# Project 3 PageRank Performance Analysis on Academic Cloud

# (due Nov. 29, 11:59 PM)

# Introduction

Academic Cloud provides a great environment for researchers and students to run applications with higher availability offered, granting users more control and flexibility to run their own applications. FutureSystems is a nationwide NSF-funded project which provides a distributed, high performance test bed to allow scientists to collaboratively develop and test innovative approaches to parallel, grid, and cloud computing. For this assignment, you will hand in a performance analysis report by running MPI/MPJ PageRank program in FutureSystems OpenStack VMs. Please reuse your MPI/MPJ PageRank program developed in Project #2A. The following sections provide detailed instructions on how to access FutureSystems.

The primary objective of this project is to test the performance of MPJ PageRank as the number of URLs increases. You may need to modify previous programs to show the total execution time.

You must have your FutureSystems account and your generated ssh private key ready for this project. If you haven’t applied for one, please sign up at [FutureSystems Portal](http://portal.futuregrid.org) [1] as soon as possible.

# Deliverables

Each team is required to turn in a report to Canvas entitled [username\_Project3.zip] with the following items:

1. **Experiment with various number of URLs** (1000, 10k, 100k, 500k, 1000k) and the following numbers in a table, then draw a human readable graph chart. Include the details of the environment where you ran each test.
   1. Instance class, number of worker nodes, number of processes per worker node, size of dataset (# of urls, # of groups), number of MPI processors (set with argument -np), MPI error threshold, and number of iterations.
   2. A speedup performance analysis chart (graph).
   3. Each execution time (of your test) must be taken as an average of three runs.
2. Findings, description and explanation of your results.
3. Feedback about using FutureSystems OpenStack.

Points will be reduced (maximum 0.5 points) if the filename or directory structure is different from instructed above.

# Project Evaluation

The point total for Project #3 is 20, where the distribution of each performance benchmark is as follows:

* 1. Experiments: design quality, accurate measurements, amount of work involved (15 points)
  2. Analysis and Report (3 points)
  3. Review/Demo (2 points)

# Performance Analysis and Writing a Report

In this project, you are going to use FutureSystems OpenStack as your test environment. You will need to copy your Project 2A back to this environment, and then execute MPI/MPJ PageRank on a very small (old) dataset to make sure everything is working correctly. Finally, run the program with the generated dataset(s) and analyze the performance. Please make notes on any interesting findings with descriptions and explanations about your results (this information is required on your report).

Each team can only use 2 VMs (2 cores per m1.medium VM) in India-OpenStack.

## Important Experiment Factors

When designing the test cases, you may configure the following factors and build your own combination:

* VM Instance class (like m1.medium)
* Number of worker nodes
* Total number of cores/CPUs/processes
* Size of dataset (# of URLs, # of groups)
* Number of MPI processes (set with argument -np)
* MPI PageRank delta threshold
* Number of iterations
* Others, if any

## Execution time and Speedup

Each test case should execute three times and take the average of those three runs. In order to calculate speedup, you may need to change the amount of work nodes and total number of cores. In addition, you may also increase or decrease the data size to achieve greater understanding about the program and system behaviors.

The definition of speedup is:

* *p* is the total number of processes
* is the execution time of the sequential [algorithm](http://en.wikipedia.org/wiki/Algorithm)
* is the execution time of the [parallel algorithm](http://en.wikipedia.org/wiki/Parallel_algorithm) with p [processors](http://en.wikipedia.org/wiki/Central_processing_unit)

**You can use the provided PageRankDataGen class to generate the URLs.**

java PageRankDataGen <output\_file> <numOfUrls> <groupid>

**For example here is the command to generate a file with 1000 URLs**

java PageRankDataGen pagerank.input.1000 1000 1

Also you can download some data from <http://salsahpc.indiana.edu/csci-p434-fall-2013/apps/PageRankInput/>.

