

# REPORT(Java, Hadoop, Swift & MPI)

<https://github.com/chintan-29/Programming-Assignment2>

## Introduction:

In this programming assignment we learn how to get experience programming with:

- Amazon Web Services, specifically the EC2 cloud
- The Hadoop framework
- The Swift parallel programming system

We had covered word count application using Java, Hadoop and Swift.

**OS : Ubuntu 14.04**

**RAM : 4GB**

**ANT version : Apache Ant 1.9.4**

**Java version : openjdk 1.7.0\_65**

**Hadoop version : Hadoop 2.5.1**

**Swift version : swift-0.95-RC6**

**MPI version : MPI 1.2.9**

## **Participation:**

**Java** : partially done by all 3 of us

**Hadoop** : Configuration and Programming(Akash Rafaliya)

**Swift** : Configuration(Chintan Patel(major) and Vashishtha Panchal(minor)),  
Programming(Chintan Patel)

**MPI** : Configuration and Programming(Vashishtha Panchal)

**Documentation** : Vashishtha Panchal(major), Akash Rafaliya(minor), Chintan Patel(minor)

## Installing virtual cluster on 1 and 17-node :

1. To create virtual cluster, first of all signup on aws.amazon.com
2. Go to EC2 section from amazon web services.
3. Create an instance by clicking on “**Launch Instance**”
4. Choose AMI whichever is required. Here, we used “**Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-3d50120d**”
5. Now you need to choose instance type according to your requirement.

6. After that configure Instance, you can select “spot instance” as well as “on demand instance”, whatever you need.
7. Add storage as per requirement.
8. Give the name to your instance.
9. Configure the security group. Here “All TCP” and “All ICMP” protocols are being covered for every instance.
10. Finally review instance and launch it.
11. When you click on launch, you are required to create a keypair. Once you create it and download it just you can go to next step.
12. Once you finished with all this procedure go to instance tab.
13. You’ll see instance is created on that page.
14. Select on instance and press “connect”.
15. Now goto your terminal and goto path wherever your key is stored.
16. Here you have to write command for example :

**“chmod 400 key.pem”**

17. Now you can configure virtual cluster using command:

**“ssh -i mpi.pem ubuntu@54.187.182.1031”**

18. This is how your virtual cluster is being started.
19. If you want to create an image of that running node, just right click on that node and click on “create image”.
20. You’ll find an image of that node in AMIs tag.
21. When you create this image it will consider as 1-node.
22. If we create 16 more images than our virtual cluster is being consist of 16-slave nodes and one master node.

## **Configuration and Installation**

### **Java**

- First we have to start instances on amazon EC2.
- We had install java on our cluster and now we had created wc.java file.
- To run java program for single node you have to write :

**javac wc.java**

**java wc**

- For multimode we had created wc\_mul.java file.
- To run java program for multi node you have to write :

**javac wc\_mul.java**

**java wc\_mul 4/8/16**

## **Hadoop**

### **Installation and Problems faced during installation :**

Here we have used **Hadoop version 2.5.1[2]** and **Java version openjdk 1.7.0\_65[1]**

#### **For single node :**

- To install Hadoop on amazon ec2 first we have to create a virtual cluster. It means we have to create an instance.
- Once we create instance we have to connect our local machine with cluster by writing command:

**ssh -i mpi.pem ubuntu@54.191.122.52**

- Here we need to create passwordless ssh, so master node can access slaves without any password.
- And then unpack the downloaded Hadoop 2.5.1 distribution. In the distribution, edit the file etc/hadoop/hadoop-env.sh to define some parameters as follows:
  - # set to the root of your Java installation
  - export JAVA\_HOME=/usr/lib/jvm/java1.7.0-openjdk-amd64
  - # Assuming your installation directory is /home/Ubuntu/Hadoop-2.5.1
  - export HADOOP\_PREFIX=/home/Ubuntu/Hadoop-2.5.1
  - Try the following command:
  - \$ bin/hadoop
  - This will display the usage documentation for the hadoop script.
  - By default, Hadoop is configured to run in a non-distributed mode, as a single Java process. This is useful for debugging.
  - We had done some several psudo-distributed operations.

- Here, we had configure 4 .xml files named core-site.xml, MapRed-site.xml, YARN-site.xml and HDFS-site.xml[2]
- First we have to format HDFS name node.  
 $\$ \text{bin/hdfs namenode -format}$
- To start all deamons we have to write command:  
 $\$ \text{sbin/start-all.sh}$
- Make the HDFS directories required to execute MapReduce jobs:

$\$ \text{bin/hdfs dfs -mkdir /user}$   
 $\$ \text{bin/hdfs dfs -mkdir /user/<username>}$

- To check demons are weather started or not we used JPS. JPS will give brief idea weather deamons are started or not.
- This is how we had configure the Hadoop for single node.

#### For multi node :

- It's the same procedure as single node for multi node. Here, you first need to create 17 nodes instances. Once you create all instances you can do the same procedure as single node.
- Now Hadoop2.5.1/etc/Hadoop in this package, there will be slaves.xml file.
- We have to configure that file.
- In master node, we have to write all private IPs of all slaves and in slave file we have to write its own private IPs.

#### Execution Process :

- Once you starts all the deamons in master and slaves node, you can run Hadoop program for 16 nodes by using the command :  
 $\$ \text{bin/hadoop jar wc-1.jar input output}$
- This command will fetch input file from HDFS file system's input directory and output will be stored in output directory of HDFS.
- To fetch file from hdfs to local machine, you need to write command:  
 $\$ \text{bin/hdfs dfs -get output output}$
- Now you can see output in output directory in local file system.

#### Name-node :

- The Name-Node is the centerpiece of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself.

**Data-node:**

- A DataNode stores data in the [HadoopFileSystem]. A functional filesystem has more than one DataNode, with data replicated across them.

**Node Manager:**

- The NodeManager (NM) is YARN's per-node agent, and takes care of the individual compute nodes in a Hadoop cluster.

**Resource Manager:**

- The ResourceManager REST API's allow the user to get information about the cluster - status on the cluster, metrics on the cluster, scheduler information, information about nodes in the cluster, and information about applications on the cluster.

**Problems :**

- While configuring Hadoop on cloud, we dealt with problem listed as below:
- We had a problem while configuring ssh.
- But by using the link [3] , we configured ssh.
- While we run program on multiple node, it gets stuck at map 0% , reduce 0%.
- Besides that few errors occurred which were being solved by “try and error”

**Snapshots :**

```

akash@akash-PC: ~/hadoop/hadoop-2.5.1
logs/hadoop-akash-secondarynamenode-akash-PC.out
starting yarn daemons
starting resourcemanager, logging to /home/akash/hadoop/hadoop-2.5.1/logs/yarn-a
akash-resourcemanager-akash-PC.out
localhost: starting nodemanager, logging to /home/akash/hadoop/hadoop-2.5.1/logs
/yarn-akash-nodemanager-akash-PC.out
akash@akash-PC:~/hadoop/hadoop-2.5.1$ jps
2935 SecondaryNameNode
3236 NodeManager
2743 DataNode
3104 ResourceManager
2587 NameNode
3529 Jps
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hdfs dfs -mkdir /user
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hdfs dfs -mkdir /user/akash
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hdfs dfs -put wiki10gb
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hadoop jar wc-1.jar wordcount wiki10gb
qqq
1460
14/10/25 01:52:41 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
14/10/25 01:52:42 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
14/10/25 01:52:42 INFO input.FileInputFormat: Total input paths to process : 1
14/10/25 01:52:43 INFO mapreduce.JobSubmitter: number of splits:78
14/10/25 01:52:43 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14
14218714201_0001
14/10/25 01:52:44 INFO impl.YarnClientImpl: Submitted application application_14
14218714201_0001
14/10/25 01:52:44 INFO mapreduce.Job: The url to track the job: http://akash-PC:
8088/proxy/application_1414218714201_0001/
14/10/25 01:52:44 INFO mapreduce.Job: Running job: job_1414218714201_0001
14/10/25 01:52:52 INFO mapreduce.Job: Job job_1414218714201_0001 running in uber
mode : false
14/10/25 01:52:52 INFO mapreduce.Job: map 0% reduce 0%
14/10/25 01:53:17 INFO mapreduce.Job: map 1% reduce 0%
14/10/25 01:53:33 INFO mapreduce.Job: map 2% reduce 0%
14/10/25 01:53:49 INFO mapreduce.Job: map 3% reduce 0%
14/10/25 01:54:13 INFO mapreduce.Job: map 4% reduce 0%
14/10/25 01:54:13 INFO mapreduce.Job: map 4% reduce 0%

