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 programing 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 Programing(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 and 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.

$ bin/hdfs namenode -format

- To start all deamons we have to write command:

$sbin/start-all.sh

- Make the HDFS directories required to execute MapReduce jobs:

$ bin/hdfs dfs -mkdir /user
$ 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 :

$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:

$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.

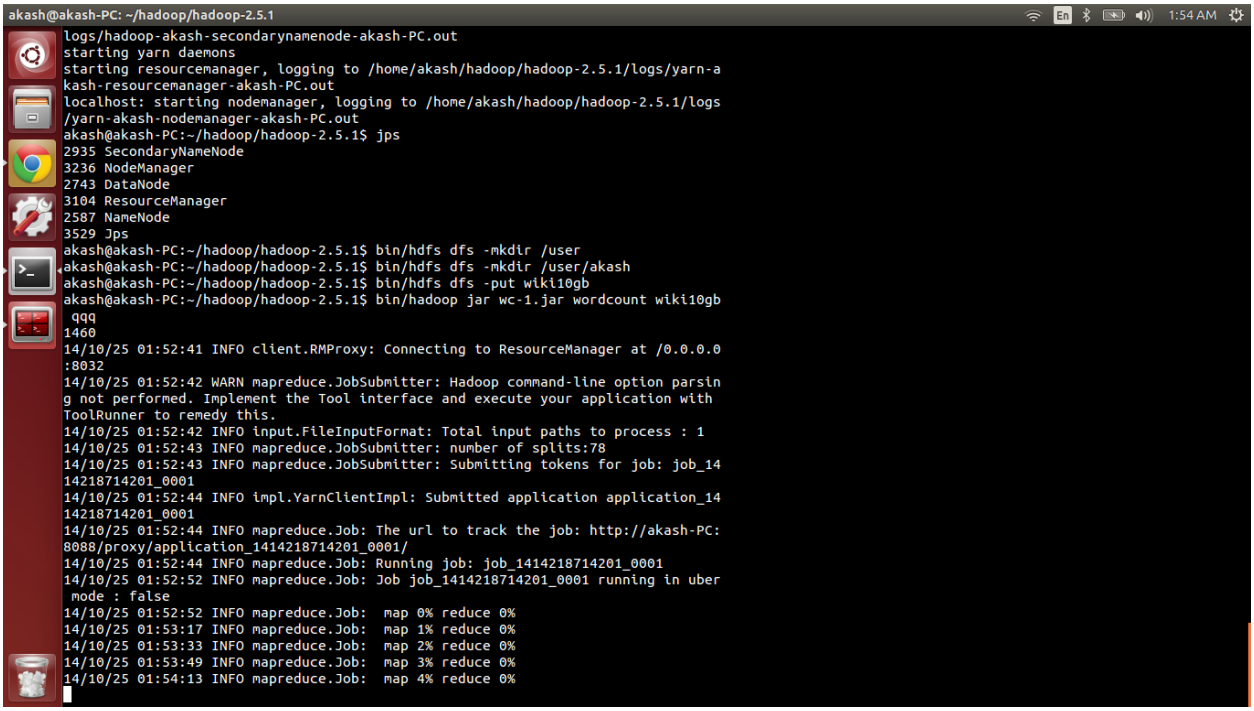
Resource Manager:

- 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.

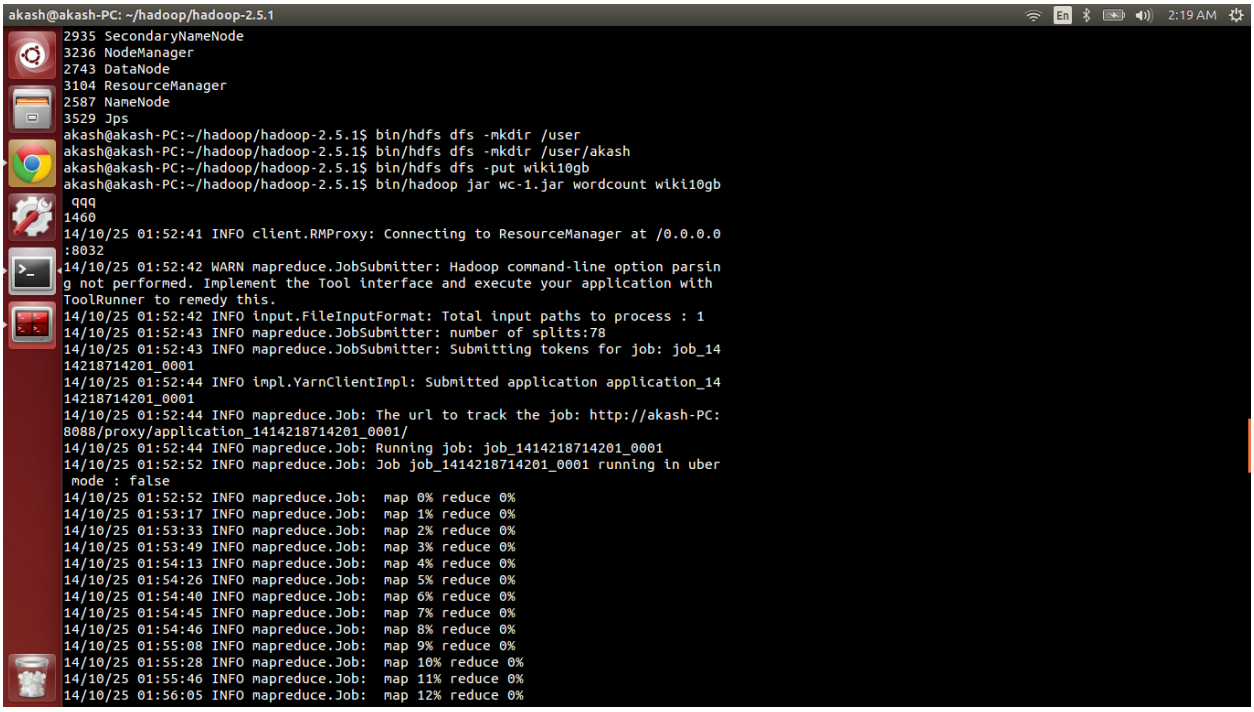
Problems :

- 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"

