Setting up an @Home Raspberry Pi 4 Cluster for OpenMPI

First, acquire 4 Raspberry Pi 4 systems. I recommend spending more to get the 4GB or 8GB memory, but any amount will work.

Get a box with cooling to house them, a small 5 port gigabit ethernet switch, and power supplies with switches. The total cost when I purchased hardware was about $500.00 in January 2021. Here's what I ordered from Amazon:

1) Qty 1 - Cloudlet CASE: for Raspberry Pi and Other Single Board Computers (Clear)

ORDER PLACED	TOTAL	SHIP TO
January 24, 2021	$75.06	Sam Siewert

Cloudlet CASE: for Raspberry Pi and Other Single Board Computers (Clear)
Return window closed on Mar 8, 2021
Buy it again | View your item

2) Qty 4 - CanaKit Raspberry Pi 4 4GB Basic Kit with PiSwitch (4GB RAM) and 64GB Micro-SD cards as needed for Raspbian (I only needed 3 as I already had one on hand)

ORDER PLACED	TOTAL	SHIP TO
January 29, 2021	$335.61	Sam Siewert

CanaKit Raspberry Pi 4 4GB Basic Kit with PiSwitch (4GB RAM)
Return window closed on Mar 2, 2021
Buy it again | View your item

SAMSUNG EVO Select 64GB microSDXC UHS-I U1 100MB/s Full HD & 4K UHD Memory Card with Adapter (MB-ME64HA)
Return window closed on Mar 2, 2021
Buy it again | View your item

3) Qty 4 – Vilros Raspberry Pi 4 & 400 Compatible Power Supply (USB-C) with on/Off Switch (I only needed 2 as I already had 2)

ORDER PLACED	TOTAL	SHIP TO
January 6, 2021	$23.58	Sam Siewert

Vilros Raspberry Pi 4 & 400 Compatible Power Supply (USB-C) with on/Off Switch
Return window closed on Feb 9, 2021
Buy it again | View your item

4) Qty 1 to 4 USB 256 GB Flash devices for a cluster shared file system – [Samsung MUF-256AB/AM FIT Plus 256GB - 300MB/s USB 3.1 Flash Drive](#)(Not necessary, but really nice to have a large cluster "shared" file system for your code and data – potentially could use software RAID or just proved a cluster shared file system for code and another for data).

Samsung MUF-256AB/AM FIT Plus 256GB - 300MB/s USB 3.1 Flash Drive
Return and product support eligibility
Buy it again | View your item

5) Qty 1 switch and 5 short ethernet cables – [NETGEAR 5-Port Gigabit Ethernet Unmanaged Switch (GS305) - Home Network Hub, Office Ethernet Splitter, Plug-and-Play, Fanless Metal Housing, Desktop or Wall Mount](#)(I picked up an extra switch for other equipment I have)

ORDER PLACED	TOTAL	SHIP TO
August 14, 2020	$69.68	Sam Siewert

2

AmazonBasics RJ45 Cat-6 Ethernet Patch Internet Cable - 5 Feet (1.5 Meters) (5-Pack)
Return eligibility
Buy it again | View your item

2

NETGEAR 5-Port Gigabit Ethernet Unmanaged Switch (GS305) - Home Network Hub, Office Ethernet Splitter, Plug-and-Play, Fanless Metal Housing, Desktop or Wall Mount
Return window closed on Sep 18, 2020
Buy it again | View your item

