High Performance Computing [1]

High Performance Computing (HPC) enables scientists and researchers to solve complex problems that require many computing capabilities. HPC typically utilizes a message passing interface (MPI) to communicate between different nodes. Traditionally, this could only be done on dedicated, hard wired HPC cluster systems, often requiring their own dedicated, long term data center facilities.

HPC in Cloud

Currently, most projects requiring HPC are still running on these legacy, hard wired, dedicated HPC clusters. Migrating such projects to a Cloud-based installation has big paybacks in the form of infrastructure savings, flexibility and management.

In this tutorial, we will build a four node HPC cluster with Open MPI on the Rackspace Cloud. Next, we will run an Open MPI application on top of our cluster and illustrate how to leverage the Cloud based technologies to rapidly build and scale an HPC cluster for real-time data processing while removing the dependency on physical infrastructure.

Open MPI

To achieve high performance clustering in the Cloud, we will use Open MPI, which is a Message Passing Interface project. It provides parallel processing, thread safety and concurrency, dynamic process spawning, and network and fault tolerance. This library is used by the world's fastest super computers and is instrumental in powering many petaflops. To find out more about Open MPI library, visit their site: http://www.open-mpi.org/.

Objective

In this tutorial, we will show you how to build an HPC cluster using the following:

1. Four Rackspace Cloud Servers
2. Open MPI

Prerequisites

The following prerequisites are expected for successful completion of this tutorial:

- Rackspace Cloud account (https://developer.rackspace.com/signup/ if you don't have one)*
- SSH client (Mac & Linux clients are fine. Windows users download PuTTY from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)
- A basic knowledge of Linux is a plus

***NOTE: The developer+ program gives you a $50/month credit to setup and run Rackspace cloud resources. The resources needed for this HPC workshop could add up to over $200/month if not deleted once we are done, so do not leave more than one smaller (1G) cloud server up and running for long or you could incur necessary charges.**

Installation Process

In this tutorial, we will be setting up a four-node cluster, running applications on it, and gauging the performance.

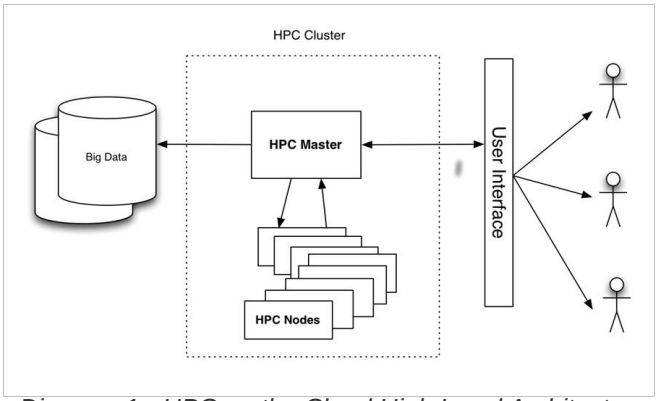

Diagram 1 - HPC on the Cloud High Level Architecture

Overview

In this tutorial, we will show you how to:

1. Create a Cloud Server (via Cloud Control Panel https://mycloud.rackspace.com/)
2. Install Open MPI
3. Use the cloud
4. Enable Clustering
5. Configure HPC
6. Create and deploy a Cloud Server image
7. Install and run a sample Open MPI enabled application

1. Create the first Cloud Server Cluster Node

Login to https://mycloud.rackspace.com and create a Cloud Server from the web interface with the following attributes.

1. Server name: *yourname*-HPC-node-01
2. Region: Northern Virginia (IAD))
3. Image (Linux): Ubuntu 12.04 LTS (Precise Pangolin, PVHVM)
4. Size: 2GB of RAM (for this workshop)
5. Click: Create Server

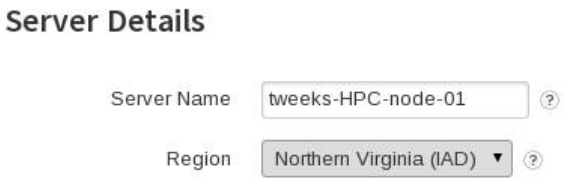

Figure 1 – Creating Your First Server

After this, the Cloud Server will begin building. During this time, a popup will appear with the Cloud Server password. Record the root password as you will need it later. Dismiss the popup window and wait for the server build to complete. Write down IP address for the server as it becomes available.

