Cluster Installation Instructions

Contents

1 Introduction

PST.tools can be run on a Beowulf cluster. What follows are instructions for building a Beowulf cluster with the software required to both compile and run PST.tools. The Ubuntu desktop linux distribution, versions 12.04 (or 14.04 soon), was chosen for these instructions because it is simple to install and provides a GUI for the comfort of novice linux users. However, there is no specific linux distribution specified as a system requirement for PST.tools. Operating system choice, and any implied deviations from these instructions, are at the user's discretion. A significant portion of these instructions have been adapted from an online tutorial by Serrano Pereira, which uses Ubuntu, was very helpful to the author and was found at

http://byobu.info/articles/Building_a_simple_Beowulf_cluster_with_Ubuntu.html

This manual does not address how to secure the cluster. However it is highly recommended that some thought goes into security before making this cluster available online, or to users who may not be completely trustworthy etc.

It is tempting to copy and paste the commands in this document directly into a terminal, rather than typing them all in. Please be careful that all the characters are indeed correct. Sometimes, certain programs like to substitute certain characters, like dashes, with extended ascii character equivalents. For example, if you copy the command for installing SSH into the terminal and receive an error stating that apt-get does not exist; it is more likely that apt-get is actually "spelt" wrong, even though it looks correct. Depending on the preferences making the pdf of the instructions the "dash" character can be changed, so cut-and-pasting from the pdf will not work.

2 Hardware Requirements

In terms of performance requirements, the types of calculations intended affect these greatly. However, it is not recommended to use computers with very different performance levels, as that can create bottlenecks. This manual assumes 2 (or more) PCs with at least one hard disk and network interface card (NIC). One of these PCs will become the master node and the other/s will become compute nodes. Optionally, the master node may have an additional NIC for accessing another network for internet access etc (external NIC). An internet connection is required to download the packages required to run PST.tools. A network switch is also required for inter-node communication.

3 Operating System Installation

The instructions provided with the operating system should be followed. This manual assumes a fresh working Ubuntu installation. There is one caveat to be aware of when installing Ubuntu in a dual boot configuration with Windows. With some modern motherboards (and, in particular, the motherboards we have now 2/2/15), UEFI options, for boot devices in the BIOS, can prevent a new Ubuntu installation from booting. If, when the computer is started, you do not see the Grub menu allowing you to select operating system and, instead, load windows by default, you might need to select a **non UEFI** boot option for the installation media and try the installation process again.

4 Master Node Setup

4.1 Network Settings

The NIC used for inter-node communication (cluster NIC) should have a **static IP number**. First you need to work out what the master node calls each NIC; they will be named eth0, eth1... etc. In a terminal, run:

ifconfig

This shows the IP numbers that are currently set. Don't use the Ubuntu GUI network editor to manually configure IPv4. Instead, edit the text file */etc/network/interfaces*

sudo nano /etc/network/interfaces

This command opens (at root level) a command line editor. Set the cluster NIC (eth1 in this example, but this will depend on the output from *ifconfig*) to static, with the following settings.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

