Ubuntu Supercomputing Virtual Cluster Setup Guide

Definition of repository

This repository and guide is designed to guide the setup of a Ubuntu supercomputing cluster. This cluster consists of a basic Ubuntu Server install that is combined with the MPICH3 system. This gives the cluster MPI capability. By default OpenMP libraries are included with GCC which is also installed in the process of setting up MPICH3.

References

Notes

Legend for this document:

Proper named objects will appear as *Proper Named Object* text

Action items and examples will appear as **Action item** text

Code, plain text to be entered, or text from the command line will appear as `code` text

Linux commands that help throughout this guide:

`cd`
change directory followed by the name of the directory to change to **Example** `cd home`

`cd ..`
go back up one directory

`dir`
list all folders and files in current directory

`nano`
command to open Nano text editor

- All commands listed at the bottom of the Nano editor are executed using `Ctrl` + `<key listed>`
- `Ctrl` + `x` - exit, `y` to confirm, and `Enter` to verify filename
- `Ctrl` + `w` - search function; enter string to search for; `Ctrl` + `w` again to search for previous string
- `Ctrl` + `c` - cursor location; used to find exact line when bug tracing

`sudo` used to execute commands using the Super User; if you receive a warning about permissions when editing or executing commands try running that same command prefixed by sudo **Example** `sudo nano /etc/hostname`

`sudo service <name of service> start stop` or `restart` used to start, stop, or restart system services **Example** `sudo service networking restart`

`shutdown -h now` immediate system halt and shutdown

`reboot` initiate a system reboot

`apt-get update` pulls all update information from the internet; does not perform an update

`apt-get upgrade` performs an update based on information received from `apt-get update` command. Using the `-y` option performs the operation without querying the user to proceed with the install.

`apt-get dist-upgrade` performs an update of the system kernel based on information received from `apt-get update` command. Using the `-y` option performs the operation without querying the user to proceed with the install.

`ls` lists all directories and files in the current directory

`ls -l` lists all files and directories in the current directory with owner information, group information, and permissions

`ls -ld` lists information about the current directory including owner information, group information, and permissions

`~` indicates the home directory for the current user

`sudo -s` Super User mode; be careful using this mode as any commands executed under Super User will reflect the **root** user and not a user account

`/` denotes the root directory of the system

`./` denotes the running of an executable file **Example** `./install.sh`

`ssh` allows ssh-ing into other systems

Information provided by the command prompt:

There is a lot of information that you can gather just by looking at the command prompt. Using `user@somecomputer:~$` as an example we will examine the command prompt. To begin with we can tell which user account is currently logged in. In this case its `user`. Next we can tell which system we are logged in to. In this example it is `somecomputer`. Next is `~` which tells us which folder we are in on the system. As stated above `~` denotes the `user` home directory. Next we can tell if we are executing commands as a user or Super User by looking at the last character. In this case it is `$` which tells us we are executing with whatever permissions `user` has.

For another example we will use `root@somecomputer:/usr/local/#`. This example shows we are signed in as `root` user on `somecomputer` and we are in the directory `/usr/local/`. The first `/` always denotes the root directory. Next we can tell we are executing commands as a Super User (root always executes as Super User) by the `#`.

Set up Cluster Head Node

Step 1 - Install Ubuntu Server

Select **Install Ubuntu Server**

Select **English**

Select **United States**

Select **No** for *Detect keyboard layout*

Select **English (US)**

Select **English (US)**

Select **enp0s3** for *Primary network adapter*

Set *hostname* to **head**

Set *New user full name* to last name and first initial

Set *Username* to last name and first initial

Choose a password for your account or use your student id number

Agree to use weak password

Select **No** for *Encrypt your home directory*

Select **Yes** for *Is this time zone correct?*

Select **Guided - use entire disk and set up LMV**

Select the default drive

Select **Yes** to *Write the changes to disks and configure LVM*

Keep the default drive size

Select **Yes** to *Write the changes to the disk*

Leave *HTTP Proxy* empty and Continue

Select **No automatic updates**

Hit *Tab* to move the cursor to **OpenSSH server** and press `Space` to select it