2. Install Open MPI

Once the server finishes building and is in Available status, SSH into it and log in using the IP address and password you recorded earlier.

```
ssh root@<Your Server IP>          # Logs you in as username@ip-address
...
root@tweeks-hpc-node-01:~#         # <-- your server's login prompt
```

NOTE: In this lab, **`bold mono text`** is what you type, `light mono` text is either comments or terminal output.

After logging in, execute the following commands:

```
apt-get update                     # This installs all the OS updates
apt-get install build-essential -y # This installs all the gss/build tools
apt-get install openmpi-bin openmpi-checkpoint openmpi-common openmpi-doc
libopenmpi-dev vim curl -y
                                   # Installs all the openmpi packages
```

3. Enable Clustering Over SSH

As mentioned earlier, Open MPI facilitates communication between nodes via SSH, therefore, we will need to enable key-based logins for SSH.

To enable clustering over ssh, run the following commands:

```
echo "StrictHostKeyChecking no" >> /etc/ssh/ssh_config  # Enable keychecking
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa -C "Open MPI" # Generate ssh keys
```

Note: You will be prompted for a passphrase twice during this process. Leave it blank.

```
Generating public/private rsa key pair.        # output of ssh-keygen
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
35:85:97:3c:98:89:8d:bc:58:96:97:41:ad:0b:a6:c8
Enter an optional comment about your key
The key's randomart image is:
+--[ RSA 2048]----+
|       . *oX..   |
|        B O.*    |
|       + ooo .   |
|      . +...     |
|   . . oS. .     |
|    E .   .      |
|                 |
|                 |
|                 |
+-----------------+
```

Copy your pub key to authorized key file and change permissions to allow SSH logins:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys      # authorize your pubkey
chmod 600 ~/.ssh/authorized_keys                     # fix permissions
```

4. Configure HPC

Now we are going to configure the master HPC node by creating a host file. To do this, ensure you are logged in to the first node over SSH and create the following file where <Your Server IP> is the IP address you used to SSH into the machine:

```
cd ~                                 # changes dir into root's home dir /root/
echo <node-01 IP>  >> mpi_hosts      # put your node's IP in a new host file
```

To verify the file, cat it out and verify the IP is correct:

```
cat mpi_hosts                        # You should see your actual node-01 IP
<node-01 IP>
```

To verify we have configured everything correctly so far, we will download, compile and use the hello_c.c from the examples included with Open MPI website. To do this, follow these steps:

```
mkdir /root/samples
cd /root/samples
wget http://svn.open-mpi.org/svn/ompi/tags/v1.6-series/v1.6.4/examples/hello_c.c
```

Now compile and run the MPI hello program

```
mpicc hello_c.c -o hello            # compile hello program
ls
hello  hello_c.c
mpirun ./hello                      # run the hello program
Hello, world, I am 0 of 1           # good output
```

If that hello program looks good, then download, build and test the second example which we will use for testing cluster connectivity (from the first node to the first node):

```
wget http://svn.open-mpi.org/svn/ompi/tags/v1.6-series/v1.6.4/examples/connectivity_c.c
mpicc connectivity_c.c -o connectivity              # Compile the connectivity test
ls                                                  # Get a directory listing
connectivity  connectivity_c.c  hello  hello_c.c
mpirun ./connectivity                               # Run the connectivity test
Connectivity test on 1 processes PASSED.ocesses PASSED.     # Looks good.
```

Now that we have confirmed that the first node is online and operational, we will finish building the cluster using an image of this first configured node, greatly saving additional setup and configuration time.

5. Create and Deploy a Cloud Server Image to create Additional Nodes

With our first node created, we are ready to set up the rest of our four node cluster by cloning the node we just created.

Login to https://mycloud.rackspace.com again and follow these steps to create an image:

1. Navigate to the servers list
2. Select the server you created for the first node
3. Click on the Actions drop down menu
4. Click Create Image. (see Figure 2 for details.)
5. When prompted, provide a meaningful name as shown in Figure 3.
6. Finally, click Create Image and wait a few minutes for the image to be created.

Figure 2 - Creating a Server Image From Your Server

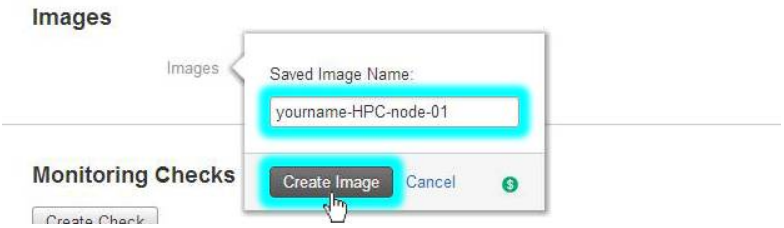

Figure 3 - Naming the Cloned Image

One the image creation completes (can take several minutes), go ahead and create a new Cloud Server called *yourname-HPC-node-02* using same procedure in step "**1. Create the first Cloud Server Cluster Node**", with the following exception; when prompted for an OS image, click the **Saved**, **Parent Server**, and the **image-name** you just created. Again, for this new node you will need to provide a meaningful name (such as *yourname-HPC-node-02*), and don't forget to record the password and IP address of this server also.

TIP: When flipping between cluster node ssh sessions, take note of the root login prompt. It will tell you who you are and where you are logged in, as seen here:

```
root@tweeks-hpc-node-01:~#
```

If you are going to flip between several systems, you might want to bring up a separate terminal for each node and order them from left to right, or use a multi-terminal poweruser utility such as screen or tmux.

For example, after creating your second node if your new node-02 IP is 10.20.30.40, and the IP of your node-01 server is <node-01 IP> -- to add the new node to the cluster, do the following:

```
# If not already logged into node-01, ssh back in as root and add the new node-02 IP to the mpi_hosts file
cd ~                                    # Puts you in to root's home dir /root/
echo 10.20.30.40 >> mpi_hosts           # Appends the new node's IP to the list of nodes
```

Or if you would prefer, edit the /root/mpi_hosts file with the nano or vim editors to add the new IP line.

Now, your host file one the node-01 system should resemble this:

```
cat mpi_hosts                           # Cats out the file
<node-01 IP>
10.20.30.40
```

If that all looks correct, then test the connectivity between the nodes, execute the following command from node-01:

```
mpirun -v -np 2 --hostfile ~/mpi_hosts /root/samples/connectivity
...
Connectivity test on 2 processes PASSED.
```

This runs the connectivity job, -v(erbosely), -np 2 runs two copies, using the hosts listed in /root/mpi_hosts.

NOTE: The first time you do this, you may get a warning such as:

Warning: Permanently added '<IP>' (ECDSA) to the list of known hosts.

This is normal and is a part of the ssh client housekeeping the first time it connects to other nodes.

If you don't get any hard errors, you have just successfully created and tested your own tiny, two node cluster "in the cloud"!

To increase the size of the cluster, add two more nodes using the same procedure, giving them the names node-03 and node-04 and adding their IPs to node-01's hosts file also.

TIP: Once you start node-03 building, you can go on to the next node-04 and begin that one building too.

Figure 4 – It can take a while to build the nodes from your image.

Once all four nodes are up, and their IPs are all plugged in to node-01's mpi_hosts file, test the connectivity between the four-node cluster by executing the following command from node-01:

```
# From node-01
mpirun -v -np 4 --hostfile ~/mpi_hosts /root/samples/connectivity
...
Connectivity test on 4 processes PASSED.
```

6. Install and Run a Sample Open MPI enabled Application

Now that we have an Open MPI cluster, let's see how it performs. We will use a simple ray tracing application that can run on a single node or on an Open MPI cluster and compare the performance.

First, we will need to install this application on all nodes of the cluster. To do this, SSH into the master node-01 and do the following:

```
# From node-01
for i in `cat mpi_hosts`; do ssh root@$i "curl -l
http://openstack.prov12n.com/files/tachyon.sh | bash"; done
```

Lots of content will scroll by, but the little *one liner* sets up a simple for loop that runs through each of our node IPs in mpi_hosts and sequentially downloads, executes and compiles the tachyon ray tracing package on each one of our nodes.

TIP: If you plan, as most do, on having many identical cluster nodes, then this sort of push-based software trickery can quickly become daunting (think versioning and node system states). You should eventually either standardize your node images by either using centralized NFS (network) OS mounts, by occasionally reprovisioning cloud images as we have done to set up our nodes here, or by using a centralized configuration control system such as Puppet, Chef or Ansible. Any of these methods can easily keep all of your software versions and configurations in sync across your nodes.

The Tachyon Parallel / Multiprocessor Ray Tracing System comes with multiple sample data files in the scenes folder, we will be using them to run our tests. First, lets run it on one node:

```
cd ~/tachyon/compile/linux-mpi
./tachyon ../../scenes/teapot.dat
```

You should see the following output:

```
Tachyon Parallel/Multiprocessor Ray Tracer   Version 0.99
Copyright 1994-2011,    John E. Stone <john.stone@gmail.com>
-------------------------------------------------------------
Scene Parsing Time:     0.0123 seconds
Scene contains 2330 objects.
Preprocessing Time:     0.0020 seconds
Rendering Progress:       100% complete
  Ray Tracing Time:     0.9733 seconds    # <---------------------
    Image I/O Time:     0.0086 seconds
```

These are powerful cloud nodes, so the whole 3D image rendered in less than 1 second. So let's increase the work load a bit by increasing the resolution of this ray traced image from 512x512 pixels to 5000x5000 pixels, which should make this single node at least break a sweat. This is easily done on our node-01 system with this command:

```
sed -i 's/512 512/5000 5000/' /root/tachyon/scenes/teapot.dat
```

Now the test works much harder, taking almost 1.4 minutes for one node to complete as seen here:

```
cd ~/tachyon/compile/linux-mpi
./tachyon ../../scenes/teapot.dat
Tachyon Parallel/Multiprocessor Ray Tracer   Version 0.99
Copyright 1994-2011,    John E. Stone <john.stone@gmail.com>
-------------------------------------------------------------
Scene Parsing Time:     0.0128 seconds
Scene contains 2330 objects.
Preprocessing Time:     0.0021 seconds
Rendering Progress:       100% complete
  Ray Tracing Time:    83.6221 seconds   # <--------------
    Image I/O Time:     0.6093 seconds
```

A much better (tougher) test job now.

Now, using another one of those for-loops we can toughen up that teapot.dat file on all of our nodes accordingly:

```
# From node-01
cd ~
for i in `cat mpi_hosts`; do ssh root@$i "sed -i 's/512 512/5000 5000/'
/root/tachyon/scenes/teapot.dat"; done
```

NOTE: No errors means it worked.

Note the ray tracing time as we will compare it to our parallel run:

```
# Run a four node raytrace from node-01
cd /root/tachyon/compile/linux-mpi/
mpirun -np 4 --hostfile ~/mpi_hosts ./tachyon ../../scenes/teapot.dat
-format BMP
```

You should see something similar to the following output:

```
Tachyon Parallel/Multiprocessor Ray Tracer   Version 0.99
Copyright 1994-2011,    John E. Stone <john.stone@gmail.com>
-------------------------------------------------------------
Scene Parsing Time:     0.0125 seconds
Scene contains 2330 objects.
Preprocessing Time:     0.0033 seconds
Rendering Progress:       100% complete
  Ray Tracing Time:    26.5774 seconds  # <----------------
    Image I/O Time:     0.5890 seconds
```

Our cluster consisted of four nodes and one CPU each, therefore, the performance improvement was almost four times greater.

Summary

In this tutorial, you learned how to create and image Cloud Servers and how to setup an HPC four node cluster using Open MPI. After setting up and configuring the cluster, you installed a small ray tracing application to demonstrate the benefits of using multiple nodes instead of one maching.

The advantages of 100% software infrastructure clusters should now be much more apparent. On a simple four node cluster (based on a clone of our first node), we saw a 4x computational increase with a heavy number crunching application. As this sort of power is scaled up to 1,000 or even 10,000 cloud nodes, you can now start taking on much much larger computational tasks -- without the need for a hard wired, inflexible, old-style dedicated and single purpose computational clusters.

[1] – Based on the CreativeCommons, Rackspace Cloud HPC knowledge base article:
http://www.rackspace.com/knowledge_center/article/high-performance-computing-cluster-in-a-cloud-environment

Authored by the Rackspace Private Cloud Team + Egle.Sigler@rackspace.com
Heavily modified by tweeks@rackspace.com for the VBI / NSF HPC Summer workshops, 2014-08-05