CS553 – Cloud Computing

Programming Assignment 2

Performance Evaluation

Karan Jeet Singh; Kanwal Preet Singh
10-25-2014

1 INTRO

Documentation and Hadoop done by – karan jeet singh (ksingh14@hawk.iit.edu)

SharedMemory and Swift done by – Kanwal preet singh (ksingh13@hawk.iit.edu)

2 SYSTEM SPECIFICATIONS

All the tests are performed by using the Amazon's EC2 cluster services, out of which c3.large type of instances were employed to run the programs.

C3.large are compute optimized nodes from amazon, they host 7 ECUs, 2 vCPUs @ 2.8GHz each, Intel Xeon E5-2680v2, 3.75 GB memory, 2 x 16 GB SSD Storage Capacity.

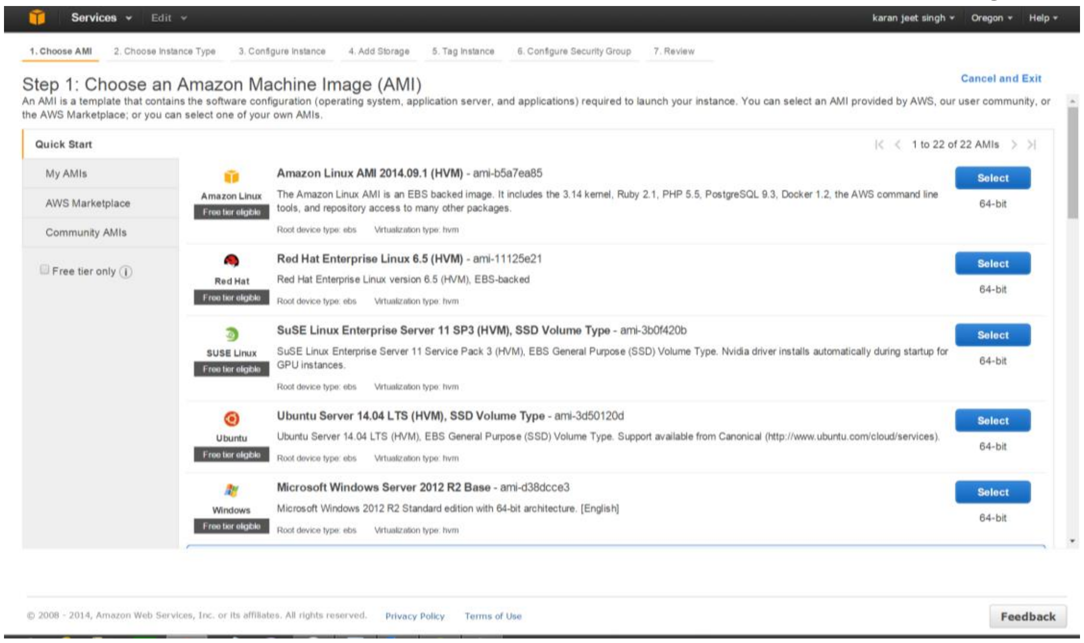
The tests were performed using Linux as its base OS, i.e. Ubuntu server 14.04 LTS (HVM).

JDK version used: 1.7

3 VIRTUAL CLUSTER (1-NODE)

The process followed to create a single node machine is as follows:

1. From the EC2 dashboard, click on Instances > Launch Instance, will lead to following screen:



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux — Free tier eligible
Amazon Linux AMI 2014.09.1 (HVM) - ami-b5a7ea85
The Amazon Linux AMI is an EBS backed image. It includes the 3.14 kernel, Ruby 2.1, PHP 5.5, PostgreSQL 9.3, Docker 1.2, the AWS command line tools, and repository access to many other packages.
Root device type: ebs Virtualization type: hvm

Red Hat — Free tier eligible
Red Hat Enterprise Linux 6.5 (HVM) - ami-11125e21
Red Hat Enterprise Linux version 6.5 (HVM), EBS-backed
Root device type: ebs Virtualization type: hvm

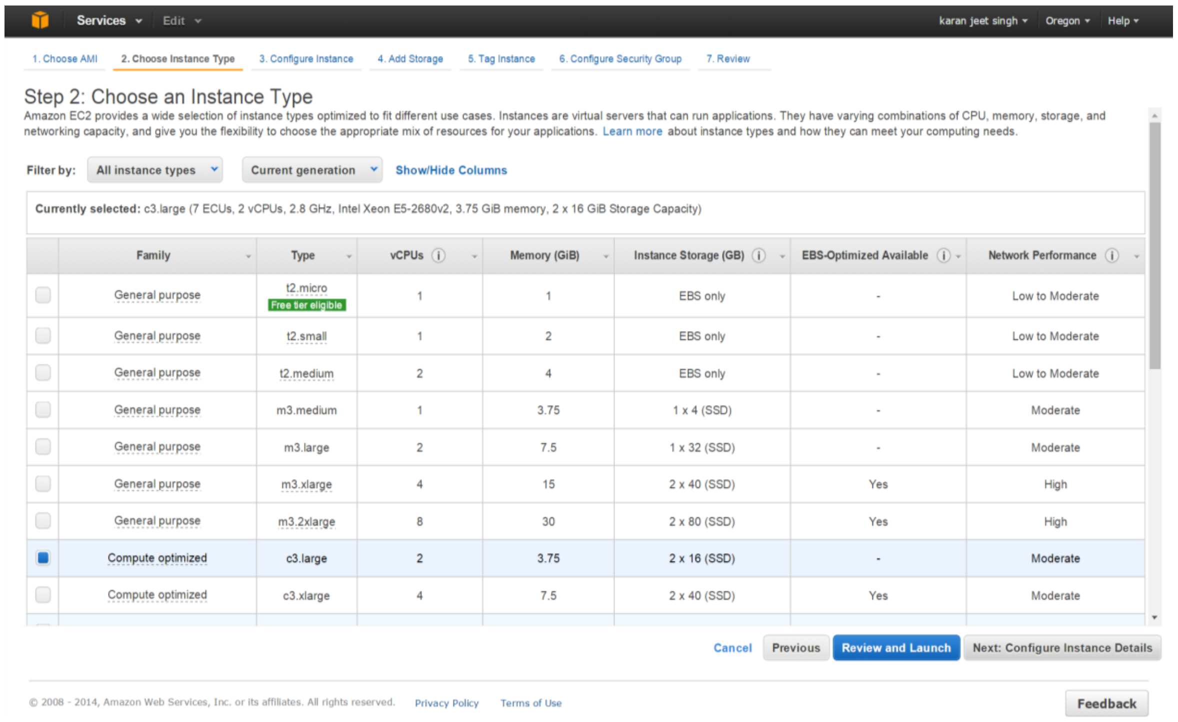
SUSE Linux — Free tier eligible
SuSE Linux Enterprise Server 11 SP3 (HVM), SSD Volume Type - ami-3b0f420b
SuSE Linux Enterprise Server 11 Service Pack 3 (HVM), EBS General Purpose (SSD) Volume Type. Nvidia driver installs automatically during startup for GPU instances.
Root device type: ebs Virtualization type: hvm

Ubuntu — Free tier eligible
Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-3d50120d
Ubuntu Server 14.04 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

Windows — Free tier eligible
Microsoft Windows Server 2012 R2 Base - ami-d38dcce3
Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]
Root device type: ebs Virtualization type: hvm

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.

