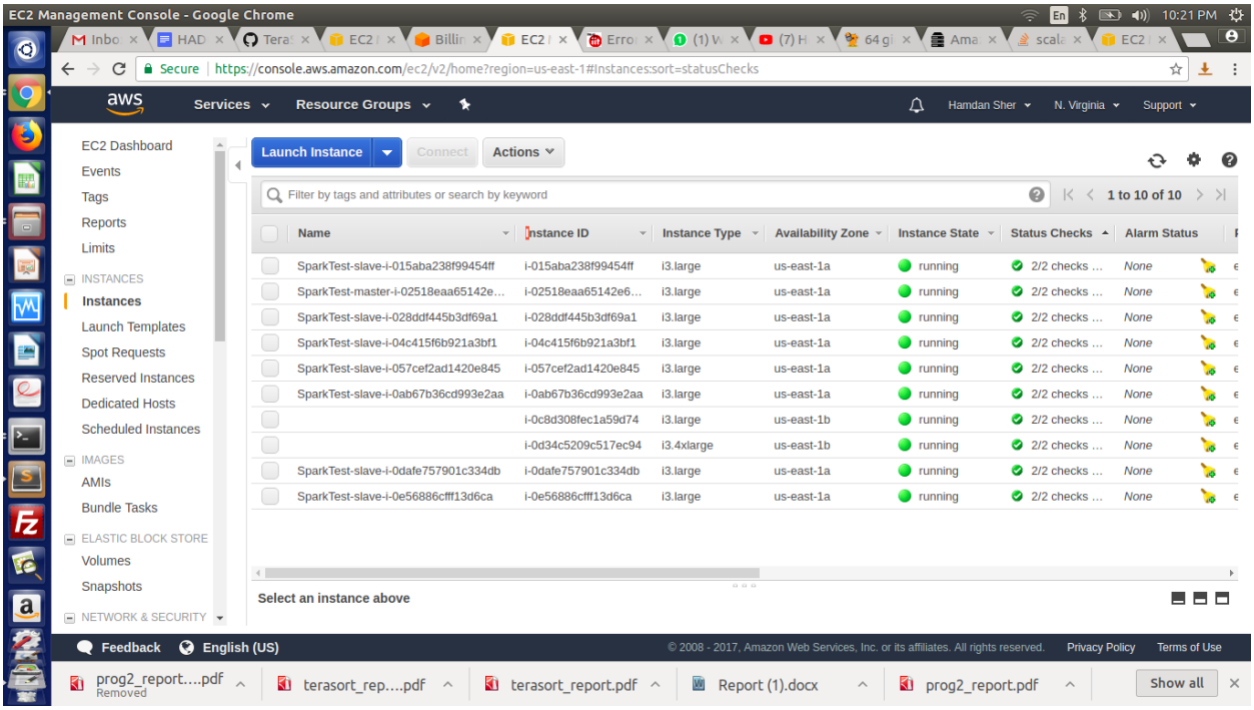
./spark-ec2 -k <awskey> -i <awskey.pem> -t <intancetype> -s <numberofslaves> [--spot-price=0.40 optional] launch Spark
I didn't use spot-price here is how I created 8 node i3.large cluster
./spark-ec2 -k spark -i spark.pem -t i3.large -s 7 launch SparkTest

It will take some time to setup cluster , here are few screen shots on it:




lastwalker@Chelsea: ~/Documents/SEMESTER-3/Cloud_Computing/A2/SPARK/spark/ec2
8:15 PM
Creating local config files...
Connection to ec2-34-235-126-195.compute-1.amazonaws.com closed.
Connection to ec2-34-235-126-195.compute-1.amazonaws.com closed.
Configuring /root/spark/conf/spark-defaults.conf
Configuring /root/spark/conf/core-site.xml
Configuring /root/spark/conf/spark-env.sh
Configuring /root/persistent-hdfs/conf/masters
Configuring /root/persistent-hdfs/conf/slaves
Configuring /root/persistent-hdfs/conf/mapred-site.xml
Configuring /root/persistent-hdfs/conf/hadoop-env.sh
Configuring /root/persistent-hdfs/conf/core-site.xml
Configuring /root/persistent-hdfs/conf/hdfs-site.xml
Configuring /root/mapreduce/hadoop.version
Configuring /root/mapreduce/conf/masters
Configuring /root/mapreduce/conf/slaves
Configuring /root/mapreduce/conf/mapred-site.xml
Configuring /root/mapreduce/conf/hadoop-env.sh
Configuring /root/mapreduce/conf/core-site.xml
Configuring /root/mapreduce/conf/hdfs-site.xml
Configuring /root/ephemeral-hdfs/conf/yarn-site.xml
Configuring /root/ephemeral-hdfs/conf/capacity-scheduler.xml
Configuring /root/ephemeral-hdfs/conf/masters
Configuring /root/ephemeral-hdfs/conf/slaves
Configuring /root/ephemeral-hdfs/conf/mapred-site.xml
Configuring /root/ephemeral-hdfs/conf/hadoop-metrics2.properties
Configuring /root/ephemeral-hdfs/conf/hadoop-env.sh
Configuring /root/ephemeral-hdfs/conf/core-site.xml
Configuring /root/ephemeral-hdfs/conf/hdfs-site.xml
Configuring /root/ephemeral-hdfs/conf/yarn-env.sh
Configuring /root/tachyon/conf/slaves
Configuring /root/tachyon/conf/workers
Configuring /root/tachyon/conf/tachyon-env.sh
Configuring /etc/httpd/conf/httpd.conf
Configuring /etc/httpd/conf.d/ganglia.conf
Configuring /etc/ganglia/gmetad.conf
Configuring /etc/ganglia/gmond.conf
Deploying Spark config files...
RSYNC'ing /root/spark/conf to slaves...
ec2-34-235-126-195.compute-1.amazonaws.com
ec2-54-211-21-138.compute-1.amazonaws.com
ec2-54-208-186-165.compute-1.amazonaws.com
ec2-34-227-191-19.compute-1.amazonaws.com
ec2-34-235-126-184.compute-1.amazonaws.com
lastwalker@Chelsea: ~/Documents/SEMESTER-3/Cloud_Computing/A2/SPARK/spark/ec2
8:16 PM
17/12/02 02:12:22 INFO util.GSet: VM type = 64-bit
17/12/02 02:12:22 INFO util.GSet: 2% max memory = 17.78 MB
17/12/02 02:12:22 INFO util.GSet: capacity = 2^21 = 2097152 entries
17/12/02 02:12:22 INFO util.GSet: recommended=2097152, actual=2097152
17/12/02 02:12:22 INFO namenode.FSNamesystem: fsOwner=root
17/12/02 02:12:22 INFO namenode.FSNamesystem: supergroup=supergroup
17/12/02 02:12:22 INFO namenode.FSNamesystem: isPermissionEnabled=false
17/12/02 02:12:22 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
17/12/02 02:12:22 INFO namenode.FSNamesystem: isAccessTokenEnabled=false accessKeyUpdateInterval=0 min(s), accessTokenLifetime=0 min(s)
17/12/02 02:12:22 INFO namenode.NameNode: Caching file names occuring more than 10 times
17/12/02 02:12:22 INFO common.Storage: Image file of size 110 saved in 0 seconds.
17/12/02 02:12:23 INFO common.Storage: Storage directory /vol/persistent-hdfs/dfs/name has been successfully formatted.
17/12/02 02:12:23 INFO namenode.NameNode: SHUTDOWN_MSG:
/**
SHUTDOWN_MSG: Shutting down NameNode at ip-172-31-4-93.ec2.internal/172.31.4.93
**/
Persistent HDFS installed, won't start by default...
[timing] persistent-hdfs setup: 00h 00m 07s
Setting up spark-standalone
RSYNC'ing /root/spark/conf to slaves...
ec2-34-235-126-195.compute-1.amazonaws.com
ec2-54-211-21-138.compute-1.amazonaws.com
ec2-54-208-186-165.compute-1.amazonaws.com
ec2-34-227-191-19.compute-1.amazonaws.com
ec2-34-235-126-184.compute-1.amazonaws.com
ec2-34-224-222-198.compute-1.amazonaws.com
ec2-34-203-212-89.compute-1.amazonaws.com
RSYNC'ing /root/spark-ec2 to slaves...
ec2-34-235-126-195.compute-1.amazonaws.com
ec2-54-211-21-138.compute-1.amazonaws.com
ec2-54-208-186-165.compute-1.amazonaws.com
ec2-34-227-191-19.compute-1.amazonaws.com
ec2-34-235-126-184.compute-1.amazonaws.com
ec2-34-224-222-198.compute-1.amazonaws.com
ec2-34-203-212-89.compute-1.amazonaws.com
./spark-standalone/setup.sh: line 22: /root/spark/sbin/stop-all.sh: No such file or directory
./spark-standalone/setup.sh: line 27: /root/spark/sbin/start-master.sh: No such file or directory
./spark-standalone/setup.sh: line 33: /root/spark/sbin/start-slaves.sh: No such file or directory
[timing] spark-standalone setup: 00h 00m 36s
Setting up tachyon
RSYNC'ing /root/tachyon to slaves...
ec2-34-235-126-195.compute-1.amazonaws.com
ec2-54-211-21-138.compute-1.amazonaws.com