```

- Shows for 1-node Hadoop is working.

```

akash@akash-PC: ~/hadoop/hadoop-2.5.1
2935 SecondaryNameNode
3236 NodeManager
2743 DataNode
3104 ResourceManager
2587 NameNode
3529 Jps
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hdfs dfs -mkdir /user
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hdfs dfs -mkdir /user/akash
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hdfs dfs -put wiki10gb
akash@akash-PC:~/hadoop/hadoop-2.5.1$ bin/hadoop jar wc-1.jar wordcount wiki10gb
qqq
1460
14/10/25 01:52:41 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
14/10/25 01:52:42 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
14/10/25 01:52:42 INFO input.FileInputFormat: Total input paths to process : 1
14/10/25 01:52:43 INFO mapreduce.JobSubmitter: number of splits:78
14/10/25 01:52:43 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14
14218714201_0001
14/10/25 01:52:44 INFO impl.YarnClientImpl: Submitted application application_14
14218714201_0001
14/10/25 01:52:44 INFO mapreduce.Job: The url to track the job: http://akash-PC:
8088/proxy/application_1414218714201_0001/
14/10/25 01:52:44 INFO mapreduce.Job: Running job: job_1414218714201_0001
14/10/25 01:52:52 INFO mapreduce.Job: Job job_1414218714201_0001 running in uber
mode : false
14/10/25 01:52:52 INFO mapreduce.Job: map 0% reduce 0%
14/10/25 01:53:17 INFO mapreduce.Job: map 1% reduce 0%
14/10/25 01:53:33 INFO mapreduce.Job: map 2% reduce 0%
14/10/25 01:53:49 INFO mapreduce.Job: map 3% reduce 0%
14/10/25 01:54:13 INFO mapreduce.Job: map 4% reduce 0%
14/10/25 01:54:26 INFO mapreduce.Job: map 5% reduce 0%
14/10/25 01:54:40 INFO mapreduce.Job: map 6% reduce 0%
14/10/25 01:54:45 INFO mapreduce.Job: map 7% reduce 0%
14/10/25 01:54:46 INFO mapreduce.Job: map 8% reduce 0%
14/10/25 01:55:08 INFO mapreduce.Job: map 9% reduce 0%
14/10/25 01:55:28 INFO mapreduce.Job: map 10% reduce 0%
14/10/25 01:55:46 INFO mapreduce.Job: map 11% reduce 0%
14/10/25 01:56:05 INFO mapreduce.Job: map 12% reduce 0%

```

- Shows for 1-node 10 GB file in Hadoop is working

```

akash@akash-PC: ~/hadoop/hadoop-2.5.1
14/10/25 02:12:57 INFO mapreduce.Job: map 87% reduce 28%
14/10/25 02:12:58 INFO mapreduce.Job: map 87% reduce 29%
14/10/25 02:13:13 INFO mapreduce.Job: map 88% reduce 29%
14/10/25 02:13:25 INFO mapreduce.Job: map 89% reduce 29%
14/10/25 02:13:43 INFO mapreduce.Job: map 90% reduce 29%
14/10/25 02:14:03 INFO mapreduce.Job: map 91% reduce 29%
14/10/25 02:14:08 INFO mapreduce.Job: map 92% reduce 29%
14/10/25 02:14:10 INFO mapreduce.Job: map 92% reduce 30%
14/10/25 02:14:22 INFO mapreduce.Job: map 93% reduce 30%
14/10/25 02:14:28 INFO mapreduce.Job: map 93% reduce 31%
14/10/25 02:14:38 INFO mapreduce.Job: map 94% reduce 31%
14/10/25 02:14:48 INFO mapreduce.Job: map 95% reduce 31%
14/10/25 02:15:03 INFO mapreduce.Job: map 96% reduce 31%
14/10/25 02:15:17 INFO mapreduce.Job: map 97% reduce 31%
14/10/25 02:15:31 INFO mapreduce.Job: map 98% reduce 31%
14/10/25 02:15:35 INFO mapreduce.Job: map 98% reduce 32%
14/10/25 02:15:37 INFO mapreduce.Job: map 99% reduce 32%
14/10/25 02:15:45 INFO mapreduce.Job: map 100% reduce 33%
14/10/25 02:15:48 INFO mapreduce.Job: map 100% reduce 67%
14/10/25 02:15:51 INFO mapreduce.Job: map 100% reduce 75%
14/10/25 02:15:54 INFO mapreduce.Job: map 100% reduce 86%
14/10/25 02:15:57 INFO mapreduce.Job: map 100% reduce 100%
14/10/25 02:15:58 INFO mapreduce.Job: Job job_1414218714201_0001 completed successfully
14/10/25 02:15:58 INFO mapreduce.Job: Counters: 50
    File System Counters
        FILE: Number of bytes read=329606679
        FILE: Number of bytes written=483139220
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=10400327756
        HDFS: Number of bytes written=83987906
        HDFS: Number of read operations=237
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Killed map tasks=1
        Launched map tasks=79
        Launched reduce tasks=1
        Data-local map tasks=79
        Total time spent by all maps in occupied slots (ms)=7043496

```

- Shows 10Gb end and it takes 20 min for 1-node

```

ubuntu@ip-172-31-30-241: ~
ubuntu@ip-172-31-30-241: ~ 162x20
Graph this data and manage this system at:
https://landscape.canonical.com/
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
40 packages can be updated.
20 updates are security updates.

Last login: Sat Oct 25 09:47:12 2014 from 208-59-152-225.c3-0.mcm-ubr1.chi-mcm.il.cable.rcn.com
Agent pid 3457
Identity added: last.pem (last.pem)
ubuntu@ip-172-31-30-241:~$ jps
3473 Jps
1475 NameNode
1699 SecondaryNameNode
1848 ResourceManager
ubuntu@ip-172-31-30-241:~$ 

ubuntu@ip-172-31-17-6: ~ 40x20
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
40 packages can be updated.
20 updates are security updates.

Last login: Sat Oct 25 09:53:27 2014 from 208-59-152-225.c3-0.mcm-ubr1.chi-mcm.il.cable.rcn.com
Agent pid 2056
Identity added: last.pem (last.pem)
ubuntu@ip-172-31-17-6:~$ jps
1854 NodeManager
2058 Jps
1274 DataNode
ubuntu@ip-172-31-17-6:~$ 

ubuntu@ip-172-31-17-7: ~ 39x20
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
40 packages can be updated.
20 updates are security updates.

Last login: Sat Oct 25 09:54:29 2014 from 208-59-152-225.c3-0.mcm-ubr1.chi-mcm.il.cable.rcn.com
Agent pid 2041
Identity added: last.pem (last.pem)
ubuntu@ip-172-31-17-7:~$ jps
2043 Jps
1841 NodeManager
1272 DataNode
ubuntu@ip-172-31-17-7:~$ 

ubuntu@ip-172-31-17-91: ~ 40x20
https://landscape.canonical.com/
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
40 packages can be updated.
20 updates are security updates.

Last login: Sat Oct 25 11:01:43 2014 from master
Agent pid 2009
Identity added: last.pem (last.pem)
ubuntu@ip-172-31-17-91:~$ jps
2011 Jps
1655 DataNode
1792 NodeManager
ubuntu@ip-172-31-17-91:~$ 

ubuntu@ip-172-31-17-92: ~ 39x20
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
40 packages can be updated.
20 updates are security updates.

Last login: Sat Oct 25 10:15:26 2014 from 208-59-152-225.c3-0.mcm-ubr1.chi-mcm.il.cable.rcn.com
Agent pid 1927
Identity added: last.pem (last.pem)
ubuntu@ip-172-31-17-92:~$ jps
1710 NodeManager
1573 DataNode
1929 Jps
ubuntu@ip-172-31-17-92:~$ 

