Building a Raspberry Pi HPC

Step 0: The Hardware

Parts list

- 3x Raspberry Pi 4 Model B — for the compute nodes
- 1x Raspberry Pi 4 Model B — for the master/login node
- 4x MicroSD Cards
- 4x USB-C power cables
- 1x 5-port 10/100/1000 network switch
- 1x 6-port USB power-supply (optional)
- 1x 128GB SSD (optional)

Step 1: Prepare Raspberry Pis

Download latest version of Raspbian Lite OS by using your MacOS/Linux terminal and type

```
wget https://downloads.raspberrypi.orgraspbian_lite_latest -O
raspbian_lite_latest.zip
```

Now extract the zipped file

```
unzip raspbian_lite_latest.zip
```

Check the directory for the contents from the extraction and find the name of the disk image file with extension .img (e.g. 2020-02-13-raspbian-buster-lite.img)

Now insert a SD/Micro-SD card inside your laptop and check for the attached devices/ mount point of the card

For MacOs:

```
diskutil list
```

For Linux:

```
lsblk
```

Let’s say it is attached to dev/disk2. First unmount the disk,

```
diskutil unmountDisk /dev/disk2
```

Then flash the image to memory card

```
sudo dd if=2020-02-13-raspbian-buster-lite.img of=/dev/disk2
```

If successful, a drive will be mounted under the name boot. Raspberry Pis usually comes with disabled SSH configuration. We don’t want that. To enable it create an empty file inside the boot directory.

For MacOS, you can find it under /Volume/boot
Now, type

```
touch ssh
```

Now, we have successfully configured a Raspbian Lite OS having ssh enabled. Let’s eject the card from the Mac

```
diskutil unmountDisk /dev/disk2
```

Repeat this process for all four memory cards. Now insert the cards to your Raspberry Pi s.

Now mark the master node to separate it from others.

Step 2: Network Setup

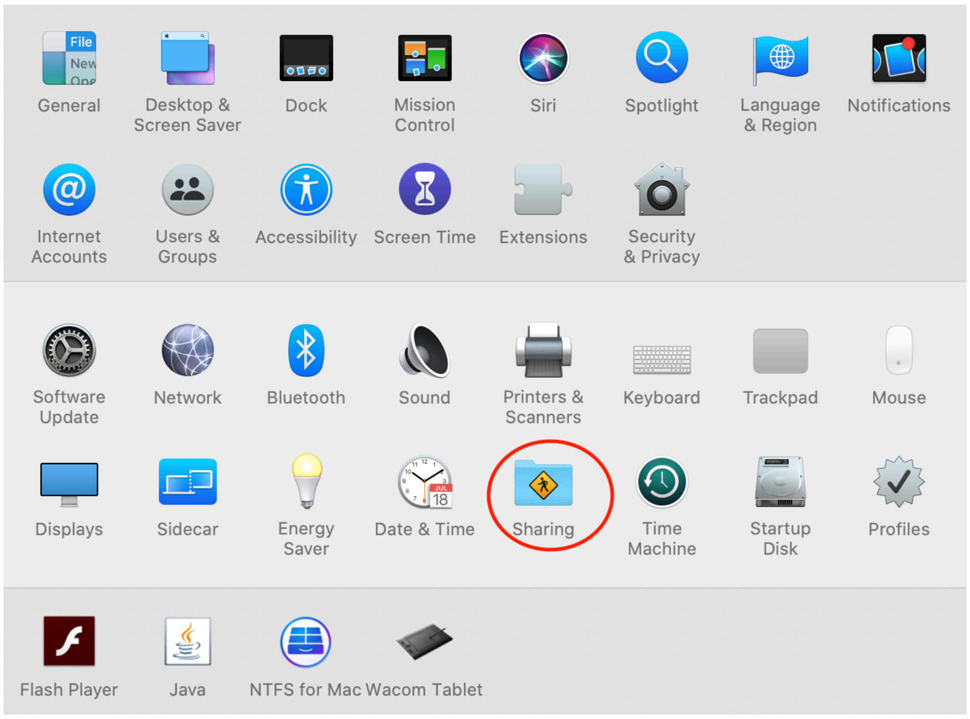
There are two different ways to do this. One with wireless/wired router with LAN ports and another way by using internet sharing facility from Mac or Windows. I’ll explain the Mac way here.