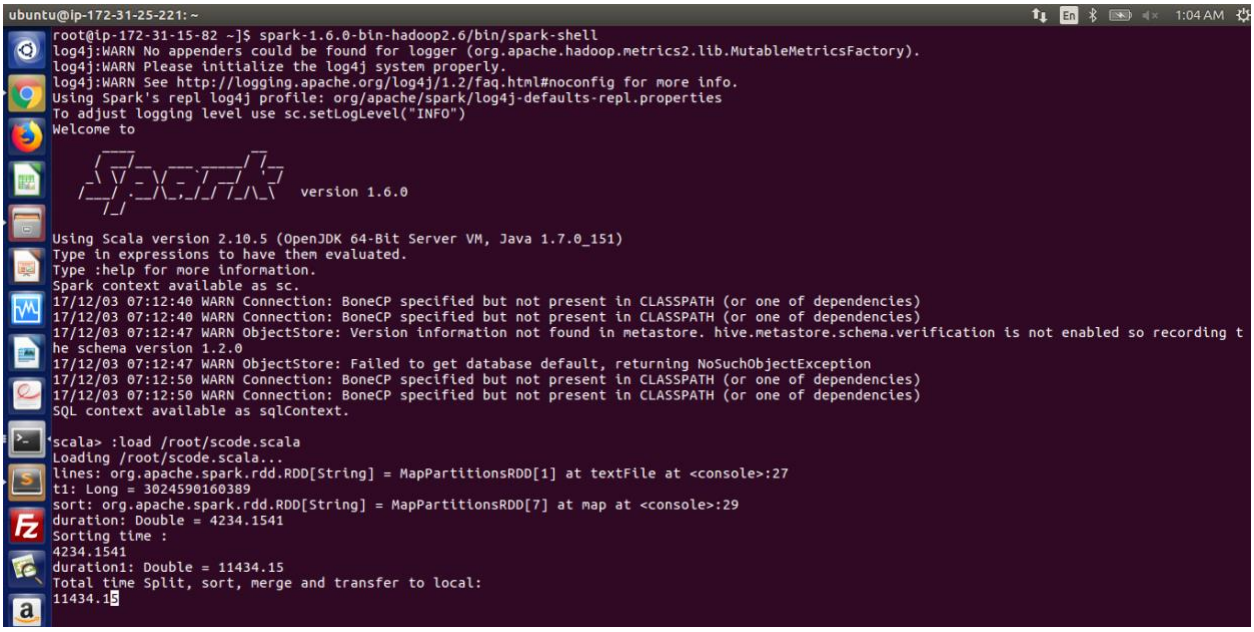
[Node: Please stop your instance when not using if you want to save money]

Login to master and go to root
Generate file from gensort and put it to hdfs using command

--> /root/ephemeral-hdfs/bin/hadoop fs -mkdir /user
--> /root/ephemeral-hdfs/bin/hadoop fs -mkdir /user/root

--> /root/ephemeral-hdfs/bin/hadoop fs -put input.txt input.txt

Go to the location where your spark-shell command is and launch spark-shell
[Note: make sure you have set up hadoop as described above in the Hadoop set up section section.
After launching spark to run the sorting code written in Scala do :
scala> :load scode.scala
////



13.large cluster master sorting time = 4234.15 sec

[5]Launch spark cluster with 8 and perform below tasks to configure spark for 1 TB and raid0 disk

go to /root/spark/sbin and stop spark
--> ./stop-all.sh

go to /root/ephemeral-hdfs/bin and stop hdfs
--> ./stop-dfs.sh

Download repo and change raid script
copy raid script to /root/ephemeral-hdfs/conf
so that it sync to all slaves

run raid script for master

go to /root/spark/conf

edit spark-env.sh
export SPARK_LOCAL_DIRS="/mnt/raid/spark"

and edit spark-defaults.conf and add below parameter
spark.local.dir /mnt/raid/tmp

edit core-site.xml

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/mnt/raid/ephemeral-hdfs</value>
</property>
```

go to /root/ephemeral-hdfs/conf

edit hdfs-site.xml

change replication to 1

```
<property>
  <name>dfs.data.dir</name>
  <value>/mnt/raid/ephemeral-hdfs/data</value>
</property>
```

edit core-site.xml

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/mnt/raid/ephemeral-hdfs</value>
</property>
```

go to /root/spark-ec2

run below command to RSYNC the above parameter copy to all slave

./copy-dir /root/ephemeral-hdfs/conf

./copy-dir /root/spark/conf

Now go to each slave and run same raid script which was edited in previous script.

go to /root/ephemeral-hdfs/bin

--> ./hadoop namenode -format

Come to master and run below command to start hdfs and spark cluster

and run script

/root/ephemeral-hdfs/bin
--> ./start-dfs.sh

/root/spark/sbin
--> start all

check with jps command you can see below item running

Master

SecondaryNameNode

TachyonMaster

Jps

NameNode

Generate 1 TB data and put it to hdfs

run ./spark-shell and run below command to sort 1 TB data

type ":paste"
and copy paste below code
and press Ctrl+D to start the execution

Val lines = sc.textFile("hdfs://ec2-52-36-141-162.us-west-2.compute.amazonaws.com:9000/user/root/input1TB.txt")

```
val sort = lines.map(_.split("  ")).map(arr => (arr(0) + "  " + arr(1),arr.mkString("  "))).sortByKey().map(_._2)
sort.saveAsTextFile("hdfs://ec2-52-36-141-162.us-west-2.compute.amazonaws.com:9000/user/root/output")
```

Performance Evaluation is given in the performance evaluation section

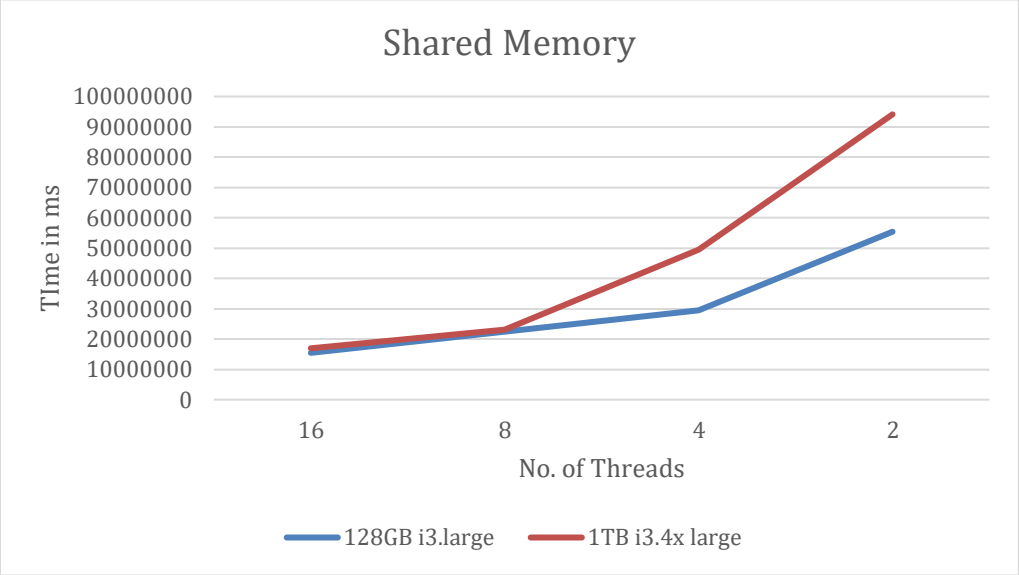
What the Code does:

file is taken from hdfs using sc.textFile than it is mapped and differentiated using double space and again map is used for other other cases well key contains double space. After that mkstring is used to get final key and value. Then used sort by key function to sort the keys used a custom mapping function for saveAsTextFile as output by sortByKey will have in tuples-to convert into string use saveAsTextFile to write it to hdfs which in turn will use above map function to write.