2. Select any OS, but for this assignment Ubuntu Server 14.04 LTS (HVM) was used. After selection, following screen will appear:
This Step lets the user choose type of instance to be deployed.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

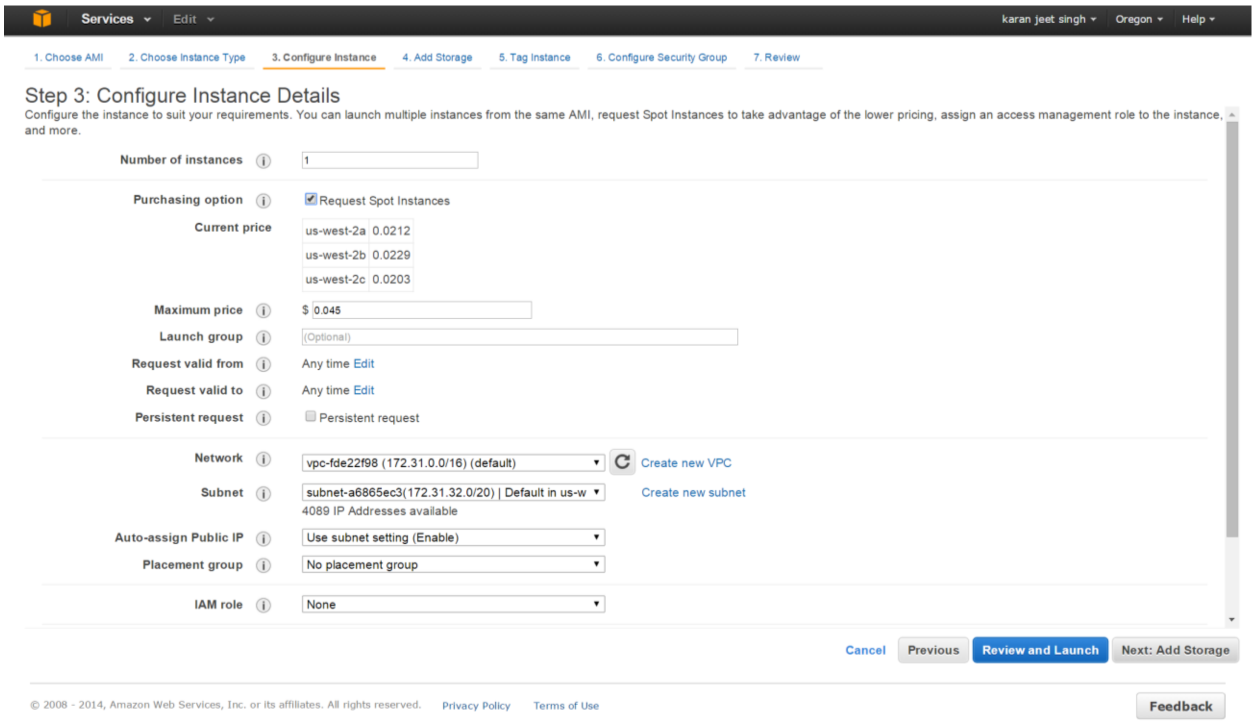
Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: c3.large (7 ECUs, 2 vCPUs, 2.8 GHz, Intel Xeon E5-2680v2, 3.75 GiB memory, 2 x 16 GiB Storage Capacity)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
☐	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
☐	General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
☐	General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
☐	General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
☐	General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High
■	Compute optimized	c3.large	2	3.75	2 x 16 (SSD)	-	Moderate
☐	Compute optimized	c3.xlarge	4	7.5	2 x 40 (SSD)	Yes	Moderate

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

3. For this node setup c3.large was selected. After that we move on to configuring further details for this instance. The first screen we get is where we can specify the number of instances we want to deploy, whether we want On-Demand instances or a Spot-request type of instance.



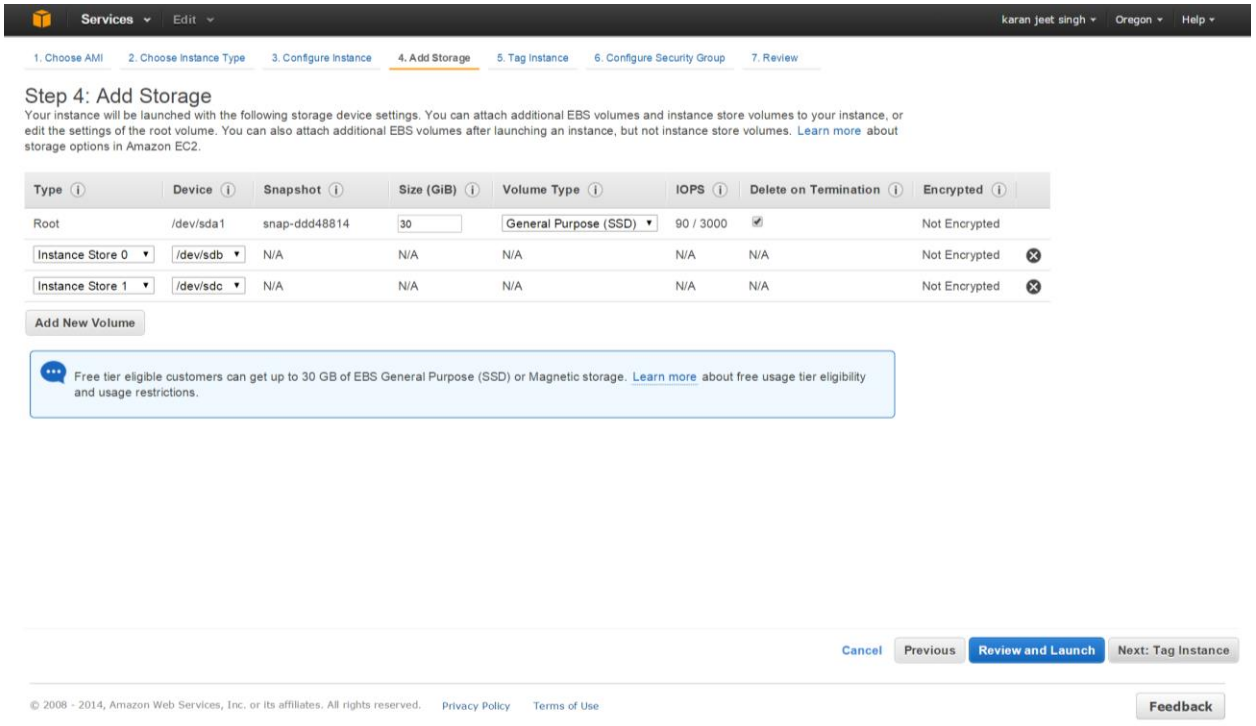
Step 3: Configure Instance Details

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
Purchasing option	☑ Request Spot Instances
Current price	us-west-2a 0.0212 us-west-2b 0.0229 us-west-2c 0.0203
Maximum price	$ 0.045
Launch group	(Optional)
Request valid from	Any time Edit
Request valid to	Any time Edit
Persistent request	☐ Persistent request
Network	vpc-fde22f98 (172.31.0.0/16) (default) — Create new VPC
Subnet	subnet-a6865ec3(172.31.32.0/20) | Default in us-w — Create new subnet 4089 IP Addresses available
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	No placement group
IAM role	None