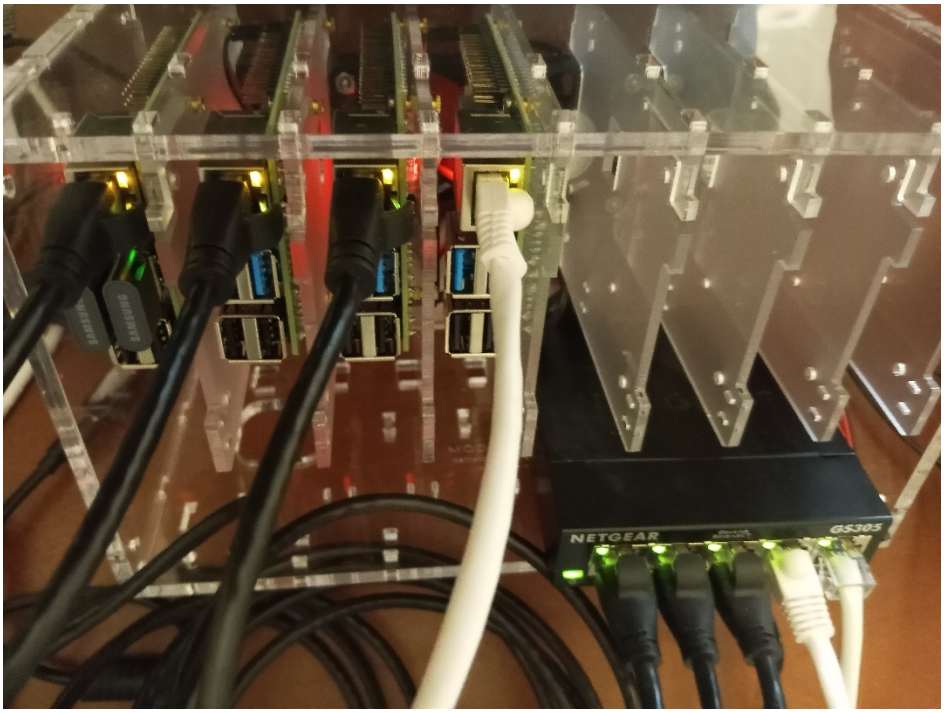
Now, flash Raspbian 64-bit using SD Card Formatter and Balena Etcher and the latest 64-bit image (https://www.raspberrypi.com/software/operating-systems/, https://phoenixnap.com/kb/enable-ssh-raspberry-pi):

2022-01-28-raspi os-bullseye-arm64.zip

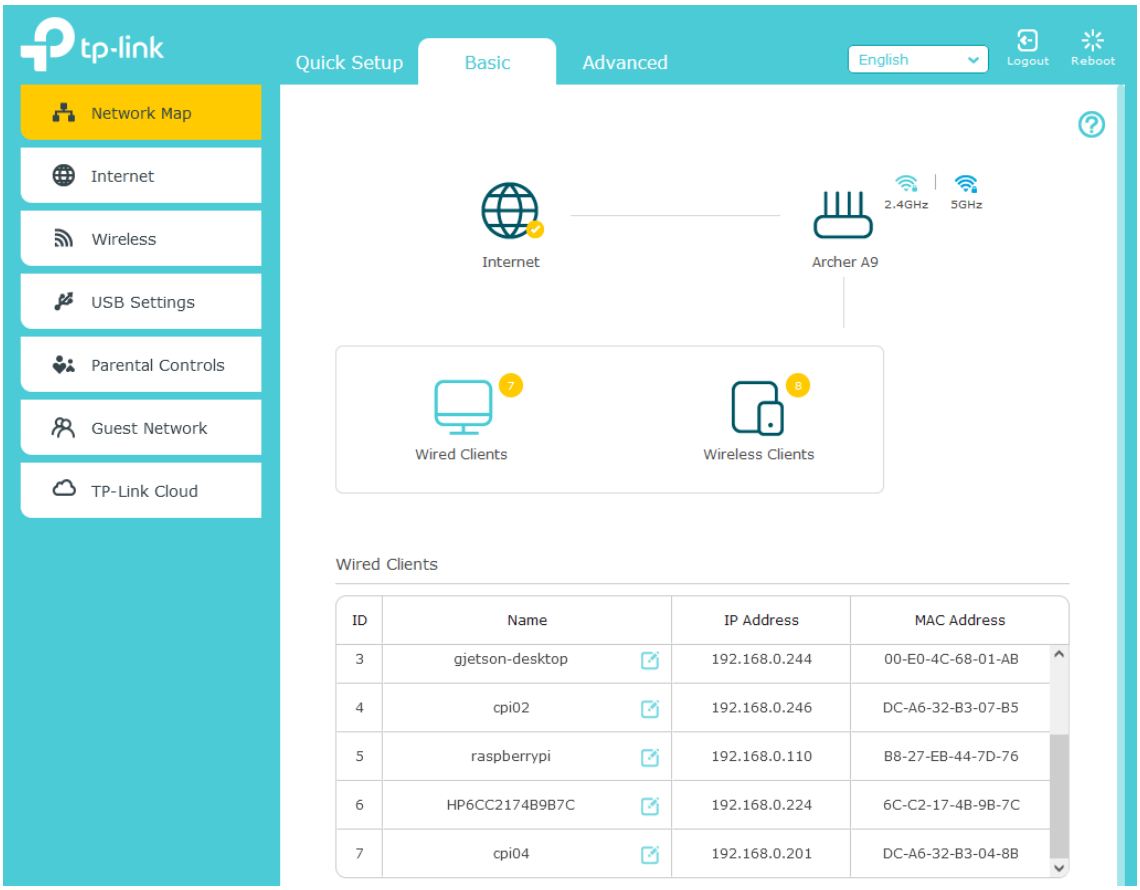
CRITICAL: Make sure you put an empty "ssh" file in each SD card boot directory of the flashed images – this enables SSH on