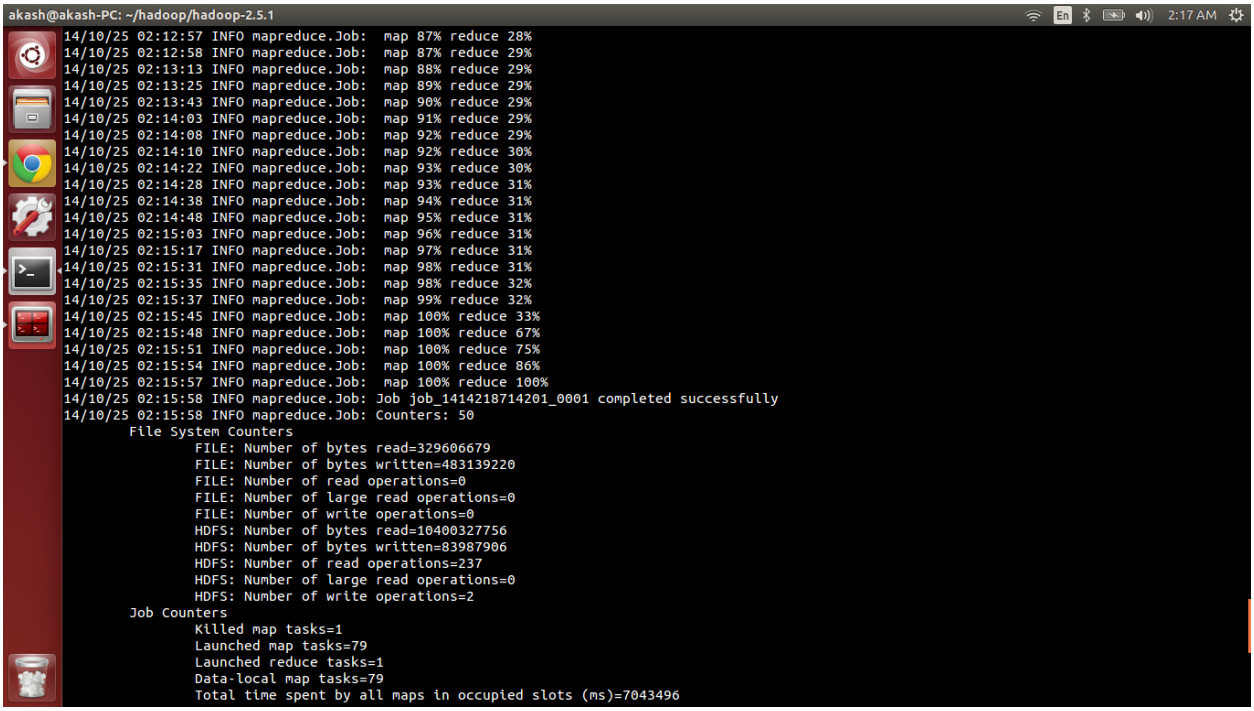
Snapshots :



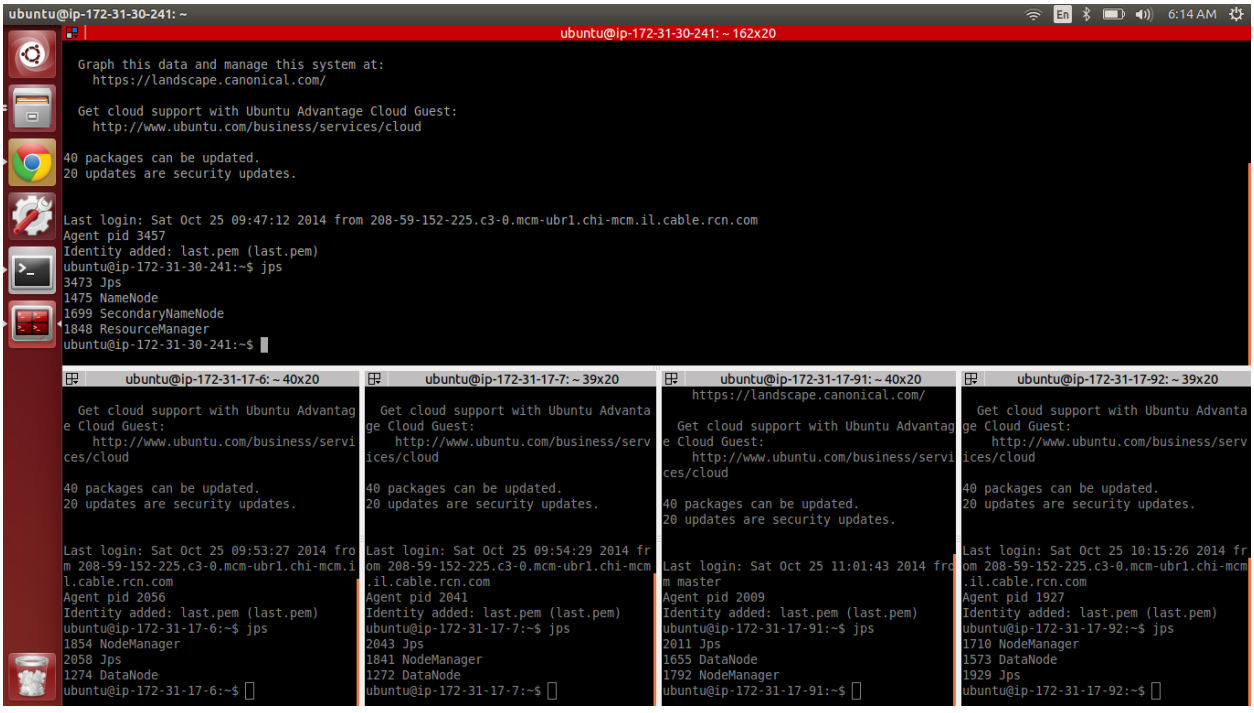
- Shows for 1-node Hadoop is working.



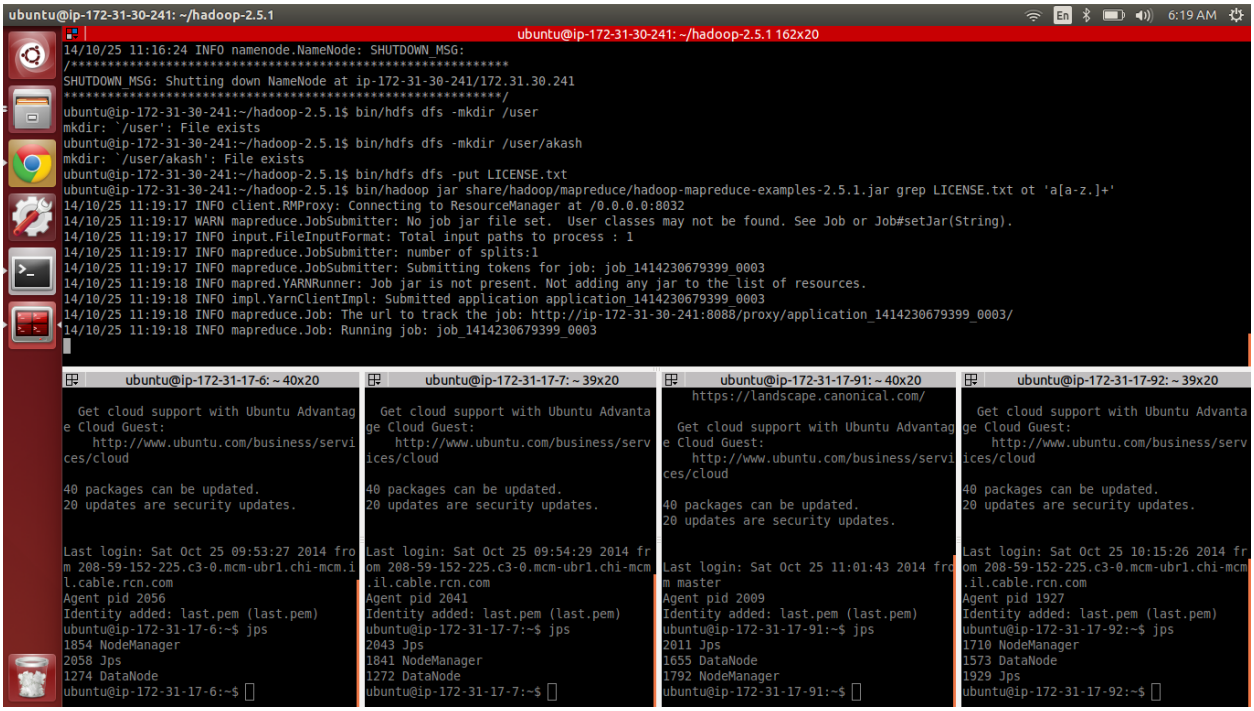
- Shows for 1-node 10 GB file in Hadoop is working