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

4. After that we go onto the screen where we specify storage details.



Step 4: Add Storage

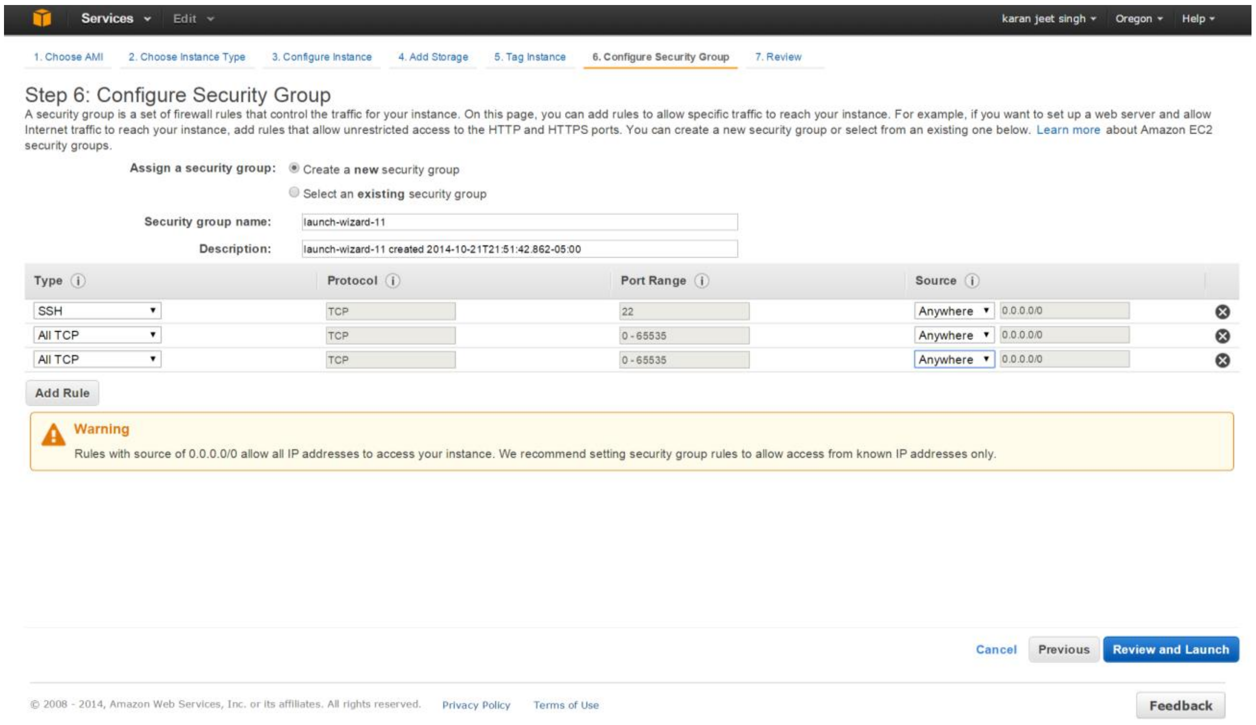
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root	/dev/sda1	snap-ddd48814	30	General Purpose (SSD)	90 / 3000	✓	Not Encrypted
Instance Store 0	/dev/sdb	N/A	N/A	N/A	N/A	N/A	Not Encrypted
Instance Store 1	/dev/sdc	N/A	N/A	N/A	N/A	N/A	Not Encrypted

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.

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.

5. Next screen allows user to specify tags for the instances, which are being deployed for easier identification, we just skip that screen and move onto the next one which helps configure the security groups and network protocol settings.



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-11

Description: launch-wizard-11 created 2014-10-21T21:51:42.862-05:00

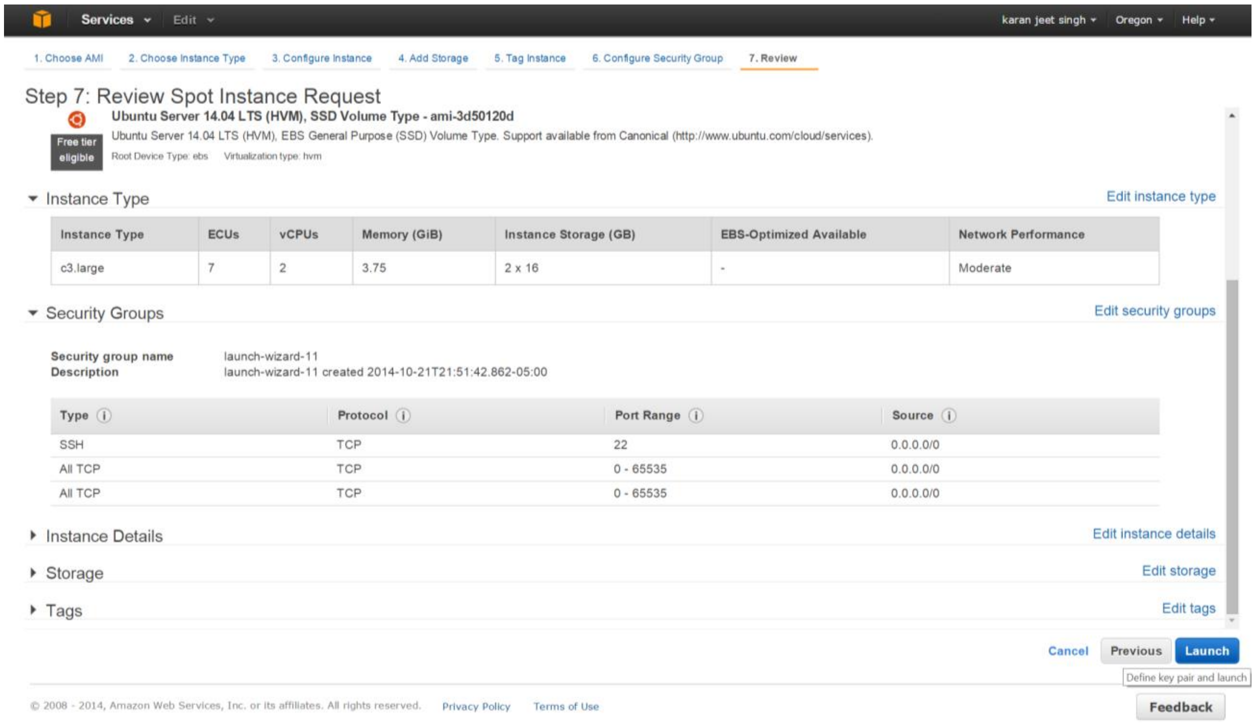
Type	Protocol	Port Range	Source	
SSH	TCP	22	Anywhere	0.0.0.0/0
All TCP	TCP	0 - 65535	Anywhere	0.0.0.0/0
All TCP	TCP	0 - 65535	Anywhere	0.0.0.0/0

Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.

6. After specifying the TCP and UDP forwarding details as specified in the screenshot above, we move onto the last screen for review, from where we hit "Launch" and the instance gets deployed after we select the ssh key-pair.



Step 7: Review Spot Instance Request

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-3d50120d

Free tier eligible

Ubuntu Server 14.04 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

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
c3.large	7	2	3.75	2 x 16	-	Moderate

Security Groups

Security group name	launch-wizard-11
Description	launch-wizard-11 created 2014-10-21T21:51:42.862-05:00

Type	Protocol	Port Range	Source
SSH	TCP	22	0.0.0.0/0
All TCP	TCP	0 - 65535	0.0.0.0/0
All TCP	TCP	0 - 65535	0.0.0.0/0

7. After initiation of node, we installed JDK version 1.7 in both shared memory word count and the Hadoop word count codes.