# The secondary network interface
auto eth1
iface eth1 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
```

The last settings are actually setting up the static IP address for the cluster NIC (eth1).

If the Master node has an external (internet/other) NIC (eth0 in this example), set it to use DHCP (which is the default) unless your external network requires otherwise. After saving these changes, reboot the computer.

4.2 Add Nodes to Hosts File

Adding each node to the hosts file allows inter-node connections via host name rather than IP address. The same hosts file information is required on all nodes, not just the master node. Edit the hosts file */etc/hosts*

sudo nano /etc/hosts

It should look similar to the example below:

127.0.0.1 localhost
192.168.1.100 master
192.168.1.101 node1
192.168.1.102 node2
192.168.1.103 node3

Make sure it doesn't look like this (which is what happens by default, need to ensure only the static IP number from step 4.1, need to get rid of the callback address 127.0.1.1):

127.0.0.1 localhost
127.0.1.1 master
192.168.1.101 node1
192.168.1.102 node2
192.168.1.103 node3

or this:

127.0.0.1 localhost
127.0.1.1 master
192.168.1.100 master
192.168.1.101 node1
192.168.1.102 node2
192.168.1.103 node3

Otherwise other nodes will try to connect to localhost (ie themselves) when trying to reach the master node.

4.3 Create User Account for MPI Jobs

The command below creates a new user with username "mpiuser" and user ID 999. Giving a user ID below 1000 prevents the user from showing up in the login screen for desktop versions of Ubuntu. It is important that all MPI users have the same username and user ID. The user IDs for the MPI users need to be the same because we give access to the MPI user on the NFS directory later. Permissions on NFS directories are checked with user IDs. Create the user like this,

sudo adduser mpiuser --uid 999

You may use a different user ID (as long as it is the same for all MPI users). Enter a password for the user when prompted. It's recommended to give the same password on all nodes so you have to remember just one password. The above command should also create a new directory */home/mpiuser*. This is the home directory for user *mpiuser* and we will use it to execute jobs on the cluster.

4.4 Update and Reboot

Before installing any packages, it is a good idea to ensure the system is up to date.

sudo apt-get update

sudo apt-get dist-upgrade

sudo reboot

The first commands checks for updated packages, the second gets them and then the system is rebooted.

4.5 Install and setup the Network File System

Files and programs used for MPI jobs (jobs that are run in parallel on the cluster) need to be available to all nodes, so we give all nodes access to a part of the file system on the master node. Network File System (NFS) enables you to mount part of a remote file system so you can access it as if it is a local directory. To install NFS, run the following command on the master node:

sudo apt-get install nfs-kernel-server

We will use NFS to share the MPI user's home directory (*/home/mpiuser*) with the compute nodes. It is important that this directory is owned by the MPI user so that all MPI users can access this directory. Since we created this home directory with the *adduser* command earlier, it is already owned by *mpiuser*.

[Aside: If you want to alter the current system, for example, if you use a different directory that is not currently owned by *mpiuser*, you must change its ownership as follows,

sudo chown mpiuser:mpiuser /path/to/shared/dir]

Now share the */home/mpiuser* directory of the master node with all other nodes. Edit the text file */etc/exports*.

sudo nano /etc/exports

Add the following line to this file,

*/home/mpiuser *(rw,sync,no_subtree_check)*

After the first install you may need to restart the NFS daemon:

sudo service nfs-kernel-server restart

This also exports the directories listed in */etc/exports*. In the future when the */etc/exports* file is modified, you need to run the following command to export the directories listed in */etc/exports*:

sudo exportfs -a

The */home/mpiuser* directory should now be shared through NFS.

The firewall is enabled by default on Ubuntu. The firewall on the master node will block access when a client tries to access an NFS shared directory. So you need to add a rule with UFW (a tool for managing the firewall) to allow access from a specific subnet. If the IP addresses in your network have the format 192.168.1.*, then 192.168.1.0 is the subnet. Run the following command to allow incoming access from a specific subnet, for the whole subnet,

sudo ufw allow from 192.168.1.0/24

4.6 Setup passwordless SSH for communication between nodes

For the cluster to work, the master node needs to be able to communicate with the compute nodes, and vice versa. Secure Shell (SSH) is usually used for secure remote access between computers. By setting up passwordless SSH between the nodes, the master node is able to run commands on the compute nodes. This is needed to run the MPI daemons on the available compute nodes.

First install the SSH server:

sudo apt-get install ssh

Now we need to generate an SSH key for all MPI users on all nodes. The SSH key is by default created in the user's home directory. Remember that in our case the MPI user's home directory (*/home/mpiuser*) is actually the same directory for all nodes; */home/mpiuser* on the master node. So if we generate an SSH key for the MPI user on one of the nodes, all nodes will automatically have an SSH key. In a terminal, run:

su mpiuser

ssh-keygen

When asked for a passphrase, leave it empty (hence passwordless SSH).

When done, all nodes should have an SSH key (the same key actually). The master node needs to be able to automatically login to the compute nodes. To enable this, the public SSH key of the master node needs to be added to the list of known hosts (this is usually a file *~/.ssh/authorized_keys*) of all compute nodes. But this is easy, since all SSH key data is stored in one location: */home/mpiuser/.ssh/* on the master node. So instead of having to copy master's public SSH key to all compute nodes separately, we just have to copy it to master's own authorized_keys file. There is a command to push the public SSH key of the currently logged in user to another computer. In a terminal, as user *"mpiuser"*, run:

ssh-copy-id localhost

Master's own public SSH key should now be copied to */home/mpiuser/.ssh/authorized_keys*. But since */home/mpiuser/* (and everything under it) is shared with all nodes via NFS, all nodes should now have master's public SSH key in the list of known hosts.

4.7 Install Open MPI

Message Passing Interface (MPI) is a standardized and portable message-passing system designed to function on a wide variety of parallel computers. There are several well-tested and efficient implementations of MPI and the Open MPI implementation is open source and has good compatabilty with the Boost C++ libraries (libopenmpi-dbg does not exist in 14.04).

sudo apt-get install openmpi-bin openssh-client openssh-server libopenmpi-dbg libopenmpi-dev

Open MPI will use the nodes listed in a hostfile located in the *mpiuser* home directory */home/mpiuser/hosts*. It is simply a text file listing the hostnames for each available node in the cluster (including the master node). Run, as the user *mpiuser*,

sudo nano ~/hosts

It's contents should be similar to this (the node list needs to be added to this file)

master
node1
node2
node3

For more detail, see *https://www.open-mpi.org/doc/v1.8/man1/mpirun.1.php*.

4.8 Install Library Dependencies

PST.tools makes use of several open source libraries; GSL, Boost, TNT/JAMA and wxwidgets.

sudo apt-get install libgsl0ldbl libgsl0-dev

sudo apt-get install libboost-all-dev

sudo apt-get install libtnt-dev libjama-dev

sudo apt-get install libwxgtk2.8-dev libwxgtk2.8-dbg

4.9 Create Local Scratch Directory

In PST.tools the default location for scratch disk on all nodes is set in the program code (for example, main.cpp).

const std::string scratch_directory="/scratch/mpiuser/PST/";

There needs to be a directory matching this path (or whatever path you place here) on all nodes. The directory should have permissions set so that the mpiuser account has full control of the directory.

To achieve this:

cd /

sudo mkdir -m a=rwx scratch

Note, this directory has full control allowed for all users! If you wish to be more precise in your assignment of access, then do so. Next, switch to the *mpiuser* and make a directory in the scratch directory that belongs to *mpiuser*.

su mpiuser

cd /scratch

mkdir mpiuser

cd mpiuser

mkdir PST

exit

4.10 Install Apt-Cacher-NG

Apt-Cacher-NG is a caching proxy server that saves downloaded packages locally on your server. Any other nodes can access the cached copies, rather than re-downloading the same packages from the internet multiple times. Any connected machine can be the Apt-Cacher-NG proxy server, provided it has internet access. In a standalone cluster, this would most likely be set up on the master node.

On the machine storing the cache, run

sudo apt-get install apt-cacher-ng

Then edit the text file */etc/apt-cacher-ng/acng.conf*

sudo gedit /etc/apt-cacher-ng/acng.conf

Ensure that the following lines are uncommented (remove # from beginning of line, if present);

CacheDir: /var/cache/apt-cacher-ng

LogDir: /var/log/apt-cacher-ng

Add a line setting the bindaddress to 0.0.0.0, after the commented line below.

 # BindAddress: localhost 192.168.7.254 publicNameOnMainInterface

BindAddress: 0.0.0.0

Check that verbose logging is enabled.

VerboseLog: 1

To run the apt-cacher service, we need to enable the pid file in the configuration.

PidFile: /var/run/apt-cacher-ng/pid

Save and close the file. Then, restart the apt-cacher-ng service,

sudo /etc/init.d/apt-cacher-ng restart

Access the report page of the apt-cacher-ng web interface using the URL below, with the IP address of the apt-cacher-ng proxy you have created.

http://192.168.1.100:3142

From the report home page we need to copy the Proxy URL. This is where all nodes will access their updates. To set each computer, including the master node, to access the proxy, edit the text file */etc/apt/apt.conf.d/02proxy*.

sudo nano /etc/apt/apt.conf.d/02proxy

Add the Proxy URL that you copied from the report page of the apt-cacher-ng web interface

Acquire::http { Proxy "http://192.168.1.100:3142"; };

and save the file.

4.11 Install Cluster SSH

Cluster SSH (CSSH) allows you to manage multiple Linux machines over SSH simultaneously. This step is optional. Lots of time can be saved installing and administering the compute nodes using this tool. However, you need to be particularly careful when executing commands on all nodes. As such, we leave the simple setup requirements of Cluster SSH to those readers who are confident to learn from its manual.

sudo apt-get install clusterssh

4.12 Cluster Boot Script

To save yourself from turning all the computers on individually, write a script that turns on all the computers via the network. If your hardware supports wake on lan, there will be a setting (somewhere) in the BIOS that turns it on and off. So ensure it is enabled. Install Wake-on-lan, which allows you to send magic packets to other computers. A magic packet will turn a computer on over the network, if a computer NIC is set appropriately and receives the correct packet.

sudo apt-get install wakeonlan

Create a "bin" folder in the home directory *~/bin*.

cd ~

mkdir bin

At next login, this will automatically be added to $PATH, so that you can call the scripts in bin without prefixing with ./ or specifying a path to the script.

Create a wake-nodes script in *~/bin*.

sudo nano ~/bin/wake-nodes

The script should be similar to below:

```
#!/bin/bash
# This script will send a magic packet to all the compute nodes so they wake on lan, using wakeonlan

wakeonlan -i 192.168.1.255 74:d4:35:b3:11:3b # node1
wakeonlan -i 192.168.1.255 74:d4:35:b3:12:43 # node2
wakeonlan -i 192.168.1.255 74:d4:35:b3:13:5d # node3
```

The IP broadcast address must match your network (192.168.1.255 in our example). The mac address of the node's cluster NIC (74:d4:35:b3:11:3b for node1 in our example) can be obtained by running *ifconfig* on each node to get its mac address (*HWaddr*) to put into the wake-nodes script. So

first create the script, leaving it empty, then setup all the nodes and get their mac addresses and then complete the script (see 6.11).

Make sure the script has execute permission. Then you can call wake-nodes in the terminal to turn on all the compute nodes (including the master node, from a separate workstation etc, if required).

sudo chmod +x wake-nodes

The first time a compute node is started, after power has been disconnected, a magic packet won't start the compute node. In this case the computer must be booted manually, **with the button**. This also needs to be done after disconnecting the power, when you reconnect power *wakeonlan* wont work the first time.

4.13 Cluster Shutdown Script

This script is only necessary **if** CSSH has not been set up. With CSSH installed, all that is required to shutdown the nodes is to log into all nodes at once and execute *node-shutdown*; which is a script you will install on each compute node, that unmounts the nfs share before calling *shutdown*.

In the absence of CSSH, to use the cluster shutdown script, create a script in the *~/bin* directory you created in the step above, called *shutdown-nodes*. This script must be run with an account that has permission to use sudo. It works better if the account has passwordless SSH access to all nodes (similarly to above but with some important differences). On the computer calling the script, either the master node or a connected workstation, logged in as the account that will call the script, run

ssh-keygen

Leave the passphrase empty. Then for each node (ie: node1, node2 ... etc), from the computer calling the script run

ssh-copy-id node1

ssh-copy-id node2 ... etc

The script:

```
#!/bin/bash
# This script will ssh into all the compute nodes and call their node-shutdown scripts, which unmount the nfs share before calling shutdown so it doesn't hang.
# DO NOT CALL IT (or manually call the compute node's shutdown script) UNTILL THE NFS SHARE HAS BEEN MOUNTED,
# OTHERWISE IT MAY MOUNT BETWEEN THE UNMOUNT AND THE SHUTDOWN AND STILL HANG
# For this to work, the correct password must be supplied in the script (note that sending the password like this, as plain text, is a security risk).
# It would be advisable to setup passwordless SSH for the “d” user (ie the username chosen at the the beginning of the Ubuntu install) to avoid having to enter the password for each ssh connection to each node.
# To do this;
#       on the master node, logged in as d, run ssh-keygen and leave the passphrase empty
#       then for each node (ie: node1, node2 ... etc), from the master node logged in as d,
```

```
#          run ssh-copy-id node1
#Alternatively, consider using CSSH for this! Simply connect to all nodes and
#run sudo  /home/d/bin/node-shutdown

ssh node1 'echo "my password" | sudo -S /home/d/bin/node-shutdown'
echo "node1 shut down"

ssh node2 'echo "my password" | sudo -S /home/d/bin/node-shutdown'
echo "node2 shut down"

ssh node3 'echo "my password" | sudo -S /home/d/bin/node-shutdown'
echo "node3 shut down"
```

You will have to put your password inside the quotes, which makes this solution less than ideal because the password is sent across the network as clear text. CSSH is a better way. Make sure the script has execute permission.