First power up your 1x 5-port 10/100/1000 network switch and connect the master node (Raspberry Pi) through network cable.

MAC OS:

Assuming your laptop is connected to internet via wifi,

- Go to System Preferences
- Choose Sharing



- Click on Internet Sharing and choose share from Wifi to Ethernet

It will start a service which allows you to share the connection from the WiFi across the Ethernet connection (through the switch) to the Raspberry Pis.

Now power up the master node and wait for couple of minutes. The network sharing service will automatically assign some ip to the master node. To access master node via SSH, we need the ip assigned to the master node. If we run “ifconfig” command on a terminal, the shared network connection showed up as an adapter called Bridge100.

```
bridge100:
flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu
1500
      options=3<RXCSUM,TXCSUM>
      ether a3:4e:50:8e:1f:63
      inet 192.168.2.1 netmask 0xffffff00 broadcast
192.168.2.255
      inet6 fe80::a45e:60ff:fe7e:1d64%bridge100 prefixlen 64
scopeid 0xe
      Configuration:
            id 0:0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
            maxage 0 holdcnt 0 proto stp maxaddr 100 timeout
1200
            root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
            ipfilter disabled flags 0x2
      member: en4 flags=3<LEARNING,DISCOVER>
```

```
            ifmaxaddr 0 port 10 priority 0 path cost 0
    Address cache:
    nd6 options=1<PERFORMNUD>
    media: <unknown type >
    status: inactive
```

Now use the following command to know ip assigned to the master node

```
arp -i bridge100 -a
```

Output:

```
? (192.168.2.2) at 3b:04:4e:3e:f7:b2 on bridge100 ifscope
permanent [ethernet]
```

It is most likely that it would assign ip starting from 192.168.2.2 to 192.168.2.255. Let's say it is 192.168.2.2

Step 3: Setting Up the Master Node

Now, log in to your master node using

```
ssh pi@192.168.2.2
```

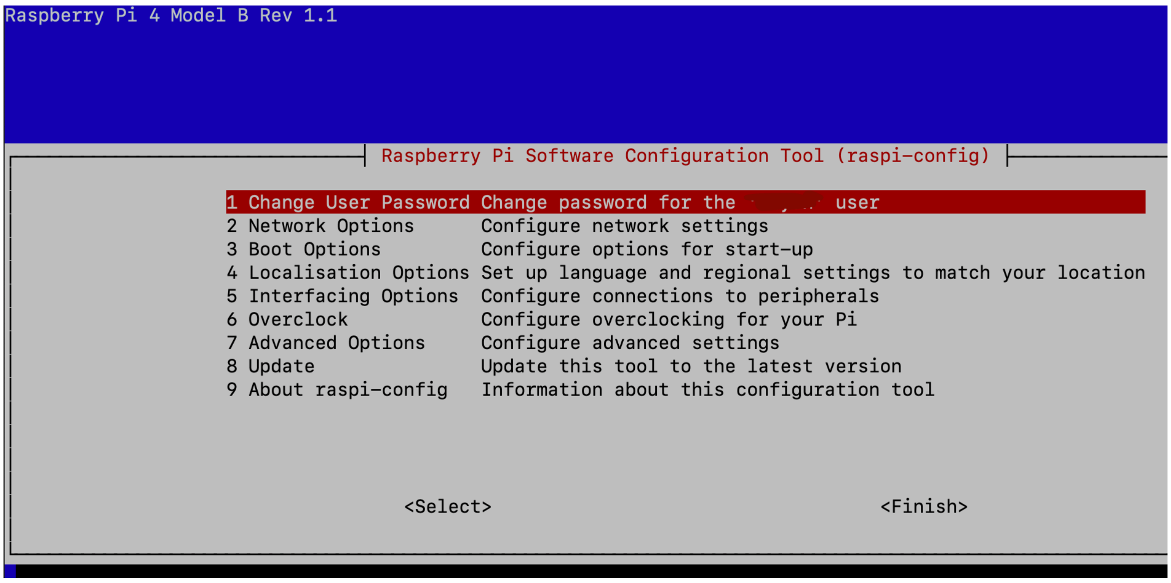
Upon connection use password **raspberry**. (Note: it is the default password)

Now, we need to configure the node before starting to use.

```
pi@raspberrypi~$ sudo raspi-config
```

It opens up the config utility. You can change the default password if you want (highly recommended). Next you should set the locale, timezone, and wifi country. Then, select finish and press enter to exit the utility.