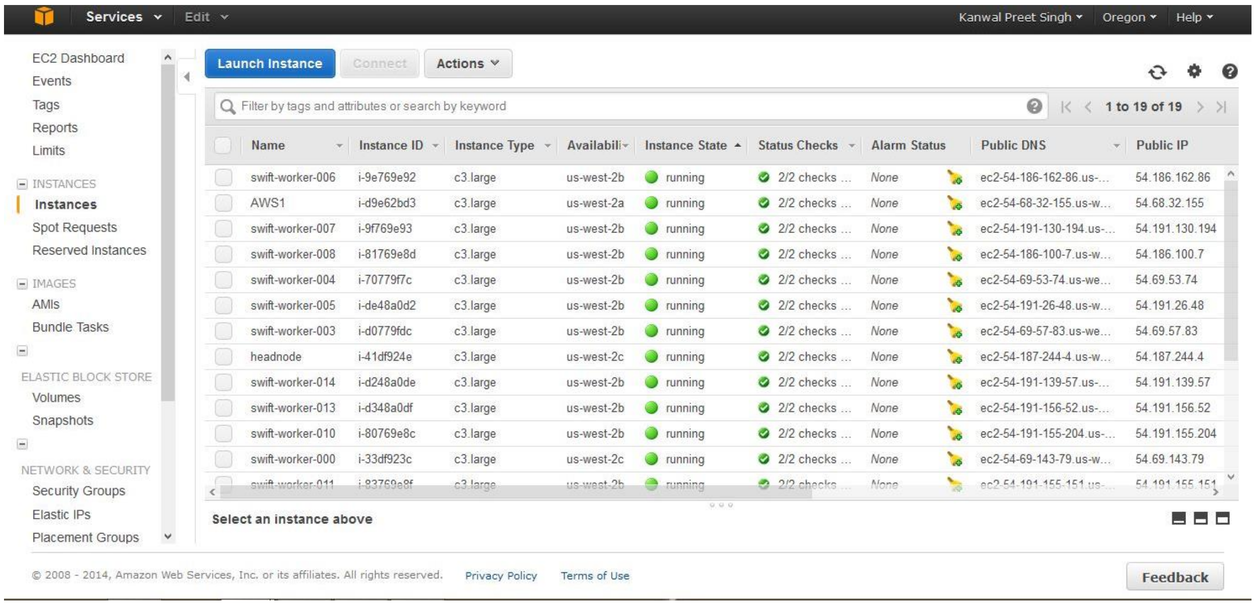
4 SHARED-MEMORY WORD COUNT

For the purpose of this exercise we developed a word count program using Java. We implemented the basic methodology behind map reduce using Java, the file is split into smaller size files which can be loaded onto the memory all at once.

JDK v1.7 was used to compile the code and create the executable jar files.

5 Virtual Cluster (17-nodes)

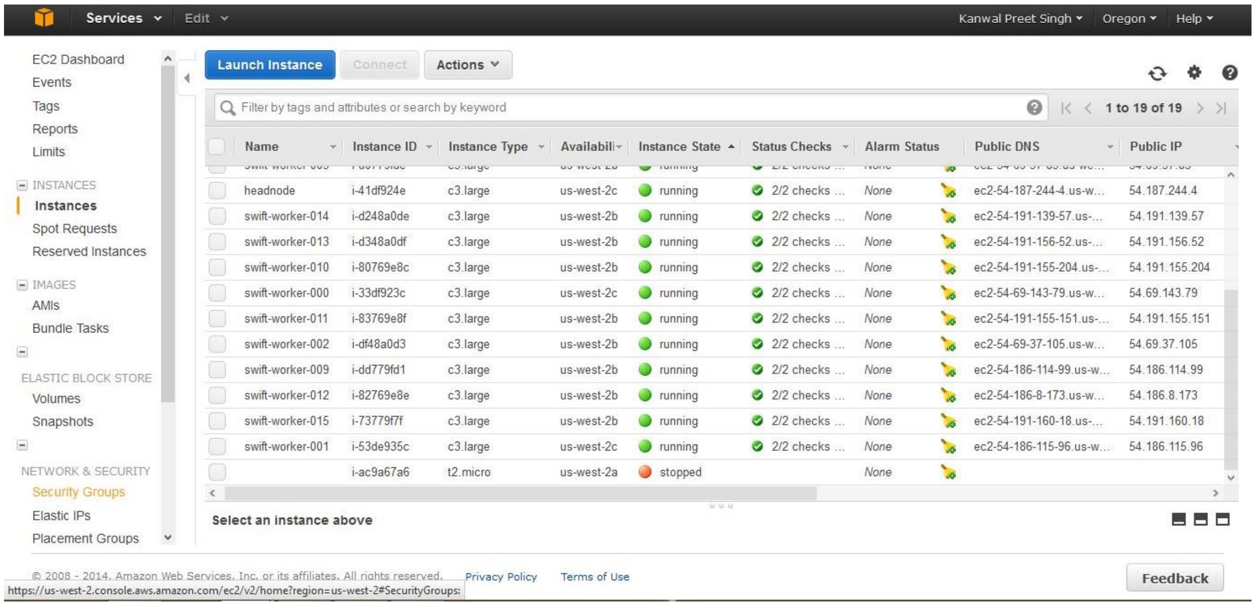
A 17-node virtual cluster can be setup just like the way single node is set up. In the step 3 of configuration, we can specify the number of clusters to be deployed which would result in launch of 17 similar nodes.



Name	Instance ID	Instance Type	Availabili	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
swift-worker-006	i-9e769e92	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-186-162-86.us-...	54.186.162.86
AWS1	i-d9e62bd3	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-68-32-155.us-w...	54.68.32.155
swift-worker-007	i-9f769e93	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-130-194.us-...	54.191.130.194
swift-worker-008	i-81769e8d	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-186-100-7.us-w...	54.186.100.7
swift-worker-004	i-70779f7c	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-69-53-74.us-we...	54.69.53.74
swift-worker-005	i-de48a0d2	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-26-48.us-w...	54.191.26.48
swift-worker-003	i-d0779fdc	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-69-57-83.us-we...	54.69.57.83
headnode	i-41df924e	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-187-244-4.us-w...	54.187.244.4
swift-worker-014	i-d248a0de	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-139-57.us-...	54.191.139.57
swift-worker-013	i-d348a0df	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-156-52.us-...	54.191.156.52
swift-worker-010	i-80769e8c	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-155-204.us-...	54.191.155.204
swift-worker-000	i-33df923c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-143-79.us-w...	54.69.143.79
swift-worker-011	i-83769e8f	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-155-151.us-...	54.191.155.151

Select an instance above

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



Name	Instance ID	Instance Type	Availabili	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
headnode	i-41df924e	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-187-244-4.us-w...	54.187.244.4
swift-worker-014	i-d248a0de	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-139-57.us-...	54.191.139.57
swift-worker-013	i-d348a0df	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-156-52.us-...	54.191.156.52
swift-worker-010	i-80769e8c	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-155-204.us-...	54.191.155.204
swift-worker-000	i-33df923c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-143-79.us-w...	54.69.143.79
swift-worker-011	i-83769e8f	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-155-151.us-...	54.191.155.151
swift-worker-002	i-df48a0d3	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-69-37-105.us-w...	54.69.37.105
swift-worker-009	i-dd779fd1	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-186-114-99.us-w...	54.186.114.99
swift-worker-012	i-82769e8e	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-186-8-173.us-w...	54.186.8.173
swift-worker-015	i-73779f7f	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-160-18.us-...	54.191.160.18
swift-worker-001	i-53de935c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-186-115-96.us-w...	54.186.115.96
	i-ac9a67a6	t2.micro	us-west-2a	stopped		None		

Select an instance above

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#SecurityGroups:

6 HADOOP

For the sake of this assignment we used Hadoop v1.2.1 to implement the word count algorithm.

We created following shell script to help us speed up the setup process of various nodes for Hadoop from scratch.