Performance Evaluation

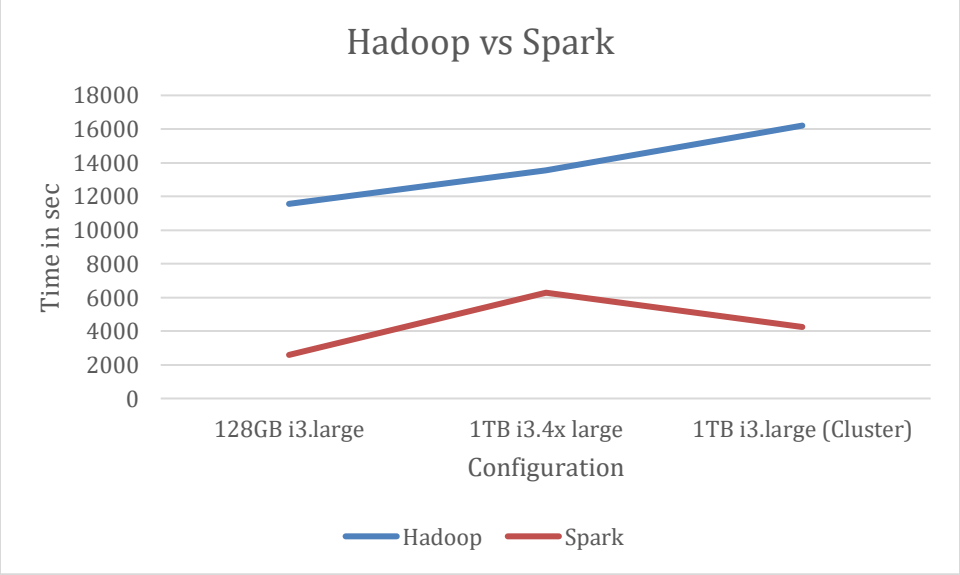
Shared Memory:

Config	16	8	4	2
128GB i3.large	15447500	22487389	29534138	55402687
1TB i3.4x large	16942600	23169513	49379459	94125548.8



Hadoop and Spark:

	Hadoop	Spark
128GB i3.large	11558	2588
1TB i3.4x large	13549	6282
1TB i3.large (Cluster)	16206	4234



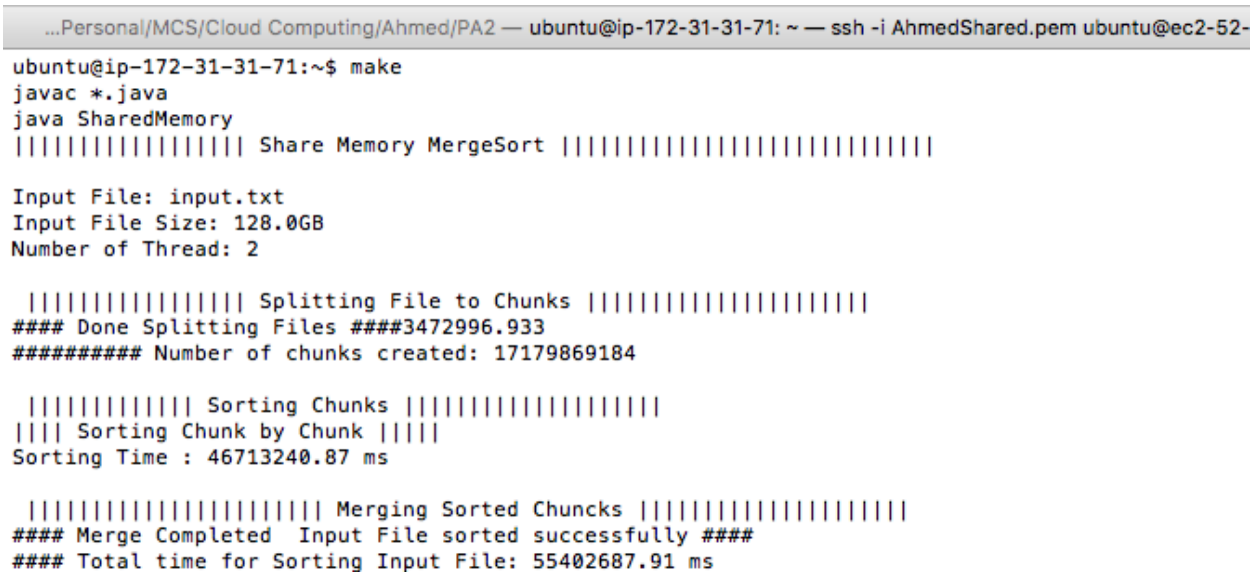
We broke our assignment into into several part as follows :-

1. Virtual Cluster (1-node i3.large): Setup virtual cluster of 1 node on Amazon EC2 using i3.large instance; generate a dataset of 128GB in size; during performance evaluation, do not include measurement of the time to generate the data, load the data, or verify that the data is sorted correctly; performance evaluation should only include the time to sort the data; after experiments (a), (b), and (c), terminate the instance.
    a. Shared-Memory TeraSort: Implement the Shared-Memory TeraSort application in your favorite language (without using Hadoop or Spark); generate the 128GB dataset and measure the time to sort it; you should make your Shared-Memory TeraSort multi-threaded to take advantage of multiple cores and SSD storage (which also requires multiple concurrent requests to achieve peak performance)

##

Observation and result of this category

128GB - 2 thread



128GB - 4 thread

```
...Personal/MCS/Cloud Computing/Ahmed/PA2 — ubuntu@ip-172-31-31-71: ~ — ssh -i AhmedShared.pem ubuntu@e
ubuntu@ip-172-31-31-71:~$ make
javac *.java
java SharedMemory
|||||||||||||||||| Share Memory MergeSort ||||||||||||||||||||||||||||||

Input File: input.txt
Input File Size: 128.0GB
Number of Thread: 4

 |||||||||||||||| Splitting File to Chunks ||||||||||||||||||||||
#### Done Splitting Files ####3394913.913
########## Number of chunks created: 8589934592

 |||||||||||||| Sorting Chunks ||||||||||||||||||||
|||| Sorting Chunk by Chunk |||||
Sorting Time : 20761440.76 ms

 |||||||||||||||||||||| Merging Sorted Chuncks ||||||||||||||||||||||
#### Merge Completed  Input File sorted successfully ####
#### Total time for Sorting Input File: 29534138.43 ms
```

128GB - 8 thread

```
...Personal/MCS/Cloud Computing/Ahmed/PA2 — ubuntu@ip-172-31-31-71: ~ — ssh -i AhmedShared.pem ubuntu@ec
ubuntu@ip-172-31-31-71:~$ make
javac *.java
java SharedMemory
|||||||||||||||||| Share Memory MergeSort ||||||||||||||||||||||||||||||

Input File: input.txt
Input File Size: 128.0GB
Number of Thread: 8

 |||||||||||||||| Splitting File to Chunks ||||||||||||||||||||||
#### Done Splitting Files ####3318586.425
########## Number of chunks created: 4294967296

 |||||||||||||| Sorting Chunks ||||||||||||||||||||
|||| Sorting Chunk by Chunk |||||
Sorting Time : 13624695.98 ms

 |||||||||||||||||||||| Merging Sorted Chuncks ||||||||||||||||||||||
#### Merge Completed  Input File sorted successfully ####
#### Total time for Sorting Input File: 22487389.18 ms
```

128GB - 16 thread

```
...Personal/MCS/Cloud Computing/Ahmed/PA2 — ubuntu@ip-172-31-31-71: ~ — ssh -i AhmedShared.pem ubuntu@ec2-52-27-24
ubuntu@ip-172-31-31-71:~$ make
javac *.java
java SharedMemory
|||||||||||||||||| Share Memory MergeSort ||||||||||||||||||||||||||||

Input File: input.txt
Input File Size: 128.0GB
Number of Thread: 16

 |||||||||||||||| Splitting File to Chunks ||||||||||||||||||||||
#### Done Splitting Files ####3243975.980
########## Number of chunks created: 2147483648

 |||||||||||||| Sorting Chunks ||||||||||||||||||||
|||| Sorting Chunk by Chunk |||||
Sorting Time : 6487959.878 ms

 |||||||||||||||||||||| Merging Sorted Chuncks ||||||||||||||||||||
#### Merge Completed  Input File sorted successfully ####
#### Total time for Sorting Input File: 15447598.00 ms
```