Note: OpenSSH package is selected when a ** * ** is shown in the box under the cursor

Press `Enter` to continue the installation

Select **Yes** to *Install the Grub boot loader to the master boot record*

Step 7

Step 2 - Set Static IP Address for Secondary Connection

Start the *Head Node* and login using the username and password created during the install process

Edit the network interfaces file:

```
sudo nano /etc/network/interfaces
```

Add the secondary interface to the file:

```
# Secondary Interface - cluster connection enp0s8
auto eno2
iface eno2 inet static
address 192.168.10.5
netmask 255.255.255.0
network 192.168.10.0
```

Edit the hosts file:

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

Add to the end of the file:

```
192.168.10.5    head
192.168.10.100  node0
```

Save and exit

Step 3 - Set up IPv4 Traffic Forwarding

Enable traffic forwarding and make it permanent:

```
sudo nano /etc/sysctl.conf
```

Add the following to the end of the file:

```
# Enable IPv4 forwarding
net.ipv4.ip_forward = 1

# Disable IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Save and exit

Enable the new rules:

```
sudo sysctl -p
```

Enter iptables rules:

```
sudo iptables -t nat -A POSTROUTING -o eno1 -j MASQUERADE
sudo iptables -t nat -A POSTROUTING -o eno2 -j MASQUERADE
sudo bash -c "iptables-save > /etc/iptables.rules"
```

Edit */etc/network/interfaces* file:

```
sudo nano /etc/network/interfaces
```

Add the following line to the end of the file:

```
pre-up iptables-restore < /etc/iptables.rules
```

Save and exit

Now reboot the system:

```
sudo reboot
```

Step 4 - Update the system packages and kernel

```
sudo apt udpate && sudo apt upgrade -y && sudo apt dist-upgrade -y
```

Step 5 - Set up SSH key

VERIFY AT THE COMMAND PROMPT THAT YOU ARE UNDER YOUR USER ACCOUNT AND NOT EXECUTING CODE AS SUPER USER OR ROOT

Generate an SSH key:

```
cd ~
ssh-keygen -t rsa -C "edison@swosu"
```

Press `Enter` to select default install location

Press `Enter` to leave passphrase blank

Press `Enter` to confirm blank passphrase

Copy SSH keys to authorized keys:

```
cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

MPI

Step 1 - Create Directories

Install some required compilers and packages:

```
sudo apt-get install make build-essential
```

Create */software* directory:

```
sudo mkdir -p /software/lib/mpich_3.2
```

Create hpc user group:

```
sudo groupadd hpc
```

Add user to hpc user group:

```
sudo usermod -aG hpc <username>
```

Take ownership of */software*:

```
sudo chown -R <username>:hpc /software
```

Change to the *mpich_3.2* directory and create *build* and *install* directories:

```
cd /software/lib/mpich_3.2

mkdir build install
```

Step 2 - Download and install

Install prerequisites, download MPICH3 package and install:

```
sudo apt install gfortran
```

Download MPICH3 package:

```
wget http://www.mpich.org/static/downloads/3.2/mpich-3.2.1.tar.gz
```

Untar the package:

```
tar xvfz mpich-3.2.1.tar.gz
```

Change to *build* directory to begin building the install:

```
cd build
```

```
/software/lib/mpich_3.2/mpich-3.2/configure  --prefix=/software/lib/mpich_3.2/install
```

Compile the install:

```
make
make install
```

Add MPI location to system environment variable PATH:

```
export PATH=$PATH:/software/lib/mpich_3.2/install/bin
```

Make the PATH change permanent by adding it to the profile file:

```
sudo nano ~/.bashrc
```

Add the following to the end of the file:

```
export PATH="$PATH:/software/lib/mpich_3.2/install/bin"
```

