# 3110 Team Project:

# Raspberry Pi Cluster

## *The League of Extraordinary IT Guys*

## Lab Section L02 Team 3

## *12/2/13*

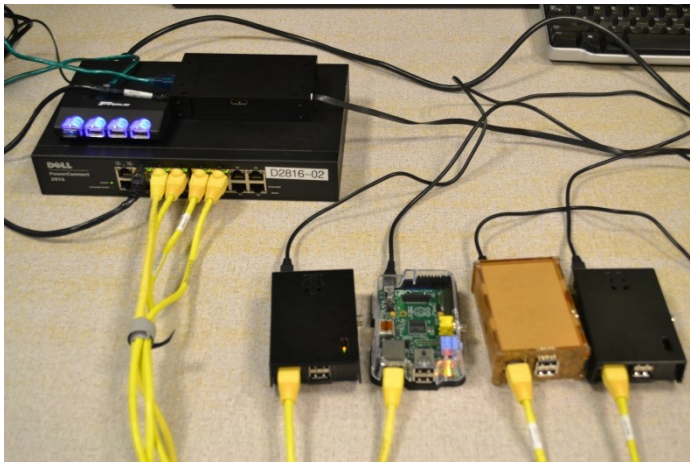
# Table of Contents

# Overview

For our project, we decided to implement a cluster of Raspberry Pis. We took on the challenge of setting up five Raspberry Pis as a demonstration for parallel computing. The rationale for choosing this as our project is that it allowed us to run the computers in parallel, as a cluster, which proved to give greater performance when running the Pis together. We used MPI (Messaging Passing Interface) to allow the Pis to communicate to complete processes. Finally, a wireless access point was utilized to remotely access the Pis through a SSH connection on a Windows laptop to effectively keep the cluster headless.

In order to facilitate the installation of MPI across the Pis, we used University of Southampton Professor Simon Cox's guide (Appendix A). Our group met on Wednesdays throughout the semester to install the cluster, as well as to evaluate the cluster through benchmarking. Michael took on the task of formatting each of our SD cards, cloning Raspbian, as well as getting MPI initialized on the Pis. Lawrence's goal was initializing a wireless access point, and then setting up the connection across all of the Pis. We all took part in the benchmarking process by researching scripts and benchmark tests, then writing the scripts and documenting our test execution results. Some proved to be beneficial in comparing CPU usage and certain calculations, while one test, which measured time differences, was not convincing due to network latency issues. Adam attempted to set up Apache, yet in the end we all figured this added very little to the theme of the overall project, so this idea was dropped, after deciding that the Apache installation proved to be unnecessary. The steps demonstrating our project are featured in the sections below, with text files, screen shots, and tables included for support.

# Raspberry Pi Cluster with Wireless SSH Access



*Supplies Used*

| |
|---|
| 5 x Raspberry Pis + power cord |
| 1 x Raspberry Pi as wireless AP |
| 6 x 8+ GB SD cards |
| 1 x Switch |
| 6 x Ethernet cables |
| 1 x Powered USB hub |

Most of the steps for setting up a Pi Cluster can be found online in Professor Simon Cox's guide (See References). A brief summary of the steps involved are detailed below.

1. Download latest version of Raspbian
2. Download latest MPI source code
3. Configure the build environment and then make and install MPI.
4. Assign a static IP address to the Pi and add it into a file called "machinefile"
5. Test that MPI installed correctly; backup SD card and clone as needed

Making the master node was by far the most time consuming process of this entire project, taking a little over 3 hours, mostly waiting for MPI to build. Once the master node was setup correctly, setting up the other nodes was fairly trivial. Assuming the SD card had a blank partition, the image file from the

master node was copied over to it, and then its IP address was changed and added to the master node's machinefile. The machinefile holds the IP address of all the nodes so MPI knows which machines to connect to. SSH was also setup on all of the machines without a passkey so that all of the machines could securely connect to each other easily. Below is a table of all the IP address used and the contents of the machinefile.

## *Table of IP Addresses*

| Name | IP Address |
|---|---|
| pimaster | 192.168.1.10 |
| node02 | 192.168.1.11 |
| node03 | 192.168.1.12 |
| node04 | 192.168.1.13 |

## *Machinefile*

| |
|---|
| 192.168.1.10 |
| 192.168.1.11 |
| 192.168.1.12 |
| 192.168.1.13 |

Once all the Pis were hooked up and running properly we began to test them using an example MPI program that calculates Pi. Below is the output of running the program on a single Pi, and the output of running it on all four Pis.