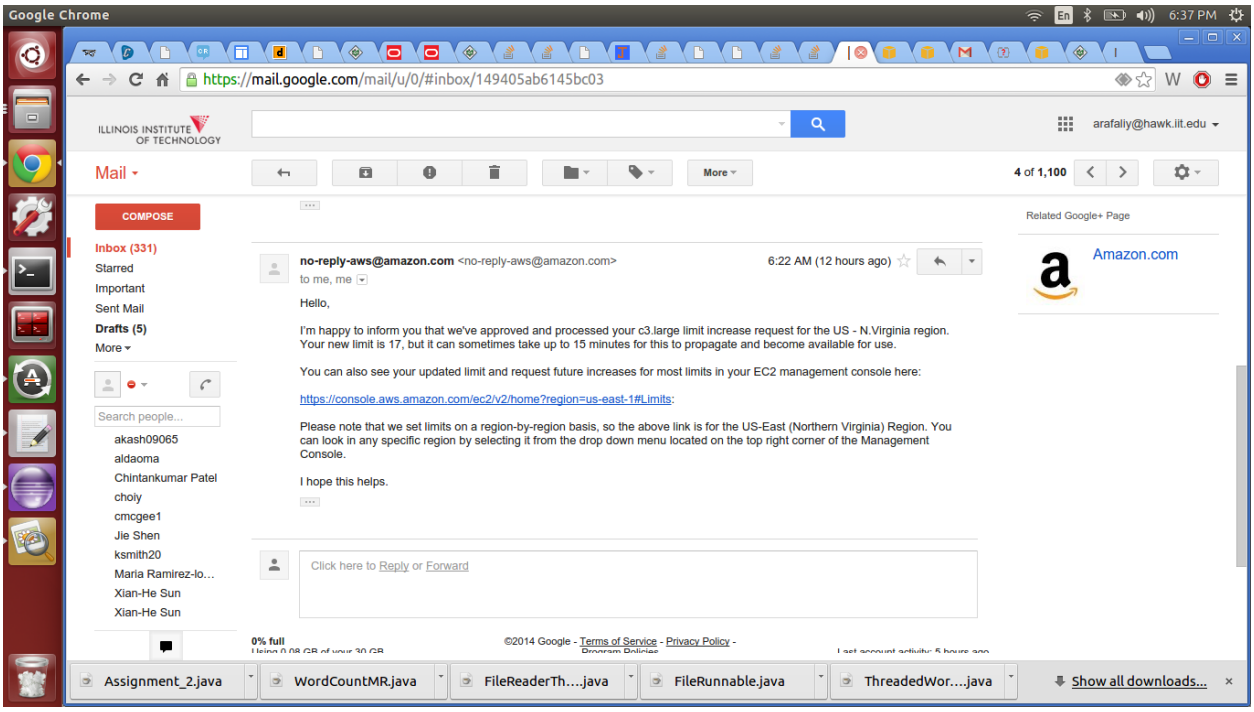
- Shows 10Gb end and it takes 20 min for 1-node



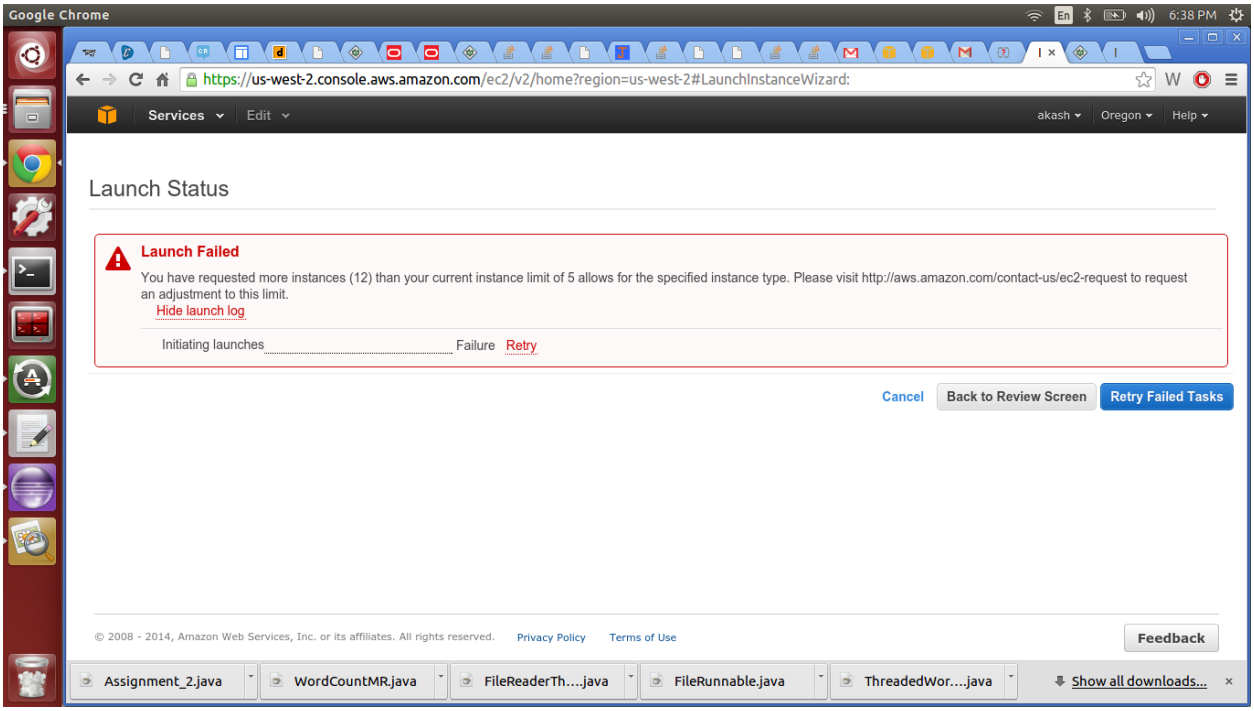
- Shows for 5 node in Hadoop configure is started.



- 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 snap-shot.