```

- Shows for 5 node in Hadoop configure is started.

```

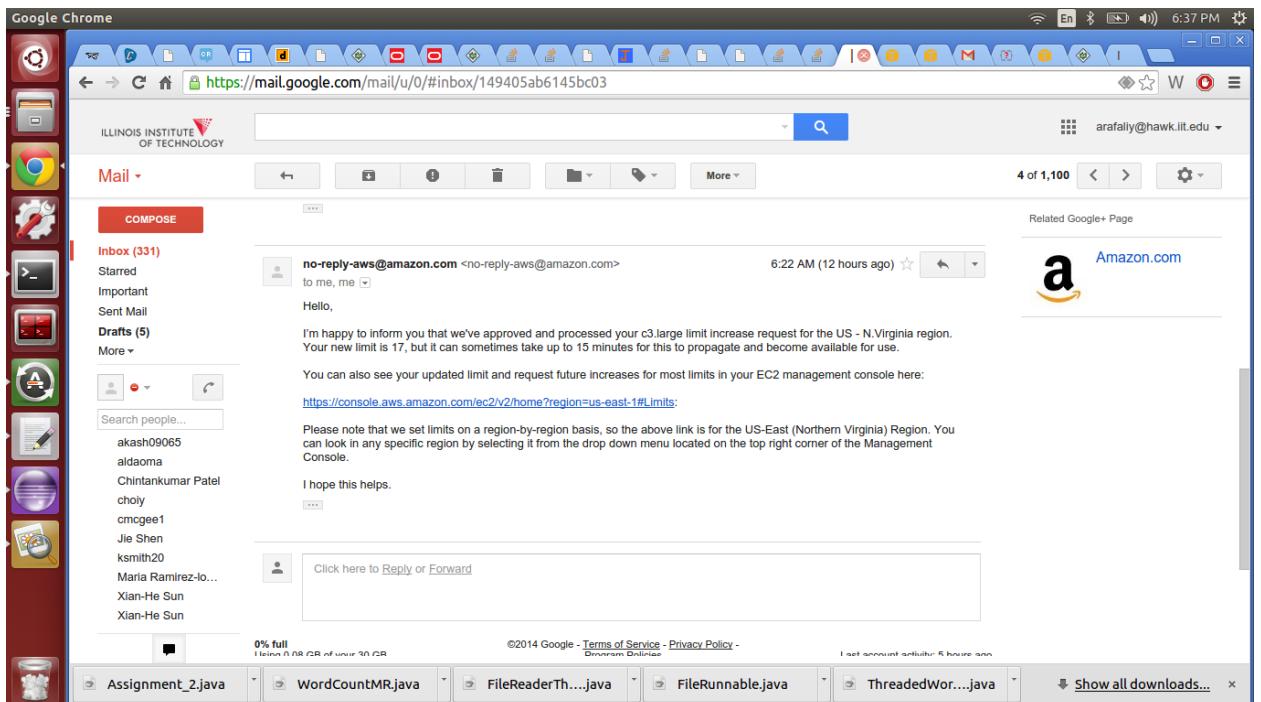
ubuntu@ip-172-31-30-241: ~/hadoop-2.5.1
14/10/25 11:16:24 INFO namenode.NameNode: SHUTDOWN_MSG:
*****Shutdown Message***** 
SHUTDOWN_MSG: Shutting down NameNode at ip-172-31-30-241/172.31.30.241
ubuntu@ip-172-31-30-241:~/hadoop-2.5.1 bin/hdfs dfs -mkdir /user
mkdir: '/user': File exists
ubuntu@ip-172-31-30-241:~/hadoop-2.5.1 bin/hdfs dfs -mkdir /user/akash
mkdir: '/user/akash': File exists
ubuntu@ip-172-31-30-241:~/hadoop-2.5.1 bin/hdfs dfs -put LICENSE.txt
ubuntu@ip-172-31-30-241:~/hadoop jar share/hadoop/mapreduce-examples-2.5.1.jar grep LICENSE.txt ot 'a[a-z]+'
14/10/25 11:19:17 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
14/10/25 11:19:17 WARN mapreduce.JobSubmitter: No job jar file set. User classes may not be found. See Job or Job#setJar(String).
14/10/25 11:19:17 INFO InputFileInputFormat: Total input paths to process : 1
14/10/25 11:19:17 INFO mapreduce.JobSubmitter: number of splits:1
14/10/25 11:19:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1414230679399_0003
14/10/25 11:19:18 INFO mapred.YARNRunner: Job jar is not present. Not adding any jar to the list of resources.
14/10/25 11:19:18 INFO impl.YarnClientImpl: Submitted application application_1414230679399_0003
14/10/25 11:19:18 INFO mapreduce.Job: The url to track the job: http://ip-172-31-30-241:8088/proxy/application_1414230679399_0003/
14/10/25 11:19:18 INFO mapreduce.Job: Running job: job_1414230679399_0003

[...]

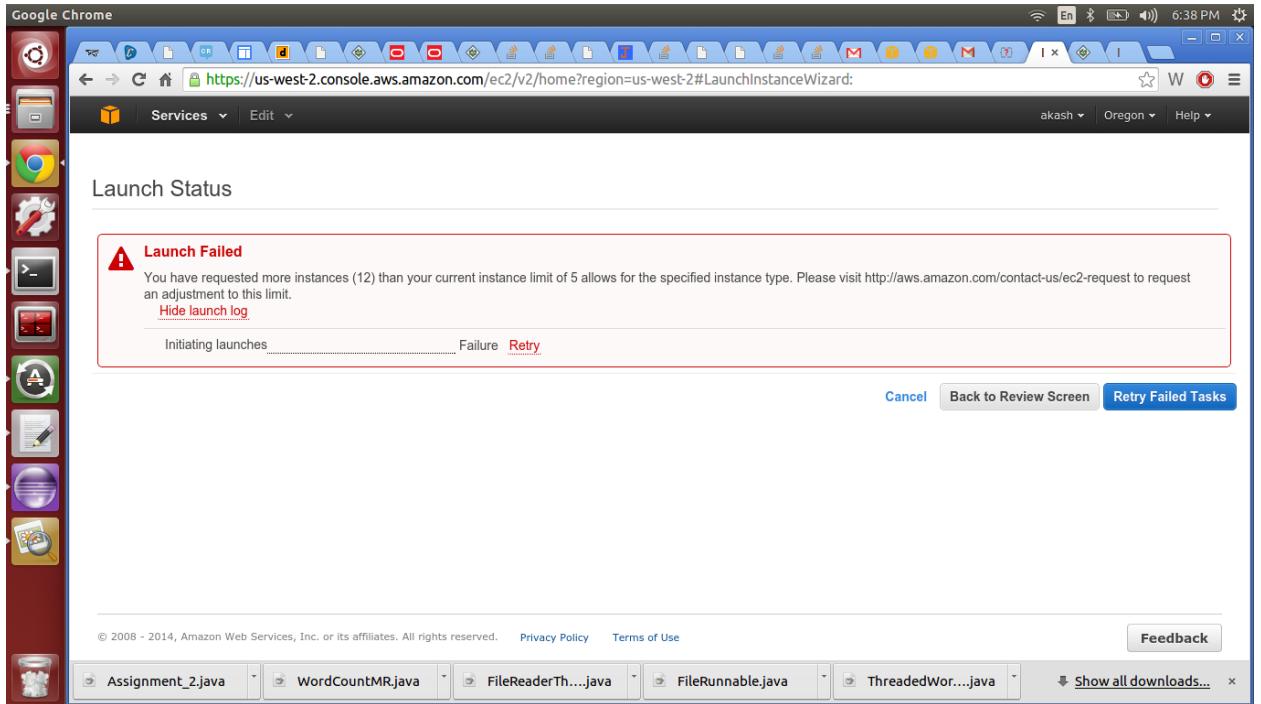
```

The screenshot shows a desktop environment with several open windows. At the top, there's a terminal window with a log of Hadoop commands. Below it are four smaller terminal windows showing the output of `apt-get update` and `apt-get upgrade` commands, indicating 40 packages can be updated and 20 are security updates. A file manager window is also visible on the left.

- Word count program is running for 10 GB.



- Here we got the permission for extra node upto 17.



- But when we try to launch it, it shows the error as shown in above snapshot.