Next, assemble the R-Pi boards with heat sinks and SD-cards and put each on the lucite carrier board to place in the case as shown here (make sure every other R-Pi is connected to a fan so the fan is powered on from the R-Pi using 5v power and ground pins based upon the R-Pi

pinout). The first picture below shows my entire home lab that I keep with my WAP (Wireless Access Point) on a small shelf including a Jetson Nano, R-Pi 3b+ and camera for computer vision. The second shows a close-up picture of the assembled 4 R-Pi 4 cluster.



Now that your R-Pi 4 cluster is booted and connected to your switch and the switch to your WAP, go to your WAP management page (mine is http://tplinkwifi.net/webpages/login.html) and login – if you forgot your login or never set a login, it is usually admin, admin, or you can reset the factory defaults so you can get in and set a more secure password (don't leave it with defaults!).

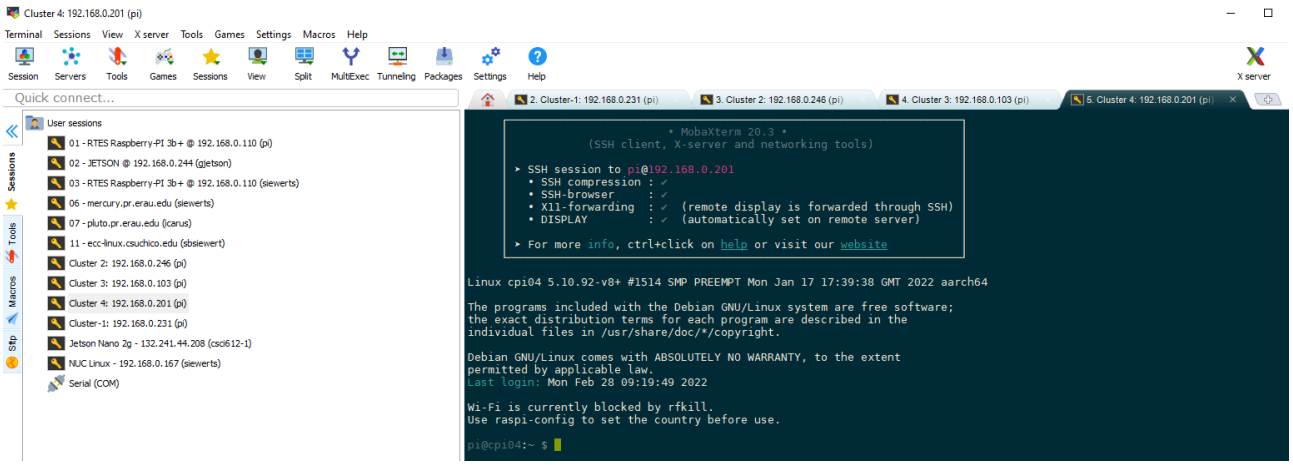


You can play with settings to make sure your WAP always assigns the R-Pi systems in your cluster (I nicknamed mine cpi01, cpi02, cpi03, cpi04) always have the same IP address, but for my WAP, DHCP always assigns them the same IP based on their 48-bit MAC address, so I just let them default, and then looked up the IP addresses they were assigned. E.g., here's what I have:

cpi01 - 192.168.0.231, DC-A6-32-B3-07-13
cpi02 - 192.168.0.246, DC-A6-32-B3-07-B5
cpi03 - 192.168.0.103, DC-A6-32-B3-07-28
cpi04 - 192.168.0.201, DC-A6-32-B3-04-8B

At this point, I would also check that you can log into each one with a tool like MobaXterm or any shell with ssh capabilities. I like to use MobaXterm eventhough I know ssh, sftp, scp, etc.

well – I just find it easier with Windows for drag and drop, remote display with graphics that is automatically set up, and enjoy the password management and session tools!