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 xvfz 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

        $ cd cloud-tutorials/ec2
    - 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

      $ source setup.sh
    - It shows like below output

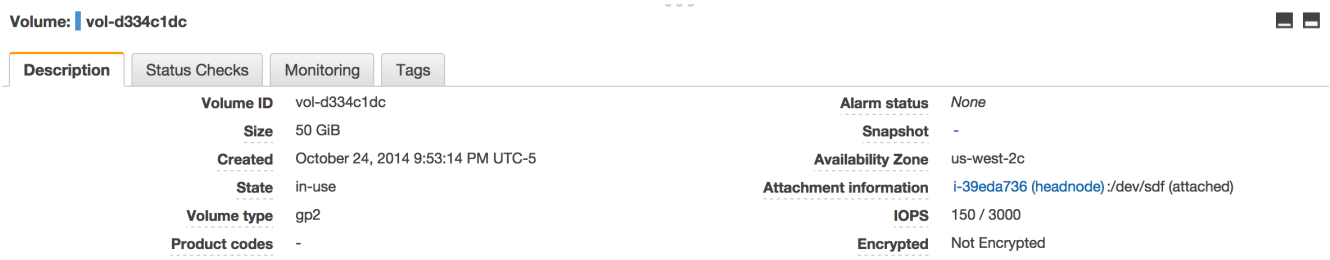
      ```
      Swift version is Swift 0.95 RC6 swift-r7900 cog-r3908
      ```

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 short it counted words in alphabetical order
      - merge.sh : bash script to get and merge output files from workernodes
      - swift.cont : swift configuration for cluster
      - inputs/ : Directory for input files
      - output/ : Directory for output files

