SCS Technical Staff Document

Maintained by: Andrew Pullin, Senior Software Designer, School of Computer Science
andrew.pullin@carleton.ca / x4338 / HP5165

Last Updated: September 18, 2018

SCS OpenStack and MPI

Table of Contents

Maintained by: Andrew Pullin, Senior Software Designer, School of Computer Science
andrew.pullin@carleton.ca / x4338 / HP5165

Last Updated: September 18, 2018

MPI on OpenStack

A common use for Openstack is MPI programming. Openstack allows the user to quickly launch a large number of instances on which to run the MPI code.

Openstack Dashboard

The Openstack dashboard provides a web interface for interacting with the resources for you course. This includes creating instances, allocating floating IPs, etc.

REMEMBER: To access the Openstack dashboard, or to access an Openstack IP address, you must be on the Carleton network. If you are off-campus, you will need to VPN.

Openstack Dashboard: https://openstack.scs.carleton.ca/

SCS Openstack Account Creation

If you have NOT setup your SCS Openstack account, you must do so using the SCS Account Create / Update tool. This tool will confirm that you are in a valid Openstack course and add you to that course.

REMEMBER: You need to run the SCS Account Create / Update tool **EVERY TERM** that you have an Openstack course. Openstack accounts are not permanent like other SCS accounts, and must be recreated every term as required.

SCS Account Creation Link: https://www.scs.carleton.ca/webacct

Steps to setup MPI

The following sections outline the steps required to setup MPI in the Openstack environment. It is based on the requirements for **COMP4009**, using the image provided by the course instructor that was created to support a number of **MPI** and **cilk** scenarios.

Launch desired number of instances (nodes)

Use the Openstack dashboard to launch the instances

- Click on **Project -> Compute -> Images** to view the images for the course. Normally the course image will appear in either the **Project** tab or the **Shared with Me** tab. Check with your faculty member if you are unsure which image to use.
- Click on the **Launch Image** action button for the image
- Fill in the required information in the **Details** tab AND the **Access & Security** tab:
    - **Details tab:** Give the instance a name (a number will be appended to each instance if you launch multiple); Select the **Flavor** based on the requirements for vCPU and Memory; Specify the **Instance Count**, which should be the number of MPI hosts you wish to create; The other fields will be pre-populated.
    - **Access & Security tab:** You most likely do NOT require a **Key Pair** as a username is provided for most course VMs; Under **Security Groups**, select both the **default** group and the **ping-ssh-web** group (or similar);
    - Click **Launch**

The launch time will vary. If an image is pre-staged on an openstack compute node, it can launch in just a few seconds. If the image has not been pre-staged, it could take 5 minutes or so to load. You can click on the **Compute -> Instances** tab to see the status of the instances you just launched

Designate one instance as the "master node"

Once the instances have launched, you need to choose one to be the "**master node**". You can click **Edit Instance** on the desired instance and change its name if you wish. The other instances will be termed "**slave nodes**"

You must also assign it a publicly accessible IP address.

1. Click **Associate Floating IP** on the desired instance
2. Select an available **IP Address** from the drop-down menu. It should be a **134.117.xxx.xxx** address; Do NOT change the **Port to be associated**;
3. Click **Associate**

The chosen instance should now show a Floating IP address matching the one you just selected.

You can now ssh to that instance using the provided credentials:

```
ssh user@134.117.31.89
```

Create RSA key to allow ssh from master node to slave nodes

Next we need to create an RSA key to allow password-less SSH between the nodes. You should make note of the **Private IP addresses** assigned to each of your instances as indicated on the dashboard. The Private IP addresses are similar to **192.168.xxx.xxx**. They may or may not be sequential depending on the IP addresses available when the instances were launched.

The following steps should be done as the **user** that will be running the MPI applications.

1. From the **master node** ssh to each of the **slave nodes** and then exit back to the **master node**. This quick step will populate with **known_hosts** file in the .ssh directory

   ```
   ssh user@192.168.13.24
   ssh user@192.168.13.25
   ssh user@192.168.13.26
   ssh user@192.168.13.27
   ```

   (Where 192.168.13.24 is assumed to be the **master node** and the other IPs are the **slave nodes**)

2. Create an RSA key:

   ```
   ssh-keygen -t rsa
   ```

   Do not change the key-file name and do not provide a passphrase. This will update the **id_rsa** private key and **id_rsa.pub** public key files in the .ssh directory.