Now, from here, I followed the fantastic 3-part guide on building an R-Pi cluster, skipping part 2 R testing unless you are interested in R, with focus on SLURM (Slurm Workload Manager, https://hpc.llnl.gov/banks-jobs/running-jobs/slurm-user-manual, https://slurm.schedmd.com/troubleshoot.html) and OpenMPI (https://www.open-mpi.org).

1) https://glmdev.medium.com/building-a-raspberry-pi-cluster-784f0df9afbd
2) https://glmdev.medium.com/building-a-raspberry-pi-cluster-aaa8d1f3d2ca
3) https://glmdev.medium.com/building-a-raspberry-pi-cluster-f5f2446702e8

Note that OpenMPI and the Intel Parallel Studio XE (now called oneAPI HPC) may have differences, but both should be able to compile and run MPI code from CSCI 551 as it is fairly basic code.

It is important that you set up a cluster NFS shared file systems for code (and data if you wish). It should look like this on your main management node:

```
pi@cpi01:~ $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root       61278092 3331964  55399376   6% /
devtmpfs         1777352       0   1777352   0% /dev
tmpfs            1943048       0   1943048   0% /dev/shm
tmpfs             777220     960    776260   1% /run
tmpfs               5120       4      5116   1% /run/lock
/dev/mmcblk0p1    258095   30520    227575  12% /boot
/dev/sda1      245581416    1116 233032596   1% /clusterfs
/dev/sdb1      245581416      48 233033664   1% /clusterdata
tmpfs             388608      24    388584   1% /run/user/1000
pi@cpi01:~ $
```

On the other 3 nodes, you should be able to mount the NFS export using “sudo mount -a” if you follow the above directions carefully, and that will look like this on cpi02, cpi03, cpi04 for example.

```
pi@cpi02:~ $ df
Filesystem                 1K-blocks    Used Available Use% Mounted on
/dev/root                   61278092 3408192  55323148   6% /
devtmpfs                     1777352       0   1777352   0% /dev
tmpfs                        1943048       0   1943048   0% /dev/shm
tmpfs                         777220     912    776308   1% /run
tmpfs                           5120       4      5116   1% /run/lock
/dev/mmcblk0p1                258095   30520    227576  12% /boot
tmpfs                         388608      24    388584   1% /run/user/1000
192.168.0.231:/clusterfs   245581824    1024 233032704   1% /clusterfs
192.168.0.231:/clusterdata 245581824       0 233033728   0% /clusterdata
pi@cpi02:~ $
```

An important part of any cluster is making sure the nodes “trust each other” for ssh with no password and with ssh-keys – for this install, a method called “munge” is used (https://github.com/dun/munge/wiki/Man-7-munge, https://computing.llnl.gov/projects/cluster-management-tools). Just make sure to follow all of these steps carefully and test as you go, rebooting where it is suggested that you do a reboot! After this, make sure to **start all of the munge and SLURM services as follows on the manager node** (for me, cpi01) with:

```
sudo systemctl enable munge
sudo systemctl start munge
sudo systemctl enable slurmd
sudo systemctl start slurmd
sudo systemctl enable slurmctld
sudo systemctl start slurmctld
```

Then on each client:

```
sudo systemctl enable munge
sudo systemctl start munge
```

You should be able to test munge from each client and see a positive response (don’t forget the password you used when you set up each R-Pi if you did in fact set one – good to have a basic password):

```
pi@cpi02:~ $ ssh pi@cpi01 munge -n | unmunge
pi@cpi01's password:
STATUS:          Success (0)
ENCODE_HOST:     cpi02 (127.0.1.1)
ENCODE_TIME:     2022-03-03 05:03:13 +0000 (1646283793)
DECODE_TIME:     2022-03-03 05:03:13 +0000 (1646283793)
TTL:             300
CIPHER:          aes128 (4)
MAC:             sha256 (5)
ZIP:             none (0)
UID:             pi (1000)
GID:             pi (1000)
LENGTH:          0

pi@cpi02:~ $
```

You should verify munge on each and every client node (for me, cpi02, cpi03, cpi04).