## Swift :

### Installation:

#### On Local Machine steps:

1. Download the file from <http://swiftlang.org/packages/swift-0.95-RC6.tar.gz>
2. Extract by running

```
$ tar xfz swift-0.95-RC6.tar.gz
```

3. Add to PATH by running

```
$ export PATH=$PATH:/home/chintan/Downloads/swift-0.95-RC6/bin
```

4. Install java jdk

```
$ sudo apt-get install openjdk-7-jdk
```

5. Download Swift tutorial

```
$ wget http://swiftlang.org/tutorials/localhost/swift-localhost-tutorial.tar.gz
```

## 6. Extract by Running

```
$ tar xvzf swift-localhost-tutorial.tar.gz
```

7.       \$ cd *swift-localhost-tutorial*

8. Run tutorial setup script

```
$ source setup.sh # You must run this with "source" !
```

## Cloud Installation steps:

1. Create Group in AWS       //chintan
2. Create User in AWS       //security1
3. Assign User to Group
4. Download credentials.csv
5. Launch instance of EC2

Instance AMI ID : ubuntu-trusty-14.04-amd64-server-20140926 (ami-c9d497f9)

Key pair name: 1234

Instance ID: i-f809cef2

Instance type: c3.large

Availability zone: us-west-2a

6. Connect instance from terminal

```
$ ssh -i 1234.pem ubuntu@54.186.69.98
```

7.       \$ exit  
to logout from instance

8. Copy your credentials.csv and 1234.pem to instance

```
$ scp -i 1234.pem ./1234.pem ubuntu@54.186.69.98:  
$ scp -i 1234.pem ./credentials.csv ubuntu@54.186.69.98:
```

## 9. Setting up launchpad instance:

- Install python and libcloud library. Since we are using an Ubuntu instance:

```
$ sudo apt-get install python python-pip
```

```
$ sudo pip install apache-libcloud
```

- Get the cloud-tutorials repository from git
- First Install git using

```
$ sudo apt-get install git
```

```
$ git clone https://github.com/yadudoc/cloud-tutorials.git
```

- Go to ec2 directory to setup cluster
- Follow steps of local installation here.

## 10. Manage Cloud resources:

Update the file *configs* in the cloned repository from the previous step, with the following information:

- AWS\_CREDENTIALS\_FILE : /home/ubuntu/credentials.csv
- AWS\_WORKER\_COUNT : 16
- AWS\_KEYPAIR\_NAME : 1234
- AWS\_KEYPAIR\_FILE : /home/ubuntu/1234.pem
- AWS\_USERNAME : Ubuntu
- AWS\_REGION : Default=us-west-2 | Do NOT change
- SECURITY\_GROUP : Default=swift\_security\_group1
- HEADNODE\_IMAGE, WORKER\_IMAGE : ami-c9d497f9
- HEADNODE\_MACHINE\_TYPE, WORKER\_MACHINE\_TYPE : t1.micro

## 11. Now run setup.sh to create cludter of 16 node.

Must source the setup script.

```
$ source setup.sh
```

12. Check AWS management console for the status of the cluster. If every instance is ready (i.e. shows "2/2 ...."), then enter the command

```
$ connect headnode",
```

and you'll ssh to the headnode.

13. We will get two folders “cloud-tutorial and :s3fs-fuse” by `$ ls` at headnode.

14. Here “cloud-tutorials” is read-only. Use below command to get access on it.

```
$ cp -r cloud-tutorials/swift-cloud-tutorial ./
```

It will create copy swift-cloud-tutorial from cloud-tutorial and change read-write permission.

- Then, do
- It shows like below output

```
Swift version is Swift 0.95 RC6 swift-r7900 cog-r3908
```

15. Go to wcount/

- Run `$ swift wordcount.swift`
- It's a parallel swift program which distributes sorting job to workers.
- Folder structure of wcount
  - `wordcount.swift` : swift program
  - `wordcount.sh` : bash script to count words and sort it counted words in alphabetical order
  - `merge.sh` : bash script to get and merge output files from workernodes
  - `swift.conf` : swift configuration for cluster
  - `inputs/` : Directory for input files
  - `outputs/` : Directory for output files

16. Now for 10Gb dataset

## 17. Create new volume at ec2 console

Volume: vol-d334c1dc			
Description	Status Checks	Monitoring	Tags
Volume ID	vol-d334c1dc	Alarm status	None
Size	50 GiB	Snapshot	-
Created	October 24, 2014 9:53:14 PM UTC-5	Availability Zone	us-west-2c
State	in-use	Attachment information	i-39eda736 (headnode):/dev/sdf (attached)
Volume type	gp2	IOPS	150 / 3000
Product codes	-	Encrypted	Not Encrypted

Attach it to head node

## 18. Now Use `$ lsblk` command to view your available disk devices and their mount points (if applicable) to help you determine the correct device name to use.

- \$ lsblk
- NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
- xvdf 202:80 0 50G 0 disk
- xvda1 202:1 0 8G 0 disk /
- Mount new volume in headnode using below sequence of commands :

```
$ sudo mkfs -t ext4 /dev/xvdf //create filesystem at disk  
$ sudo mkdir data  
$ sudo mount /dev/xvdf data // it will mount disk at data directory in headnode
```

## 19. Now put wcount directory in newly mounted drive

## 20. Download 10Gb dataset in inputs directory

```
$ wget https://s3.amazonaws.com/cs-553/wiki10gb.xz  
$ xz -d wiki10gb.xz
```

## 21. Now we have done split this 10gb file using below bash command instead of inserting into swift program. Because it takes more time to split.

*\$ split -b 1000000000 wiki10gb output-  
It will split file into 1GB parts*

22. Now run swift program you will get one merged output file and other parts which are created by workernodes.
23. Download output files to local machine using below command

```
chintan@ubuntu:~/Desktop$ scp -i 1234.pem  
ubuntu@54.191.35.109:/home/ubuntu/data/swift-cloud-  
tutorial/wcount/output/final_output.out  
/home/chintan/Desktop
```

## Problems faced during installation :

While running setup.sh from ec2 folder get authentication error as below screen shot.

-- Solved by creating group in aws and assign user to this group

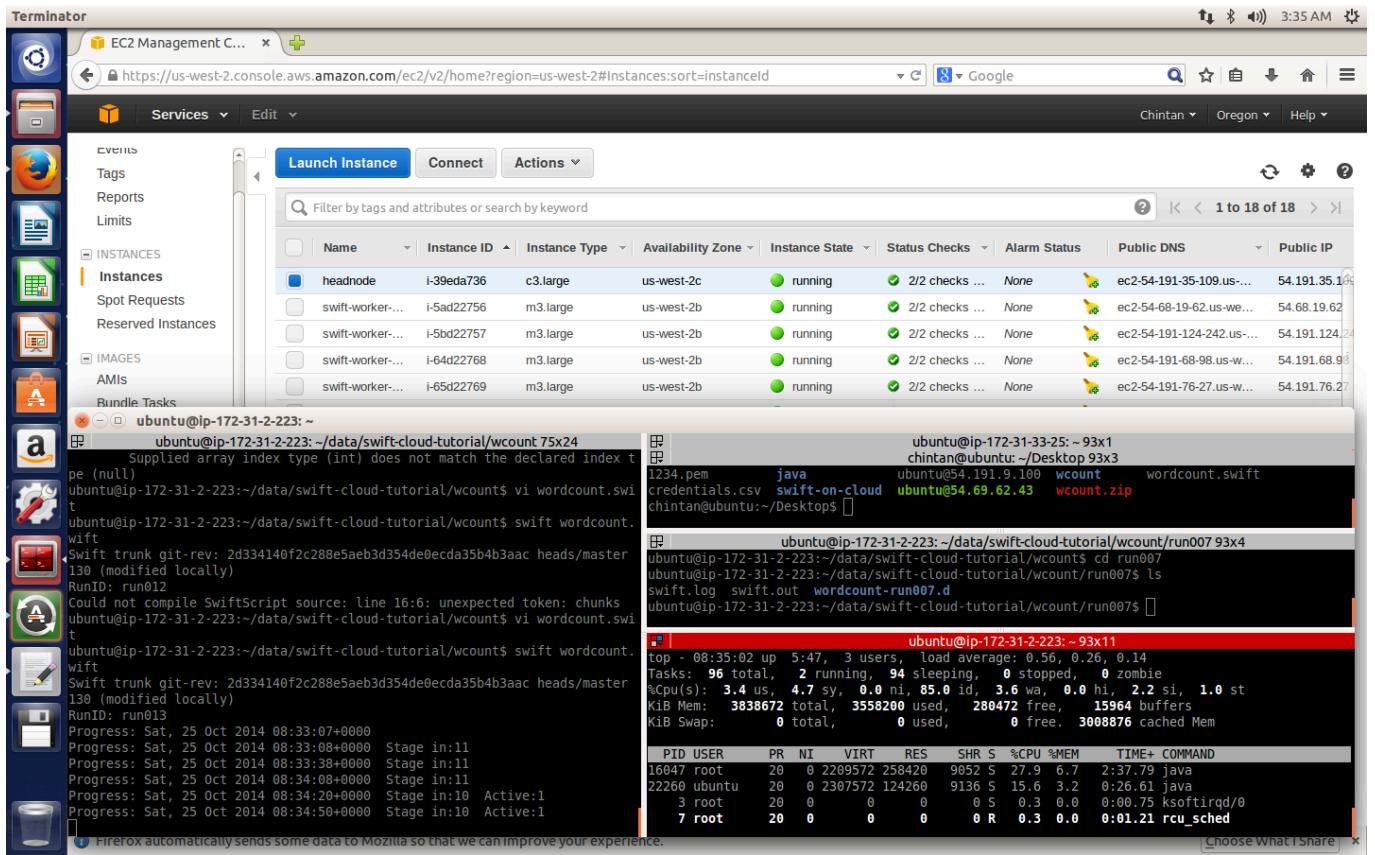
```
Current workers    : 0
Workers requested : 16
Starting worker(s) swift-worker-000 swift-worker-001 swift-worker-002 swift-work
  swift-worker-004 swift-worker-005 swift-worker-006 swift-worker-007 swift-worke
swift-worker-009 swift-worker-010 swift-worker-011 swift-worker-012 swift-worker
swift-worker-014 swift-worker-015
Traceback (most recent call last):
  File "./aws.py", line 241, in <module>
    configs, driver = init()
  File "./aws.py", line 223, in init
    aws_create_security_group(ec2_driver, configs)
  File "./aws.py", line 56, in aws_create_security_group
    current      = driver.ex_list_security_groups()
  File "/usr/local/lib/python2.7/dist-packages/libcloud/compute/drivers/ec2.py",
2719, in ex_list_security_groups
    response = self.connection.request(self.path, params=params).object
  File "/usr/local/lib/python2.7/dist-packages/libcloud/common/base.py", line 68
request
    response = responseCls(**kwargs)
  File "/usr/local/lib/python2.7/dist-packages/libcloud/common/base.py", line 11
init
    raise Exception(self.parse_error())
Exception: UnauthorizedOperation: You are not authorized to perform this operati
Traceback (most recent call last):
  File "./aws.py", line 241, in <module>
    configs, driver = init()
  File "./aws.py", line 223, in init
    aws_create_security_group(ec2_driver, configs)
  File "./aws.py", line 56, in aws_create_security_group
    current      = driver.ex_list_security_groups()
```