# FutureSystems OpenStack

As mentioned above, virtual clusters offer more power to run applications on a shared environment. OpenStack is a virtual infrastructure management tool that provides the functions to manage and access those virtual resources. With the help of OpenStack, we can easily obtain a private virtual cluster (Cloud) on a set of machines. FutureSystems has deployed OpenStack and provided service to its users for years. Figure 1 shows the compute resources available from the FutureSystems project. We will obtain VMs and run applications on India-OpenStack in the following section.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | System Type | # Nodes | # CPUs | # Cores | TFLOPS | RAM (GB) | Storage (TB) | Site |
| juliet | SuperMicro HPC Cluster (available Summer 2015) | 128 | 256 | 3456 |  | 16384 | 1024 HDD; 50 SSD | DSC Internal |
| india | IBM iDataPlex | 128 | 256 | 1024 | 11 | 3072 | 335 | DSC |
| bravo | HP Proliant | 16 | 32 | 128 | 1.7 | 3072 | 128 | DSC |
| delta | SuperMicro GPU Cluster | 16 | 32 | 192 |  | 1333 | 144 | DSC |
| echo | SuperMicro ScaleMP Cluster | 16 | 32 | 192 | 2 | 6144 | 192 | DSC |
| madrid | Dell HPC Cluster | 8 | 32 | 128 |  | 384 | 28.8 HDD; 6.8 SSD; | DSC Internal |
| tempest | HP Proliant HPC Custer | 32 | 128 | 768 |  | 1536 | 25 | DSC Internal |

Fig. 1 Overview of FutureSystems Compute Resources

This section shows the steps to access India Cluster HeadNode on FutureSystems. From there, we obtain a set of VM nodes as our virtual cluster.

1. Access FutureSystems India with your ssh private key.

ssh –i [ssh private key] [username]@india.futuresystems.org

Enter passphrase for key 'ssh\_id\_rsa':

[mengli@i136 ~]$

Here, “i136” is the headnode (first login node) of India cluster. We will obtain an OpenStack Virtual Cluster from this node. **DO NOT** run your MPI program on the “i136” headnode.

1. Load the OpenStack tool.

$ module load openstack

1. An initial openrc file is currently created for you automatically and can be activated with:

$ source **~/.cloudmesh/clouds/india/kilo/openrc.sh**

1. Test the nova client tools :

# see available virtual images

$ nova image-list

+--------------------------------------+--------------------------+--------+--------------------------------------+

| ID | Name | Status | Server |

+--------------------------------------+--------------------------+--------+--------------------------------------+

| 9eb8416d-1313-4748-a832-5fe0ecbbdffc | Ubuntu-14.04-64 | ACTIVE | |

| f51bd217-f809-46a1-9cdb-604d977ad4e9 | Ubuntu-15.10-64 | ACTIVE | |

| 1a80ac5b-4e57-479d-bed6-42e1448e6785 | cirros | ACTIVE | |

| 2cb17a9f-f62d-4ba2-805c-d988bcbf7c66 | fg486/ubuntu-mpj | ACTIVE | d64d3720-d0e8-40c8-bbda-6d3ce44f8bbf |

| 870fc84f-5c64-44e3-9fd6-88016c7bb4e9 | nist-nbis-03Nov2015 | ACTIVE | b5494855-b669-45c6-8828-b1775dbc5899 |

| 27417e29-f178-401c-bd3f-9eea716ea801 | stock-analysis-02Nov2015 | ACTIVE | 578ab013-1240-419d-bee1-3e31b8381c35 |

+--------------------------------------+--------------------------+--------+--------------------------------------+

# see available instance type

$ nova flavor-list

+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+

| ID | Name | Memory\_MB | Disk | Ephemeral | Swap | VCPUs | RXTX\_Factor | Is\_Public |

+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+

| 1 | m1.tiny | 512 | 1 | 0 | | 1 | 1.0 | True |

| 2 | m1.small | 2048 | 20 | 0 | | 1 | 1.0 | True |

| 3 | m1.medium | 4096 | 40 | 0 | | 2 | 1.0 | True |

| 4 | m1.large | 8192 | 80 | 0 | | 4 | 1.0 | True |

| 5 | m1.xlarge | 16384 | 160 | 0 | | 8 | 1.0 | True |

+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+

#Everything is fine if you see outputs similar to above.