Save and exit

Step 3 - Create Node List

Create a list of nodes for MPI to use:

```
cd ~

sudo nano nodelist
```

Add the *head node* ip address to the file:

```
192.168.10.5
```

Step 4 - Test MPI

```
cd ~
```

Test 1

```
mpiexec -f nodelist hostname
```

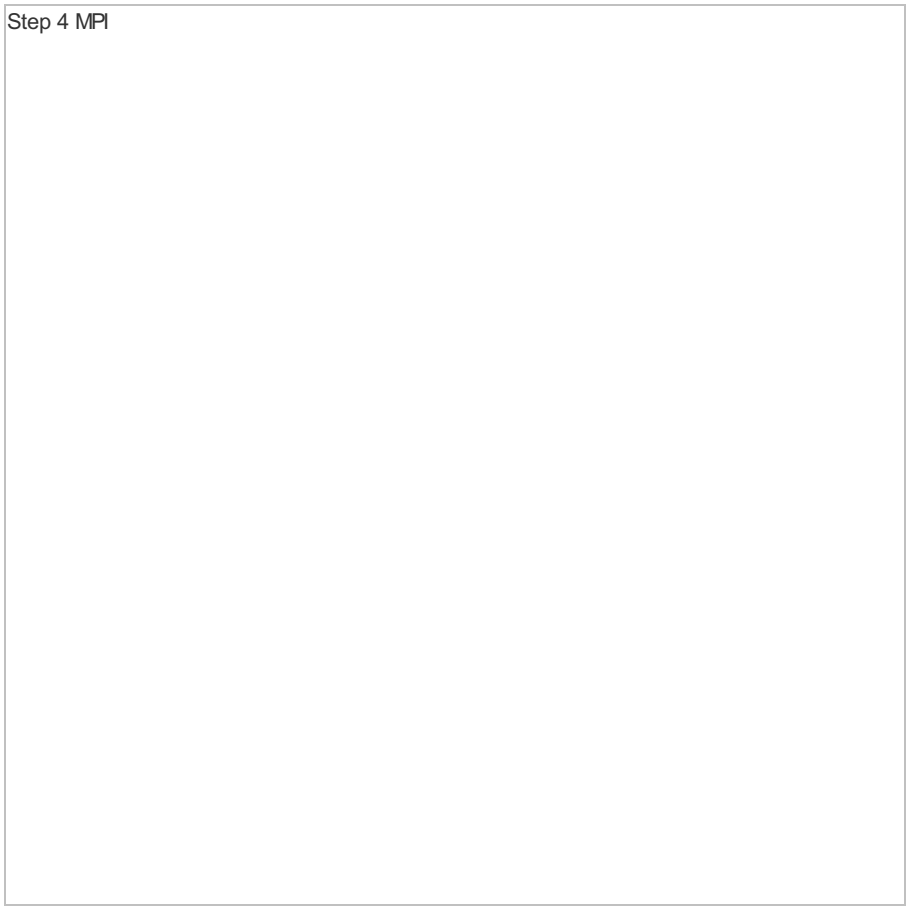
Output:

```
head
```

Test 2

```
mpiexec -f nodelist -n 2 /software/lib/mpich_3.2/build/examples/cpi
```

Ouput:



Set up Cluster Compute Node

Edit */etc/network/interfaces* file:

```
sudo nano /etc/network/interfaces
```

Edit the file by changing or adding the following:

```
iface eno1 inet static
address 192.168.10.100
netmask 255.255.255.0
gateway 192.168.10.5
dns-nameservers 8.8.8.8
```

Save and exit

Edit */etc/hostname* file:

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

Change:

```
head
```

To:
NOTE: This will be a sequential number that increases as you add nodes to the cluster.

```
node0
```

Save and exit

Edit */etc/hosts* file:

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

Change:

```
head
```

To:
NOTE: This will be a sequential number that increases as you add nodes to the cluster.

```
node0
```

Save and exit

Reset the network connection to apply the settings:

```
sudo ifdown eno1
sudo ifup eno1
```

Check the ip address of eno1:

```
ifconfig
```

Check that *eno1* connection has the ip address *192.168.10.100*.