→ After connect to head node can't able to run swift program

Error because of directory is read-only.

-- solved using command : \$ cp -r cloud-tutorials/swift-cloud-tutorial

## Snap-shots :



## 1.1 AWS console and running swift program for 10gb in headnode



```

ubuntu@ip-172-31-33-25: ~/cloud-tutorials/ec2
ubuntu@ip-172-31-33-25: ~/cloud-tutorials/ec2$ ./list_instances.sh
NAME          | STATUS   | EXTERNAL_IP
launchpad     | RUNNING  | 54.186.214.90
swift-worker-014 | RUNNING  | 54.191.124.242
swift-worker-002 | RUNNING  | 54.191.71.244
swift-worker-008 | RUNNING  | 54.191.84.13
swift-worker-005 | RUNNING  | 54.191.78.167
headnode       | RUNNING  | 54.191.35.109
swift-worker-011 | RUNNING  | 54.68.192.83
swift-worker-004 | RUNNING  | 54.191.76.27
swift-worker-012 | RUNNING  | 54.69.163.160
swift-worker-013 | RUNNING  | 54.191.124.32
swift-worker-009 | RUNNING  | 54.187.123.87
swift-worker-007 | RUNNING  | 54.191.86.2
swift-worker-015 | RUNNING  | 54.68.19.62
swift-worker-000 | RUNNING  | 54.68.111.94
swift-worker-003 | RUNNING  | 54.191.68.98
swift-worker-001 | RUNNING  | 54.191.70.197
swift-worker-006 | RUNNING  | 54.191.78.21
swift-worker-010 | RUNNING  | 54.191.94.17
ubuntu@ip-172-31-33-25:~/cloud-tutorials/ec2$ stop_n_workers 4
Stopping 4 instances :
swift-worker-003
swift-worker-001
swift-worker-006
swift-worker-010
Deleting node : swift-worker-003
Deleting node : swift-worker-001
Deleting node : swift-worker-006
Deleting node : swift-worker-010
ubuntu@ip-172-31-33-25:~/cloud-tutorials/ec2$ 

```

```

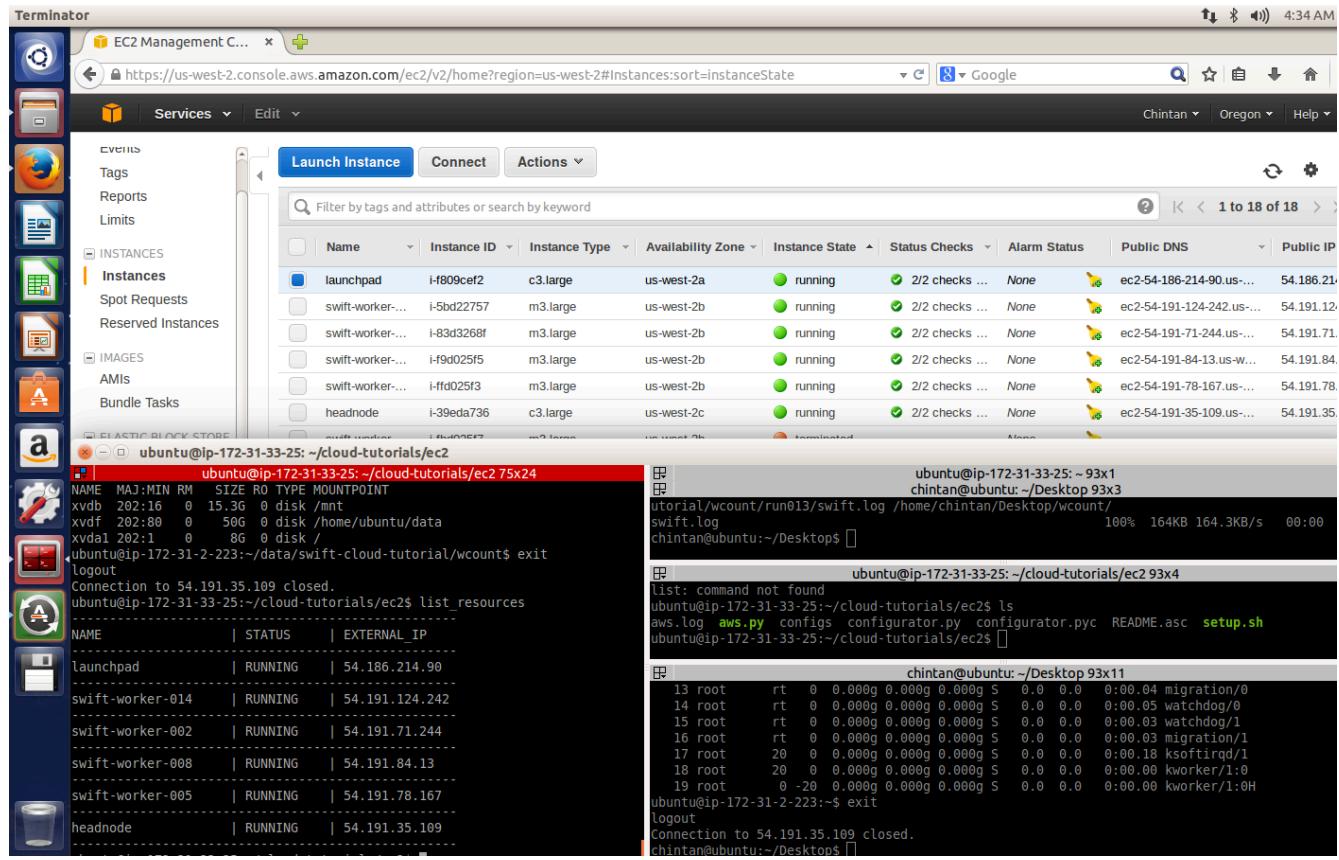
ubuntu@ip-172-31-33-25: ~ 94x1
chintan@ubuntu:~/Desktop 94x9
1234.pem      java      ubuntu@54.191.9.100  wcount      wordcount.swift
credentials.csv swift-on-cloud  ubuntu@54.69.62.43  wcount.zip
chintan@ubuntu:~/Desktop$ 

ubuntu@ip-172-31-2-223: ~/data/swift-cloud-tutorial/wcount/run007 94x11
ubuntu@ip-172-31-2-223:~/data/swift-cloud-tutorial/wcount$ cd run007
ubuntu@ip-172-31-2-223:~/data/swift-cloud-tutorial/wcount/run007$ ls
swift.log  swift.out  wordcount-run007.d
ubuntu@ip-172-31-2-223:~/data/swift-cloud-tutorial/wcount/run007$ 

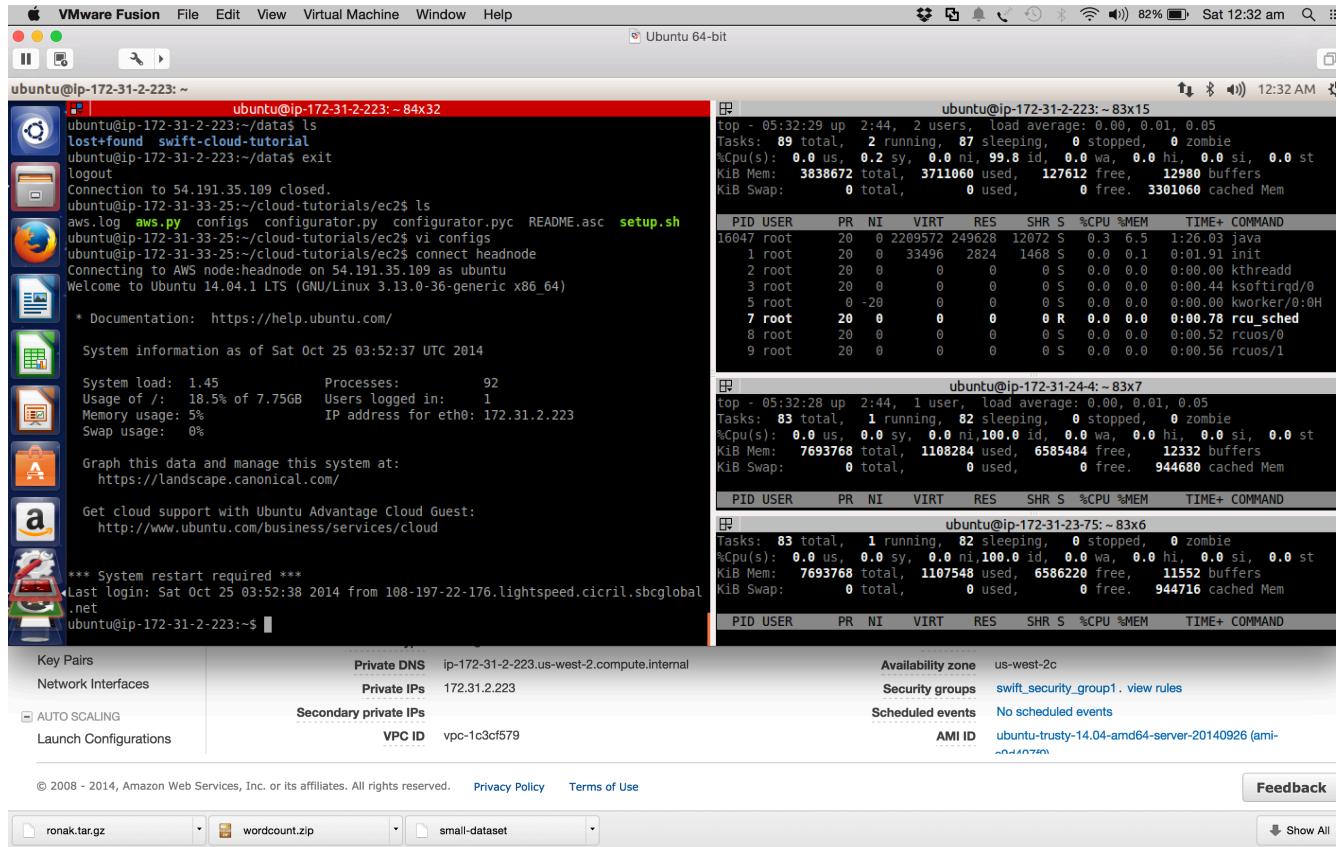
ubuntu@ip-172-31-2-223: ~ 94x25
top - 08:44:18 up 5:56, 2 users, load average: 0.01, 0.07, 0.11
KiB Mem: 3838672 total, 3683192 used, 155488 free, 15276 buffers
KiB Swap: 0 total, 0 used, 0 free. 3303192 cached Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
22363 ubuntu 20 0 23.0m 1.6m 1.1m R 0.3 0.0 0:00.62 top