1. Set up the VM keypair and allow ssh access to VM (you only need to do this step one time):

After testing the nova client, add an ssh keypair using nova keypair-add for future access to the VM. Note that without setting this correctly, you will not able to ssh into your VMs:

Command: nova keypair-add [new public key name]> [new private key name]

# create your own Nova ssh keypair for your future started instance (do this only for the first time)

$ nova keypair-add $USER-fg486-vm-sshPublickey > ~/.ssh/$USER-fg486-vm-sshPrivatekey

$ chmod 600 ~/.ssh/$USER-fg486-vm-sshPrivatekey

$ nova keypair-list

+------------------------------+-------------------------------------------------+

| Name | Fingerprint |

+------------------------------+-------------------------------------------------+

| $USER-fg486-vm-sshPublickey | 93:50:96:f8:57:2c:c4:2a:7a:c7:9f:c9:f6:07:c5:b4 |

+------------------------------+-------------------------------------------------+

In the next step we need to make sure that the security groups allow us to log into the VMs. To do so we create the following policies as part of our default security policies. Note that when you are in a group project this may already have been done for you by another group member. We will add ICMP and port 22 on default group:

# add security group for SSH connection

$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0

$ nova secgroup-list-rules default

|+-------------+-----------+---------+-----------+--------------+

| IP Protocol | From Port | To Port | IP Range | Source Group |

+-------------+-----------+---------+-----------+--------------+

| | | | | default |

| | | | | default |

| icmp | -1 | -1 | 0.0.0.0/0 | |

| tcp | 22 | 22 | 0.0.0.0/0 | |

+-------------+-----------+---------+-----------+--------------+

## Start and log into an OpenStack VM

Finally, we start two m1.medium OpenStack VMs and construct our cluster.

# set the net list variable for using the internal network

$ NET\_ID=$(nova net-list | awk ‘/ fg486-net / {print $2}’)

command: nova boot --flavor [instance type] --image [image name/id] –nic [net-id=net-uuid] --key\_name [VM public key name] --num-instances [amount of starting instances] [instances prefix]

# Boot 2 OpenStack Nova virtual compute nodes

$ nova boot --flavor m1.medium --image "fg486/ubuntu-mpj" --nic net-id=$NET\_ID --key\_name $USER-fg486-vm-sshPublickey $USER-001

$ nova boot --flavor m1.medium --image "fg486/ubuntu-mpj" --nic net-id=$NET\_ID --key\_name $USER-fg486-vm-sshPublickey $USER-002

Here, please refer to [FutureSystems OpenStack Manual](http://cloudmesh.github.io/introduction_to_cloud_computing/iaas/openstack.html) [6] for details about the available instance classes. Please check and wait for the instance status to become “ACTIVE”. Once you get the VM running, you will get an internal IP assigned to each instance, for example, “10.0.4.9” and “10.0.4.10”. At the end, you can log in as an ubuntu user with your created ssh private key.

# See the instances running and the instances you start (they have the userid)

$ nova list

Now login to one of the instances using its private IP address.

$ ssh -i ~/.ssh/$USER-fg486-vm-sshPrivatekey ubuntu@10.0.4.10

It will take a while for you to be able to login.

If you cannot login to the instances using private IP addresses, you need to associate a floating IP address to each instance.

First, create a floating IP address:

$ nova floating-ip-create ext-net

+--------------------------------------+-----------------+-----------+----------+---------+

| Id | IP | Server Id | Fixed IP | Pool |

+--------------------------------------+-----------------+-----------+----------+---------+

| 59e79084-55da-4b4b-a59a-fab8c00d50e3 | 149.165.159.176 | - | - | ext-net |

+--------------------------------------+-----------------+-----------+----------+---------+

Then, add the IP address to your instance:

$ nova floating-ip-associate $USER-001 149.165.159.176

Check your floating ip list to see if the ip address is added to your instance:

$ nova floating-ip-list

+--------------------------------------+-----------------+--------------------------------------+-----------+---------+

| Id | IP | Server Id | Fixed IP | Pool |

+--------------------------------------+-----------------+--------------------------------------+-----------+---------+

| 59e79084-55da-4b4b-a59a-fab8c00d50e3 | 149.165.159.176 | f054143d-10b4-43e5-ad5d-f0d524a9822a | 10.0.4.9 | ext-net |

| d402efc8-d102-4396-9262-a56fe7499652 | 149.165.159.178 | 478e3efa-6338-4f3b-acf3-654490e0d2e6 | 10.0.4.10 | ext-net |

+--------------------------------------+-----------------+--------------------------------------+-----------+---------+

Then you can login to the instances using their public IP addresses.

$ ssh -i ~/.ssh/$USER-fg486-vm-sshPrivatekey ubuntu@149.165.159.176

## Terminate your instance(s) after finishing your task

Due to the FutureSystems OpenStack capacity, there are limited resources allotted to our class. Therefore, please shut down the VM when finished running your test in order to let others run theirs.

