Introduction - Parallel Computing (from pi supercomputing rsh.odt)

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.

Other Programs

Let's say you have now either written or downloaded a program from another location and wish to test it on the Pi cluster.

If the program is not already on the master Pi, then use a program such as WinSCP to copy the source code to the master Pi. You can either create a new working directory under /home/pi (~/ as user pi) or use the mpi_testing directory previously created. This document will assume ~/mpi_testing is used.

Log on to the master Pi as user 'pi' and change to the testing directory. Verify that your source code is in the directory. Now you can compile the program on the master pi to verify that it compiles wihtout errors. If there are errors, you will have to correct them, which is beyond the scope of this document.

cd ~/mpi_testing
mpicc -o *progname progname.c*

If the program compiles without errors, you can test it on the master Pi thusly:

mpiexec -n 4 *./progname*

You don't have to use 4 threads, but you should use more than 1 to test the parallel processing. Note that you cannot just use the program name but must use the more specific pathname *./progname* due to the manner in which mpiexec searches for the programs (especially important when using

muliple Pi's in a cluster).

To test the program on several Pi's in a cluster, first ensure that your machinefile is in the working directory and verify that the machines you want to use are listed in the machinefile.

You must copy the file to the node and then compile it on that node before you can execute the program using that node.

cat *progname.c* | ssh *pi@node_ip* "mkdir *working_dir*; *cat >working_dir/progname.c*"

cat *progname.c* | ssh *pi@node_ip* "cat >*working_dir/progname.c*"

where the *progname.c* is the program source file's name, *node_ip* is the IP address of the node (i.e. 10.1.1.4x) and *working_dir* is the working directory on the node. If the working directory does not yet exist, you must create it first as in the first example.

Now you can compile the program on each node:

ssh pi@*node_ip* "cd *working_dir*; /home/rpimpi/mpich2-install/bin/mpicc -o *progname progname.c; ls -l*"

Again, you should compile without errors. The 'ls' command after the compile simply shows you the executable has been created. Note that you must provide the full pathname to the mpicc command or the remote node will report 'bash: mpicc: command not found'. I don't yet know why this is the case (a remote 'echo $PATH$' shows the bin directory in path), but giving the full path works, so we won't worry about this for now.

Once the program has been compiled on all nodes, you can execute the program on the master node.

mpiexec -f machinefile -n 4 *./progname*

Shell Scripts

The above commands work, but it would be nice to have the copy and compile process automated such that you only provide the necessary information to copy and compile the program, such as the last octet of the node IP (nodes share the same subnet) and the program's name (omitting the .c) such as the following:

./mpicpcc.sh *progname node_octet*

The batch file mpicpcc.sh can be created on the master node in whichever directory you choose (user's root, user's bin, or working directory). Assuming the working directory will be mpi_testing, the file is as follows:

```
#!/bin/bash
#copy and compile an MPI c program to another cluster node


#test number of arguments
#if [ $# -lt 2 ]; then
#test for empty arguments instead
#if [ -z $1 ] || [ -z $2 ]
```

```
then
        echo "usage: ./mpicpcc.sh progname node_octet"
else
#copy the source file to the node's MPI testing directory
        cat $1.c | ssh pi@0.1.1.$2 "cat >mpi_testing/$1.c"
#compile the source file on the node
        ssh pi@0.1.1.$2 "cd mpi_testing; /home/rpimpi/mpich2-install/bin/mpicc -o $1 $1.c; ls -l"
fi
```

Shell Script Update

After working with the system for a bit longer and using the above batch files, I have modified the scripts into two scripts – one to compile locally or remotely, and one to exeucte a program compiled locally by copying the binary to each node and then running mpiexec. This means that the compile script is now only really used in 'local' mode.

Both scripts have been renamed and are shown below:

mpic.sh ***(to be added)***

mpix.sh***(to be added)***

Update – October 9, 2012

On a website about MPI 2 I saw the 'machinefile' file named '.mpihosts'. I like this as it would allow me to put the file in the user (pi) home directory and then invoke it from mpix.sh from any working directory without requiring multiple copies of the 'machinefile' file.

On the currently built system, this must be done in a couple of steps:

First, on the master node, move and reanme machinefile from a working directory like 'mpi_testing' to ~/ .mpihosts'.

cd ~; mv mpi_testing/machinefile .mpihosts; more .mpihosts

Next find all other copies of the file 'machinefile' on the master node.

find . -name machinefile -print

This should give you a list of full pathnames to 'machinefile'. In my current case this was just the other testing directory './jburkardt'.

Now copy .mpihosts to all other nodes and remove all copies of machinefile. First on the master:

rm ./jburkardt/machinefile

And for all nodes

ssh pi@10.1.1.42 "mv mpi_testing/machinefile .mpihosts; rm ./jburkardt/machinefile;

find . -name machinefile -print; ls -al"

Do the same for the other nodes in the cluster.

Now only on the master node, edit ./bin/mpix.sh and change 'machinefile' to '~/.mpihosts'. Run one of the test programs to verify the changes are correct.