##

b. Hadoop TeraSort: Install Hadoop (including the HDFS distributed file system); turn off replication in order to have lower storage requirement; you must setup your own Hadoop cluster, and cannot use the Amazon Elastic MapReduce (EMR) available from Amazon; all Hadoop components should be configured on this 1 node; load the 128GB dataset into HDFS; implement the Hadoop TeraSort application, and evaluate its performance on 1 node

##

Observation and result of this category

- Gensort 128GB file generated:

```
ubuntu@ip-172-31-9-44: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-212-113-254.us-west-2.compute.amazonaws.com
ubuntu@ip-172-31-9-44:~$ ./gensort -a 1374389534 input128.txt
reall   41m7.456s
user    4m15.488s
sys     0ms
```

- File moved to HDFS

```
...ersonal/MCS/Cloud Computing/Ahmed/PA2 — ubuntu@ip-172-31-9-44: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-212-113-254.us-west-2.compute.
ubuntu@ip-172-31-9-44:~$ time hdfs dfs -copyFromLocal sortInput.txt /input4
reall   46m24.768s
user    17m26.528s
sys     3m44.768si
```

- File Sorted Output

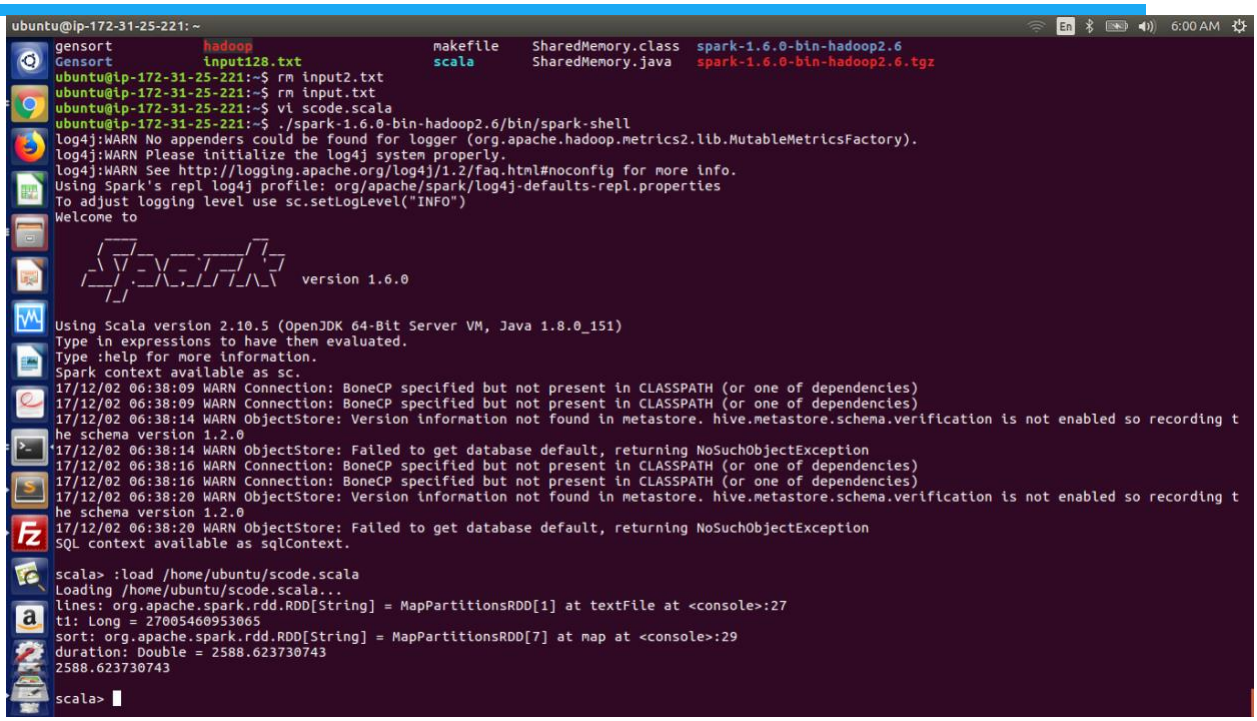
```
ubuntu@ip-172-31-9-44: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-212-113-254.us-west-2.compute.amazonaws.com
ubuntu@ip-172-31-9-44:~$ time hadoop jar hterasort.jar /input4 /output5

17/12/03 18:54:23 INFO mapreduce.Job:  map 100% reduce 100%
17/12/03 18:54:23 INFO mapreduce.Job: Job job_local1949456664_0001 completed successfully
17/12/03 18:54:23 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=1051417243136
                FILE: Number of bytes written=1682648899712
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=5905752016
                HDFS: Number of bytes written=1073741800
                HDFS: Number of read operations=118
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=11
        Map-Reduce Framework
                Map input records=343597376
                Map output records=343597376
                Map output bytes=1073741800
                Map output materialized bytes=1095216684
                Input split bytes=720
                Combine input records=343597376
                Combine output records=343597376
                Reduce input groups=343597376
                Reduce shuffle bytes=1095216684
                Reduce input records=343597376
                Reduce output records=343597376
                Spilled Records=40032212254
                Shuffled Maps =256
                Failed Shuffles=0
                Merged Map outputs=256
                GC time elapsed (ms)=213767
                Total committed heap usage (bytes)=126013669376
        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=1073770472
        File Output Format Counters
                Bytes Written=1073741800


real    192m38.506s
user    150m22.120s
sys     14m6.892s
```

##

c. Spark TeraSort: Install Spark (including the HDFS distributed file system); you must setup your own Spark cluster, and cannot use EMR to launch the Spark cluster; load the 128GB dataset into HDFS (unless its already loaded); implement the Spark TeraSort application, and evaluate its performance on 1 node

##

128GB i3.large



##

2. Virtual Cluster (1-node i3.4xlarge): Setup virtual cluster of 1 node on Amazon EC2 using i3.4xlarge instance; generate a dataset of 1TB in size; during performance evaluation, do not include measurement of the time to generate the data, load the data, or verify that the data is sorted correctly; performance evaluation should only include the time to sort the data; after experiments (a), (b), and (c), terminate the instance
    a. Shared-Memory TeraSort: Evaluate Shared-Memory TeraSort application and measure the time to sort the 1TB dataset on 1 node (don't forget to increase the number of threads to make use of the 16-core instance)

##

Observation and result of this category

1TB - 2 thread

```
ubuntu@ip-172-31-2-37: ~ — ssh -i thursdaykey.pem ubuntu@ec2-54-191-154-143.us-west-2.compute.ama...
ubuntu@ip-172-31-2-37:~$ make
javac *.java
java SharedMemory
|||||||||||||||||| Share Memory MergeSort ||||||||||||||||||||||||||||||

Input File: input.txt
Input File Size: 1000.0GB
Number of Thread: 2

 |||||||||||||||| Splitting File to Chunks ||||||||||||||||||||||
#### Done Splitting Files ####3809134.024
########## Number of chunks created: 17179869184

 |||||||||||||| Sorting Chunks ||||||||||||||||||||
|||| Sorting Chunk by Chunk |||||
Sorting Time : 39848995.62 ms

 |||||||||||||||||||||||| Merging Sorted Chuncks ||||||||||||||||||||||
#### Merge Completed  Input File sorted successfully ####
#### Total time for Sorting Input File: 49379459.04 ms
```

1TB - 4 thread

```
ubuntu@ip-172-31-2-37: ~ — ssh -i thursdaykey.pem ubuntu@ec2-54-191-154-143.us-west-2.compute.ama...
ubuntu@ip-172-31-2-37:~$ make
javac *.java
java SharedMemory
|||||||||||||||||| Share Memory MergeSort ||||||||||||||||||||||||||||||

Input File: input.txt
Input File Size: 1000.0GB
Number of Thread: 4

 |||||||||||||||| Splitting File to Chunks ||||||||||||||||||||||
#### Done Splitting Files ####3723493.669
########## Number of chunks created: 8589934592

 |||||||||||||| Sorting Chunks ||||||||||||||||||||
|||| Sorting Chunk by Chunk |||||
Sorting Time : 20636086.98 ms

 |||||||||||||||||||||||| Merging Sorted Chuncks ||||||||||||||||||||||
#### Merge Completed  Input File sorted successfully ####
#### Total time for Sorting Input File: 30257858.64 ms
```