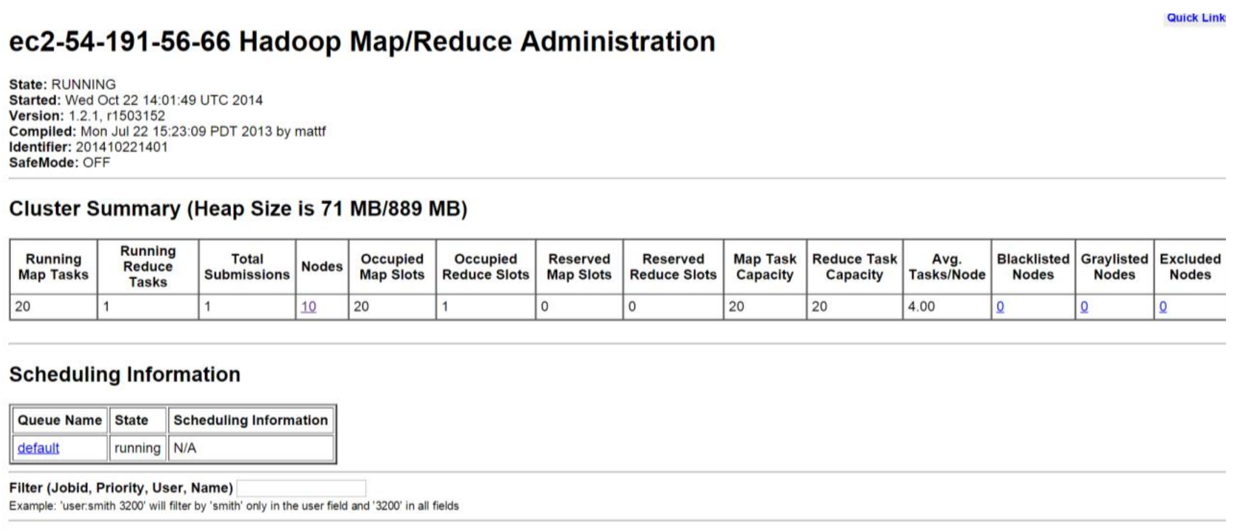
configure.sh

6.1 INSTALLING HADOOP

In order to install Hadoop:

1. Install JDK v7
2. Setup hosts file to use the public DNS name rather than localhost
3. Edit .bashrc for environment variables.
4. Download and extract Hadoop-1.2.1
5. Make a local directory for hdfs to use.
6. Edit Hadoop-env.sh, hdfs-site.xml, and mapred-site.xml
7. Configure conf/masters, and conf/slaves file.

Screen shot for 10 node setup:



Quick Link

ec2-54-191-56-66 Hadoop Map/Reduce Administration

State: RUNNING
Started: Wed Oct 22 14:01:49 UTC 2014
Version: 1.2.1, r1503152
Compiled: Mon Jul 22 15:23:09 PDT 2013 by mattf
Identifier: 201410221401
SafeMode: OFF

Cluster Summary (Heap Size is 71 MB/889 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
20	1	1	10	20	1	0	0	20	20	4.00	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

7 Hadoop Word Count

Hadoop word count employs the logic of map reduce to implement the word count. A file is split into hash tables, and then a sorted version of it is merged together to produce a final count. Hadoop lets the user use n number of nodes to create the hash tables, i.e. for mapping the input string.

7.1 Configure Hadoop

Configuring Hadoop requires changes in following 6 files:

1. Conf/masters
2. Conf/slaves
3. Conf/Hadoop-env.sh
4. Conf/core-site.xml
5. Conf/hdfs-site.xml
6. Conf/mapred-site.xml

“masters” file hosts information about the master nodes in the cluster, i.e. the namenode and the secondary namenode.

“slaves” file in the master node hosts the address of all the slave nodes accessible via the master nodes which are then accessed and made part of the cluster, the slaves file in the slave node just has its own address in there so that the node knows that it is acting as a slave.

“Hadoop-env.sh” is used to set environment variables to be used by the Hadoop itself, for this exercise we define the path to the jdk in it.

“core-site.xml” it specifies the information about the master node, i.e. the namenode, and also the path to the directory which is used by Hadoop file system to create a shareable storage.

“hdfs-site.xml” specifies the count of node where the replication block for hdfs is to be created. This number basically determines how many nodes can connect to the master node.

“mapred-site.xml” specifies the path on which the map reduce job tasks are to be run, the address mentioned here should be accessible via all the slave nodes and the secondary namenode as well.

Hadoop map reduce work by dividing the task in hand to different workers, who are also known as slaves, and all the slaves are controlled by one primary node known as master node. When a file is read by master name node, it distributes the workload to the slaves, and then when they are done with their part of processing, the master node sorts the data by managing the data flow between the slaves. After slaves receive the sorted data, it is merged together into a single file which in result is the output file.

While on shared platform, different ports are used to access the file system, and these are configured in the files mentioned above:

- In core-site.xml, it is configured which port is used by the namenode for File System Metadata operations.
- In mapred-site.xml, it is configured which port is used by the JobTracker for Job submission, and Task Tracker heartbeats.

The number of mappers and reducers are also configurable by using below mentioned tags in the mapred-site.xml file:

```
<property>
    <name>mapred.tastracker.map.tasks.maximum</name>
    <value>2</value>
</property>
<property>
    <name>mapred.tasktracker.reduce.tasks.maxmimum</name>
    <value>2</value>
</property>
```

In such a way we can set the number of map and reduce task to be 2 each for each node.

8 SWIFT

In this assignment we used Swift 0.95-RC6 to implement the word count algorithm.

8.1 INSTALLING SWIFT

In order to install swift on the cloud (18 node cluster):

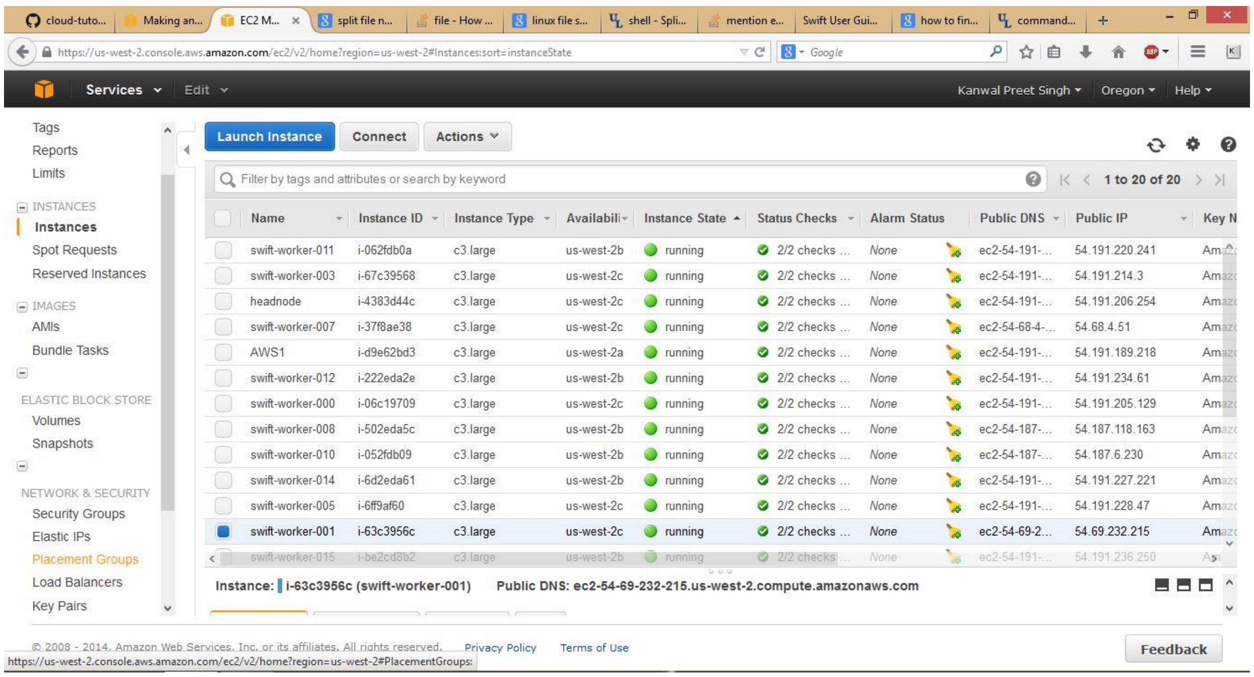
Setting up the Launchpad

1. Install JDK v7
2. Install python
3. Install git
4. Download swift 0.95-RC6.tar.gz and extract it
5. Edit .bashrc for environment variables to add Java and swift
6. Clone the Github repository cloud-tutorials
7. In the EC2 folder of cloud-tutorials edit the configs files
8. Run setup.sh

configs

The configs file mentions the headnode, no. of worker, the image to be used for creating the headnode and workers. The security group attached to the headnode and workers and the key details for the ssh connection.