```
sudo chmod +x shutdown_nodes
```

5 Development Environment Setup

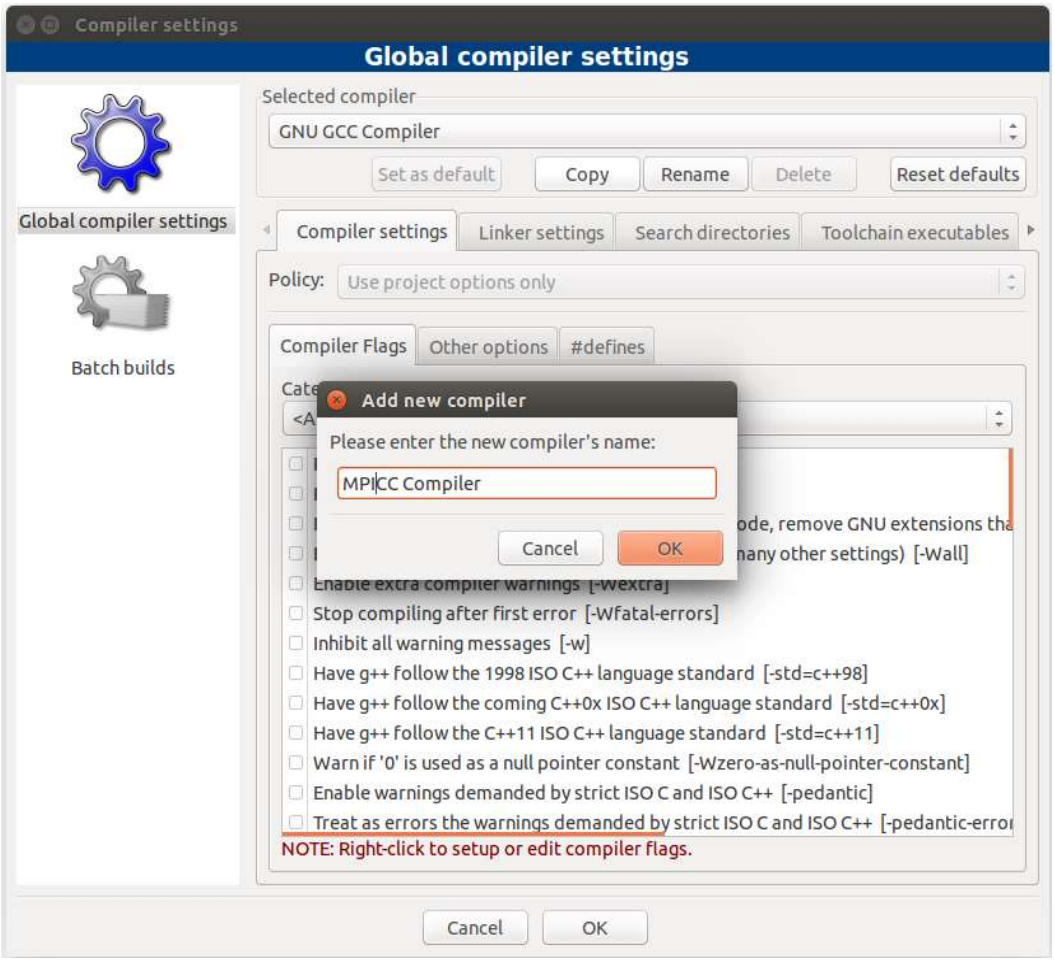
While it is not strictly necessary to install the full development on the cluster, it is highly recommended that PST.tools is compiled on the master node when running on a cluster, to ensure that the program runs as expected. However, alternate methods of compiling on the master node are left to the user's discretion.

To get an up-to-date version of wxFormbuilder, the PPA needs to be added to the repositories.

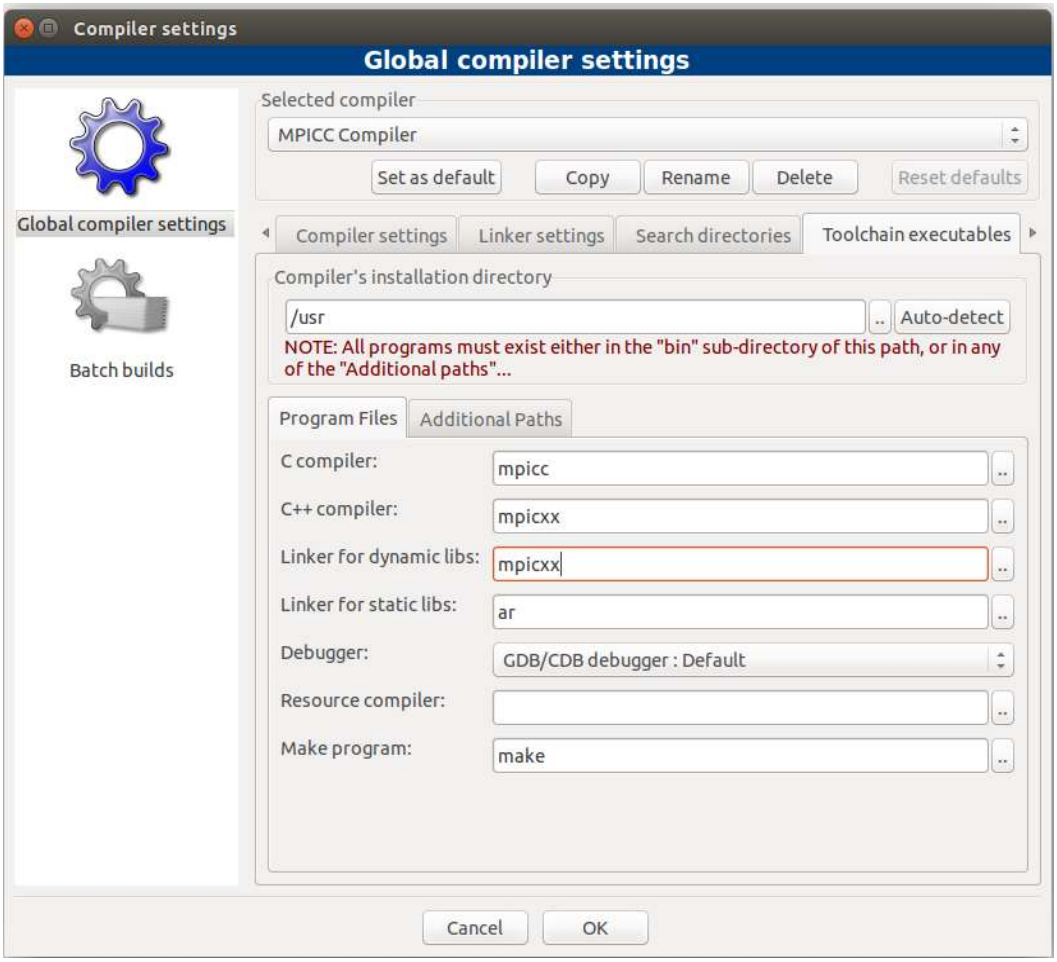
```
sudo add-apt-repository ppa:wxformbuilder/release