A key step after installing SLURM is to use systemctl to enable it and start it – you may have to do this after booting your cluster if you did not automate the start of this service.

```
pi@cpi01:~ $  sudo  systemctl enable slurmctld
Synchronizing state of slurmctld.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable slurmctld
pi@cpi01:~ $ sudo systemctl start slurmctld
pi@cpi01:~ $ sinfo
PARTITION  AVAIL  TIMELIMIT  NODES  STATE NODELIST
picluster*    up   infinite      3    unk cpi[02-04]
pi@cpi01:~ $
```

Assuming SLURM is installed correctly (in **/etc/slurm**) as shown, you should be able to use SLURM to install OpenMPI on all nodes or run any command on all nodes once you have tested it. Note that SLURM used to be installed in /etc/slurm-llnl, but it has been simplified to /etc/slurm now. The "llnl" stood for Lawrence Livermore National Labs. It has been deprecated.

```
pi@cpi01:~ $ ls /etc/slurm/
cgroup_allowed_devices_file.conf  cgroup.conf  plugstack.conf  plugstack.conf.d  slurm.conf
pi@cpi01:~ $
```

Work your way thought the above installation instructions.

After installing SLURM based on instructions and then restarting my cluster, I had trouble with the STATE for the client nodes showing as "down", so I researched this and found that it is best to have **ReturnToService=2 set in your /etc/slurm/slurm.conf (see man slurm.conf).**

When you have SLURM successfully installed, you can run "sinfo" on your main R-Pi 4 that hosts your cluster filesystem. The "sinfo" is a key test to make sure your cluster is read to run:

```
pi@cpi01:/etc/slurm $ sinfo
PARTITION  AVAIL  TIMELIMIT  NODES  STATE NODELIST
picluster*    up   infinite      3   idle cpi[02-04]
pi@cpi01:/etc/slurm $
```

When you get your cluster into the "idle" state successfully, it is now ready to go! So, try a simple command on all cluster nodes:

```
pi@cpi01:/etc/slurm $ srun --nodes=3 hostname
cpi03
cpi02
cpi04
pi@cpi01:/etc/slurm $
```

Make sure you have installed OpenMPI on all cluster nodes – the installation instructions have a slick way to do this using SLURM to install it on all nodes in parallel, which is great! Make sure you do this.

Now, you are ready to download test code from the CSCI 551 code site, including:

1) hello_cluster - https://www.ecst.csuchico.edu/~sbsiewert/csci551/code/hello_cluster/, https://www.ecst.csuchico.edu/~sbsiewert/csci551/code/hello_cluster.zip
2) Pacheco's example code - https://www.ecst.csuchico.edu/~sbsiewert/csci551/code/MPI_Examples/, https://www.ecst.csuchico.edu/~sbsiewert/csci551/code/MPI_Examples.zip
3) Other MPI examples - csci551/code/PP-Quinn-Source/mpi/

Here is an example of **build on cpi02 in /clusterfs on my @Home cluster of the hello_cluster code**. Note that you should not build on cpi01 as it is the manager node.

```
pi@cpi02:/clusterfs/hello_cluster $ make clean
rm -f *.o *.NEW *~
rm -f greetings greetingscpp ranksum ranksumfan ranksumtree ranksumbutterfly ranksumall ranksumreduce ranksumallreduce ranksca
ttergather piseriesreduce piseriessimp
pi@cpi02:/clusterfs/hello_cluster $ make
mpicc -g -Wall -I.  -o greetings greetings.c
mpicxx -g -Wall -I.  -o greetingscpp greetings.cpp
mpicc -g -Wall -I.  -o ranksum ranksum.c
mpicc -g -Wall -I.  -o ranksumfan ranksumfan.c
mpicc -g -Wall -I.  -o ranksumtree ranksumtree.c
mpicc -g -Wall -I.  -o ranksumbutterfly ranksumbutterfly.c
mpicc -g -Wall -I.  -o ranksumall ranksumall.c
mpicc -g -Wall -I.  -o ranksumreduce ranksumreduce.c
mpicc -g -Wall -I.  -o ranksumallreduce ranksumallreduce.c
mpicc -g -Wall -I.  -o rankscattergather rankscattergather.c
mpicc -g -Wall -I.  -o piseriesreduce piseriesreduce.c
mpicc -g -Wall -I.  -o piseriessimp piseriessimp.c  -lm
pi@cpi02:/clusterfs/hello_cluster $
```