## *Single Node Calculate Pi*

| |
|---|
| *Process 0 of 1 is on Pimaster* |
| *pi is approximately 3.1415926544231341, Error is 0.0000000008333410* |
| *wall clock time = 0.002069* |

*All Nodes Calculate Pi*

```
Process 0 of 4 is on pimaster

Process 2 of 4 is on node02

Process 3 of 4 is on node03

Process 1 of 4 is on node01

pi is approximately 3.1415926544231239, Error is 0.0000000008333307

wall clock time = 0.015739
```

In this example, you can see the error goes down by .0000000000000000103 when all four Pis are used, and the value of Pi changes just slightly as a result, but the wall clock time goes up. Wall clock time is a measure of network latency and is one of the issues we encountered while trying to test the Pi cluster.

The first way we tried to show the Pis doing an intensive task more efficient than a single Pi running the same task was through a random number generation and sort script. We generated 100,000 random numbers and then later sorted them in ascending order after they were placed in a file. Below is the script that was written for generating the 100,000 random numbers.

```
#!/bin/bash

COUNTER=0

while [ $COUNTER -lt 100000 ]; do

        od -vAn -N4 -tu < /dev/urandom >> /home/pi/random.txt

        let COUNTER=COUNTER+1

done

sort -h /home/pi/random.txt >> /home/pi/random.txt
```

Running the script along with the time command resulted in the following data.

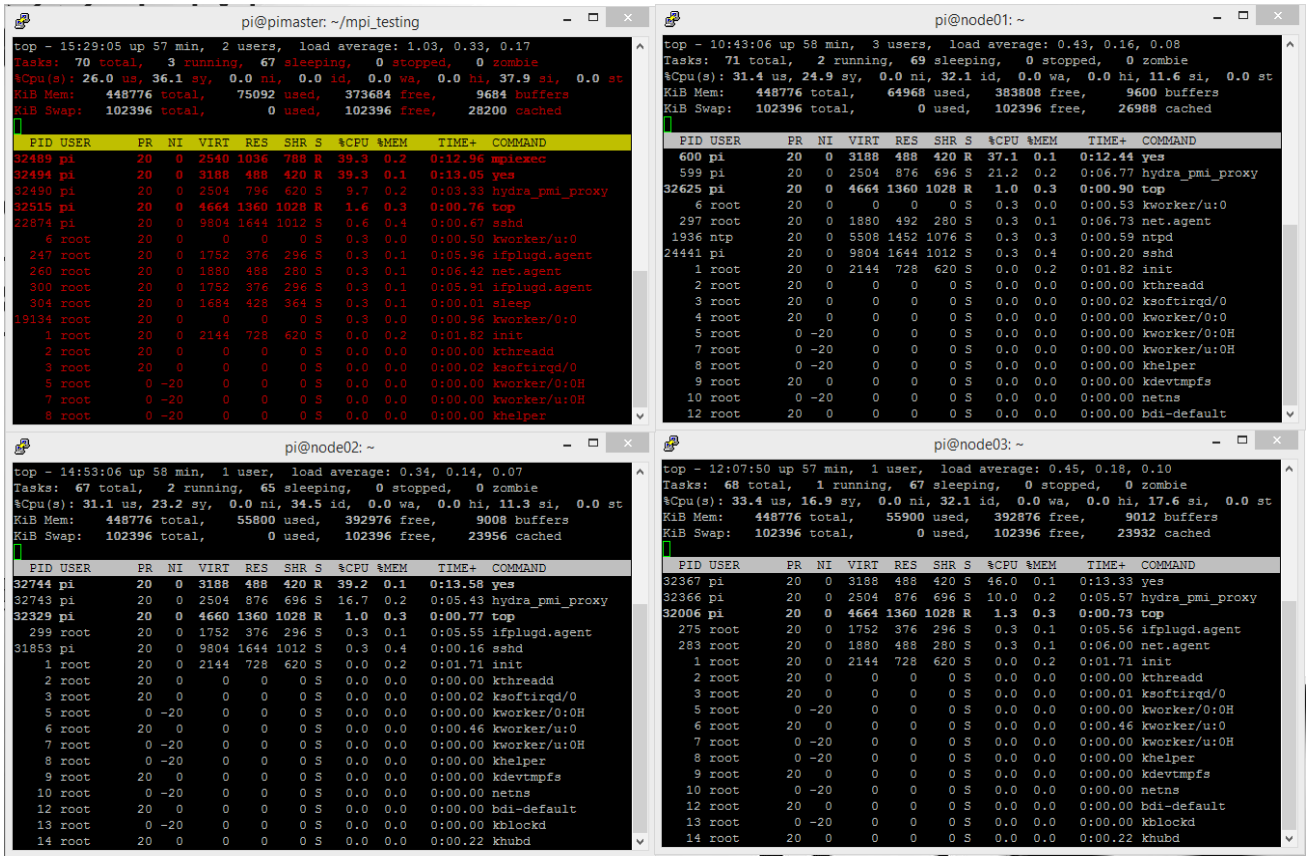
| **TIMES Generating Numbers:** | **MASTER PI** | **3 PI CLUSTER** |
|---|---|---|
| REAL | 22m37.220s | 22m29.485s |
| USER | 4m20.080s | 3m14.390s |
| SYS | 5m35.250s | 3m26.480s |
| | | |
| **SORTING:** | | |
| REAL | 2.292s | 2.974s |
| USER | 2.100s | 2.590s |
| SYS | .050s | .150s |