sudo apt-get update
```

Install wxFormBuilder, GNU C++ compiler and Code::Blocks from the Ubuntu Software Center. The wxFormBuilder is used for the GUI (depending, may not be necessary). Code::Blocks is a cross platform, open source Integrated Development Environment (IDE) for C++. Before you can compile PST.tools you will need to add the Open MPI compiler to Code::Blocks.

In Code::Blocks, go to Settings/Compiler... which loads the Global compiler settings. With the GNU GCC Compiler selected, click copy and change the name to MPICC Compiler.



Select the Toolchain executables tab and change to below



If you have been provided with a codeblocks project file, open it. It should have all the source files added and the correct linker settings in the project build options. Otherwise, create a new project. Select console application as the type and C++ as the language. Give it a title and location to save. Select the MPICC compiler option and make sure the release configuration checkbox is ticked. Right click on your project name in the management pane on the left and select add files. Add all of the source code files (including EasyBMP.cpp and EasyBMP.h, which are in the EasyBMP_1.06 subfolder). In the file main.cpp, make sure the line

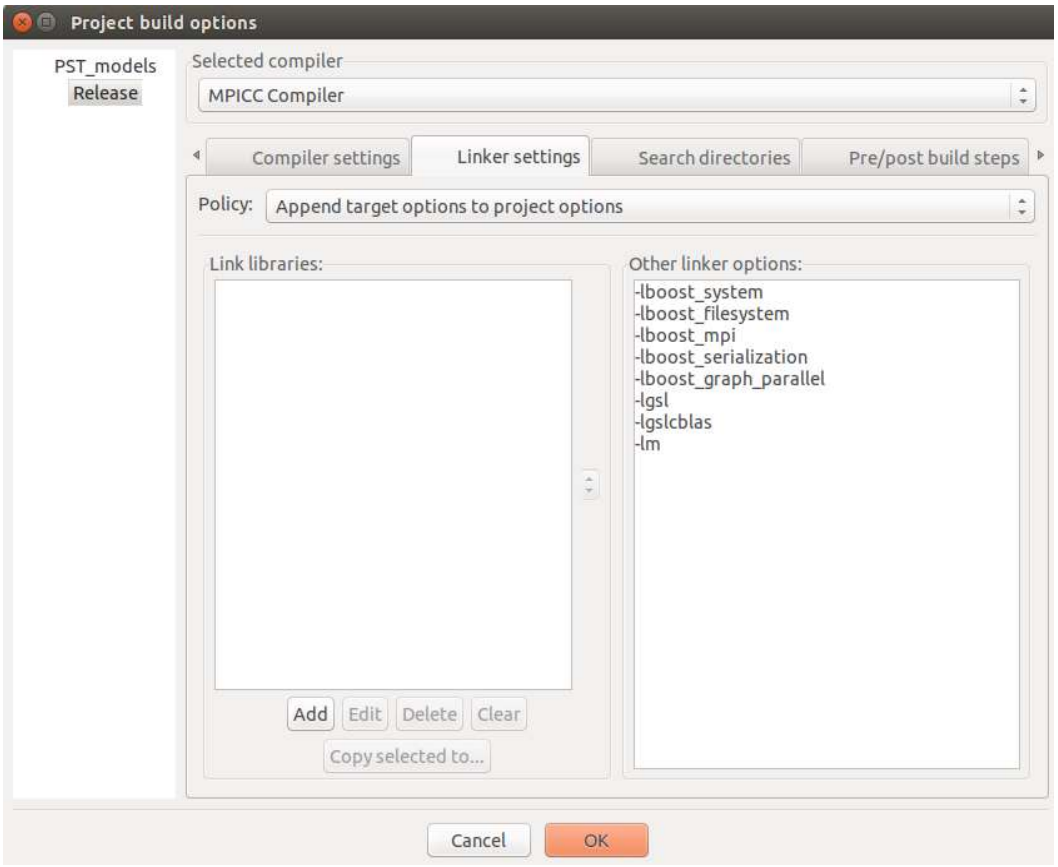
#define USING_MPI 1

is not commented out, otherwise the MPI code will not be compiled.

You will need to set some linker settings for your project. Go to Project/Build Options... In the linker settings tab enter the following linker options

-lboost_graph_parallel
-lboost_mpi
-lboost_system

-lboost_serialization
-lboost_filesystem
-lgsl
-lgslcblas
-lm



Now rebuild the entire project by clicking the button with 2 blue arrows making a circle (like a recycling symbol). The executable file will be created in the *./bin/release* folder inside the project folder.

5.1 Directory Structure

The compiled executable file, and the required input files and directory structure, must be placed in the */home/mpiuser/* directory, usually in a bin folder, so that they are available to all nodes. In our example, the top level folder will be */home/mpiuser/bin/PST*.

su mpiuser

mkdir ~/bin

mkdir ~/bin/PST

At the top level are the executable and three folders; input, output and config.

mkdir ~/bin/PST/input

mkdir ~/bin/PST/output

mkdir ~/bin/PST/config

To copy your compiled executable to the top level directory, run the following command from the *Release* directory containing the executable.

sudo cp PST.tools /home/mpiuser/bin/PST/PST.tools

The config folder contains elements.csv, a file that is read when *PST.tools* is launched (additional configuration files may be stored there in future versions). If you need to copy *elements.csv* to its destination from another user's home directory, you can use a command like this (from the directory containing *elements.csv*).

sudo cp elements.csv /home/mpiuser/bin/PST/config/elements.csv

The input directory holds text files that are read when *PST.tools* is launched and the output directory is where the results are written. Future versions may allow these paths to be customised by the user.

6 Compute Node Setup

Having set up the master node, we now need to set up each of the compute nodes.

6.1 Network Settings

The compute node NIC should have a static IP number. Don't use the Ubuntu GUI network editor to manually configure IPv4. Instead, edit the text file */etc/network/interfaces*

Set the cluster NIC to static, with the following settings, so that each node has a unique IP number.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.1.101
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    gateway 192.168.1.100
    dns-nameservers <IP number of the gateway/DNS of the external network> (This is optional, use if internet connection sharing is enabled on the master node, this is the dns that is assigned by dhcp on the external NIC, or ask your system administrator)
```

After saving these changes, reboot the machine.

Ubuntu desktop has some startup delays when it thinks the network isn't responding... These can be removed by editing the text file */etc/init/failsafe.conf* and commenting out all the "waiting for network config..." and sleeps etc.