Screen shot for 17 node setup:



https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=instanceState

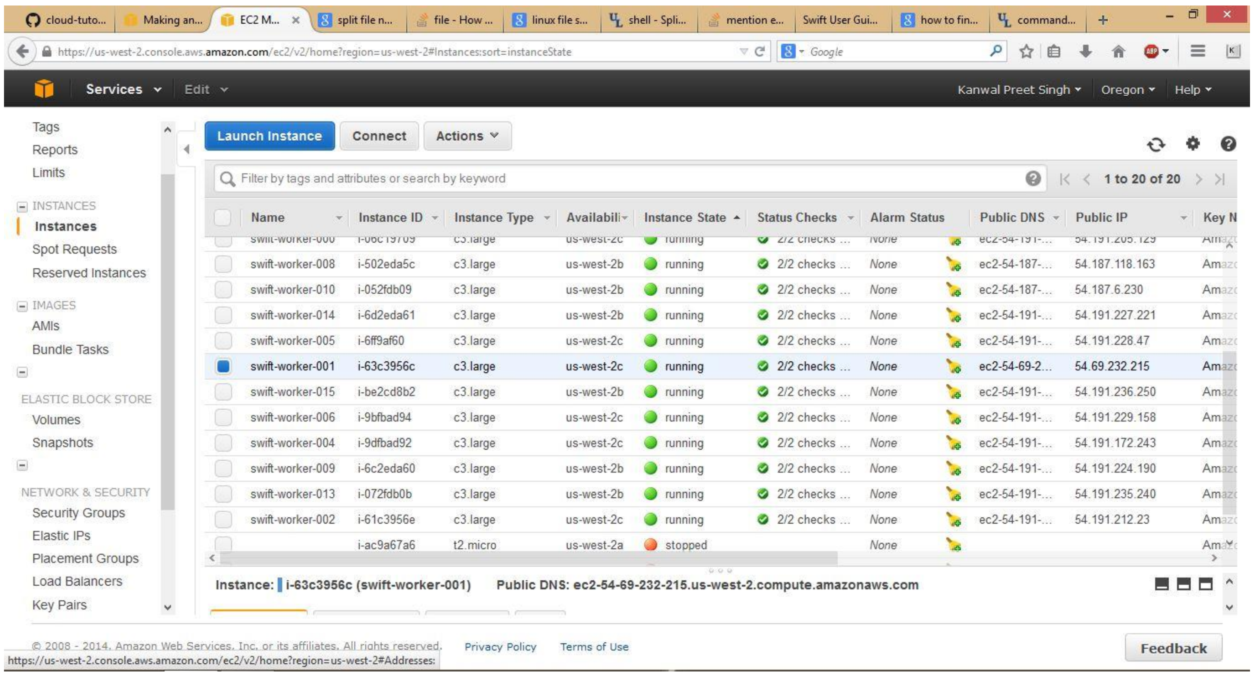
1 to 20 of 20

Name	Instance ID	Instance Type	Availabili	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key N
swift-worker-011	i-062fdb0a	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.220.241	Ama
swift-worker-003	i-67c39568	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.214.3	Amaz
headnode	i-4383d44c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.206.254	Amaz
swift-worker-007	i-37f8ae38	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-68-4-...	54.68.4.51	Amaz
AWS1	i-d9e62bd3	c3.large	us-west-2a	running	2/2 checks ...	None	ec2-54-191-...	54.191.189.218	Amaz
swift-worker-012	i-222eda2e	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.234.61	Amaz
swift-worker-000	i-06c19709	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.205.129	Amaz
swift-worker-008	i-502eda5c	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-187-...	54.187.118.163	Amaz
swift-worker-010	i-052fdb09	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-187-...	54.187.6.230	Amaz
swift-worker-014	i-6d2eda61	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.227.221	Amaz
swift-worker-005	i-6ff9af60	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.228.47	Amaz
swift-worker-001	i-63c3956c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-2...	54.69.232.215	Amaz
swift-worker-015	i-be2cd8b2	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.236.250	A

Instance: i-63c3956c (swift-worker-001) **Public DNS:** ec2-54-69-232-215.us-west-2.compute.amazonaws.com

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#PlacementGroups:



https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=instanceState

1 to 20 of 20

Name	Instance ID	Instance Type	Availabili	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key N
swift-worker-000	i-06c19709	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.205.129	Amaz
swift-worker-008	i-502eda5c	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-187-...	54.187.118.163	Amaz
swift-worker-010	i-052fdb09	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-187-...	54.187.6.230	Amaz
swift-worker-014	i-6d2eda61	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.227.221	Amaz
swift-worker-005	i-6ff9af60	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.228.47	Amaz
swift-worker-001	i-63c3956c	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-69-2...	54.69.232.215	Amaz
swift-worker-015	i-be2cd8b2	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.236.250	Amaz
swift-worker-006	i-9bfbad94	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.229.158	Amaz
swift-worker-004	i-9dfbad92	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.172.243	Amaz
swift-worker-009	i-6c2eda60	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.224.190	Amaz
swift-worker-013	i-072fdb0b	c3.large	us-west-2b	running	2/2 checks ...	None	ec2-54-191-...	54.191.235.240	Amaz
swift-worker-002	i-61c3956e	c3.large	us-west-2c	running	2/2 checks ...	None	ec2-54-191-...	54.191.212.23	Amaz
	i-ac9a67a6	t2.micro	us-west-2a	stopped		None			Amaz

Instance: i-63c3956c (swift-worker-001) **Public DNS:** ec2-54-69-232-215.us-west-2.compute.amazonaws.com

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Addresses:

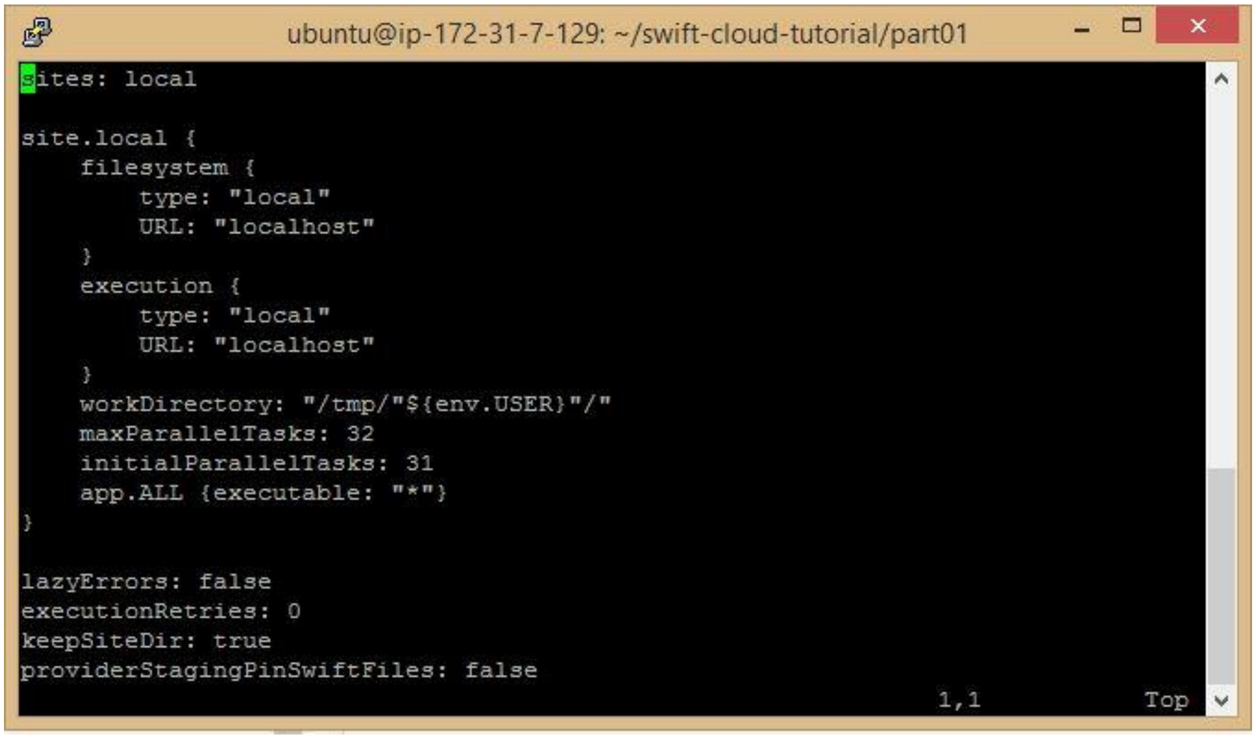
9 SWIFT WORD COUNT

In swift we use the shell scripts to implement the word count logic.