After it became apparent that the random number script wasn't the best way to measure the Pi's computing performance we found running the command "yes > /dev/null" was a far better approach to put a load on the machines and effectively monitor the load decreasing as more machines were added. The reason this works best is because instead of doing a time sensitive task which can be ruined by network latency, writing to /dev/null continuously is not time sensitive and allows a large CPU load to put on the machines. Below are the results we documented in terms of CPU usage while running this command.

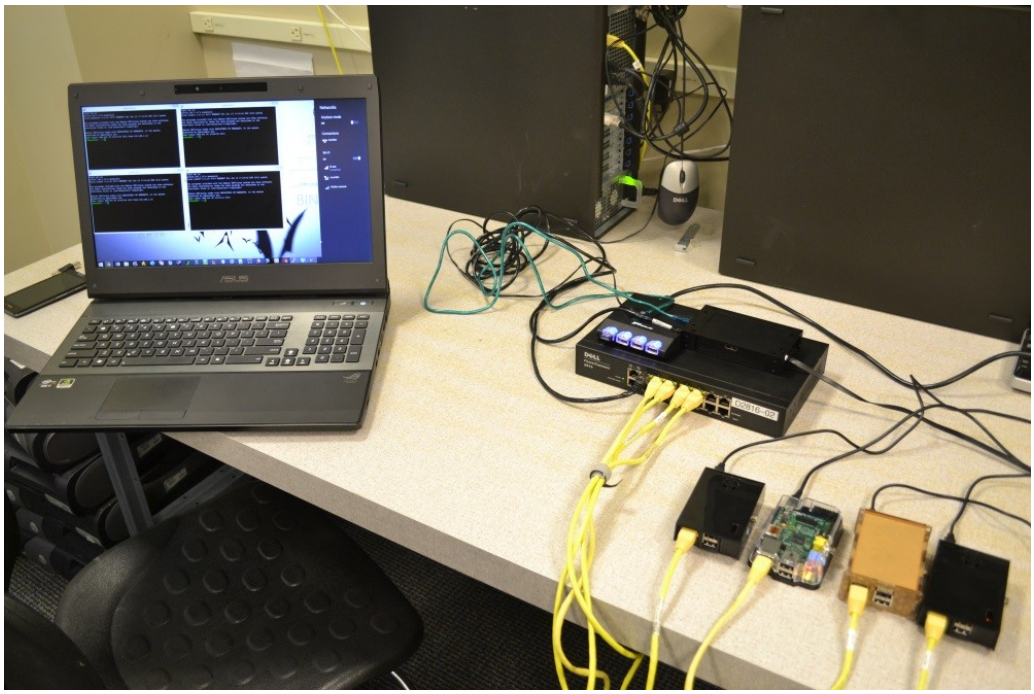
| 2nd Benchmark: Stress Test | | CPU USAGE(AVG) |
|---|---|---|
| yes > /dev/null & | MASTER PI | 0.925 |
| | 2 PI'S | 0.496 |
| | 3 PI'S | 0.452 |
| | 4 PI'S | 0.382 |

Included below is our documentation of the SSH process from Lawrence's laptop as well as a picture of the complete project setup.