Now, lets run greetings with mpiexec and mpirun -n 4 (see **man mpirun**):

```
pi@cpi02:/clusterfs/hello_cluster $ mpirun -n 4 -host cpi02:4 ./greetings
Hello from process 0 of 4 on cpi02
Hello from process 1 of 4 on cpi02

Hello from process 2 of 4 on cpi02

Hello from process 3 of 4 on cpi02

pi@cpi02:/clusterfs/hello_cluster $ mpiexec ./greetings
Hello from process 0 of 4 on cpi02
Hello from process 1 of 4 on cpi02

Hello from process 2 of 4 on cpi02

Hello from process 3 of 4 on cpi02

pi@cpi02:/clusterfs/hello_cluster $
```

Note that you can take out the “Intel” include and library paths in the Makefile for the R-Pi cluster.

```
pi@cpi02:/clusterfs/MPI_Examples $ make clean
rm -f *.o *.NEW *~
rm -f bubble odd_even vector_add trap mpi_many_msgs mpi_hello mpi_mat_vect_mult mpi_mat_vect_time mpi_odd_even mpi_output mpi_
vector_add mpi_trap1 mpi_trap2 mpi_trap3 mpi_trap4
pi@cpi02:/clusterfs/MPI_Examples $ make
gcc -g -Wall -I.  -o bubble bubble.c
gcc -g -Wall -I.  -o odd_even odd_even.c
gcc -g -Wall -I.  -o vector_add vector_add.c
gcc -g -Wall -I.  -o trap trap.c
mpicc -g -Wall -O2 -I.  -o mpi_many_msgs mpi_many_msgs.c
mpicc -g -Wall -O2 -I.  -o mpi_hello mpi_hello.c
mpicc -g -Wall -O2 -I.  -o mpi_mat_vect_mult mpi_mat_vect_mult.c
mpicc -g -Wall -O2 -I.  -o mpi_mat_vect_time mpi_mat_vect_time.c
mpicc -g -Wall -O2 -I.  -o mpi_odd_even mpi_odd_even.c
mpicc -g -Wall -O2 -I.  -o mpi_output mpi_output.c
mpicc -g -Wall -O2 -I.  -o mpi_vector_add mpi_vector_add.c
mpicc -g -Wall -O2 -I.  -o mpi_trap1 mpi_trap1.c
mpicc -g -Wall -O2 -I.  -o mpi_trap2 mpi_trap2.c
mpicc -g -Wall -O2 -I.  -o mpi_trap3 mpi_trap3.c  -lm
mpicc -g -Wall -O2 -I.  -o mpi_trap4 mpi_trap4.c  -lm
```

We can run the example trap4 as follows:

```
pi@cpi02:/clusterfs/MPI_Examples $ mpiexec ./mpi_trap4
Enter a, b, and n
0.0
3.14159
1000000
With n = 1000000 trapezoids, our estimate
of the integral from 0.000000 to 3.141590 = 1.999999999994840e+00
pi@cpi02:/clusterfs/MPI_Examples $
```

Running on multiple cluster nodes using SLURM for greetings in /clusterfs/hello_cluster:

1) Make a script to run the MPI program greetings

```
#!/bin/bash

cd $SLURM_SUBMIT_DIR

# Print the node that starts the process
echo "Master node: $(hostname)"

# Run our program using OpenMPI.
# OpenMPI will automatically discover resources from SLURM.

mpirun ./greetings
~
~
```

2) Submit the batch SLURM job for the MPI program specified in the sub_mpi.sh bash script

```
pi@cpi01:/clusterfs/hello_cluster $ sbatch --nodes=3 --ntasks-per-node=4 sub_mpi.sh
Submitted batch job 17
pi@cpi01:/clusterfs/hello_cluster $
```