First we connect to the Launchpad, from the Launchpad we run the ‘setup.sh’ script which initializes the workers and the headnode using the ‘configs’ file. When the headernode and workers are initialized we connect to the headernode. After connecting to the header node we copy the swift-cloud-tutorial folder to the root directory, run the ‘setup.sh’ file again.

For running the code on 1 node:

All the script files, input file and the swift file were moved to the part01 folder in the swift-cloud-tutorial folder. The files were first split using the splitfile.sh script to split the main file into 16 files which are then saved to the ‘input’ folder of the current location. The swift then takes input files from this ‘input’ folder then runs the word count script on individual file and dumps its output in the ‘outwc’ folder. The files in the ‘outwc’ contains the word count of individual files. These files are then merged to produce the final file which contains the word count of the original file. All the processing takes place on the head node as site.conf file in the part folder specifies to the swift to execute the process on the local site.



ubuntu@ip-172-31-7-129: ~/swift-cloud-tutorial/part01

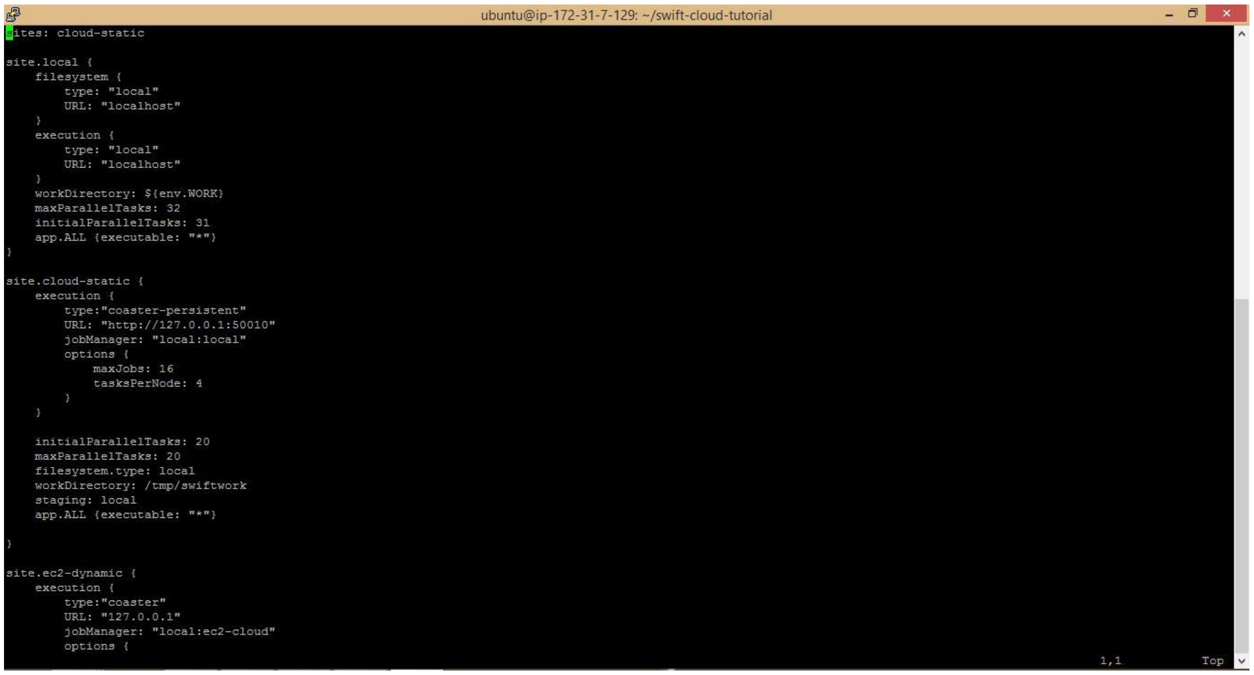
```
sites: local

site.local {
    filesystem {
        type: "local"
        URL: "localhost"
    }
    execution {
        type: "local"
        URL: "localhost"
    }
    workDirectory: "/tmp/"${env.USER}"/"
    maxParallelTasks: 32
    initialParallelTasks: 31
    app.ALL {executable: "*"}
}

lazyErrors: false
executionRetries: 0
keepSiteDir: true
providerStagingPinSwiftFiles: false
                                                              1,1           Top
```

For running the code on 16 nodes:

All the script files, input file and the swift file were moved to the part05 folder in the swift-cloud-tutorial folder. The files were first split using the splitfile.sh script to split the main file into 16 files which are then saved to the ‘input’ folder of the current location. The swift then takes input files from this ‘input’ folder then sends the individual file to every worker. The worker receives the input file runs the swift logic on it and returns the output file. The output files are dumped in the ‘outwc’ folder. The files in the ‘outwc’ contains the word count of individual files. These files are then merged to produce the final file which contains the word count of the original file. The site.conf file mentions the no. of workers and the site of execution of the tasks.



ubuntu@ip-172-31-7-129: ~/swift-cloud-tutorial

```
sites: cloud-static

site.local {
    filesystem {
        type: "local"
        URL: "localhost"
    }
    execution {
        type: "local"
        URL: "localhost"
    }
    workDirectory: ${env.WORK}
    maxParallelTasks: 32
    initialParallelTasks: 31
    app.ALL {executable: "*"}
}

site.cloud-static {
    execution {
        type:"coaster-persistent"
        URL: "http://127.0.0.1:50010"
        jobManager: "local:local"
        options {
            maxJobs: 16
            tasksPerNode: 4
        }
    }

    initialParallelTasks: 20
    maxParallelTasks: 20
    filesystem.type: local
    workDirectory: /tmp/swiftwork
    staging: local
    app.ALL {executable: "*"}

}

site.ec2-dynamic {
    execution {
        type:"coaster"
        URL: "127.0.0.1"
        jobManager: "local:ec2-cloud"
        options {
                                                                        1,1           Top
```