```

## 1.2 All list\_resources



### 1.3 Headnode after mount 50Gb disk (lsblk command)



## 1.4 Connect headnode with it's IP in AWS console

The screenshot shows a VMware Fusion window running an Ubuntu 64-bit guest OS. The terminal window contains several configuration scripts and system monitoring output.

**Configuration Scripts (Terminal Output):**

- Setting AWS credentials file:

```
ubuntu@ip-172-31-33-25:~/cloud-tutorials/ec2
ubuntu@ip-172-31-33-25:~/cloud-tutorials/ec2$4x32
Set following config with an absolute path to the credentials file
AWS_CREDENTIALS_FILE=/home/ubuntu/credentials.csv
```

- Setting worker count:

```
# Set the number of worker nodes required (limited by your account quotas)
AWS_WORKER_COUNT=16
```

- Setting keypair name:

```
# Set the name for the keypair used to ssh to instances
# A key will be generated in path if none are found
AWS_KEYPAIR_NAME=1234
```

- Setting keypair file:

```
# Set following config with an absolute path to where you want the keypair file
AWS_KEYPAIR_FILE=/home/ubuntu/1234.pem
```

- Setting default user:

```
#ubuntu is the default user on Ubuntu nodes
AWS_USERNAME=ubuntu
```

- Setting Headnode image:

```
# Ubuntu Server 14.04 LTS (PV), SSD Volume Type - ami-c9d497f9 (64-bit)
HEADNODE_IMAGE=ami-c9d497f9
```

- Setting Headnode instance type:

```
# Headnode instance type, refer to Amazon instance types page for reference
# Legal machine types : t1.micro | m1.small | m1.medium | m1.large
HEADNODE_MACHINE_TYPE=c3.large
```

- Setting Worker image:

```
# Ubuntu Server 14.04 LTS (PV), SSD Volume Type - ami-c9d497f9 (64-bit)
WORKER_IMAGE=ami-c9d497f9
```

- Setting Worker instance type:

```
# Headnode instance type, refer to Amazon instance types page for reference
# Legal machine types : t1.micro | m1.small | m1.medium | m1.large
WORKER_MACHINE_TYPE=m3.large
```

- Setting Worker init script:

```
# The following allows a script to executed on the worker at boot time
# WORKER_INIT_SCRIPT=foo.sh
# Do not change following parameters
```

- Setting AWS Region:

```
AWS_REGION=us-west-2
```

- Configuring "configs" directory:

```
"configs" 34L, 1273C
```

**System Monitoring (top Command Output):**

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
16047	root	20	0	2209572	249628	12872	S	0.3	6.5	1:25.96 java
17396	ubuntu	20	0	23540	1568	1148	R	0.3	0.0	0:05.78 top
1	root	20	0	33496	2824	1468	S	0.0	0.1	0:01.91 init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00 kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.44 ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 kworker/0:0H
7	root	20	0	0	0	0	R	0.0	0.0	0:00.78 rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.52 rcuos/0

**Amazon CloudWatch Metrics (Metrics Tab):**

Key Pairs	Private DNS	ip-172-31-2-223.us-west-2.compute.internal	Availability zone	us-west-2c
Network Interfaces	Private IPs	172.31.2.223	Security groups	swift_security_group1. view rules
Auto Scaling	Secondary private IPs		Scheduled events	No scheduled events
Launch Configurations	VPC ID	vpc-1c3cf579	AMI ID	ubuntu-trusty-14.04-amd64-server-20140926 (ami- and so on)