*Screenshot from Windows laptop running top via SSH*



*Full project setup: Raspberry Pi Cluster, Raspberry Pi Wireless AP, and Windows Laptop running SSH wirelessly*



# Raspberry Pi Wireless Access Point

The next process involved setting up a wireless access point for the Pi cluster. Featured below are the supplies used and details about our access point configuration.

## Required Supplies

1 x Raspberry PI & Required Cables

1 x 4GB SD card with latest Raspbian Image installed

1 x Wireless USB Adapter (Edimax EW-7811Un )

## Access Point Configuration Table

| | |
|---|---|
| SSID | PI-Net |
| Security Type | WPA-PSK |
| WPA Key | PI-Net_PASS |
| RPI eth0 IP | 192.168.1.15 |
| RPI wlan0 IP | 192.168.2.1 |
| Access Point Subnet | 192.168.2.0/24 |
| Access Point IP Range | 192.168.2.10-192.168.2.50 |

There were few problems encountered in this process. The only minor issue revolved around static vs. dynamic IP addressing. Thus, the steps for successfully initiating the access point are presented below.

### Steps Performed

1. Connect the wireless USB adapter to the Pi.
2. Connect the Pi to the Internet through the Ethernet Port.
3. Update Raspbian.
    a. *Sudo apt-get update*
4. Install Hostapd and DHCP Server.
    a. *Sudo apt-get install hostpad isc-dhcp-server*
5. Configure the DHCP server to allow WIFI connections to automatically be assigned an IP.
    a. Edit the */etc/dhcp/dhcpd.conf* file
        i. *Sudo nano /etc/dhcp/dhcpd.conf*
        ii. Comment of the lines beginning with *option domain-name* & *option domain-name-servers*
        iii. Uncomment the *authoritative* line.
        iv. Add the following lines to the end of the file.

```
subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.10 192.168.2.50;
    option broadcast-address 192.168.2.255;
    default-lease-time 600;
```

max-lease-time 7200;
option domain-name "RPINet";
option routers 192.168.2.1 ;
}

b. Edit the */etc/default/isc-dhcp-server* file
    i. *Sudo nano /etc/ default/isc-dhcp-server*
    ii. Add *wlan0* inside of the quotations after the *INTERFACES* line.

6. Configure *wlan0* to have a static IP address
    a. Edit the */etc/network/interfaces* file
        i. *sudo nano /etc/network/interfaces*
        ii. Comment out any wlan0 configuration lines.
        iii. Add the following lines.

allow-hotplug wlan0

iface wlan0 inet static
address 192.168.2.1
netmask 255.255.255.0

        iv. Assign the wifi adapter's IP address.
*sudo ifconfig wlan0 192.168.2.1*

7. Configure the Access Point
    a. Create the Hostapd configuration file
        *i.* *sudo nano /etc/hostapd/hostapd.conf*
        *ii.* Add the following lines.

interface=wlan0
ssid=PI-Net
channel=1

macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=PI-Net_PASS
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
driver=rtl871xdrv
hw_mode=g

    b. Configure Hostapd to use the configuration file.
        *i.* *sudo nano /etc/default/hostapd*
        *ii.* Uncomment the DAEMON_CONF line and add */etc/hostapd/hostapd.conf* inside the quotations.
    *c.* Enable NAT.
        *i.* *sudo nano /etc/sysctl.conf*
        *ii.* Add the following line to the end of the file.
*net.ipv4.ip_forward=1*
    d. Start NAT IP forwarding at boot.
        i. *sudo sh –c "echo 1>/proc/sys/net/ipv4/ip_forward"*
    e. Configure the network translation between *eth0* & *wlan0*.

i. *sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE*
ii. *sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT*
iii. *sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT*

*f.* Automatically configure the network translation to run at boot.
i. *sudo sh -c "iptables-save > /etc/iptables.ipv4.nat*
ii. Edit the */etc/network/interfaces* file adding the following line.
*up iptables-restore < /etc/iptables.ipv4.nat*

8. Configure the Access Point to run as a daemon.
   a. Start Hostapd and DHCP server.
      i. *sudo service hostapd start*
      ii. *sudo service isc-dhcp-server start*
   *b.* Make Hostapd and DHCP server start at boot.
      i. *sudo update-rc.d hostapd enable*
      ii. *sudo update-rc.d isc-dhcp-server enable*
9. Disconnect the Ethernet cable from the Internet source and connect it to the cluster switch.
10. Reboot
    *a. sudo reboot*

Once the access point has been set up successfully, the steps for accessing the Pi cluster are listed below.