A snapshot of the utility screen is provided below.



```
Raspberry Pi 4 Model B Rev 1.1

                 ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

          1 Change User Password Change password for the [redacted] user
          2 Network Options      Configure network settings
          3 Boot Options         Configure options for start-up
          4 Localisation Options Set up language and regional settings to match your location
          5 Interfacing Options  Configure connections to peripherals
          6 Overclock            Configure overclocking for your Pi
          7 Advanced Options     Configure advanced settings
          8 Update               Update this tool to the latest version
          9 About raspi-config   Information about this configuration tool


                      <Select>                                 <Finish>
```

Next,

SYSTEM UPDATE AND UPGRADE

```
pi@raspberrypi ~> sudo apt-get update && sudo apt-get upgrade
```

Now, we need to decide the hostnames for master node as well as cluster nodes. I would recommend to stick with the usual ones. Set "**master**" for master node and for cluster nodes starting from "**node01**" to "**node03**" (for 3 node cluster). Use the following command to set it up on master node.

```
pi@raspberrypi ~> sudo hostname master     # choice of yours
```

Change "**raspberrypi**" to "**master**" by editing the hostname file.

```
pi@raspberrypi ~> sudo nano /etc/hostname
```

Now edit the hosts file

```
pi@raspberrypi ~> sudo nano /etc/hosts
```

Add the following at the bottom of the existing information

```
127.0.1.1       master
192.168.2.2     master
192.168.2.3     node01
192.168.2.4     node02
```

```
192.168.2.5     node03
```

NETWORK TIME:

Now, since we are planning for a HPC system that uses a SLURM scheduler and the Munge authentication, we need to make sure that the system time is accurate. For that purpose we can install ntpdate package to periodically sync the system time in the background.

```
pi@raspberrypi ~> sudo apt install ntpdate -y
```

To apply the effect of changes that have been made so far reboot the system using the following command

```
sudo reboot
```

After, successful reboot, login to the master node again using ssh.

NEXT STOP, SHARED STORAGE:

The concept of cluster is based on idea of working together. In order to do so, they need to have access to the same files. We can arrange this mounting an external SSD drive (not necessary but convenient and faster) and exporting that storage as a network file system (NFS). It would allow us to access the files from all nodes.

First, insert the external storage into your master node. Now login to your master node using ssh and use the following command to see the dev location and mount point of your storage.

```
pi@master ~> lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     105:0    0  7.4G  0 disk
├─mmcblk0p1 105:1    0 43.8M  0 part /boot
└─mmcblk0p2 105:2    0  7.4G  0 part /
sda         3:16     0 59.2G  0 disk
└─sda1      3:17     0 59.2G  0 part
```

In our case, the main partition of the external storage is mounted at /dev/sda1
Before, using it as a NFS drive, we need to format it properly in ext4 file system. Use the following command to do that.

```
sudo mkfs.ext4 /dev/sda1
```

Now, we need to create a directory where the storage will be mounted. We can choose any name for that. But, lets choose something that is easy to remember "**nfsdrive**",

```
sudo mkdir /nfsdrive
sudo chown nobody.nogroup -R /nfsdrive
sudo chmod 777 -R /nfsdrive
```

Now, we need to configure to mount the storage during boot. For, that we need the the UUID of the storage. We can find that using the following command,

```
pi@master ~> blkid
```

Now, copy the number from /dev/sda1. It'll look like

```
UUID="78543e7a-4hy6-7yea-1274-01e0ff974531"
```

Now, open and edit fstab to mount the drive on boot.

```
pi@master ~> sudo nano /etc/fstab

Add the following line:

UUID=78543e7a-4hy6-7yea-1274-01e0ff974531 /nfsdrive ext4
defaults 0 2
```

All done, now we can mount the drive using the following command,

```
pi@master ~> sudo mount -a
```

If it fails for some reason, reboot the master node and try again. If it is still not working double check the process and look for typo.