1TB - 8 thread

```
ubuntu@ip-172-31-2-37: ~ — ssh -i thursdaykey.pem ubuntu@ec2-54-191-154-143.us-west-2.compute.ama...
ubuntu@ip-172-31-2-37:~$ make
javac *.java
java SharedMemory
|||||||||||||||||| Share Memory MergeSort ||||||||||||||||||||||||||||

Input File: input.txt
Input File Size: 1000.0GB
Number of Thread: 8

 |||||||||||||||| Splitting File to Chunks ||||||||||||||||||||||
#### Done Splitting Files ####3639778.758
########## Number of chunks created: 4294967296

 |||||||||||||| Sorting Chunks ||||||||||||||||||||
|||| Sorting Chunk by Chunk |||||
Sorting Time : 13449035.88 ms

 |||||||||||||||||||||||| Merging Sorted Chuncks ||||||||||||||||||||
#### Merge Completed  Input File sorted successfully ####
#### Total time for Sorting Input File: 23169513.78 ms
```

1TB - 16 thread

```
ubuntu@ip-172-31-2-37: ~ — ssh -i thursdaykey.pem ubuntu@ec2-54-191-154-143.us-west-2.compute.ama...
ubuntu@ip-172-31-2-37:~$ make
javac *.java
java SharedMemory
|||||||||||||||||| Share Memory MergeSort ||||||||||||||||||||||||||||

Input File: input.txt
Input File Size: 1000.0GB
Number of Thread: 16

 |||||||||||||||| Splitting File to Chunks ||||||||||||||||||||||
#### Done Splitting Files ####3557946.78
########## Number of chunks created: 2147483648

 |||||||||||||| Sorting Chunks ||||||||||||||||||||
|||| Sorting Chunk by Chunk |||||
Sorting Time : 7115892.67 ms

 |||||||||||||||||||||||| Merging Sorted Chuncks ||||||||||||||||||||
#### Merge Completed  Input File sorted successfully ####
#### Total time for Sorting Input File: 16942687.98 ms
```

##

b. Hadoop TeraSort: Configure HDFS with no replication; evaluate Hadoop TeraSort application and measure the time to sort the 1TB dataset on 1 node (don't forget to increase the number of mappers and reducers to make use of the 16-core instance)

##

Observation and result of this category

- Gensort 1TB generated

```
ubuntu@ip-172-31-9-44: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-212-113-254.us-west-2.compute.amazonaws.com
ubuntu@ip-172-31-23-103:$ time ./gensort -a 10995116278 input.txt
real    110m44.192s
user    67m57.184s
sys     0m0.534s

ubuntu@ip-172-31-3-103:$ ls -lash input.txt
1T -rwxrwxr-x 1 ubuntu ubuntu 1T Dec  3 13:36 input.txt
```

- Move file to DFS

```
ubuntu@ip-172-31-9-44: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-212-113-254.us-west-2.compute.amazonaws.com
ubuntu@ip-172-31-23-103:~$ time hdfs dfs -copyFromLocal input.txt /input3

real    37m41.565s
user    28m40.724s
sys     0m0.820s
ubuntu@ip-172-31-23-103:~$ ls -lash input.txt
1T -rwxrwxr-x 1 ubuntu ubuntu 1T Dec  3 14:18 input.txt
```

- Input File Sorted

```
ubuntu@ip-172-31-9-44: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-212-113-254.us-west-2.compute.amazonaws.com
ubuntu@ip-172-31-23-103:$ time hadoop jar hterasort.jar /input /output
17/12/02 13:40:48 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.
17/12/02 13:40:48 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionI
17/12/02 13:40:48 INFO input.FileInputFormat: Total input files to process : 1
17/12/02 13:40:48 INFO mapreduce.JobSubmitter: number of splits:32
```

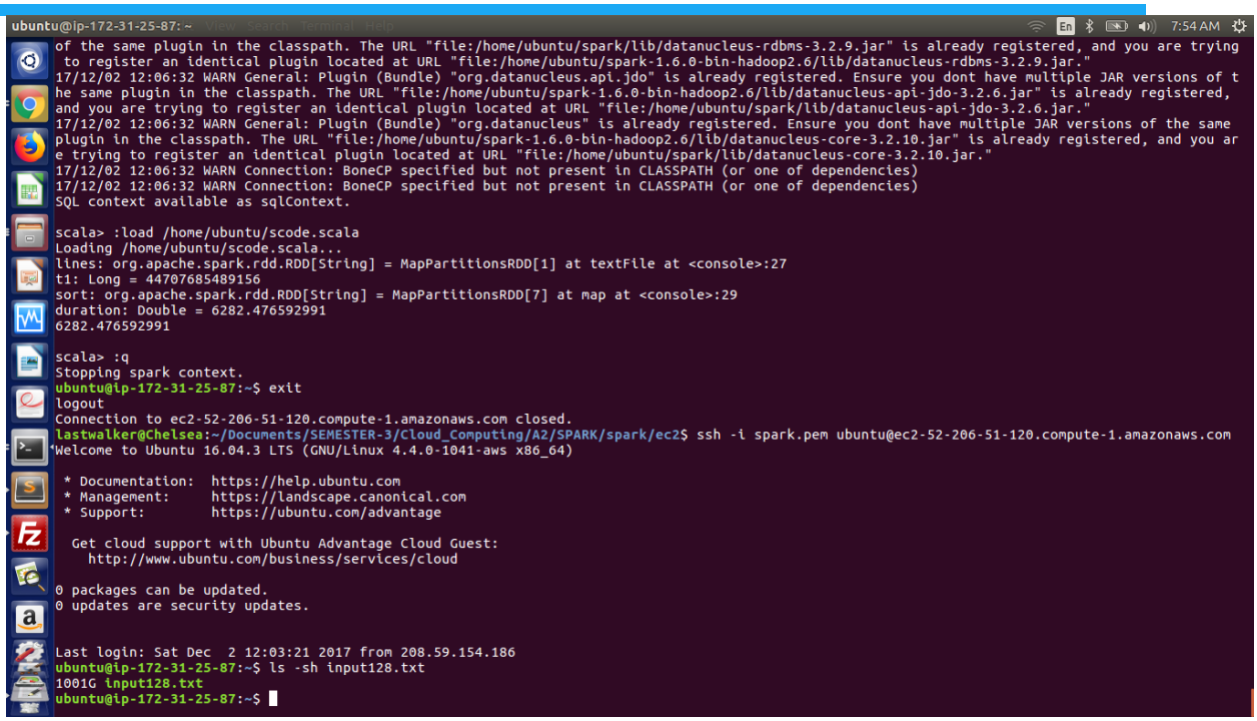
```
ubuntu@ip-172-31-23-103:~$ time hadoop jar hterasort.jar /input4 /output5 >out2 2>&1

17/12/03 18:54:23 INFO mapreduce.Job:  map 100% reduce 100%
17/12/03 18:54:23 INFO mapreduce.Job: Job job_local1949456664_0001 completed successfully
17/12/03 18:54:23 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=8411337945088
                FILE: Number of bytes written=13461191197696
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3785587042256
                HDFS: Number of bytes written=6611028262600
                HDFS: Number of read operations=726526
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=67727
        Map-Reduce Framework
                Map input records=34359737667655
                Map output records=3435973765677
                Map output bytes=10667773741800
                Map output materialized bytes=10975425216684
                Input split bytes=23040
                Combine input records=34359737667655
                Combine output records=3435973765677
                Reduce input groups=3435934567376
                Reduce shuffle bytes=10952164532684
                Reduce input records=34359737667655
                Reduce output records=3435973765677
                Spilled Records=400322532512254
                Shuffled Maps =2048
                Failed Shuffles=0
                Merged Map outputs=2048
                GC time elapsed (ms)=309767
                Total committed heap usage (bytes)=195826013669376
        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=1095350878208
        File Output Format Counters
                Bytes Written=1095350878208


real    225m49.506s
user    179m12.120s
sys     13m670.092s
```