MPI

Step 1 - Create Directories

Install some required compilers and packages:

```
sudo apt-get install make build-essential
```

Create */software* directory:

```
sudo mkdir -p /software/lib/mpich_3.2
```

Create hpc user group:

```
sudo groupadd hpc
```

Add user to hpc user group:

```
sudo usermod -aG hpc <username>
```

Take ownership of */software*:

```
sudo chown -R <username>:hpc /software
```

Change to the *mpich_3.2* directory and create *build* and *install* directories:

```
cd /software/lib/mpich_3.2

mkdir build install
```

Step 2 - Download and install

Install prerequisites, download MPICH3 package and install:

```
sudo apt install gfortran
```

Download MPICH3 package:

```
wget http://www.mpich.org/static/downloads/3.2/mpich-3.2.1.tar.gz
```

Untar the package:

```
tar xvfz mpich-3.2.1.tar.gz
```

Change to *build* directory to begin building the install:

```
cd build
```

```
/software/lib/mpich_3.2/mpich-3.2/configure  --prefix=/software/lib/mpich_3.2/install
```

Compile the install:

```
make
make install
```

Add MPI location to system environment variable PATH:

```
export PATH=$PATH:/software/lib/mpich_3.2/install/bin
```

Make the PATH change permanent by adding it to the profile file:

```
sudo nano ~/.bashrc
```

Add the following to the end of the file:

```
export PATH="$PATH:/software/lib/mpich_3.2/install/bin"
```

Save and exit

Step 3 - Create Node List

Create a list of nodes for MPI to use:

```
cd ~

sudo nano nodelist
```

Add the *head node* ip address to the file:

```
192.168.10.5
```

Step 4 - Test MPI

```
cd ~
```

Test 1

```
mpiexec -f nodelist hostname
```

Output:

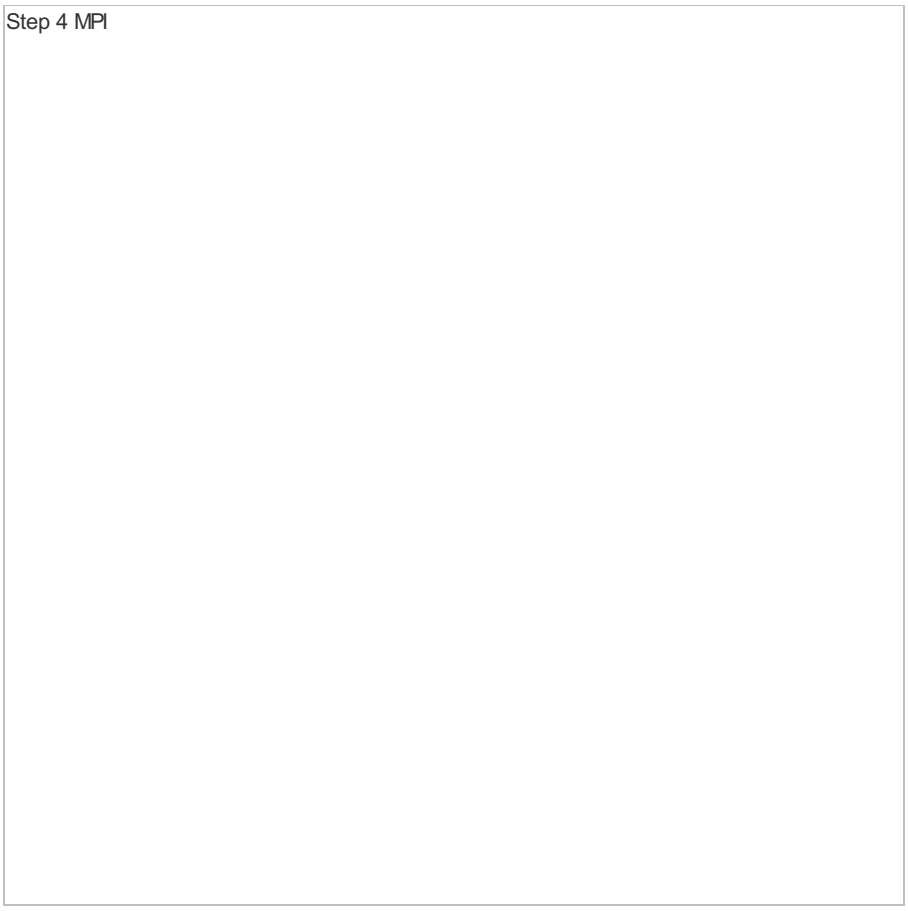
```
head
```

Test 2

```
mpiexec -f nodelist -n 2 /software/lib/mpich_3.2/build/examples/cpi
```

Ouput:

Step 4 MPI



Test the cluster so far

On the *Head Node* edit the nodelist:

```
cd ~

sudo nano nodelist
```

Add to the end of the file:

```
192.168.10.100
```

Save and exit

Test MPI:

```
cd ~
```

Test 1

```
mpiexec -f nodelist hostname
```

Output:

```
head
```

Test 2

```
mpiexec -f nodelist -n 2 /software/lib/mpich_3.2/build/examples/cpi
```

Ouput:

Step 4 MPI

Note: Each process shows which node it was executed on. You should see both head and node0 displayed, plus any others that you setup after this guide. This shows that MPI is sending and executing the script on both nodes in the cluster.

Congratulations! This cluster is ready to execute MPI code.

Adding more nodes

After completing the guide to this point you can now repeat the node setup process for setting up more compute nodes. There are three things to sequence. Those are: hostname, hostname in hosts file, and ip address in the interfaces file. Also make sure to add

any new nodes to the hosts file on all nodes and share the ssh key from the head node to all nodes for ease of access on the command line.

Slurm on Head Node

Step 1 - Install needed packages

Execute:

```
sudo apt-get install slurm-wlm slurmctld slurmd
```

Step 2 - Develop configuration file

Edit */etc/slurm-llnl/slurm.conf* and add or edit to match the follow ing:

```
#Virtual cluster slurm.conf
#
ControlMachine=head
ControlAddr=192.168.10.5
#
#MailProg=/bin/mail
MpiDefault=none
#MpiParams=ports=#-#
ProctrackType=proctrack/pgid
ReturnToService=2
SlurmctldPidFile=/var/run/slurm-llnl/slurmctld.pid
#SlurmctldPort=6817
SlurmdPidFile=/var/run/slurm-llnl/slurmd.pid
#SlurmdPort=6818
SlurmdSpoolDir=/var/lib/slurm
SlurmUser=slurm
#SlurmdUser=root
StateSaveLocation=/var/lib/slurm
SwitchType=switch/none
TaskPlugin=task/none
#
#
# TIMERS
#KillWait=30
#MinJobAge=300
#SlurmctldTimeout=120
#SlurmdTimeout=300
#
#
# SCHEDULING
FastSchedule=1
SchedulerType=sched/backfill
#SchedulerPort=7321
SelectType=select/linear
#
#
# LOGGING AND ACCOUNTING
AccountingStorageType=accounting_storage/none
ClusterName=raspi2
#JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/none
#SlurmctldDebug=3
SlurmctldLogFile=/var/log/slurm/slurmctld.log
#SlurmdDebug=3
SlurmdLogFile=/var/log/slurm/slurmd.log
#
```

```
#
# COMPUTE NODES
NodeName=head State=UNKNOWN
NodeName=node1 Procs=1 State=UNKNOWN
PartitionName=TEST Default=YES Nodes=head,node1 State=UP
```

==POSSIBLE CHANGES==

```
sudo mkdir /var/lib/slurm

sudo chown -R slurm:slurm /var/lib/slurm

sudo mkdir /var/log/slurm

sudo chown -R slurm:slurm /var/log/slurm
```

==END CHANGES==

Step 4 - Create Munge authentication keyboard

`sudo /usr/sbin/create-munge-key`

Step 5 - Fix Munge issue so it will boot

`sudo systemctl edit --system --full munge`

Change this line:

`ExecStart=/usr/sbin/munged`

To:

`ExecStart=/usr/sbin/munged --syslog`

Save and exit.

```
sudo systemctl start munge
```

Note: The *systemctl enable munge* may show a failed notification but its fine. Just move to the next command.