## Accessing the Pi Cluster

Steps Performed

1. Secure Shell Login from a Windows machine.
   a. Download PuTTY SSH.
   b. Connect to the Pi Cluster's Wireless Access Point.
   c. Start PuTTY SSH.
   d. Enter the IP address of the master Pi node and click *open*.
   e. Log in using the master node's credentials.
2. Secure Copy between a Windows machine and the Pi cluster.
   a. Download PuTTY SCP.
   b. Open a Windows Command Prompt.
   c. Execute the PuTTY *pscp* command in the following format.
   *pscp user@192.168.1.x:/directory/file C:\directory*
   d. Enter the source user's password.

# Conclusion

In this group project, we were able to successfully install MPI and run the Pis as a cluster. Once this was complete, we executed multiple benchmarking tests on the cluster to demonstrate performance improvements in both a calculation and CPU usage. Fortunately, there were very few issues that we encountered through the span of our project. Initially, we ran into problems with the size of our SD card since the partition sizes were too small for the original image to be cloned onto them. There were also some other issues with the wireless access point which revolved around the use of dynamic IP vs. static IP addresses. Overall our project was a success, and we achieved our goal of setting up the Pis in a cluster environment.

# Appendix A

## Steps to make Raspberry Pi Supercomputer

## Prof Simon Cox

### First steps to get machine up

1. Get image from
http://www.raspberrypi.org/downloads
I originally used: 2012-08-16-wheezy-raspbian.zip
Updated 30/11/12: 2012-10-28-wheezy-raspbian.zip

My advice is to to check the downloads page on raspberrypi.org and use the latest version.

2. Use win32 disk imager to put image onto an SD Card (or on a Mac e.g. Disk Utility/ dd)
http://www.softpedia.com/get/CD-DVD-Tools/Data-CD-DVD-Burning/Win32-Disk-Imager.shtml
You will use the "Write" option to put the image from the disk to your card

3. Boot on Pi

4. Expand image to fill card using the option on screen when you first boot. If you don't do this on first boot, then you need to use
$ sudo raspi-config
http://elinux.org/RPi_raspi-config

5. Log in and change the password
http://www.simonthepiman.com/beginners_guide_change_my_default_password.php
$ passwd

6. Log out and check that you typed it all OK (!)
$ exit

7. Log back in again with your new password

Building MPI so we can run code on multiple nodes

8. Refresh your list of packages in your cache
$ sudo apt-get update

9. Just doing this out of habit, but note not doing any more than just getting the list (upgrade is via "sudo apt-get upgrade").

10. Get Fortran… after all what is scientific programming without Fortran being a possibility?
$ sudo apt-get install gfortran

11. Read about MPI on the Pi. This is an excellent post to read just to show you are going to make it by the end, but don't type or get anything just yet- we are going to build everything ourselves:
http://westcoastlabs.blogspot.co.uk/2012/06/parallel-processing-on-pi-bramble.html

Note there are a few things to note here:
a) Since we put Fortran in we are good to go without excluding anything
b) The packages here are for armel and we need armhf in this case… so we are going to build MPI ourselves

12. Read a bit more before you begin:
http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-1.4.1-installguide.pdf

Note: As the version of MPICH2 updates, you are better to go to:
http://www.mpich.org/documentation/guides/
and get the latest installer's Guide.

We are going to follow the steps from 2.2 (from the Quick Start Section) in the guide.

13. Make a directory to put the sources in
$ mkdir /home/pi/mpich2

$ cd ~/mpich2

14. Get MPI sources from Argonne.
$ wget http://www.mcs.anl.gov/research/projects/mpich2/downloads/tarballs/1.4.1p1/mpich2-1.4.1p1.tar.gz

[Note that as the MPI source updates, you can navigate to:
http://www.mpich.org/downloads/ to get the latest stable release version for MPICH2]

15. Unpack them.
$ tar xfz mpich2-1.4.1p1.tar.gz

[Note: You will need to update this as the version of MPICH2 increments]

16. Make yourself a place to put the compiled stuff – this will also make it easier to figure out what you have put in new on your system. Also you may end up building this a few times…
$ sudo mkdir /home/rpimpi/

$ sudo mkdir /home/rpimpi/mpich2-install
[I just chose the "rpimpi" to replace the "you" in the Argonne guide and I did the directory creation in two steps]