16. Now for 10Gb dataset

17. Create new volume at ec2 console



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

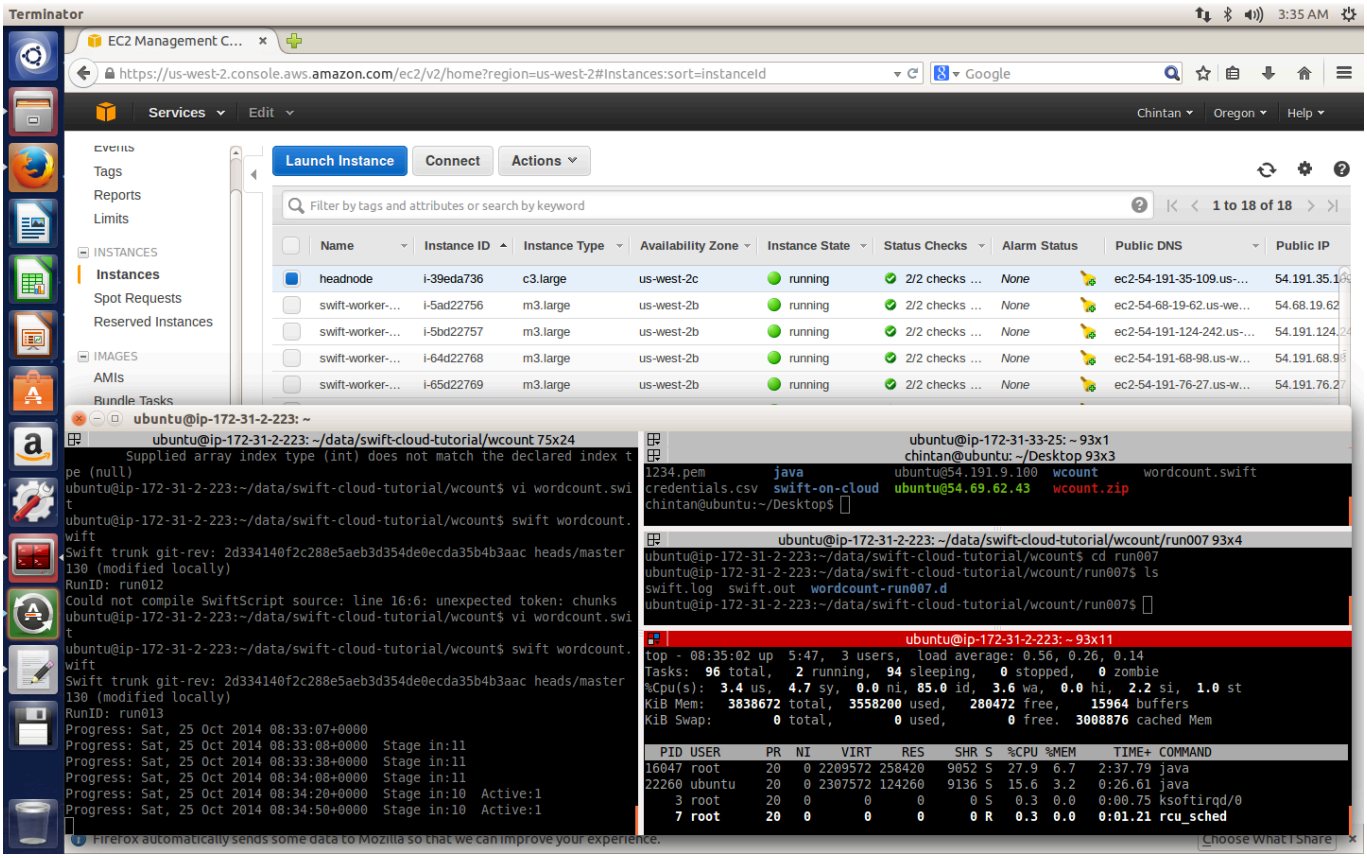
```
Current workers   : 0
Workers requested : 16
Starting worker(s) swift-worker-000 swift-worker-001 swift-worker-002 swift-work
 swift-worker-004 swift-worker-005 swift-worker-006 swift-worker-007 swift-worke
swift-worker-009 swift-worker-010 swift-worker-011 swift-worker-012 swift-worker
wift-worker-014 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

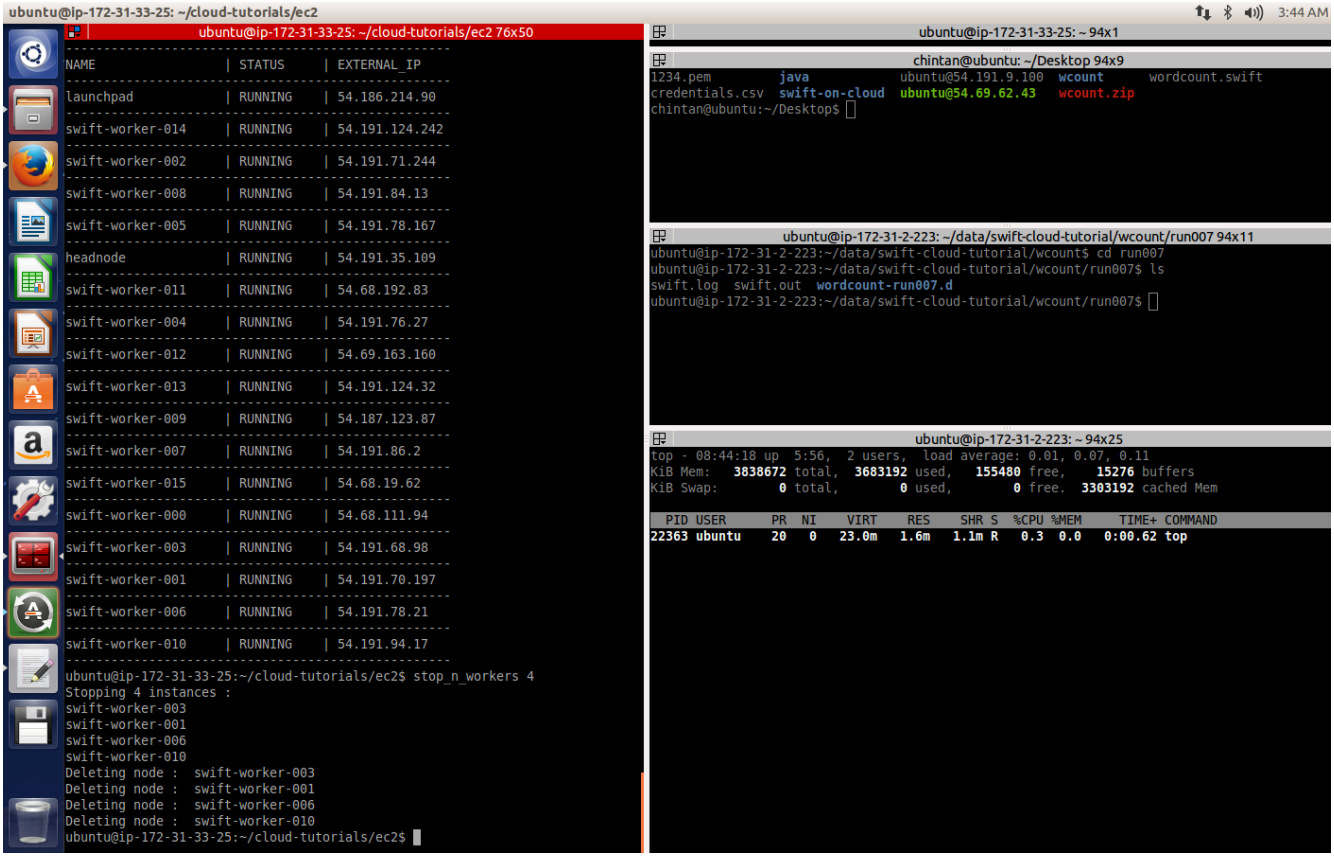
Error because of directory is read-only.

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