   You should now go into the .ssh directory and list the contents to verify that these key files are there

   ```
   cd ~/.ssh

   ls -la
   ```

3. We now want to be able to ssh to all of the nodes (master and slave nodes) from the master node, so copy the public key using the following command:

   ```
   ssh-copy-id user@192.168.13.24
   ssh-copy-id user@192.168.13.25
   ssh-copy-id user@192.168.13.26
   ssh-copy-id user@192.168.13.27
   ```

Now, when you **ssh** or **scp** from the **master node** to the **slave nodes**, you will no longer require a password

Allow ssh between all nodes

If you want to be able to ssh between all nodes, you can perform these additional steps, which are performed on the master node and continue from the steps in the previous section (this also eliminates the need to perform step 3 above – *ssh-copy-id*).

1. We want this public key (**id_rsa.pub**) to be the default authorized public key, so we copy it to the **authorized_keys** file:

   ```
   cat id_rsa.pub >> authorized_keys
   ```

2. There should now be 4 files in the .ssh directory:

   ```
   authorized_keys  id_rsa  id_rsa.pub  known_hosts
   ```

3. Copy the **.ssh** directory to each of the slave nodes:

   ```
   scp -r .ssh user@192.168.13.25:~
   scp -r .ssh user@192.168.13.26:~
   scp -r .ssh user@192.168.13.27:~
   ```

Now, when you **ssh** or **scp** between any of the **master** or **slave nodes**, you will no longer require a password.

Update hosts file

This is an optional step. Rather than using an IP address, it may be easier to modify the **/etc/hosts** file so that you can identify nodes by easy to understand names.

1. Edit the **/etc/hosts** file using **sudo** (using nano or some other editor)**:**

   ```
   sudo nano /etc/hosts
   ```

   Add the following entries to the file

   ```
   192.168.13.24 master
   192.168.13.25 slave01
   192.168.13.26 slave02
   192.168.13.27 slave03
   ```

2. Copy the new hosts file to each of the systems. Because this file can only be modified by root (sudo), you will need to copy the file to the users home directory on the target system, then change the permissions and copy it to the /etc/hosts file on the target system:

   ```
   scp /etc/hosts slave01:~
   scp /etc/hosts slave02:~
   scp /etc/hosts slave03:~
   ```

3. SSH onto each of the slave nodes

   ```
   ssh slave01
   …
   ```

   And run the following commands to change the ownership of the file and copy it over the /etc/hosts file:

   ```
   chown root:root hosts
   mv hosts /etc/hosts
   ```

Update system hostnames

An additional step would be to rename the systems so that you can easily tell what system you are currently on:

1. Edit the /etc/hostname file and change the hostname to whatever hostname is desired; usually you would make it match the names specified for the /etc/hosts file in the previous section.

   ```
   sudo nano /etc/hostname
   ```

2. Reboot the system

   ```
   sudo reboot
   ```

Test an MPI "Hello World" application

The final step is to test an actual MPI application.

A sample "**Hello World**" application is provided here: https://carleton.ca/scs/wp-content/uploads/MPI.zip

1. Download the zip file to the master node:

   ```
   wget https://carleton.ca/scs/wp-content/uploads/MPI.zip
   ```

2. Unzip the file which will create an **MPI** directory.

   ```
   unzip MPI.zip
   ```

   There will be a **hostfile** a **Makefile** and a **MPI_hello.c** file.

3. Go into the **MPI** directory and type the **make** command to compile the **hello** application.

   ```
   make
   ```

   This will add a fourth executable application called "**hello**"

4. Copy the "**hello**" application to the **master node's** home directory and the home directory of each of the **slave nodes**

   ```
   cp hello ~
   scp hello slave01:~
   scp hello slave02:~
   scp hello slave03:~
   ```

5. Update the **hostfile** in the MPI directory to reflect the names of each of the nodes as indicated in the **/etc/hosts** file. If you did not update the **/etc/hosts** file, then you will need to use the IP addresses instead. The contents of the **hostfile** should look like this:

   ```
   master
   slave01
   slave02
   slave03
   ```

6. Now, run an MPI test to confirm that the code works on the **master node**. *(The mpirun command will have the -np option to specify the number of processors to use, the --hostfile option to point to a file that lists the MPI nodes, and finally a COMMAND at the end, in this case **hello**)*

   Move to the users home directory:

   ```
   cd ~
   ```

   Run the **mpirun** command (*-np 1* tells it to run on one processor):

   ```
   mpirun -np 1 --host master hello
   ```

   The output should be similar to this:

   ```
   Hello World!, I am processor with rank 0 on node master.
   ```

7. Now repeat it using the **hostfile** to run it on all of the nodes (*-np 4* tells it to run on at least four processors, which will ensure it runs on all four of our nodes):

   ```
   mpirun -np 4 --hostfile MPI/hostfile hello
   ```

   With the output being similar to:

   ```
   Hello World!, I am processor with rank 0 on node master.
   Hello World!, I am processor with rank 2 on node slave02.
   Hello World!, I am processor with rank 1 on node slave01.
   Hello World!, I am processor with rank 3 on node slave03.
   ```