ENABLE NFS SHARE

Now, we have a storage that can be shared but we need to install NFS server on master node in order to do so.

```
pi@master ~> sudo apt install nfs-kernel-server -y
```

Now, edit /etc/exports and add the following line to export

```
/nfsdrive
192.168.2.0/24(rw,sync,no_root_squash,no_subtree_check)
```

Remember, depending upon the IP address schema used on your local network, the ip will be different for setup. For example, if your master node ip is 10.0.0.1, then you need to replace the ip with 10.0.0.0/24.

Now, we can update the configuration of the NFS kernel with the following command,

```
sudo exportfs -a
```

One of the tasks for NFS to work remains unfinished which we will do in the next section.

Step 4: Setting Up the Worker Nodes

At this point, we already have the ssh access to all the worker nodes. But before configuring anything we need to prepare the worker nodes first. After connecting all the available nodes to the same network (just plug the Ethernet cables into the network switch), If we repeat one of the command in step 2,

```
arp -i bridge100 -a
```

We should be able to see the assigned IPs for the worker nodes like the following,

```
? (192.168.2.2) at 3b:04:4e:3e:f7:b2 on bridge100 ifscope
permanent [ethernet]
? (192.168.2.3) at 1a:54:3e:7b:27:ac on bridge100 ifscope
permanent [ethernet]
? (192.168.2.4) at 4c:b6:3h:66:g3:2a on bridge100 ifscope
permanent [ethernet]
? (192.168.2.5) at 0a:41:3h:bc:2a:k3 on bridge100 ifscope
permanent [ethernet]
```

The bracketed number is the assigned IP and the 12 digit address is our respective MAC address of raspberry pis.

Note: Do not worry if they are not in order and wait for a while if they do not appear at first. Repeat the command until you see all of them.

Now, we will assign some name to each worker node to refer them easily. We decide the following,

192.168.2.2: master (already assigned)

192.168.2.3: node01

192.168.2.3: node02

192.168.2.4: node03

Let's prepare them one by one. Log into node01 by using the following command,

```
ssh pi@192.168.2.3
```

It will ask for a password, use the default one "**raspberry**." It will open up a terminal, the same that you had for the master. Now configure the node using

```
pi@raspberrypi~$ sudo raspi-config
```

It opens up the config utility. Next you should set the locale, timezone, and wifi country. Then, select finish and press enter to exit the utility. Exactly same what you did for the master node, except the password.

Now let's update the hostname.

```
pi@raspberrypi ~> sudo hostname node01
```

Change "**raspberrypi**" to "**node01**" by editing the hostname file.

```
pi@raspberrypi ~> sudo nano /etc/hostname
```

Now edit the hosts file

```
pi@raspberrypi ~> sudo nano /etc/hosts
```

Add the following

```
127.0.1.1       node01
192.168.2.2     master
192.168.2.3     node01
192.168.2.4     node02
192.168.2.5     node03
```

Next,

SYSTEM UPDATE AND UPGRADE

```
pi@raspberrypi ~> sudo apt-get update && sudo apt-get upgrade
```

Now reboot the system to apply the effect of changes that have been made so far.

```
pi@raspberrypi ~> sudo reboot
```

After the reboot, login to the system again.