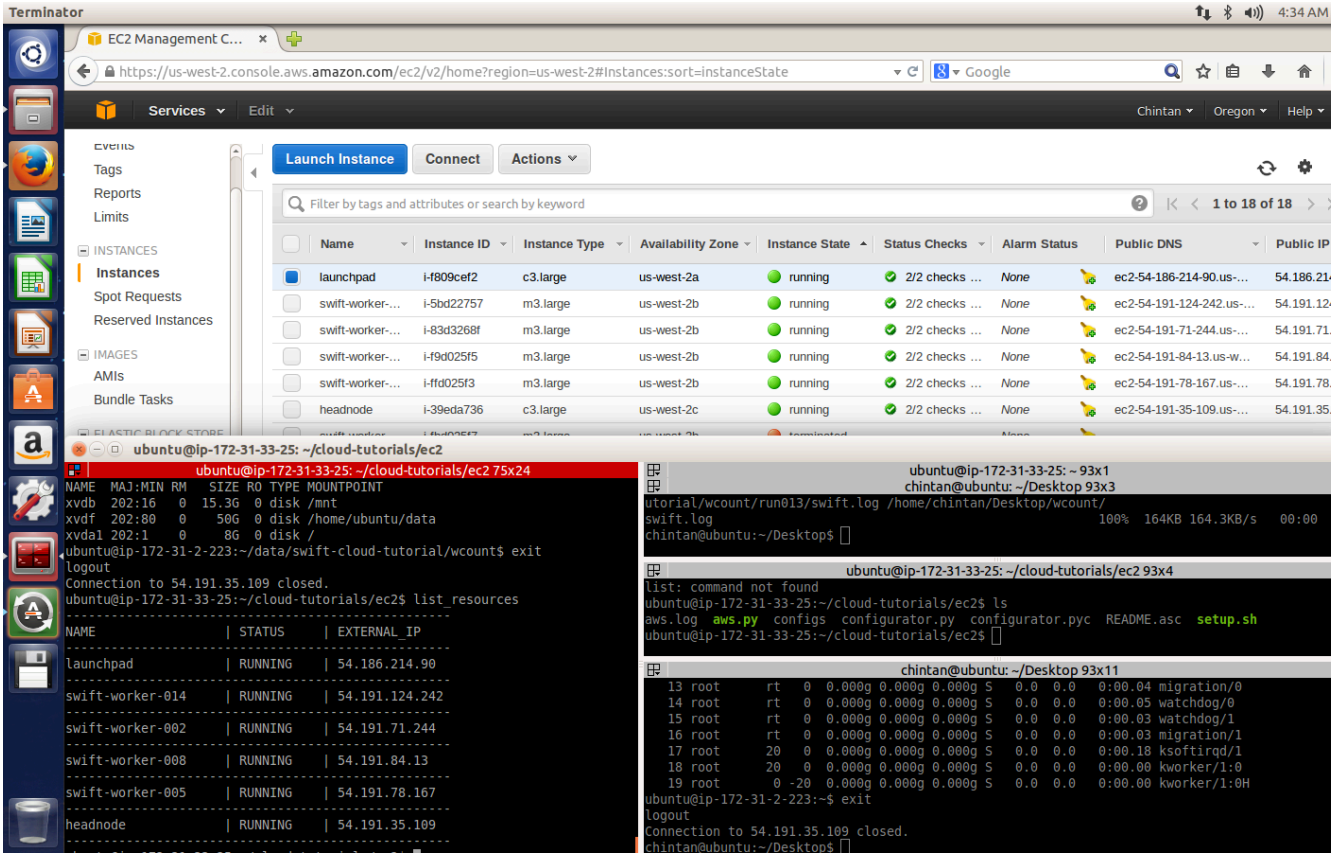
Snap-shots :



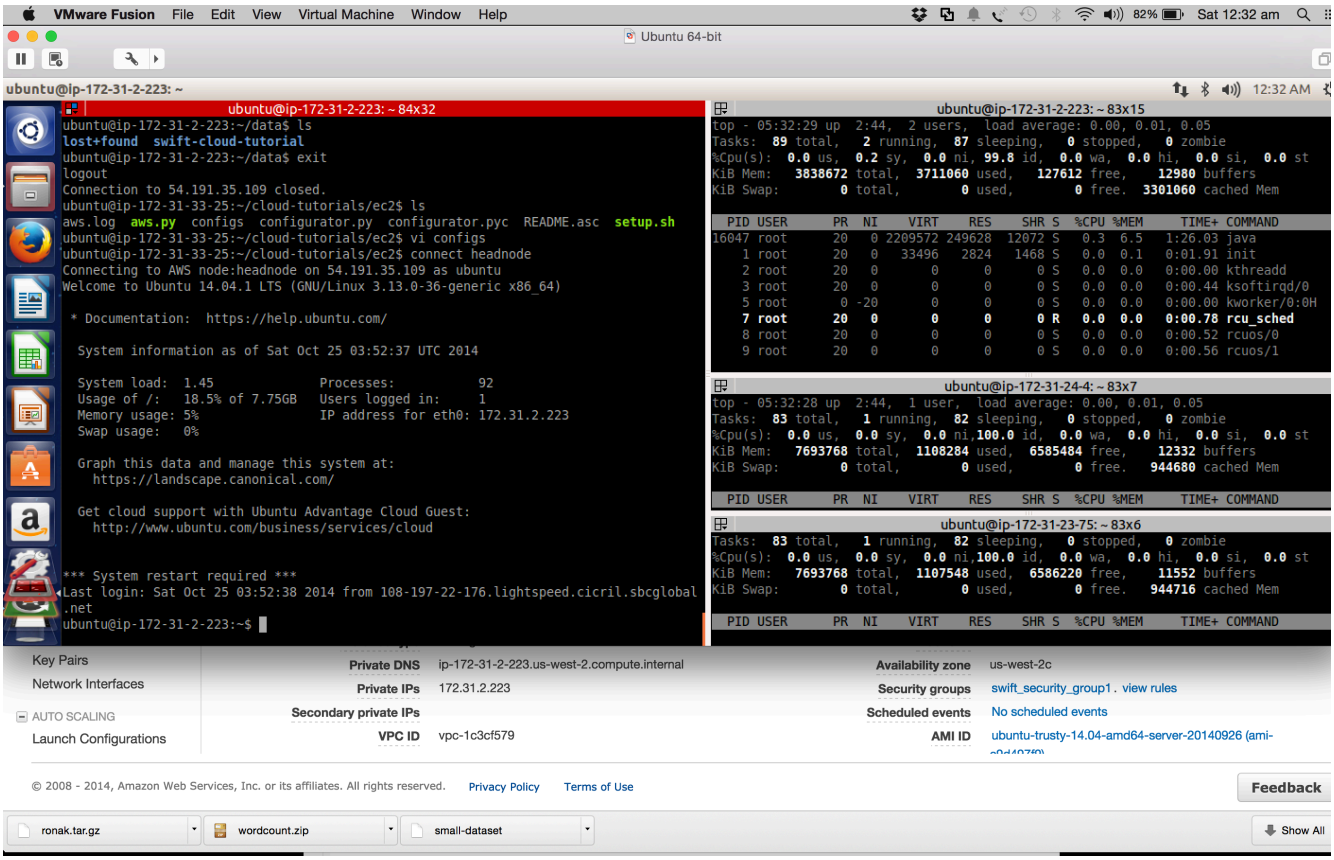
1.1 AWS console and running swift program for 10gb in headnode



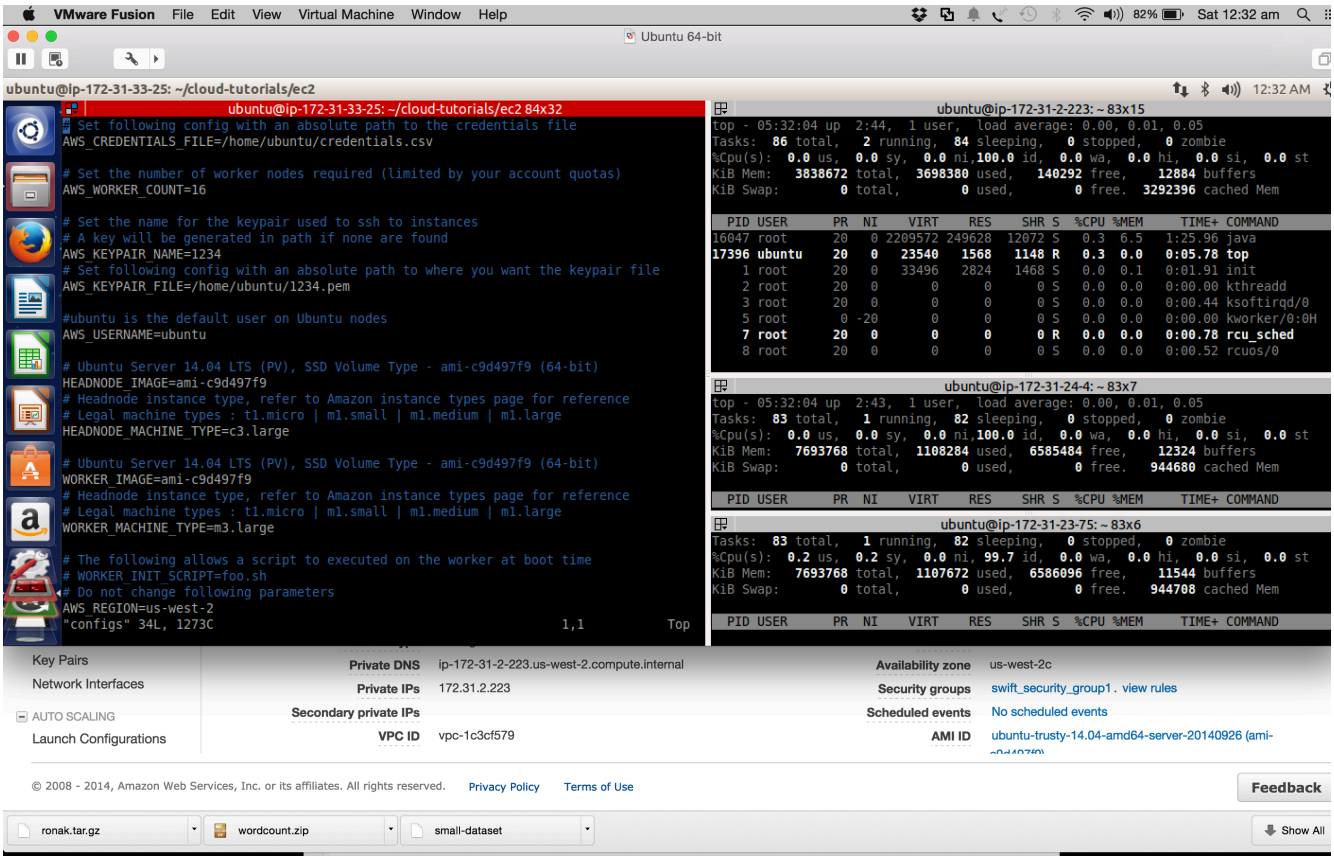
1.2 All list_resources



1.3 Headnode after mount 50Gb disk (lsblk command)



1.4 Connect headnode with it's IP in AWS console



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

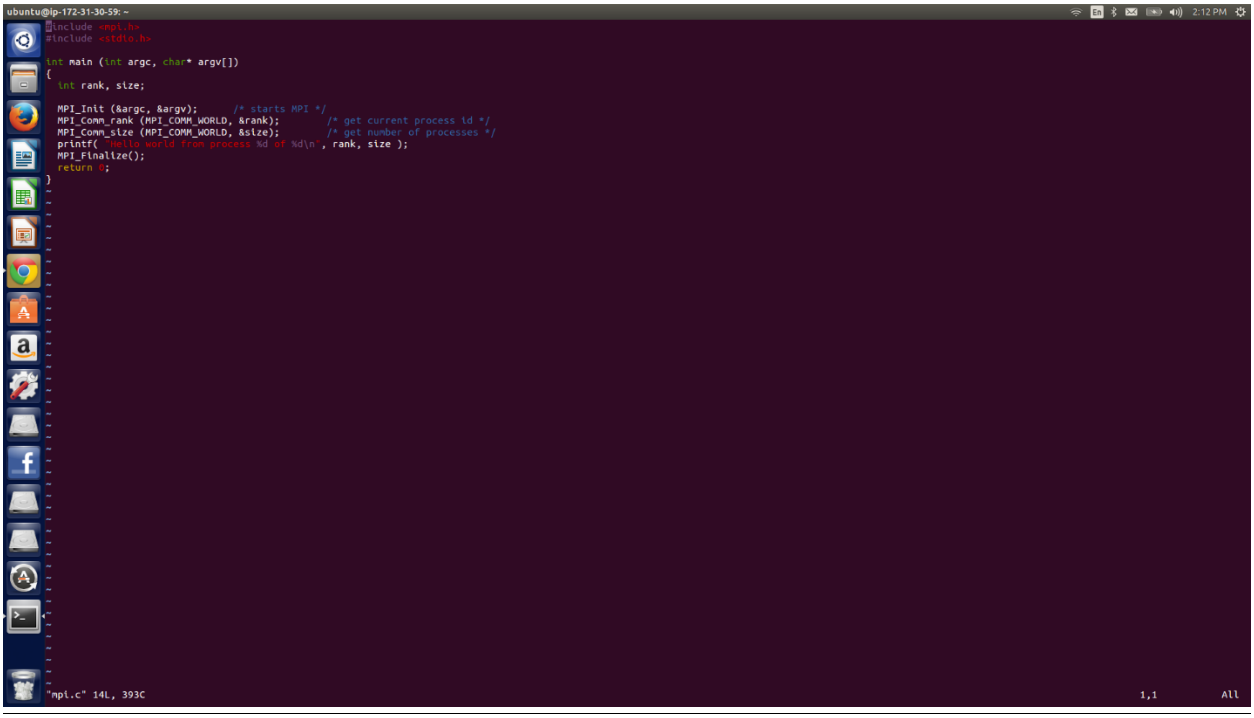
- To run that program using command:

mpirun –np 16 /.m1

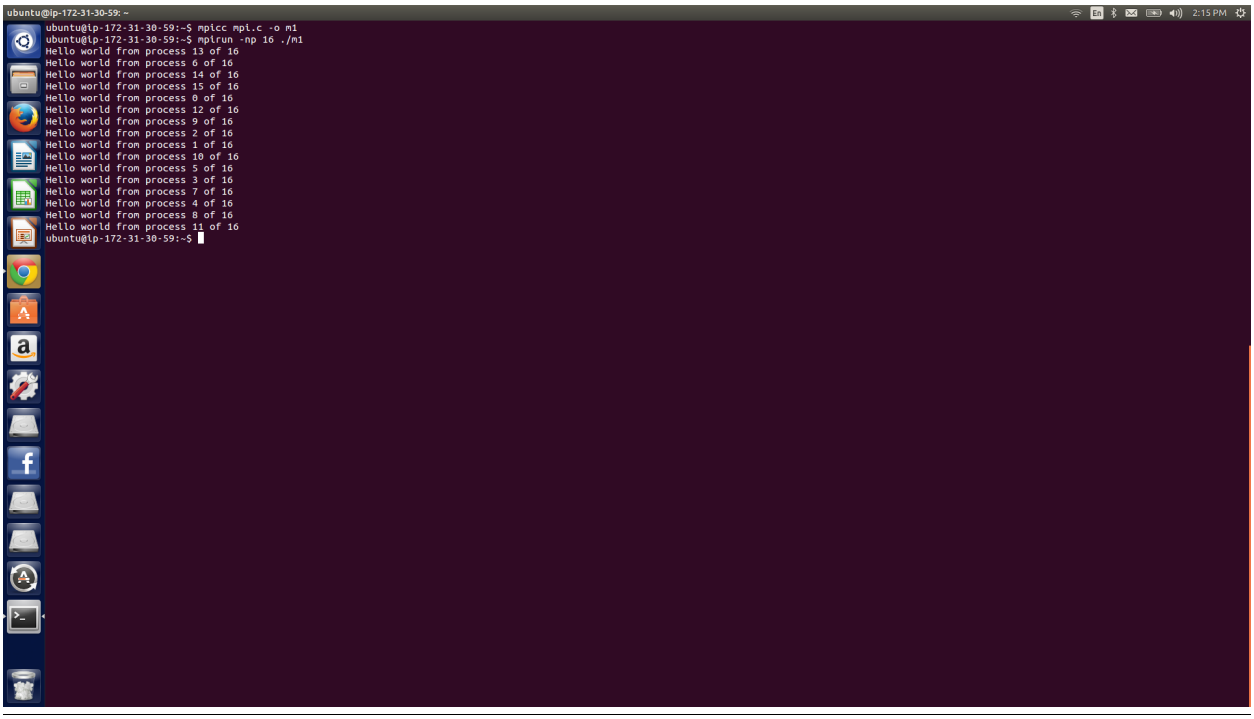
- Then we tried to do sorting of word for MPI program.[3] But we found some problem to with character array.