3) Look for output in the corresponding SLUM job file (17 here):

```
pi@cpi01:/clusterfs/hello_cluster $ ls *slurm*
slurm-16.out  slurm-17.out
pi@cpi01:/clusterfs/hello_cluster $
```

```
6. Cluster-1: 192.168.0.231 (pi)    3. Cluster 2: 192.168.0.246 (pi)
Master node: cpi02
Hello from process 0 of 12 on cpi02
Hello from process 1 of 12 on cpi02

Hello from process 2 of 12 on cpi02

Hello from process 3 of 12 on cpi02

Hello from process 4 of 12 on cpi03

Hello from process 5 of 12 on cpi03

Hello from process 6 of 12 on cpi03

Hello from process 7 of 12 on cpi03

Hello from process 8 of 12 on cpi04

Hello from process 9 of 12 on cpi04

Hello from process 10 of 12 on cpi04

Hello from process 11 of 12 on cpi04
```

It is perhaps simpler to set up ssh-key-gen on the R-Pi cluster like we do on the Intel NUC cluster as documented here - https://www.ecst.csuchico.edu/~sbsiewert/csci551/README-cluster.html

Given ssh-keygen and ssh-copy-id pi@cpi02 and so forth for all other nodes cpi03 and cpi04, we can now just use mpirun to distributed ranks over all the nodes including the management node. This seems a bit easier for interactive MPI development compared to SLURM. It does not require the SLURM wrapper script and allows for work on cpi01 using MPI tools (installed on cpi01 with **sudo apt install openmpi-bin openmpi-common libopenmpi3 libopenmpi-dev -y**). Simply building in /clusterfs, shared via NSF mount by all nodes, allows for build and run on cpi01.

Note that mpirun has a different syntax than it does for Intel Parallel Studio XE, but very similar:

mpirun -n 16 -host cpi01:4 -host cpi02:4 -host cpi03:4 -host cpi04:4 ./greetings

It is also possible to use a host file. Here's an example run using all 16 cores in the new R-Pi cluster:

```
pi@cpi01:/clusterfs/hello_cluster $ make clean
rm -f *.o *.NEW *~
rm -f greetings greetingscpp ranksum ranksumfan ranksumtree ranksumbutterfly ranksumall ranksumreduce ranksumallreduce ranksca
ttergather piseriesreduce piseriessimp
pi@cpi01:/clusterfs/hello_cluster $ make
mpicc -g -Wall -I.  -o greetings greetings.c
mpicxx -g -Wall -I.  -o greetingscpp greetings.cpp
mpicc -g -Wall -I.  -o ranksum ranksum.c
mpicc -g -Wall -I.  -o ranksumfan ranksumfan.c
mpicc -g -Wall -I.  -o ranksumtree ranksumtree.c
mpicc -g -Wall -I.  -o ranksumbutterfly ranksumbutterfly.c
mpicc -g -Wall -I.  -o ranksumall ranksumall.c
mpicc -g -Wall -I.  -o ranksumreduce ranksumreduce.c
mpicc -g -Wall -I.  -o ranksumallreduce ranksumallreduce.c
mpicc -g -Wall -I.  -o ranksscattergather ranksscattergather.c
mpicc -g -Wall -I.  -o piseriesreduce piseriesreduce.c
mpicc -g -Wall -I.  -o piseriessimp piseriessimp.c  -lm
pi@cpi01:/clusterfs/hello_cluster $ mpirun -n 16 -host cpi01:4 -host cpi02:4 -host cpi03:4 -host cpi04:4 ./greetings
Hello from process 0 of 16 on cpi01
Hello from process 1 of 16 on cpi01

Hello from process 2 of 16 on cpi01

Hello from process 3 of 16 on cpi01

Hello from process 4 of 16 on cpi02

Hello from process 5 of 16 on cpi02

Hello from process 6 of 16 on cpi02

Hello from process 7 of 16 on cpi02

Hello from process 8 of 16 on cpi03

Hello from process 9 of 16 on cpi03

Hello from process 10 of 16 on cpi03

Hello from process 11 of 16 on cpi03

Hello from process 12 of 16 on cpi04

Hello from process 13 of 16 on cpi04

Hello from process 14 of 16 on cpi04

Hello from process 15 of 16 on cpi04

pi@cpi01:/clusterfs/hello_cluster $
```