##

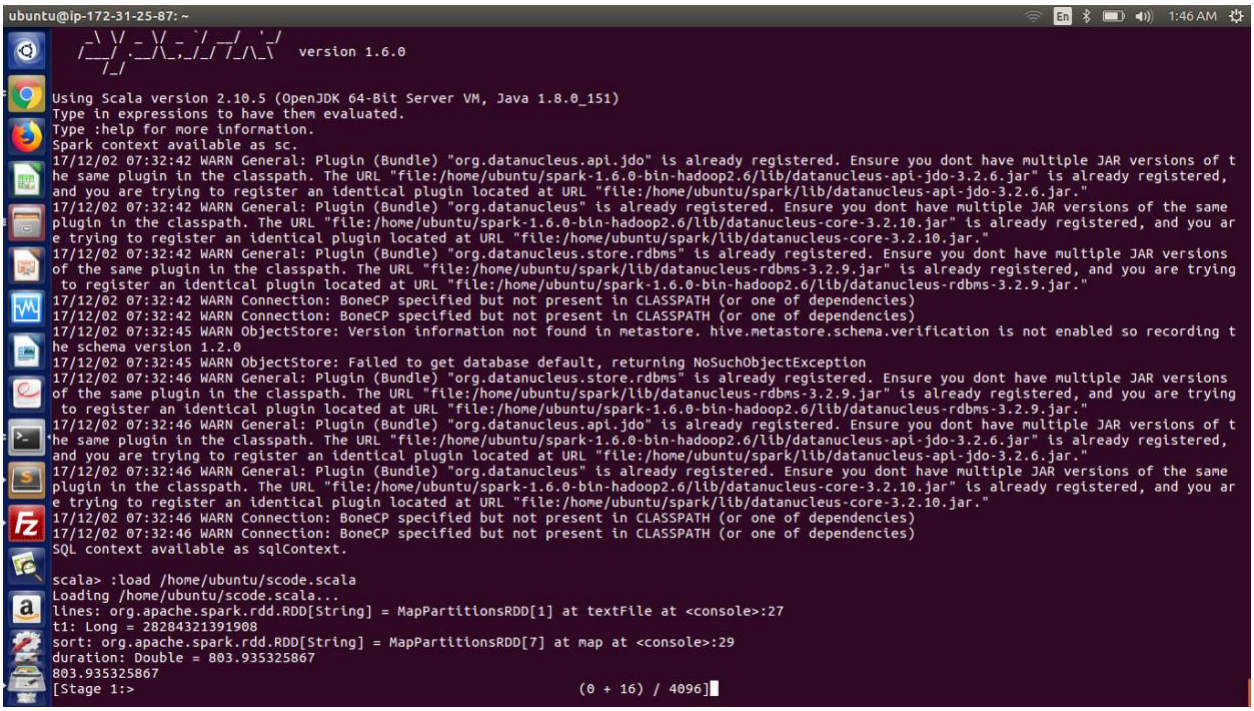
c. Spark TeraSort: Configure HDFS with no replication; evaluate Spark TeraSort application and measure the time to sort the 1TB dataset on 1 node (don't forget to increase the parallelism to make use of the 16-core instance)

##

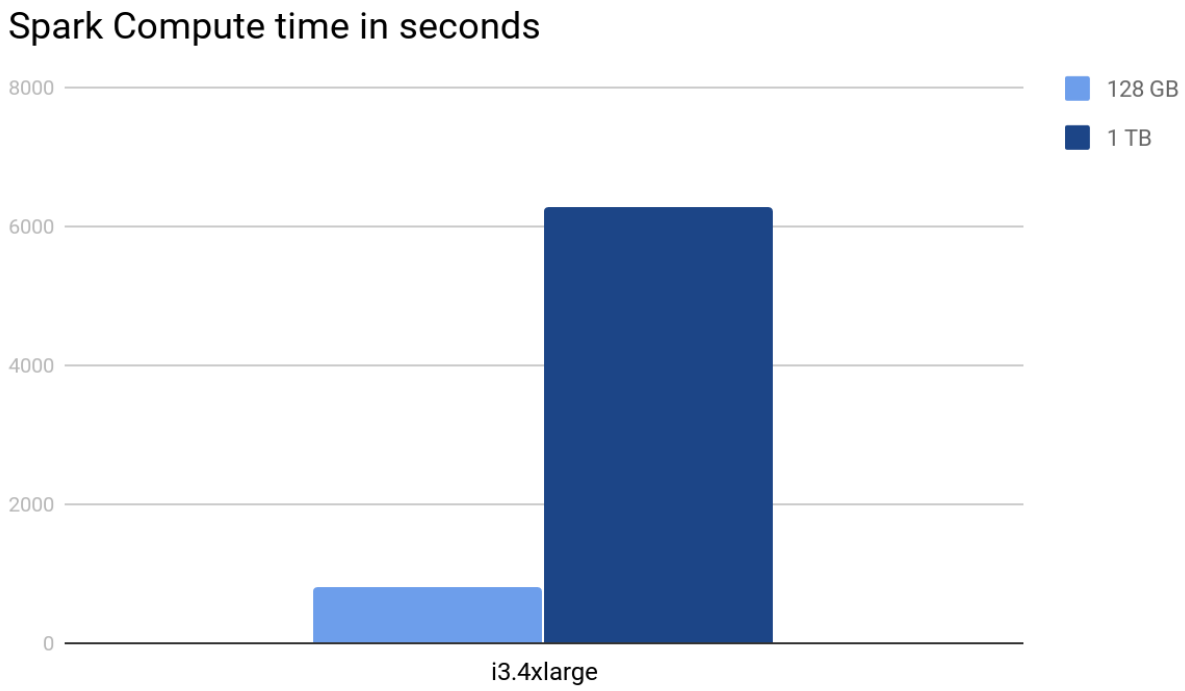
1 TB Single node i3.4xlarge



1 TB on 12.4xlarge : 6282.47 sec compute time.



128 GB on i3.4x : 803.8 seconds compute time



##

3. Virtual Cluster (1-node i3.large): Setup virtual cluster of 8 nodes on Amazon EC2 using i3.large instance types; generate a dataset of 1TB in size; during performance evaluation, do not include measurement of the time to generate the data, load the data, or verify that the data is sorted correctly; performance evaluation should only include the time to sort the data; after experiments (a) and (b), terminate the instance
    a. Hadoop TeraSort: Configure Hadoop to span all 8 nodes, and HDFS with no replication; evaluate Hadoop TeraSort application and measure the time to sort the 1TB dataset.

##

Observation and result of this category

```
[ubuntu@ip-172-31-6-96:~$ hadoop jar hadoop-2.7.4/share/hadoop/mapreduce/hadoop-mapreduce-ex]
amples-2.7.4.jar teragen 85800345 /teraInput8GB
17/12/03 01:41:48 INFO client.RMProxy: Connecting to ResourceManager at ec2-34-216-149-36.u
s-west-2.compute.amazonaws.com/172.31.6.96:9076
17/12/03 01:41:49 INFO terasort.TeraSort: Generating 85800345 using 2
17/12/03 01:41:50 INFO mapreduce.JobSubmitter: number of splits:2
17/12/03 01:41:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1512265063012
_0001
17/12/03 01:41:50 INFO impl.YarnClientImpl: Submitted application application_1512265063012
_0001
17/12/03 01:41:51 INFO mapreduce.Job: The url to track the job: http://ec2-34-216-149-36.us
-west-2.compute.amazonaws.com:9078/proxy/application_1512265063012_0001/
17/12/03 01:41:51 INFO mapreduce.Job: Running job: job_1512265063012_0001
```

```
PA2 — ubuntu@ec2-34-215-219-28: ~ — ssh -i AhmedShared.pem ubuntu@ec2-34-215-2...
s-west-2.compute.amazonaws.com/172.31.6.96:9076
17/12/02 07:36:25 INFO impl.YarnClientImpl: Killed application application_1512196130380_00
01
Killed job job_1512196130380_0001
[ubuntu@ip-172-31-6-96:~$ hadoop job -list                                              ]
DEPRECATED: Use of this script to execute mapred command is deprecated.
Instead use the mapred command for it.

17/12/02 07:36:30 INFO client.RMProxy: Connecting to ResourceManager at ec2-34-215-219-28.u
s-west-2.compute.amazonaws.com/172.31.6.96:9076
Total jobs:0
                  JobId      State            StartTime      UserName           Queue      P
riority  UsedContainers  RsvdContainers  UsedMem         RsvdMem         NeededMem
AM info
```

```
...ersonal/MCS/Cloud Computing/Ahmed/PA2 — ubuntu@ip-172-31-6-96: ~ — ssh -i AhmedShared.pem ubuntu@ec2-52-35-230-240.us-west-2.compute.am
17/12/02 12:22:32 INFO mapreduce.Job: Job job_local717214462_0001 completed successfully
17/12/02 12:22:32 INFO mapreduce.Job: Counters: 45
        File System Counters
                FILE: Number of bytes read=58526155002345
                FILE: Number of bytes written=93991308536775
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=390784688917066
                HDFS: Number of bytes written=7900410385070
                HDFS: Number of read operations=770750
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=90305
        Map-Reduce Framework
                Map input records=4450074413
                Map output records=4450074413
                Map output bytes=335007441600
                Map output materialized bytes=4450074413
                Input split bytes=512000
                Combine input records=4450074413
                Combine output records=4450074413
                Reduce input groups=323435932453
                Reduce shuffle bytes=45427802345
                Reduce input records=4450074413
                Reduce output records=4450074413
                Spilled Records=4523513376
                Shuffled Maps =2048
                Failed Shuffles=0
                Merged Map outputs=2048
                GC time elapsed (ms)=243890
                Total committed heap usage (bytes)=144651859986
        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=1099511627776
        File Output Format Counters
                Bytes Written=1099511627776

real    270m6.902s
user    267m34.756s
sys     0m54.543s
ubuntu@ip-172-31-6-96:~$
```