- Hence we tried to solve using link [2], but due to less time we didn’t figure out how to define character array. So right now we are submitting MPI-hello-world program for single node.

Screen-shots:

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

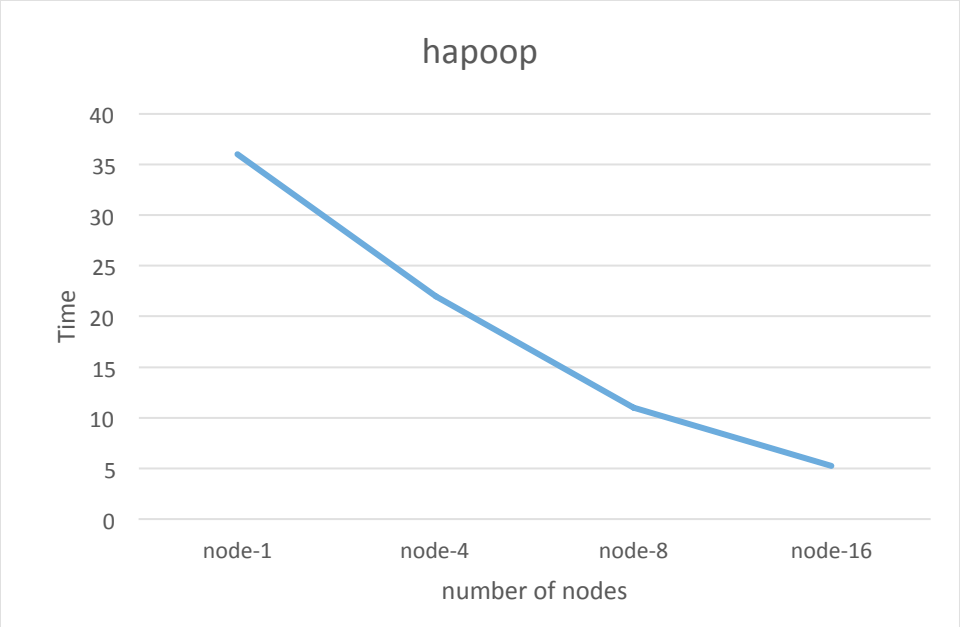


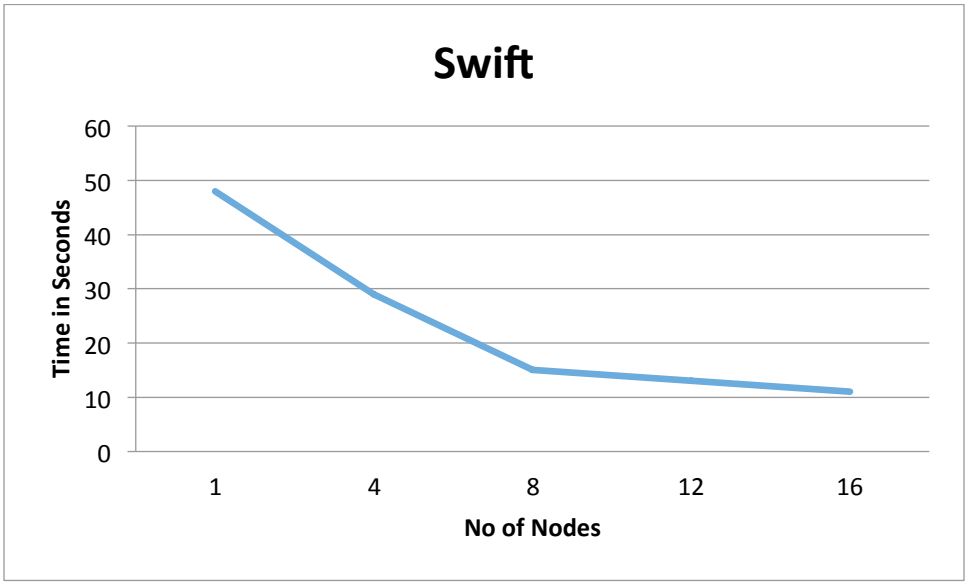
- And the below given snap-shot is the output of the given mpi.c program on cloud.

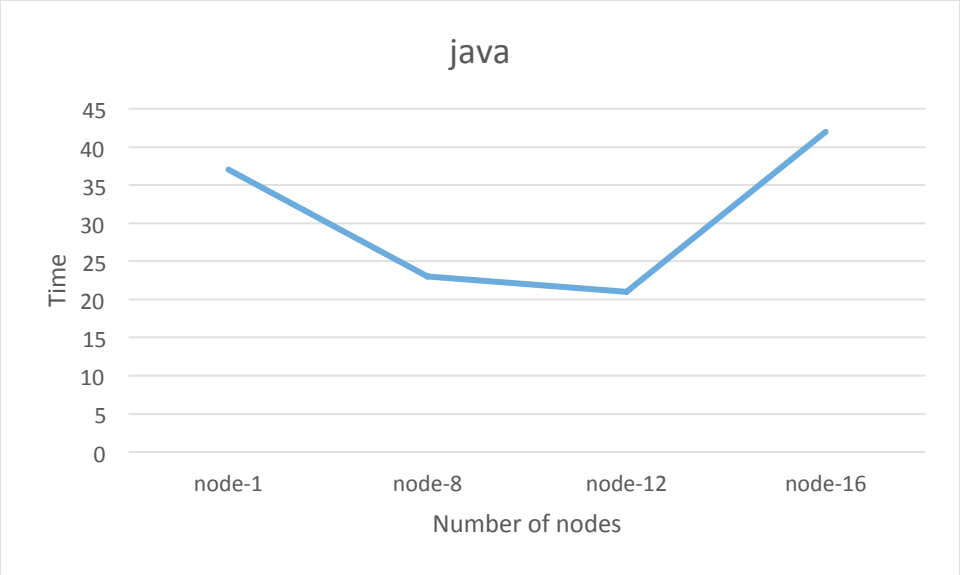


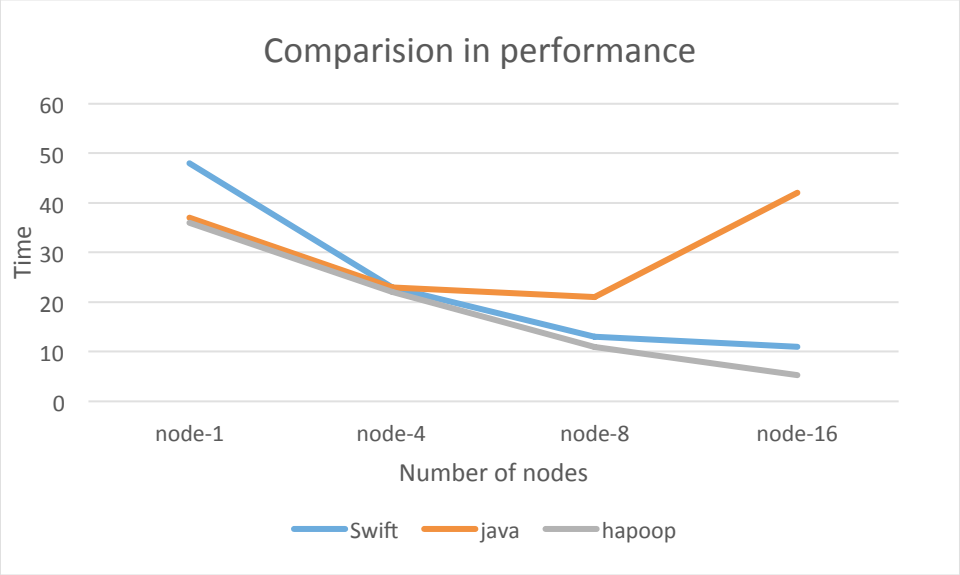
Performance:

- 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** .


hapoop
40
35
30
25
20
15
10
5
0
Time
node-1
node-4
node-8
node-16
number of nodes


Swift
60
50
40
30
20
10
0
Time in Seconds
1
4
8
12
16
No of Nodes


java
45
40
35
30
25
20
15
10
5
0
Time
node-1
node-8
node-12
node-16
Number of nodes


Comparision in performance
60
50
40
30
20
10
0
Time
node-1
node-4
node-8
node-16
Number of nodes
Swift
java
hapoop

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

Swift References:

[1] http://swift-lang.org/main/
[2] 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.

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.dailyfreecode.com/code/sort-words-604.aspx
[4] http://mpitutorial.com/launching-an-amazon-ec2-mpi-cluster/