17. Make a build directory (so we keep the source directory clean of build things)
mkdir /home/pi/mpich_build

18. Change to the BUILD directory
$ cd /home/pi/mpich_build

19. Now we are going to configure the build
$ sudo /home/pi/mpich2/mpich2-1.4.1p1/configure -prefix=/home/rpimpi/mpich2-install

[Note: You will need to update this as the version of MPICH2 increments]

20. Make the files
$ sudo make

21. Install the files
$ sudo make install

22. Add the place that you put the install to your path
$ export PATH=$PATH:/home/rpimpi/mpich2-install/bin

Note to permanently put this on the path you will need to edit .profile

$nano ~/.profile

… and add at the bottom these two lines:

# Add MPI to path
PATH="$PATH:/home/rpimpi/mpich2-install/bin"

23. Check whether things did install or not
$ which mpicc
$ which mpiexec

24. Change directory back to home and create somewhere to do your tests
$ cd ~
$ mkdir mpi_testing
$ cd mpi_testing

25. Now we can test whether MPI works for you on a single node
mpiexec -f machinefile -n <number> hostname
where machinefile contains a list of IP addresses (in this case just one) for the machines

a) Get your IP address
$ ifconfig
b) Put this into a single file called machinefile

26. $ nano machinefile
Add this line:
192.168.1.161
[or whatever your IP address was]

27. If you use
$ mpiexec -f machinefile –n 1 hostname

Output is:
raspberrypi

28. Now to run a little C code. In the examples subdirectory of where you built MPI is the famous CPI example. You will now use MPI on your Pi to calculate Pi:
$ cd /home/pi/mpi_testing
$ mpiexec -f machinefile -n 2 ~/mpich_build/examples/cpi

Output is:
Process 0 of 2 is on raspberrypi
Process 1 of 2 is on raspberrypi
pi is approximately 3.1415926544231318, Error is 0.0000000008333387

29. We now have a master copy of the main node of the machine with all of the installed files for MPI in a single place. We now want to clone this card.

30. Shutdown your Pi very carefully
$ sudo poweroff

Remove the SD Card and pop it back into your SD Card writer on your PC/ other device. Use Win32 disk imager (or on a Mac e.g. Disk Utility/ dd) to put the image FROM your SD Card back TO your PC:
http://www.softpedia.com/get/CD-DVD-Tools/Data-CD-DVD-Burning/Win32-Disk-Imager.shtml

You will use the "Read" option to put the image from the disk to your card

Let us call the image "2012-08-16-wheezy-raspbian_backup_mpi_master.img"

31. Eject the card and put a fresh card into your PC/other device. Use win32 disk imager to put image onto an SD Card (or on a Mac e.g. Disk Utility/ dd)
http://www.softpedia.com/get/CD-DVD-Tools/Data-CD-DVD-Burning/Win32-Disk-Imager.shtml

You will use the "Write" option to put the image from the disk to your card and choose the "2012-08-16-wheezy-raspbian_backup_mpi_master.img" image you just created.

[Note that there are probably more efficient ways of doing this – in particular maybe avoid expanding the filesystem in step 4 of the first section.]

32. Put the card into your second Pi and boot this. You should now have two Raspberry Pis on. Unless otherwise stated, all the commands below are typed from the Master Pi that you built first.

Using SSH instead of password login between the Pis

33. Sort out RSA to allow quick log in. This is the best thing to read:
http://steve.dynedge.co.uk/2012/05/30/logging-into-a-rasberry-pi-using-publicprivate-keys/
In summary (working on the MASTER Pi node)
$ cd ~
$ ssh-keygen -t rsa –C "raspberrypi@raspberrypi"
This set a default location of /home/pi/.ssh/id_rsa to store the key

Enter a passphrase e.g. "myfirstpicluster". If you leave this blank (not such good security) then no further typing of passphrases is needed.
$ cat ~/.ssh/id_rsa.pub | ssh pi@192.168.1.162 "mkdir .ssh;cat >> .ssh/authorized_keys"

34. If you now log into your other Pi and do
$ ls –al ~/.ssh
You should see a file called "authorized_keys" – this is your ticket to 'no login heaven' on the nodes

35. Now let us add the new Pi to the machinefile. (Log into it and get its IP address, as above)
Working on the Master Raspberry Pi (the first one you built):
$ nano machinefile

Make it read
192.168.1.161
192.168.1.162
[or whatever the two IP addresses you have for the machines are]

36. Now to run a little C code again. In the examples subdirectory of where you built MPI is the famous CPI example. First time you will need to enter the passphrase for the key you generated above (unless you left it blank) and also the password for the second Pi.
$ cd /home/pi/mpi_testing
$ mpiexec -f machinefile -n 2 ~/mpich_build/examples/cpi

Output is:
Process 0 of 2 is on raspberrypi
Process 1 of 2 is on raspberrypi
pi is approximately 3.1415926544231318, Error is 0.0000000008333387

If you repeat this a second time you won't need to type any passwords in. Hurray.