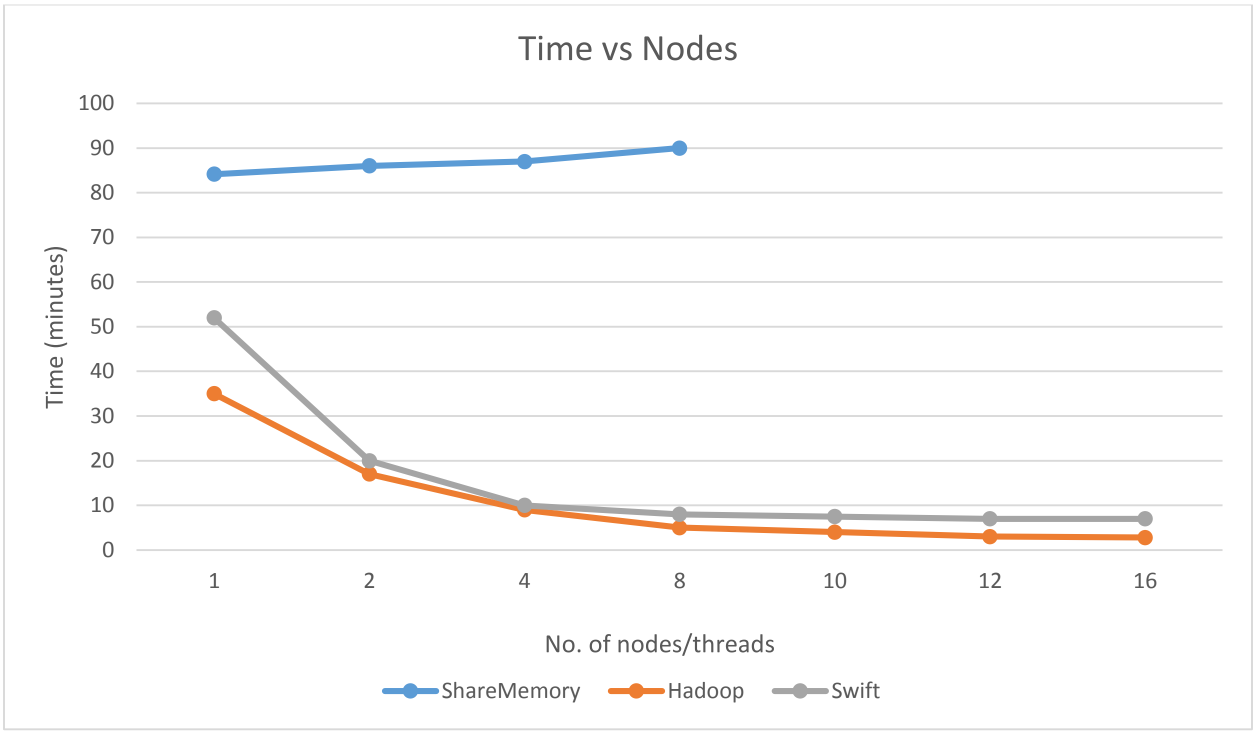
10 PERFORMANCE EVALUATION

Scaling experiments were performed on word count program in different environments to look for the best possible approach to do the same.

SharedMemory-wordcount was implemented using java and was scaled from 1 thread run to max of 8 threads on the c3.large instance.

Wordcount-hadoop was implemented using mapreduce on Hadoop and this was scaled from 1 node processing to the max of 16 nodes.

Wordcount-swift was implemented using swift/T and was scaled from 1 node processing to the max of 16 nodes.



10.1 Shared memory

The shared memory word count program was made using Java. The ConcurrentHashMaps functionality was used to maintain a single hash table for all the threads, the effects of which are shown in our scaling experiments.

Even when we increase the number of threads from 1 to 8, the performance doesn't change much because the ConcurrentHashMaps become a bottle neck for us.

The speed up graphs also depict the same verdict that there is no change in performance as we jump from 1 thread to more on the c3.large instance

10.2 Hadoop

The word count in Hadoop is implemented by using map reduce functionality.

Increase in number of nodes show clear effect on the performance of the application, the Hadoop system proves out to be very scalable and highly efficient even as we increase the number of nodes.

The speed up graph shows a very little change in the trajectory of the performance of Hadoop, i.e. adding more nodes to the scenario would definitely improve the speed at an equal rate.

10.3 Swift/T

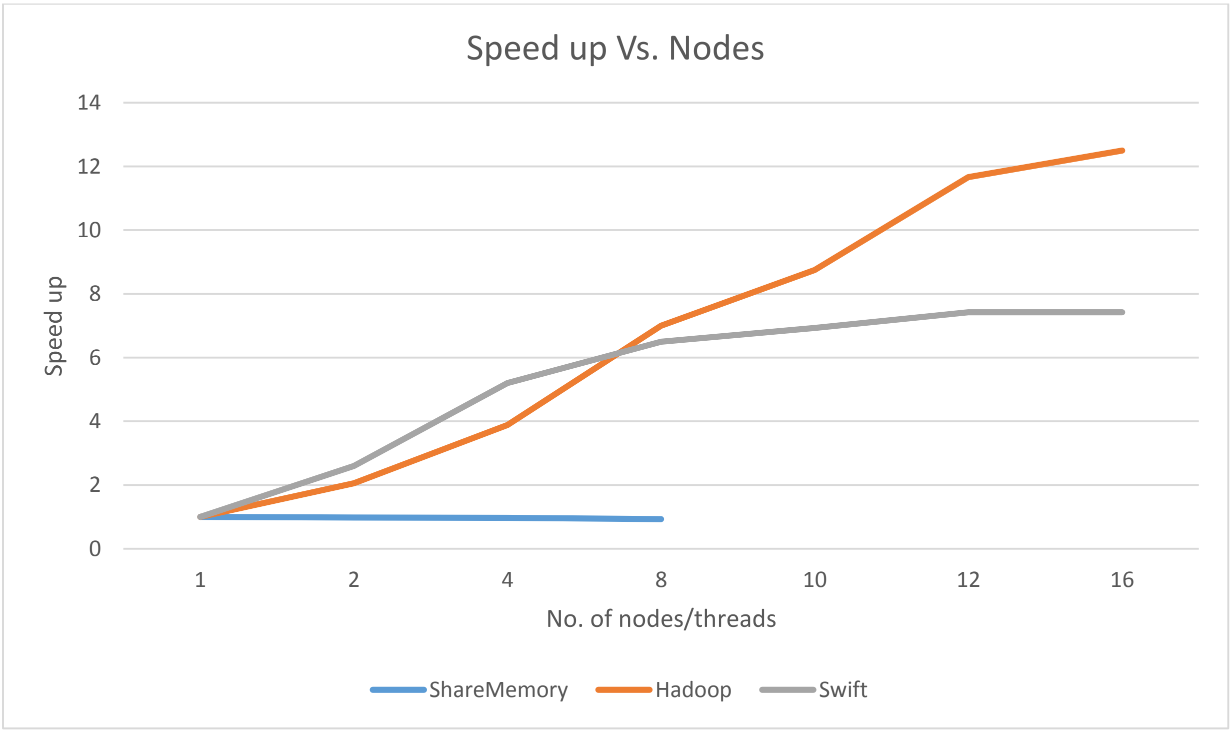
The word count in swift is implemented by combining shell scripting with swift code. The shell scripting is used to split the input file into smaller files, which are then fed to the slaves. At the end, when slaves are done processing, the output files are merged into one using scripts.

In swift, increase in the number of nodes show great signs of improvements, the time it takes to process the file using 1 node is significantly higher than it takes in 16.

The speed up graph shows that the increase in nodes improve the system drastically but even so the change in performance as we go from 8 nodes to 16 nodes is not as much as 1 to 2 nodes.

10.4 Comparison

The performance comparison of SharedMemory, Hadoop, and Swift prove that Hadoop is the best when it comes to scalability. The Speed up for it is much more linear and shows drastic change in performance with every node added.



Conclusion

Upon evaluating all the above mentioned data and experiments, we notice that both Hadoop and Swift turn out to be better than Shared Memory application.

At single node, Hadoop was the best performer. The only factor that causes the difference in performance between Hadoop and swift at single node is that Hadoop sorts its data before it merges them into a hash table. This helps in quicker formation of hash table and hence is the game changer.

At 16 nodes, Hadoop again turns out to be the best performer. The sorting again shows its effect on the result and also the fact that the different nodes in Hadoop make use of hdfs to pick and store data, this helps in easier and quicker data management between the nodes.

Extrapolating the results to 100 or even more number of nodes reveal that Hadoop will continue to be the best performer.