6.2 Compute Node Grub Settings

The following steps prevent the grub boot menu on the compute node from sticking when no keyboard is attached, which is likely for compute nodes. Edit the text file */etc/default/grub*

sudo nano /etc/default/grub

Add the following line

GRUB_RECORDFAIL_TIMEOUT=2

Save and close the file. To apply this change to the boot menu. In a terminal, run:

sudo update-grub

6.3 Apply Apt-Cacher-NG Proxy Setting

To set each computer to access the proxy, edit the text file */etc/apt/apt.conf.d/02proxy*.

sudo nano /etc/apt/apt.conf.d/02proxy

Add the Proxy URL that you copied from the report page (see last step in 4.10) of the apt-cacher-ng web interface, in the master node apt-cacher-ng step above, and save the file.

Acquire::http { Proxy "http://192.168.1.100:3142"; };

6.4 Update and Reboot

Before installing any packages, it is a good idea to ensure the system is up to date.

sudo apt-get update

sudo apt-get dist-upgrade

sudo shutdown –r now

6.5 Install SSH

First install the SSH server:

sudo apt-get install ssh

If preferable, you should be able to ssh to the compute node from the master node, or elsewhere, and complete install from there (perhaps CSSH could be useful, but only if you know what you're doing). In terminal, on the master node, run:

ssh 192.1681.101 (or use host name if master node host file is set up, step 4.2, which it should be by now).

6.6 Setup Hosts File and Create MPI User

Make the contents of the text file */etc/hosts*, on each compute node, the same as on the master node, see above.

sudo nano /etc/hosts

Create user account for MPI jobs, as above.

sudo adduser mpiuser --uid 999

6.7 Install and setup the Network File System

In order to make it possible to mount a Network File System on the compute nodes, the nfs-common package needs to be installed on all compute nodes:

sudo apt-get install nfs-common

Assuming the master node is set up already, the following command checks that the */home/mpiuser/* directory is being shared correctly (using whatever host name you gave the master node).

showmount -e master

sudo mount master:/home/mpiuser /home/mpiuser

Check to see if the contents on master node are shown in the *mpiuser* home directory. If not, reboot and try again.

Now make the mount happen automatically on boot by editing the text file */etc/fstab*.

sudo nano /etc/fstab

Add the following line

master:/home/mpiuser /home/mpiuser nfs nfsvers=3

Reboot compute node and check the contents of */home/mpiuser* to see if the master node contents are shown.

The first time mpiuser connects from the master node to a compute node, via SSH, there will be a question asked about the authenticity of the new host and whether you want to continue connecting. When an MPI job is started, if this question is asked for each compute node then the job will fail. So the very first time you install the cluster, you should manually SSH to each node, from the master node, as the mpiuser. There are two reasons for this; the first is to answer the authenticity question so that each node is added to the list of known hosts, the second it to confirm that the nfs share of /home/mpiuser is mounting correctly on each node.

6.8 Install Open MPI and Library Dependencies

Install Open MPI on **all** compute nodes.

sudo apt-get install openmpi-bin openssh-client openssh-server libopenmpi-dbg libopenmpi-dev

PST.tools makes use of several open source libraries; GSL, Boost and TNT/JAMA.

sudo apt-get install libgsl0ldbl libgsl0-dev

sudo apt-get install libboost-all-dev

sudo apt-get install libtnt-dev libjama-dev

6.9 Create Local Scratch Directory

In PST.tools the default location for scratch disk on **all** nodes is set in main.cpp.

const std::string scratch_directory="/scratch/mpiuser/PST/";

There needs to be a directory matching this path (or whatever path you place here) on all nodes. The directory should have permissions set so that the mpiuser account has full control of the directory.

To achieve this:

cd /

sudo mkdir -m a=rwx scratch

Note, this directory has full control allowed for all users! If you wish to be more precise in your assignment of access (which is recommended) then do so. Next, switch to the *mpiuser* and make a directory in the scratch directory that belongs to *mpiuser*.

su mpiuser

cd /scratch

mkdir mpiuser

cd mpiuser

mkdir PST

6.10 Create Restart and Shutdown Scripts

Set up bin directory for main user and put reboot and shutdown scripts in there. These scripts unmount the nfs shafe before calling shutdown so it doesn't hang if the network is brought down before the nfs shares unmount properly. The shutdown script is as follows. (The master node script calls this script on every compute node).

```
#!/bin/bash
# This script will shutdown a compute node properly by unmounting the nfs share first so that the shutdown procedure doesn't hang.

sudo umount -f -l /home/mpiuser
echo "nfs mount /home/mpiuser has been unmounted"
sudo shutdown -h now
```

The reboot script is as follows.

```
#!/bin/bash
# This script will reboot a compute node properly by unmounting the nfs share first so that the
shutdown procedure doesn't hang.

sudo umount -f -l /home/mpiuser
echo "nfs mount /home/mpiuser has been unmounted"
sudo shutdown -r now
```

Make sure the scripts have execute permission.

sudo chmod +x node-shutdown

sudo chmod +x node-reboot

6.11 Get NIC MAC Address for Cluster Boot Script

Run ifconfig to get mac address (HWaddr) to put into wake-nodes script (see 4.12).

The first time a compute node is started, after power has been disconnected, a magic packet won't start the compute node. The computer must be booted manually, with the button, after disconnecting the power.