NFS MOUNT

To access the storage that we shared on master node from individual worker nodes, we need to install and configure NFS services.

```
sudo apt install nfs-common -y
```

Now, we need to create the same directory in order to mount the storage.

```
pi@node01 ~> sudo mkdir /nfsdrive
pi@node01 ~> sudo chown nobody.nogroup /nfsdrive
pi@node01 ~> sudo chmod -R 777 /nfsdrive
```

To allow automatic mounting we need to edit the fstab file for each node. Use the following command to edit,

```
pi@node01 ~> sudo nano /etc/fstab
```

And add the following line below the existing texts

```
192.168.2.2:/nfsdrive    /nfsdrive    nfs    defaults   0 0
```

Now, use the following to finish the mounting

```
pi@node01 ~> sudo mount -a
```

To check whether the shared storage is working. Open a new terminal window and login to your master node. The create a blank file.

```
ssh pi@192.168.2.2
```

```
pi@master ~> cd /nfsdrive
pi@master ~> touch nas_test.dat
```

Now go back to the node01 terminal and check the contents of your shared directory

```
pi@node01 ~> cd /nfsdrive
pi@node01 ~> ls
```

If you see nas_test.dat file here, means you have successfully created a Network File System. If you can't see, you may have to reboot the node once.

Now repeat the process for rest of the worker nodes. Remember to replace "**node01**" word with their respective node numbers.

Step - 5: Configuring SLURM on master Node

Slurm is an open source, and highly scalable cluster management and job scheduling system. It can be used for both large and small Linux clusters. Let's install it on our Pi cluster.

```
pi@master ~> sudo apt install slurm-wlm -y
```

Upon successful installation, we need to configure slurm,

```
pi@master ~> cd /home/pi
pi@master ~> cp /usr/share/doc/slurm-client/examples/
slurm.conf.simple.gz .
pi@master ~> gzip -d slurm.conf.simple.gz
pi@master ~> sudo mv slurm.conf.simple /etc/slurm-llnl/
slurm.conf
```

Now edit the configuration file by searching for the keyword on the left (e.g. "*SlurmctlHost"*) and edit the line as per the information provide below,

```
pi@master ~> sudo nano /etc/slurm-llnl/slurm.conf

SlurmctldHost=master(192.168.2.2)
SelectType=select/cons_res
SelectTypeParameters=CR_Core
ClusterName=cluster
```

Now we need to add the node information as well as partition at the end of the file. Delete the example entry for the compute node and add the following configurations for the cluster nodes:

```
NodeName=master NodeAddr=192.168.2.2 CPUs=4 State=UNKNOWN
NodeName=node01 NodeAddr=192.168.2.3 CPUs=4 State=UNKNOWN
NodeName=node02 NodeAddr=192.168.2.4 CPUs=4 State=UNKNOWN
NodeName=node03 NodeAddr=192.168.2.5 CPUs=4 State=UNKNOWN
PartitionName=picluster Nodes=node[01-03] Default=YES
MaxTime=INFINITE State=UP
```

Now we need to create a configuration for cgroup support

```
pi@master ~> sudo nano /etc/slurm-llnl/cgroup.conf
```

Now, add the following,

```
CgroupMountpoint="/sys/fs/cgroup"
CgroupAutomount=yes
CgroupReleaseAgentDir="/etc/slurm-llnl/cgroup"
AllowedDevicesFile="/etc/slurm-llnl/
cgroup_allowed_devices_file.conf"
ConstrainCores=no
TaskAffinity=no
ConstrainRAMSpace=yes
ConstrainSwapSpace=no
ConstrainDevices=no
AllowedRamSpace=100
AllowedSwapSpace=0
MaxRAMPercent=100
MaxSwapPercent=100
MinRAMSpace=30
```

Now, we need to whitelist system devices by creating the file

```
pi@master ~> sudo nano /etc/slurm-llnl/
cgroup_allowed_devices_file.conf
```

Now add the following lines,

```
/dev/null
/dev/urandom
/dev/zero
/dev/sda*
/dev/cpu/*/*
/dev/pts/*
/nfsdrive*
```

Now we need to set the same for all nodes. To do that we need to copy these files to the shared storage.

```
pi@master ~> sudo cp /etc/slurm-llnl/*.conf /nfsdrive
pi@master ~> sudo cp /etc/munge/munge.key /nfsdrive
```

All done, now we need to enable and start SLURM Control Services and munge,

```
pi@master ~> sudo systemctl enable munge
pi@master ~> sudo systemctl start munge

pi@master ~> sudo systemctl enable slurmd
pi@master ~> sudo systemctl start slurmd

pi@master ~> sudo systemctl enable slurmctld
pi@master ~> sudo systemctl start slurmctld
```

To ensure smooth operation, we need to reboot the system at this point.