**Bottom Navigation Bar:**

- Feedback
- Show All

## 1.5 configs file in ec2

MPI

## **Installation and Problems faced during installation :**

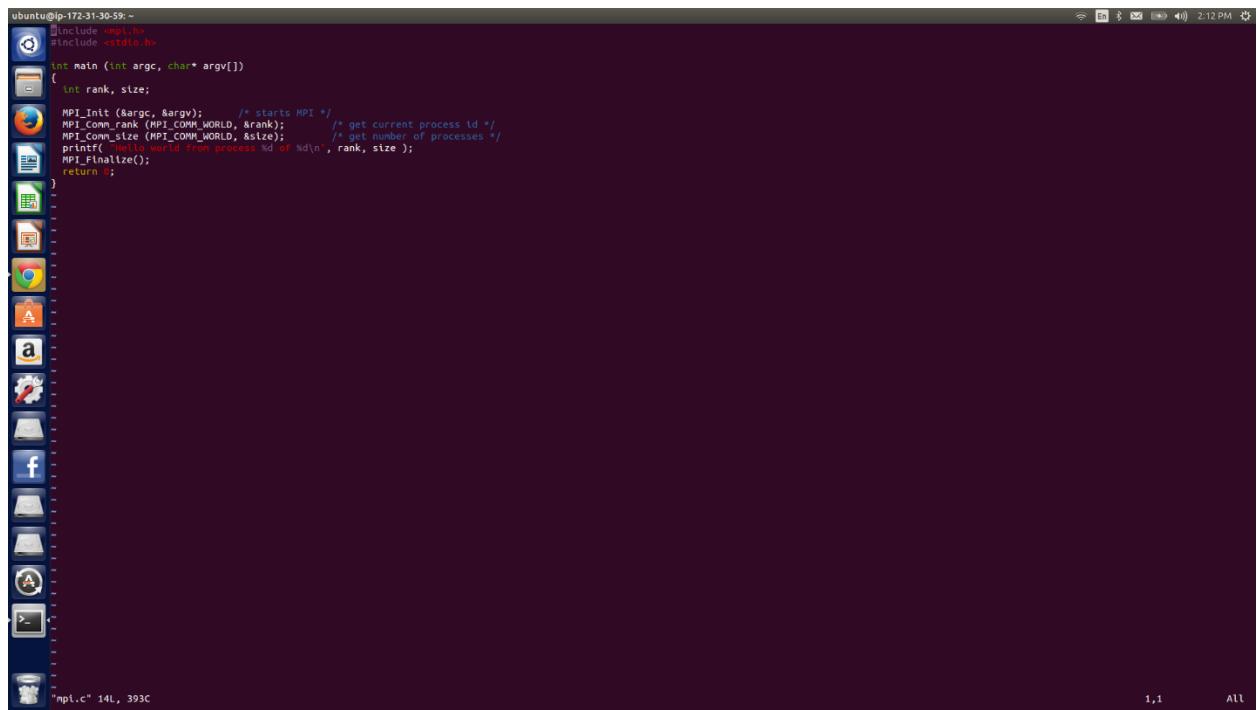
- To configure the MPI we have used some reference links, first we tried to configure on local machine and it ran perfectly for a simple hello-world program.[4]
  - When we tried to run on cloud, we have to give AWS\_ACCESS\_KEY, AWS\_SECRET\_KEY and also USER\_ID and key-pair location.[4] When I tried to configure it on cloud and when I started star-cluster it shows error with some location not found for key pair. So, we tried to run MPI program another way.
  - We used [1] to configure on Ubuntu. So at first we'd created a cluster on amazon ec2 for a single node, on cluster we installed mpich2 using command:

```
sudo apt-get install libcr-dev mpich2 mpich2-doc
```

- Then we had created mpi.c file using vi editor which has MPI code to print “hello-world” on amazon cloud.[1]
- Once we created mpi.c, we compile that file using command:  
**mpicc mpi.c -o m1**
- To run that program using command:  
**mpirun -np 16 ./m1**
- Then we tried to do sorting of word for MPI program.[3] But we found some problem to with character array.
- Hence we tried to solve using link [2], but due to less time we didn’t figure out how to define character array. So right now we are submitting MPI-hello-world program for single node.

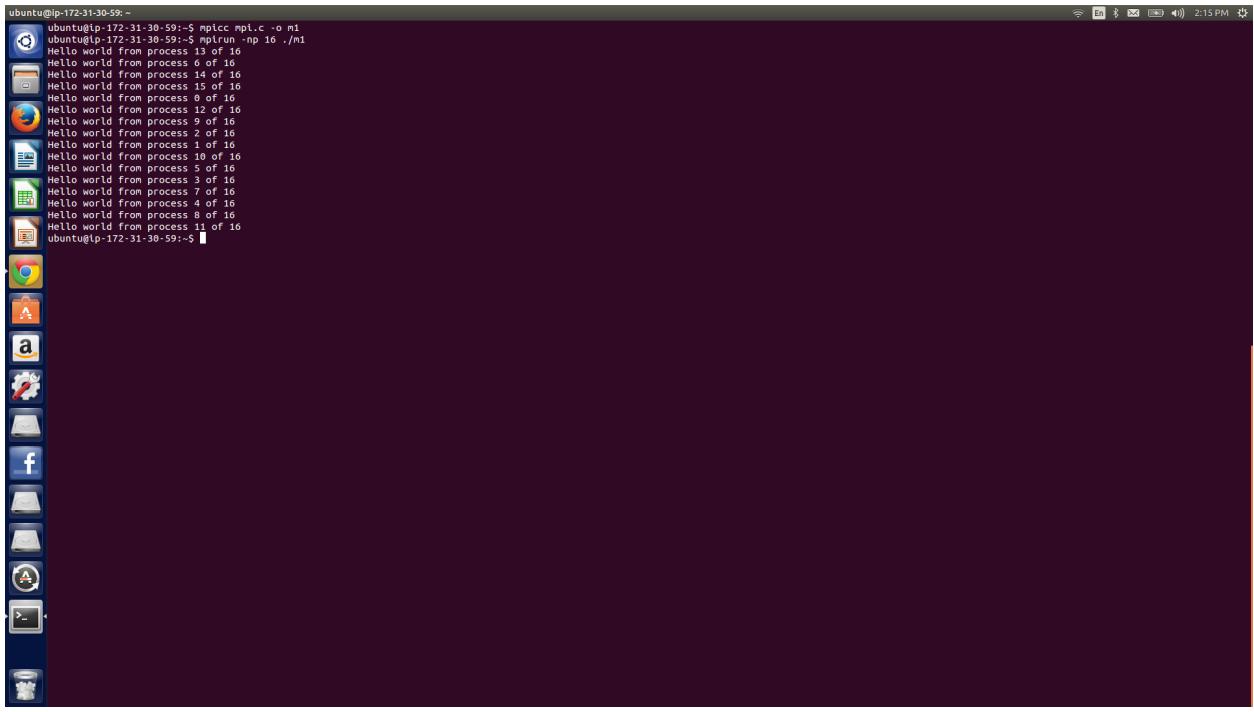
### **Screen-shots:**

- The below given is the snap-shot of mpi.c program in which I had written a code.



```
ubuntu@ip-172-31-30-59: ~
└─Include <mpi.h>
└─Include <stdio.h>
int main (int argc, char* argv[])
{
    int rank, size;
    MPI_Init (&argc, &argv);           /* starts MPI */
    MPI_Comm_rank (MPI_COMM_WORLD, &rank);      /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, &size);      /* get number of processes */
    printf( "Hello world from process %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
1,1     All
```

- And the below given snap-shot is the output of the given mpi.c program on cloud.