Command: nova delete [instance ID]

$ nova list

$ nova delete $USER-001

# 6. Running MPJ on two instances

You need to install MPJExpress on the FG VM. And you need to scp your MPJ PageRank solution to the VMs before you can run the program. To SCP the program to VMs, first you need to send the files to FG head node.

scp <your project in a zip file> username@india.futuresystems.org:

Then login to FG and scp the files to both VMs you created. For example, here is a command used to scp the files. Note you need to change these commands to suit your file names, user names, ip addresses, etc.

$ scp -i ~/.ssh/$USER-fg486-vm-sshPrivatekey project2A.zip ubuntu@10.0.4.9:

$ scp -i ~/.ssh/$USER-fg486-vm-sshPrivatekey projects2A.zip ubuntu@10.0.4.10:

Unzip the program in the VMs. You will need to compile the MPJ PageRank here. Make sure to have your program in the same place in both VMs.

For example here Project2 is in /home/ubuntu/Project2A in both VMs.

Also you need to compile your program in both VMs using the command:

$ javac -cp .:$MPJ\_HOME/lib/mpj.jar MPIPageRank.java

**Single Machine**

To execute the program with MPI, use the following command in the homework directory:

mpjrun.sh -np 8 MPIPageRank pagerank.input.1000.0 output.txt .001 .85 30

-np means the number of processes. In the above case you are creating 8 processes and executing the algorithm in parallel, but this all happens in a single machine. If it reports “java.net.UnknownHostException”, modify /etc/hosts by adding a pair of ip-address and hostname:

sudo vim /etc/hosts

Then add the pair of ip-address and hostname, like:

10.0.4.9 xxx-001

## Run MPJ with multi-nodes mode

First, copy the private key to each instance:

$ scp -i ~/.ssh/$USER-fg486-vm-sshPrivatekey ~/.ssh/$USER-fg486-vm-sshPrivatekey ubuntu@149.165.159.176:.ssh/id\_rsa

Test if you can login to one instance from the other:

$ ssh 10.0.4.9

Now go to the MPJ Folder.

$ cd mpj-v0\_38/bin

Before running any MPJ job in multiple machines, you will have to make a “machines” file, which has the worker hostname written line by line, e.g. 10.0.4.9 and 10.0.4.10. Note that the filename must be “machines.” You need to put the IP address of the current machine at the bottom of the “machines” file. In FutureSystems these are the private IPs of the machines created, as shown below where the machines file is in $USER-001 (10.0.4.9):

$ vi machines

10.0.4.10

10.0.4.9

Once you have set up the “machines” file, you can use mpjboot to start MPJ daemons on worker nodes. Here, an IU passphrase may be required as you are starting the mpj daemons remotely.

$ mpjboot machines

Check if it works.

$ jps

1737 WrapperSimpleApp

1757 Jps

If the WarpperSimpleApp doesn’t exist, then the mpjboot command doesn’t work.

If the mpjboot command doesn’t work, please use the command below. You will need to run this command in both VMs.

$ mpjdaemon\_linux\_x86\_64 start

You will also need to copy the “machines” file to the Project2A directory.

Then you can run MPJ PageRank by adding “-dev niodev” parameters to start your application across worker nodes. Note: the only difference from the previous command you use to run the MPJ in a single machine is the –dev niodev from the previous command. Now go to your Project2A folder and execute the following command:

mpjrun.sh -dev niodev -np 8 MPIPageRank pagerank.input.1000.0 pagerank.out .001 .85 30

After you are done experimenting, execute the following command to clean up the resources.

mpjhalt.sh machines

# 7. Acknowledgement

FutureGrid was a national scale, NSF-funded project which provided a capability that made it possible for researchers to tackle complex research challenges in computer science related to the use and security of grids and clouds.

# 8 References

1. <https://portal.futuresystems.org/>
2. [http://manual.FutureSystems.org/openstackgrizzly.html](http://manual.futuregrid.org/openstackgrizzly.html)
3. <http://mpj-express.org/docs/guides/linuxguide.pdf>
4. <http://en.wikipedia.org/wiki/PageRank>
5. <http://salsahpc.indiana.edu/csci-p434-fall-2013/apps/PageRankInput/>
6. <http://cloudmesh.github.io/introduction_to_cloud_computing/iaas/openstack.html>