##########

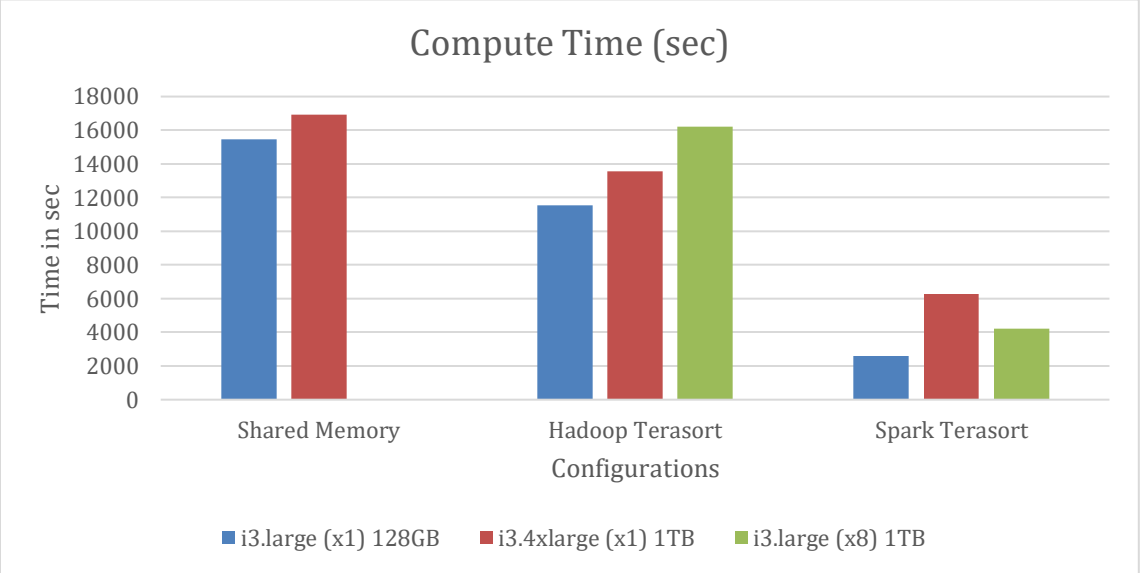
Performance Evaluation Table

Experiment (instance/dataset)	Shared Memory TeraSort	Hadoop TeraSort	Spark TeraSort	MPI TeraSort
Compute Time (sec) [1xi3.large 128GB]	15447.5	11558.4	2588.624	N/A
Data Read (GB) [1xi3.large 128GB]	128	128	128	N/A
Data Write (GB) [1xi3.large 128GB]	128	128	128	N/A
I/O Throughput (MB/sec) [1xi3.large 128GB]	16.97	22.68	101.29	N/A
Compute Time (sec) [1xi3.4xlarge 1TB]	16942.6	13549.2	6282.47	N/A
Data Read (GB) [1xi3.4xlarge 1TB]	128	128	128	N/A
Data Write (GB) [1xi3.4xlarge 1TB]	128	128	128	N/A
I/O Throughput (MB/sec) [1xi3.4xlarge 1TB]	123.78	154.78	333.81	N/A
Compute Time (sec) [8xi3.large 1TB]	N/A	16206.7	4234.1	N/A
Data Read (GB) [8xi3.large 1TB]	N/A	1024	1024	N/A
Data Write (GB) [8xi3.large 1TB]	N/A	1024	1024	N/A
I/O Throughput (MB/sec) [8xi3.large 1TB]	N/A	129.4	495.3	N/A

Speedup (weak scale)	7.3	5.1	4.89	N/A
Efficiency (weak scale)	91%	71.3	61%	N/A

Compute Time (sec):

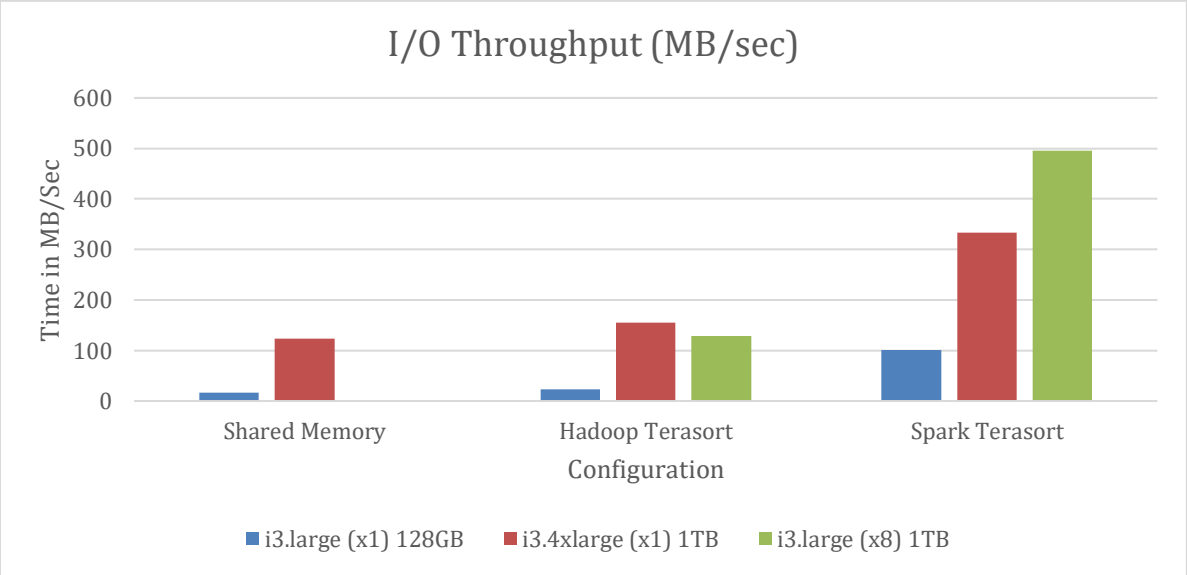
	i3.large (x1)	i3.4xlarge (x1)	i3.large (x8)
Shared Memory	15447.5	16943	N/A
Hadoop Terasort	11558.4	13549	16206.7
Spark Terasort	2588.624	6282	4234.1



Blue bar : 128 GB , Orange bar : 1 TB , Grey bar : 1 TB cluster

I/O Throughput:

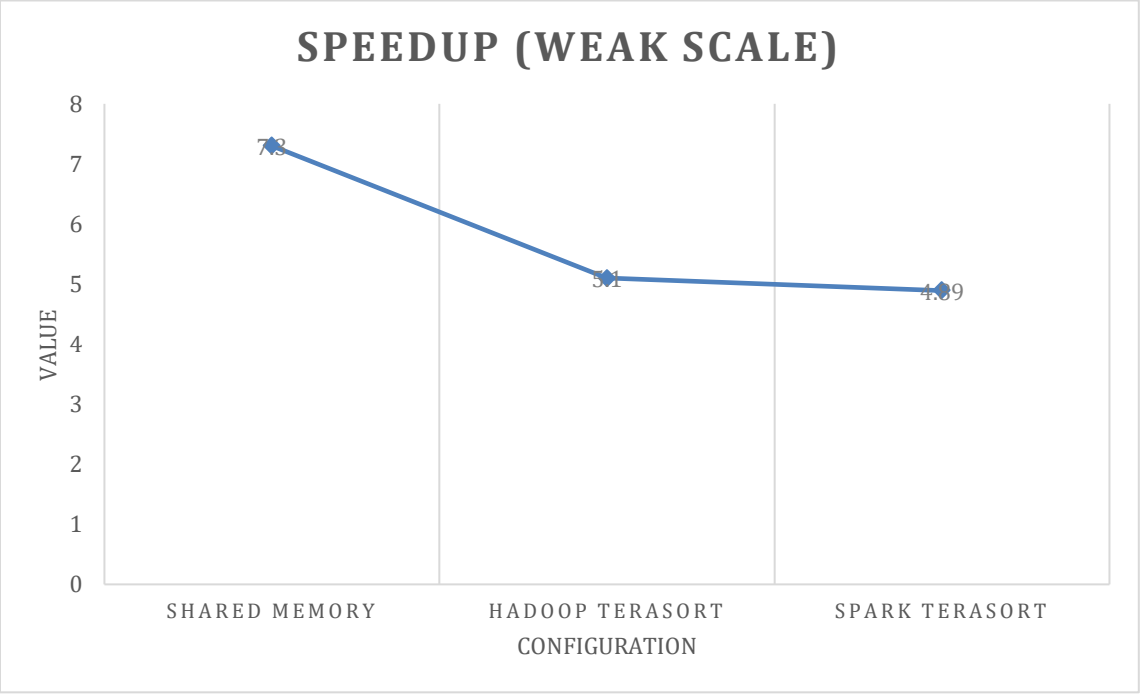
	i3.large (x1)	i3.4xlarge (x1)	i3.large (x8)
Shared Memory	16.97	124	
Hadoop Terasort	22.68	155	129.4
Spark Terasort	101.29	334	495.3



Blue bar : 128 GB , Orange bar : 1 TB , Grey bar : 1 TB cluster

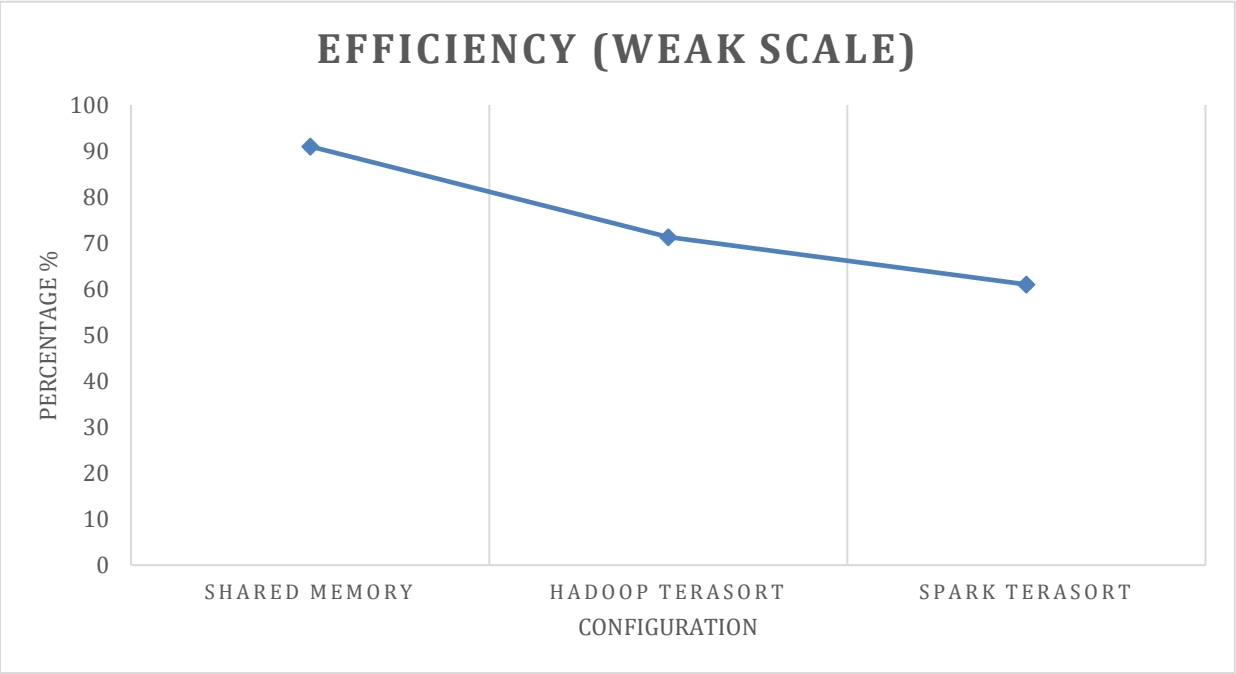
Speedup:

Configuration	Speedup
Shared Memory	7.3
Hadoop Terasort	5.1
Spark Terasort	4.89



Efficiency:

Configuration	Efficiency (%)
Shared Memory	91
Hadoop Terasort	71.3
Spark Terasort	61



Conclusion

We ran few experiments and tried to calculate computation time, throughput, speed and efficiency of different configuration provided for this assignment. We ran our tests on i3.large and i3,4xlarge instances and the above contents represents our reading and observations.

- We came to a conclusion that Apache Spark is faster than Hadoop and external sort shared memory.
- Spark can run as a standalone or on top of Hadoop YARN, where it can read data directly from HDFS.
- Apache Spark processes data in-memory while Hadoop MapReduce persists back to the disk after a map or reduce action, so Spark should outperform Hadoop MapReduce.
- Spark needs a lot of memory. Much like standard DBs, it loads a process into memory and keeps it there until further notice, for the sake of caching. If Spark runs on Hadoop YARN with other resource-demanding services, or if the data is too big to fit entirely into the memory, then there could be major performance degradations for Spark.
- MapReduce, however, kills its processes as soon as a job is done, so it can easily run alongside other services with minor performance differences.
- Spark performs better when all the data fits in the memory, especially on dedicated clusters; Hadoop MapReduce is designed for data that doesn't fit in the memory and it can run well alongside other services.
- Spark has comfortable APIs for Java, Scala and Python, and also includes Spark SQL (formerly known as Shark) for the SQL savvy. Thanks to Spark's simple building blocks, it's easy to write user-defined functions. It even includes an interactive mode for running commands with immediate feedback.
- Hadoop MapReduce is written in Java and is infamous for being very difficult to program. Pig makes it easier, though it requires some time to learn the syntax, and Hive adds SQL compatibility to the plate. Some Hadoop tools can also run MapReduce jobs without any programming.
- Spark is easier to program and includes an interactive mode; Hadoop MapReduce is more difficult to program but many tools are available to make it easier.
- Spark has excellent performance and is highly cost-effective thanks to in-memory data processing. It's compatible with all of Hadoop's data sources and file formats, and thanks to friendly APIs that are available in several languages, it also has a faster learning curve. Spark even includes graph processing and machine-learning capabilities.

- Hadoop MapReduce is a more mature platform and it was built for batch processing. It can be more cost-effective than Spark for truly Big Data that doesn't fit in memory and also due to the greater availability of experienced staff. Furthermore, the Hadoop MapReduce ecosystem is currently bigger thanks to many supporting projects, tools and cloud services.

References

1. http://www.ordinal.com/gensort.html
2. https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/terasort/package-summary.html
3. https://sparkour.urizone.net/recipes/spark-ec2/
4. https://www.ec2instances.info/
5. https://github.com/viyatgandhi/TeraSort-Local-Hadoop-MR-Spark
6. https://letsdobigdata.wordpress.com/2014/01/13/setting-up-hadoop-multi-node-cluster-on-amazon-ec2-part-1/
7. https://letsdobigdata.wordpress.com/2014/01/13/setting-up-hadoop-1-2-1-multi-node-cluster-on-amazon-ec2-part-2/
8. https://www.xplenty.com/blog/apache-spark-vs-hadoop-mapreduce/
9. https://www.ec2instances.info/
10. https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/terasort/package-summary.html
11. http://ant.apache.org/bindownload.cgi
12. http://www.oracle.com/technetwork/java/javase/downloads/index.html
13. http://hadoop.apache.org/docs/current1/mapred_tutorial.html

14. http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html
15. http://spark.apache.org/downloads.html
16. http://spark.apache.org/docs/latest/cluster-overview.html
17. http://www.ordinal.com/gensort.html
18. http://sortbenchmark.org
19. http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf

Special thanks to professor Ioan Raicu and his teaching assistants TAs (George M & J Peng)for helping us with lectures, concepts and proper guidance.

It was fun working in this really interesting assignment we learned alot about Big-Data technologies such as Hadoop and Spark.

THANK YOU