Step 6 - Enable Slurm Controller

`sudo systemctl enable slurmctld`

Reboot:

`sudo reboot`

Slurm on Compute Node

On *head node*

Step 1 - Copy Slurm configuration file and Munge key to *node1* home directory:

`sudo cat /etc/munge/munge.key | ssh <username>@node1 "cat > ~/munge.key"`

`sudo cat /etc/slurm-llnl/slurm.conf | ssh <username>@node1 "cat > ~/slurm.conf"`

On *compute node*

Step 2 - Install Slurm

`sudo apt-get install slurmd slurm-client`

Step 3 - Copy the configuration files to proper locations

```
sudo cp ~/munge.key /etc/munge/
sudo cp ~/slurm.conf /etc/slurm-llnl/
```

Step 4 - Fix Munge issue so it will boot

```
sudo systemctl edit --system --full munge
```

Change this line:

`ExecStart=/usr/sbin/munged`

To:

`ExecStart=/usr/sbin/munged --syslog`

Save and exit.

```
sudo systemctl enable munge

sudo systemctl start munge
```

Note: The *systemctl enable munge* may show a failed notification but its fine. Just move to the next command.

Step 5 - Enable Slurm daemon

`sudo systemctl enable slurmd`

Step 6 - Set Slurm folder permissions

``

sudo mkdir /var/lib/slurm

sudo chow n -R slurm:slurm /var/lib/slurm

sudo mkdir /var/log/slurm

sudo chown -R slurm:slurm /var/log/slurm
``

Step 7 - Reboot both nodes

Execute on both nodes:

```
sudo reboot
```

Save Your Cluster Snapshot

Once your cluster is working properly you will want to take a snapshot of all nodes. This will allow you to work forward from here but to have a restore point if things don't work out with future changes.

Step 1 - Shutdown All nodes

Execute the shutdown on all nodes:

```
sudo shutdown -h now
```

Step 2 - Snapshot Your Nodes

In VirtualBox right click the node in the left column

In the upper right hand corner of VirtualBox click **Snapshots**

Click the left purple camera icon to take a snapshot of the current machine state

Give the node a name that includes its node name and stage **Example** `Head Node (MPI Stage)` or `Head Node (HADOOP/MPI Stage)`

Click **OK** and you are done

Do this for all nodes and you are safe to begin making changes and producing

Note: You can snapshot the node anywhere you want by following these instructions. In this case take advantage of the description box after naming the snapshot.

Troubleshooting

Host Verification Key Error

In case of *host verification key* error when executing MPI follow the steps for deleting and regenerating SSH keys.

On *Head Node* as user delete previous SSH keys:

```
rm -rf ~/.ssh
mkdir ~/.ssh
```

Generate new SSH keys:

```
cd ~
ssh-keygen -t rsa -C "cluster@swosu"
```

`Enter` to select default install location

`Enter` to leave passphrase blank

`Enter` to confirm blank passphrase

Copy new SSH keys to local system and nodes:

```
cat /home/<username>/.ssh/id_rsa.pub >> /home/<username>/.ssh/authorized_keys
cat ~/.ssh/id_rsa.pub | ssh <username>@192.168.10.100 "cat >> .ssh/authorized_keys"
```

Save new SSH keys to keychain:

```
ssh-agent bash
ssh-add
```

Restore VirtualBox snapshot

In VirtualBox right click the node in the left column

In the upper right hand corner of VirtualBox click **Snapshots**

Select the snapshot you wish to restore from the list

Click the second icon with the loopback green arrow to restore that snapshot

You will be prompted if you want to save a copy of the current machine state. This is a personal choice and is advised if you think you may resolve the situation causing the restore later.

Note: Remember the rule to save and save often!