Step - 6: Configuring SLURM on Compute Nodes

We have successfully configured the master node, we need to do the same on compute nodes. Now, log into the one of the nodes and install slurm

```
pi@node01 ~> sudo apt install slurmd slurm-client -y
```

Upon installation, we need to copy the configuration files from the shared storage to the node.

```
pi@node01 ~> sudo cp /nfsdrive/munge.key /etc/munge/munge.key
pi@node01 ~> sudo cp /nfsdrive/*.conf /etc/slurm-llnl/
```

Similar to the master node, we need to enable and start slurm daemon and munge on the nodes.

```
pi@node01 ~> sudo systemctl enable munge
pi@node01 ~> sudo systemctl start munge

pi@node01 ~> sudo systemctl enable slurmd
pi@node01 ~> sudo systemctl start slurmd
```

Now, we need to verify whether our the SLURM controller can successfully authenticate with the client nodes using munge. In order to do that, we need to login to master node and use the following command,

```
pi@master ~> ssh pi@node01 munge -n | unmunge
```

Upon successful operation, you should get output something similar to the following,

```
ssh pi@node01 munge -n | unmunge
pi@node01's password:
STATUS:           Success (0)
ENCODE_HOST:      master (127.0.1.1)
ENCODE_TIME:      2020-08-30 22:45:00 +0200 (1598820300)
DECODE_TIME:      2020-08-30 22:45:00 +0200 (1598820300)
TTL:              300
CIPHER:           aes128 (4)
MAC:              sha256 (5)
ZIP:              none (0)
UID:              pi (1001)
GID:              pi (1001)
LENGTH:           0
```

Sometime, you might get an error, which indicates that you may have failed to copy the exact munge key to the nodes.

Now repeat this process on all the other nodes.

Step - 7: Test SLURM

Login to master node using ssh and type the following command

```
pi@master ~>sinfo
```

You should get an output something like this

```
PARTITION  AVAIL  TIMELIMIT  NODES  STATE NODELIST
picluster*    up   infinite      3   idle node[01-03]
```

To resume the nodes

```
sudo scontrol update NodeName=node[01-03] state=resume
```

You can simply run a task to ask the hostname for each node

```
pi@master ~>srun --nodes=3 hostname
```

It will give you an output similar to

```
node02
node03
node01
```

Step - 8: Powering On and Off (Cluster)

Write a shell script with the following lines of codes and save it as clusterup.sh

```
#!/bin/bash
sudo scontrol update NodeName=node[01-03] state=resume
#sudo scontrol update NodeName=node01 state=resume
sinfo
echo "Nodes up and running"
```

Now, you need to setup password less super user access to perform the next action. To do that efficiently, we need to create admin groups

```
sudo groupadd admin
```

Now add your users (or yourself) to that group

```
sudo usermod —a —G admin pi
```

Now edit sudoers file

```
sudo vim /etc/sudoers
```

Add these lines or edit accordingly

```
# User privilege specification
root  ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL
%admin      ALL=(ALL) ALL
# See sudoers(5) for more information on "#include"
directives:
%admin      ALL=(ALL) NOPASSWD: ALL
```

REPEAT this process for each node. Starting from admin group add.

Write a shell script with the following lines of codes and save it as clusterdown.sh

```
#!/bin/bash
echo "WiPi Cluster Shutdown"
echo "===================="
sudo scontrol update NodeName=node[01-03] state=down
reason="power down"
ssh node01 "sudo halt"
ssh node02 "sudo halt"
ssh node03 "sudo halt"
echo "Nodes disconnected and shutting down"
echo "Do you want to shutdown master node too?"
echo "Press 'y' to continue or q to abort"
count=0
while : ; do
read -n 1 k <&1
if [[ $k = y ]] ; then
printf "\nShutting down Master Node\n"
sudo halt
elif [[ $k = q ]] ; then
printf "\nShutdown aborted\n"
break
else
((count=$count+1))
printf "\nWrong Key Pressed\n"
echo "Press 'q' to abort"
fi
done
```

