After following the instructions in **pi_supercomputer_southampton.pdf** document, there are some corrections.

Preparation and Notes

Once the new full SD image has been saved on the PC and written to a new card you can begin to create nodes for the cluster. However, you cannot simply boot a new node with the master node connected to the network because each new node first powers up with the same static IP as the master node. Disconnect the master node from the network, then configure all new node in the cluster. Once all nodes have their proper static IP's, reconnect the master node to complete the cluster configuration.

Current recommended practice is to install the freshly copied SD cards into all nodes in the cluster including the master node, then power up ALL nodes in the cluster with ALL NODES DISCONNECTED from the network switch.

IMPORTANT NOTE: to edit any file except pi's .profile, you must use '**sudo vi *filename***' or the file will be read only.

Master Node

Connect the master node to the network switch and ssh into the master node as user pi.

- edit .profile and correct the comment on the second last line from 'App to "Add'
- NOTE: editing files in /etc will require root access via '**sudo vi *filename***'
- edit /etc/hostname – change it to the new host name (i.e. rshpi01 for master node)
- edit /etc/hosts – change the last line from raspberrypi to the new host name (rshpi01)
- edit ~/mpi_testing/machinefile and add all static ip's for all nodes that will be in the cluster (i.e. 10.1.1.41 is in the file, add 10.1.1.42, 10.1.1.43, 10.1.1.44, etc.)
- reboot the node with '**sudo shutdown -r now**'
- reconnect to the master node and verify all changes
- DISCONNECT the master node from the network by unplugging the network cable from the switch

All Other Cluster Nodes

WITH THE MASTER NODE DISCONNECTED from the network, connect the next node in sequence to the network.

Node numbering is based on static IP address (last octet). Numbering is sequential starting with the master node (i.e. 10.1.1.41), so node 2 would be 10.1.1.42 with machine name rshpi02.

- edit .profile and correct the comment on the second last line from 'App to "Add'
- NOTE: editing files in /etc will require root access via '**sudo vi *filename***'
- edit /etc/hostname – change it to the new host name (i.e. rshpi0x)
- edit /etc/hosts – change the last line from raspberrypi to the new host name (rshpi0x)
- edit /etc/network/interfaces – change the static IP to the proper one (i.e. 10.1.1.4x)
- reboot the new node with '**sudo shutdown -r now**'
- check the static IP has changed by pinging the node's new static IP.

Leave the node connected to the network as you progress to the next node. When complete, all

nodes EXCEPT the master node should be connected to the network and visible via ping.

Cluster Complete

Now that the cluster is configured to this point, reconnect the master node to the network and ssh to the master node, logging in as user pi. All subsequent work will be done on or through the master node.

- From the master node, ssh to each node in the cluster. This will store the key for the node in the master node's keystore (~/.ssh/known_hosts) for the user pi.
- Create a public private key pair on the master node for cluster operations: **ssh-keygen -t rsa -C "user@email.com"** (substitute your email address here). Accept the default filename and press enter when prompted for a passphrase. This is less secure, but this cluster won't be exposed to the internet.
- Copy this key to all other nodes on the cluster: **cat ~/.ssh/id_rsa.pub | ssh pi@10.1.1.4x "mkdir .ssh;cat >> .ssh/authorized_keys"**. You will have to enter the nodes password for user pi for each node.
- Test the changes by using ssh to login to each node as user pi. You should no longer be asked for a password.
- Copy the machinefile you edited on the master node to each node in the cluster: **cat ~/mpi_testing/machinefile | ssh pi@10.1.1.4x "cat >mpi_testing/machinefile"**
- Verify machinefile on each node: **ssh pi@10.1.1.4x "ls -l mpi_testing; cat mpi_testing/machinefile"**

Note: After using this configuration for some time, I have now moved and renamed the file 'machinefile' from /mpi_testing to the user's home directory: "**mv ~/mpi_testing/machinefile ~/.mpihosts**". Changing the name from machinefile to .mpihosts follows the spirit of host files more closely, and placing the file in the user's home directory makes it easier to find and use from other working directories. Please see the "pi supercomputer programming rsh" document for further information.

Parallel Computing

Test the parallel computing configurations (completed on the master SD image as per the Southhampton document) by running the cpi test program:

cd ~/mpi_testing
mpiexec -f machinefile -n X ~/mpich_build/examples/cpi

where -n X is the number of nodes in the cluster (i.e. 4). You should see output from each node in the cluster. Due to the keygen operation above, no password should be required to access all the nodes in the cluster for the operation.

IMPORTANT NOTE: Before you can run a parallel application on the cluster, it must be compiled and the executable present on ALL machines in the cluster. The easiest method to accomplish this is to use the following commands:

First, on the master node:

cd ~/mpich_build/examples; sudo make icpi; sudo make hellow

Then on each node in the cluster:

ssh pi@10.1.1.4x "cd ~/mpich_build/examples; sudo make icpi; sudo make hellow"

Let's test hellow to confirm all the nodes have the compiled application:

cd ~/mpi_testing; mpiexec -f machinefile -n X ~/mpich_build/examples/hellow

The program should execute on all nodes and display a greeting from each node in the cluster.

NOTE: the examples directory contains a Makefile to build several example applications. However, at this time I have only tested icpi and hellow. The application pmandel generates compiler warnings and/or errors, and seems to require other components to be compiled. At this time it has not been successfully tested.

Alternatively, you could create the following two shell scripts to compile the main examples and another to run them:

make_tests.sh

```
#compile suite of example programs for mpiexec
cd ~/mpich_build/examples
sudo make hellow
sudo make parent
sudo make child
sudo make srtest
sudo make icpi
```

run_tests.sh

```
#test primary examples on current nodes (4)
mpiexec -f machinefile -n 4 ~/mpich_build/examples/cpi
mpiexec -f machinefile -n 4 ~/mpich_build/examples/hellow
mpiexec -f machinefile -n 4 ~/mpich_build/examples/srtest
mpiexec -f machinefile -n 4 ~/mpich_build/examples/icpi
 #mpiexec -f machinefile -n 1 ~/mpich_build/examples/parent : -n 3
~/mpich_build/examples/child
```

(NOTE: parent/child is commented out at this time as the execution fails with communication errors)

C Programming

The raspian linux operating system provides a C (and a C++) compiler as part of the standard operating system. To demonstrate the C compiler, issue the following commands:

cd ~
mkdir testing
cd testing

Create the file 'hello.c' using vi so that it looks as follows:

```
#include <stdio.h>

int main(void) {
  printf("Hello from Pi!\n");
  return 0;
}
```

Be sure to leave a blank line after the closing curly brace or the compiler may complain. Compile and test the program:

```
cc -o hello hello.c
./hello
```

You should see the output "Hello from Pi!" on the command line.

Future Tasks?

- rename the default user 'pi' to 'richard'. It would be easier to create a new user 'richard', but I'd prefer to rename 'pi' instead. However, this is not a simple process.