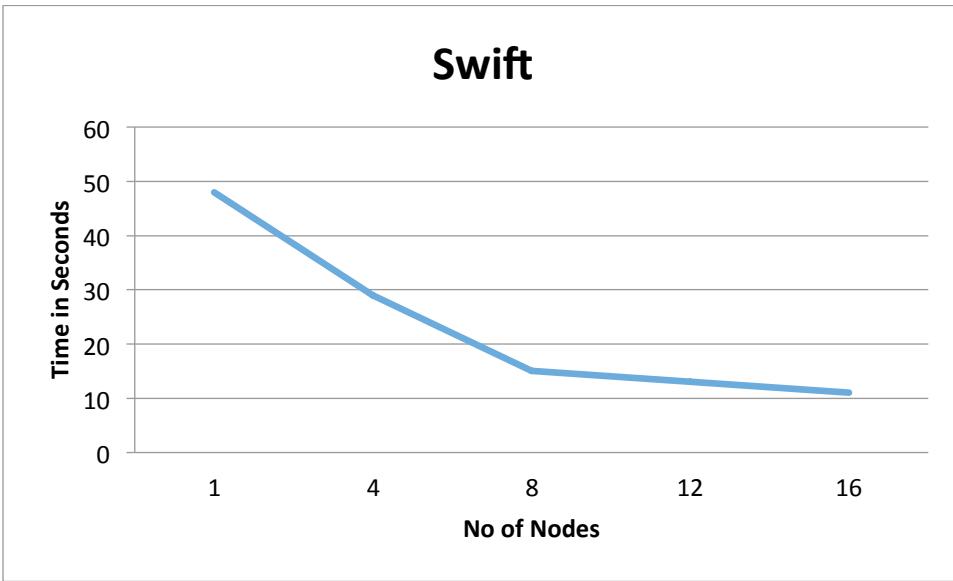
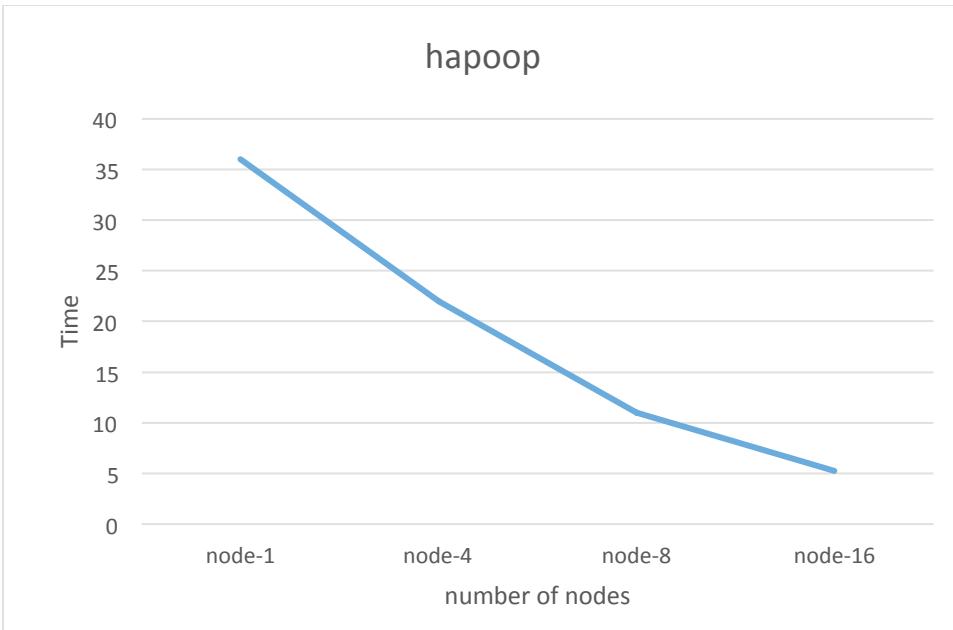


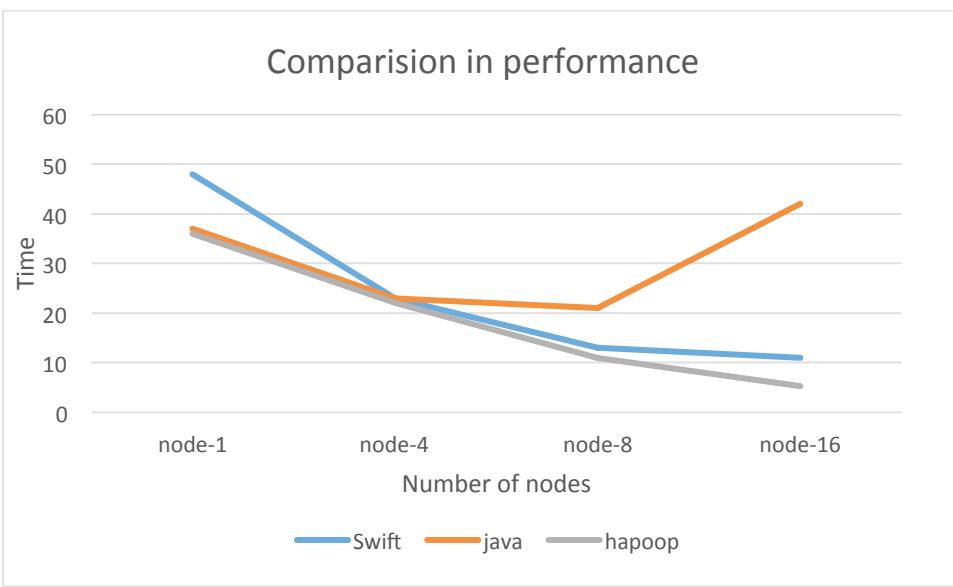
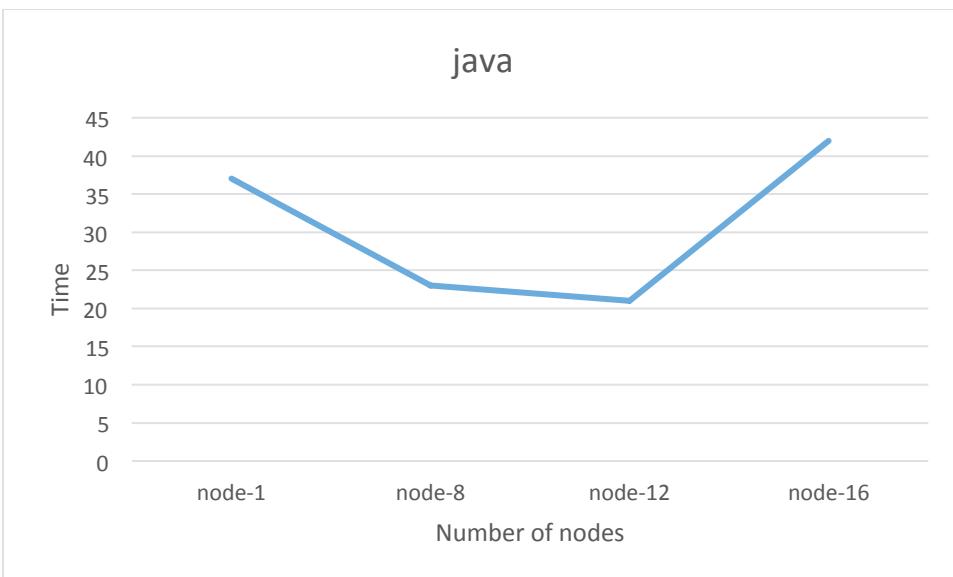
A screenshot of a Linux terminal window titled "ubuntu@ip-172-31-30-59: ~". The terminal displays the output of a MPI program. The command run was "mpicc mpi.c -o m1" followed by "mpicxx -o m2 m1" and then "mpirun -np 16 ./m1". The output shows 16 parallel processes each printing "Hello world from process [process\_id] of 16".

```
ubuntu@ip-172-31-30-59: ~
ubuntu@ip-172-31-30-59: ~ mpicc mpi.c -o m1
ubuntu@ip-172-31-30-59: ~ mpirun -np 16 ./m1
Hello world from process 13 of 16
Hello world from process 6 of 16
Hello world from process 14 of 16
Hello world from process 15 of 16
Hello world from process 9 of 16
Hello world from process 10 of 16
Hello world from process 11 of 16
Hello world from process 2 of 16
Hello world from process 1 of 16
Hello world from process 10 of 16
Hello world from process 5 of 16
Hello world from process 12 of 16
Hello world from process 7 of 16
Hello world from process 4 of 16
Hello world from process 8 of 16
Hello world from process 11 of 16
ubuntu@ip-172-31-30-59: ~
```

## Performance:

- Here, we have to calculate performance for Hadoop, Swift and Java.
- The first three graph shows the individual performance on each node of Hadoop, swift and java.
- And the forth graph shows the comparision which system is better.
- Here, for one node hadoop takes 36mins, while java takes 37mins and swift takes 47mins.
- And for 16 nodes Hadoop takes only 5.25 mins while swift takes 11 mins and java takes 42 mins.
- So Hadoop is better for both nodes 1 as well as 16.
- If we have to predict the performance from the given example than for 100 nodes than Hadoop would take the least time around **64 sec**.
- And for 1000 nodes Hadoop would take time around **70 ms** .





## **References**

### **Hadoop References :**

- [1] <http://wiki.apache.org/hadoop/HadoopJavaVersions>
- [2] <http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html>
- [3] <http://letsdobigdata.wordpress.com/>
- [4] [http://en.wikipedia.org/wiki/Apache\\_Hadoop](http://en.wikipedia.org/wiki/Apache_Hadoop)

### **Swift References:**

- [1] <http://swift-lang.org/main/>
- [2] [http://en.wikipedia.org/wiki/Swift\\_\(parallel\\_scripting\\_language\)](http://en.wikipedia.org/wiki/Swift_(parallel_scripting_language))
- [3] <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html>
- [4] [https://github.com/yadudoc/cloud-tutorials.git.](https://github.com/yadudoc/cloud-tutorials.git)

### **MPI References :**

- [1] <https://jetcracker.wordpress.com/2012/03/01/how-to-install-mpi-in-ubuntu/>
- [2] <http://condor.cc.ku.edu/~grobe/docs/intro-MPI-C.shtml>
- [3] <http://www.dailycancode.com/code/sort-words-604.aspx>
- [4] <http://mpitutorial.com/launching-an-amazon-ec2-mpi-cluster/>