Note that we have NOT changed the hostnames yet (so yes, the above IS running on the two machines, but they both have the same hostname at the moment).

37. If you change the hostname on your second machine (see Appendix 1 "Hostname Script") and run:
$ mpiexec -f machinefile -n 2 ~/mpich_build/examples/cpi

Output:
Process 0 of 2 is on raspberrypi
Process 1 of 2 is on iridispi002
Now you can see each process running on the separate nodes.

## Scripts and other things to do

1. Power off the worker Pi and eject the card
$ sudo poweroff
We now have a copy of the WORKER nodes of the machine with all of the installed files for MPI in a single place. We now want to clone this card- as it has the ssh key on it in the right place. Shutdown your Pi very carefully
$ sudo poweroff

2. Remove the SD Card and pop it back into your SD Card writer on your PC/ other device. Use Win32 disk imager (or on a Mac e.g. Disk Utility/ dd) to put the image FROM your SD Card back to your PC:
http://www.softpedia.com/get/CD-DVD-Tools/Data-CD-DVD-Burning/Win32-Disk-Imager.shtml

You will use the "Read" option to put the image from the disk to your card
Let us call the image "2012-08-16-wheezy-raspbian_backup_mpi_worker.img"

3. Eject the card and put a fresh card into the machine. Use win32 disk imager to put image onto an SD Card (or on a Mac e.g. Disk Utility/ dd)
http://www.softpedia.com/get/CD-DVD-Tools/Data-CD-DVD-Burning/Win32-Disk-Imager.shtml

You will use the "Write" option to put the image from the disk to your card and choose the "2012-08-16-wheezy-raspbian_backup_mpi_master.img" image you just created.

[Note that there are probably more efficient ways of doing this – in particular maybe avoid expanding the filesystem in step 4 of the first section.]

## Hostname Script

If you want to rename each machine, you can do it from the Master node using:
ssh pi@192.168.1.162 'sudo echo "iridispi002" | sudo tee /etc/hostname'
ssh pi@192.168.1.163 'sudo echo "iridispi003" | sudo tee /etc/hostname'
ssh pi@192.168.1.164 'sudo echo "iridispi004" | sudo tee /etc/hostname'

You should then reboot each worker node
If you re-run step ( 36) above again, you will get:
$ mpiexec -f machinefile -n 2 ~/mpich_build/examples/cpi

Output:
Process 0 of 2 is on raspberrypi
Process 1 of 2 is on iridispi002
pi is approximately 3.1415926544231318, Error is 0.0000000008333387
This shows the master node still called raspberrypi and the first worker called iridispi002

## Using Python

There are various Python bindings for MPI. This guide just aims to show how to get ONE of them working.
1. Let us use mpi4py. More info at
http://mpi4py.scipy.org/
http://mpi4py.scipy.org/docs/usrman/index.html
$ sudo apt-get install python-mpi4py

2. We also want to run the demo so let us get the source too
$ cd ~
$ mkdir mpi4py
$ cd mpi4py
$ wget http://mpi4py.googlecode.com/files/mpi4py-1.3.tar.gz
$ tar xfz mpi4py-1.3.tar.gz
$ cd mpi4py-1.3/demo

3. Repeat steps 1 and 2 on each of your other nodes (we did not bake this into the system image)

4. Run an example (on your master node)
$ mpirun.openmpi -np 2 -machinefile /home/pi/mpi_testing/machinefile python helloworld.py

Output is:
Hello, World! I am process 0 of 2 on raspberrypi.
Hello, World! I am process 1 of 2 on iridispi002.

5. $ mpiexec.openmpi -n 4 -machinefile /home/pi/mpi_testing/machinefile python helloworld.py

Output is:
Hello, World! I am process 2 of 4 on raspberrypi.
Hello, World! I am process 3 of 4 on iridispi002.
Hello, World! I am process 1 of 4 on iridispi002.
Hello, World! I am process 0 of 4 on raspberrypi.