Find out the MAC address of your cluster NIC (replace eth if with your interface name, eth0, eth1, ..):

ifconfig eth | grep "HWaddr" | awk '{print $5}'

Shutdown the machine. You should be able to wake it up using:

wakeonlan your_mac

7 Booting the Cluster

Start the master node first. Once the master node has started, run the *wake-nodes* script to boot the compute nodes. It is important that the master node finishes booting before the compute nodes are started. Otherwise, the */home/mpiuser* nfs share on the master node may not be mounted properly. It is good practice to test that the share is mounted when you boot the cluster. To do this, from the master node, as *mpiuser*, ssh to each compute nodes. If you are asked for a password then the share has not mounted. If it doesn't mount, MPI calculations will not work.

8 Running PST.tools on the Cluster

MPI calculations must be run as the user *mpiuser*. On the master node, switch to the user *mpiuser*;

su mpiuser

Then go to the directory containing the executable.

cd ~/bin/PST

To run *PST.tools* on the cluster you need to call *mpirun*.

mpirun -np 8 --hostfile ~/hosts ./PST.tools

In this example the job will run on 8 cores and will use the nodes listed in the hostfile *~/hosts*. *~/hosts* is simply a text file listing the hostnames for each available node in the cluster (including the master node). It was setup in the master node Open MPI setup step above. The last argument is the path to the *PST.tools* executeable itself. For more detail, see *https://www.open-mpi.org/doc/v1.8/man1/mpirun.1.php*.

9 Optional Installation Tasks

9.1 Internet Connection Sharing

These optional steps allow you to share the master node internet connection with the compute nodes. For more detail, see *https://help.ubuntu.com/community/Internet/ConnectionSharing.* It assumes the master node has an external NIC for internet access. The first step is to work what the master node calls each NIC; they will be named eth0, eth1... etc. In a terminal, run:

ifconfig

The NIC with IP address *192.168.1.100* is the cluster NIC and the external NIC should be listed with an IP address on the network providing internet access. In the following commands we assume *eth0* is the external NIC and *eth1* is the cluster NIC.

Configure iptables for NAT translation so that packets can be correctly routed through the Ubuntu gateway. In a terminal, run:

sudo iptables -A FORWARD -o eth0 -i eth1 -s 192.168.1.0/24 -m conntrack --ctstate NEW -j ACCEPT

sudo iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

sudo iptables -t nat -F POSTROUTING

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

IPtables settings need to be set-up at each boot (they are not saved automatically). Save the iptables with the following terminal command:

sudo iptables-save | sudo tee /etc/iptables.sav

So that the iptables are loaded each boot, edit the text file */etc/rc.local* and add the following line before the *"exit 0"* line:

iptables-restore < /etc/iptables.sav

Configure the gateway for routing between two interfaces by enabling IP forwarding. In a terminal, run:

sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"

Edit the text file */etc/sysctl.conf* and uncomment:

#net.ipv4.ip_forward=1

It should now read:

net.ipv4.ip_forward=1

9.2 Modify .bashrc

This optional step is purely for aesthetics. A user might find it useful to have a coloured command prompt, to either find where the last command was in a long screen of output, or as a visual cue as to which machine or user account they are logged into. Open */home/.bashrc*

uncomment *force_color_prompt=yes*

see customised versions of these saved in new customised files folder. There is a version for *d* and a version for *mpiuser*. The *mpiuser* one will automatically work when *mpiuser* connects, via SSH, to a compute node because the home directory is common. To get the same behaviour for the *d* user on compute node s; as the user *d*, ssh to compute node

cp /home/mpiuser/.bashrc ~/.bashrc

nano .bashrc

Then fix the colour of (note one character difference before \u …) the following part of the file:

if ["$color_prompt" = yes]; then

PS1='${debian_chroot:+($debian_chroot)}\[\033[01;35m\]\u\[\033[01;30m\]@\[\033[01;32m\]\h\[\033[01;30m\]:\[\033[01;34m\]\w\[\033[01;30m\]\$ \[\033[00m\]'

Change it to match the following:

if ["$color_prompt" = yes]; then

PS1='${debian_chroot:+($debian_chroot)}\[\033[01;31m\]\u\[\033[01;30m\]@\[\033[01;32m\]\h\[\033[01;30m\]:\[\033[01;34m\]\w\[\033[01;30m\]\$ \[\033[00m\]'

You can also edit the version in /root, so that when you use real root "*sudo -i*", you can have a bright red cursor.

if ["$color_prompt" = yes]; then

PS1='${debian_chroot:+($debian_chroot)}\[\033[01;31m\]\u@\h:\w\$ \[\033[00m\]'

9.3 Installing Temperature Sensor

This optional step installs a program that lets you monitor the temperature of your CPU etc. This may be particularly useful during setup, to establish whether there is sufficient ventilation for your computers or whether your overclock is too aggressive etc.

sudo apt-get install lm-sensors hddtemp

I chose not to have it start automatically on boot, as recommended. This can be changed later by running *dpkg-reconfigure hddtemp*.

sudo sensors-detect

Answer yes to everything. Restart the computer.

sudo apt-get install psensor