Now make these scripts executable using the following command

```
pi@master ~>chmod a+x clusterup.sh
pi@master ~>chmod a+x clusterup.sh
```

Each time you power on your cluster, run this script at the startup using the following command.

```
pi@master ~>./clusterup.sh
```

Each time you need to power off your cluster, run this script at the end using the following command.

```
pi@master ~>./clusterdown.sh
```

Step - 9: Password-less SSH

Now, we'll set up password-less SSH on master node

```
pi@master ~> ssh-keygen -t rsa
```

This would ask you for input, each time press enter key to proceed. After successful operation, the output will look like the following

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_rsa):
```

```
Created directory '/home/pi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pi/.ssh/id_rsa.
Your public key has been saved in /home/pi/.ssh/id_rsa.pub.
The key fingerprint is:
9b:98:c7:86:17:0a:1e:32:95:65:ee:1c:0f:48:48:ef pi@beira
The key's randomart image is:
+---[RSA 2048]----+
| .... o          |
|  .o *           |
|    = +          |
|   o o +         |
|  o E o S        |
|   + o * +       |
|    . = B        |
|       +         |
|                 |
+-----------------+
```

Now copy your rsa key to all the nodes

```
pi@master ~> ssh-copy-id pi@node01

pi@master ~> ssh-copy-id pi@node02

pi@master ~> ssh-copy-id pi@node03
```

Step - 10: OpenMPI

OpenMPI is the Open sourced Message Passing Interface. In short it is a very abstract description on how messages can be exchanged between different processes. It will allow us to run a job across multiple nodes connected to the same cluster.

```
pi@master ~>sudo su -
#srun —-nodes=3 apt install openmpi-bin openmpi-common
libopenmpi3 libopenmpi-dev -y
```

Note: the number 3 was chosen based on our available nodes.

If you are interested in using master node as well, you need to install the same for master node too.

```
pi@master ~>sudo apt install openmpi-bin openmpi-common
libopenmpi3 libopenmpi-dev -y
```

Now create a host file to run MPI jobs

```
pi@master ~>nano hostfile
```

Now add the following (Change the ip addresses accordingly)

```
192.168.2.2:4
192.168.2.3:4
192.168.2.4:4
192.168.2.5:4
```

NOTE: The last number “4” represents the number of cores(processors) in each CPU.

Now, we are ready to use MPI on our cluster. Let’s test a sample script.

Create a hello world program in C and save it as hello_mpi.c

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);

    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Hello world from processor %s, rank %d"
           " out of %d processors\n",
           processor_name, world_rank, world_size);
    // Finalize the MPI environment.
    MPI_Finalize();
}
```

Now, compile the program using mpicc

```
mpicc hello_mpi.c
```

This would create an executable name a.out
You can run the executable using the following command

```
mpirun -np 3 -hostfile hostfile ./a.out
```

Now, let’s test the same using SLURM job manager. In order to do so, first we have to create a job script. Create a file named hello_mpi.sh and enter the following lines

```
#!/bin/bash
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=4
#SBATCH --partition=picluster
cd $SLURM_SUBMIT_DIR
mpicc hello_mpi.c -o hello_mpi
mpirun ./hello_mpi
```

NOTE: The number 3 represents the number of nodes available in your cluster and the number 4 represents the number of cores(processors) available in each node.

To submit a job use the following command

```
sbatch hello_mpi.sh
```

To view the status of any job

```
squeue -u pi
```

NOTE: pi is your username

References:

1. https://medium.com/@glmdev/building-a-raspberry-pi-cluster-784f0df9afbd
2. https://medium.com/@glmdev/building-a-raspberry-pi-cluster-aaa8d1f3d2ca
3. https://medium.com/@glmdev/building-a-raspberry-pi-cluster-f5f2446702e8
4. https://epcced.github.io/wee_archlet/
5. https://scw-aberystwyth.github.io/Introduction-to-HPC-with-RaspberryPi/
6. https://github.com/colinsauze/pi_cluster
7. https://magpi.raspberrypi.org/articles/build-a-raspberry-pi-cluster-computer
8. https://www.hydromag.eu/~aa3025/rpi/