6. These are handy to remove things if your attempts to get mpi4py don't quite pan out
$ sudo apt-get install python-mpi4py
$ sudo apt-get autoremove

## Keygen script commands

cat ~/.ssh/id_rsa.pub | ssh pi@192.168.1.161 "cat >> .ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub | ssh pi@192.168.1.162 "cat >> .ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub | ssh pi@192.168.1.163 "cat >> .ssh/authorized_keys"
etc. for sending out the key exchanges if you want to do this again having generated a new key

## Getting Pip for Raspberry Pi

1. We can install Pip, which gives us a nice way to set up Python packages (and uninstall them too).
More info is at:
http://www.pip-installer.org/en/latest/index.html
http://www.pip-installer.org/en/latest/installing.html
$ cd ~
$ mkdir pip_testing
$ cd pip_testing

2. A prerequisite for pip is “distribute” so let’s get that first and then install pip. The sudo is because the installation of these has to run as root.
$ curl http://python-distribute.org/distribute_setup.py | sudo python
$ curl https://raw.github.com/pypa/pip/master/contrib/get-pip.py | sudo python

Notes on making MPI Shared Libraries for Raspberry Pi
MPI libraries can also be built “shared” so that they can be dynamically loaded. This gives a library file that ends in “.so” etc. not “.a” and we can do that by building those MPI libraries again. This is a repeat of steps above, but written out again using the suffix “_shared” on the directory names.

1. Make a directory to put the sources in
$ mkdir /home/pi/mpich2_shared
$ cd ~/mpich2_shared
2. Get MPI sources from Argonne.
$ wget http://www.mcs.anl.gov/research/projects/mpich2/downloads/tarballs/1.4.1p1/mpich2-1.4.1p1.tar.gz

[Note that as the MPI source updates, you can navigate to:
http://www.mpich.org/downloads/ to get the latest stable release version]

3. Unpack them.
$ tar xfz mpich2-1.4.1p1.tar.gz
[Note: You will need to update this as the version of MPICH2 increments]

4. Make yourself a place to put the compiled stuff – this will also make it easier to figure out what you have put in new on your system.
$ sudo mkdir /home/rpimpi_shared/
$ sudo mkdir /home/rpimpi_shared/mpich2-install_shared
[I just chose the “rpimpi_shared” to replace the “you” in the Argonne guide and I made the directory creation in two steps]

5. Make a build directory (so we keep the source directory clean of build things)
$ mkdir /home/pi/mpich_build_shared

6. Change to the BUILD directory
$ cd /home/pi/mpich_build_shared

7. Now we are going to configure the build
$ sudo /home/pi/mpich2_shared/mpich2-1.4.1p1/configure -prefix=/home/rpimpi_shared/mpich2-install_shared --enable-shared
[Note: You will need to update this as the version of MPICH2 increments]

8. Make the files
$ sudo make

9. Install the files
$ sudo make install

10. Finally add the place that you put the install to your path
$ export PATH=$PATH:/home/rpimpi_shared/mpich2-install_shared/bin

Note to permanently put this on the path you will need to edit .profile
$ nano ~/.profile
… and add at the bottom these two lines:
# Add MPI Shared to path
PATH="$PATH:/home/rpimpi_shared/mpich2-install_shared/bin

# References


Conroy, Dave. "Turn Your Raspberry Pi Into a WiFi Hotspot with Edimax Nano USB EW-7811Un (RTL8188CUS Chipset)." *DaveConroycom RSS*. N.p., 10 July 2013. Web. 28 Oct. 2013.

“HPL Software." netlib.org. N.p.. Web. 1 Dec 2013. <http://www.netlib.org/benchmark/hpl/software.html>.

Ladyada. "Setting up a Raspberry Pi as a WiFi Access Point." *Setting-up-a-raspberry-pi-as-a-wifi-access-point*. N.p., n.d. Web. 28 Oct. 2013.

"Message Passing Interface." *wikipedia*. N.p.. Web. 1 Dec 2013. <http://en.wikipedia.org/wiki/Message_Passing_Interface>.

"Steps to make Raspberry Pi Supercomputer." *http://www.southampton.ac.uk/~sjc/raspberrypi/*. N.p. Web. 1 Dec 2013. http://www.southampton.ac.uk/~sjc/raspberrypi/pi_supercomputer_southampton.htm"

Tokar, Jacek. "Hotspot – WiFi Access Point." *Raspberry at Home*. N.p., 4 June 2013. Web. 28 Oct. 2013.