Creating a High-Performance Computer for Dummies

Thomas Boggs, Cameron Kane, Tim Lanzi

Fall 2018

Contents

1 Introduction

If you're reading this, then prepare for an exciting, and sometimes painful, journey into creating your own supercomputer! This manual will be available on GitHub: https://github.com/camerk/wwMpiGanglia. This is also the repository where there is ongoing development on a automated setup script that sets up a High-Performance stack after the initial install of Debian 7. If you have any questions about this manual or the accompanying script, feel free to open an issue on that page. If you have an interest in DIY projects, resources to building your own Ethernet cables will be in the appendices. When you have reached the MPI programming section, we suggest also looking into purchasing the textbook *Using MPI: Portable Parallel Programming with the Message Passing Interface, by Gropp, Lusk, and Skjellum.* This textbook is extremely useful in helping one learn how to not only code for MPI, but to also understand what is happening when the code is running.

1.1 Hardware

We highly recommend getting an SSD (Solid State Drive) for the master node's operating system and fast storage. A small SSD will suffice for the OS, /boot and swap space. If your budget allows, a larger SSD to cover other heavily used partitions will be a nice quality of life update, but not necessary. About 1 TB of HDD space is suggested. However, depending on how much data you plan to use, you can choose less or more space. For the master node, you will need two Ethernet ports. You can purchase a relatively cheap PCIe card for the second port. Any decommissioned desktop towers that are still in working order will work for compute nodes. No hard drives will be needed for these. If you are using decommissioned/old computers, we suggest re-seating and re-plugging in any internal hardware that may have been loosened over time. You will also need an Ethernet cable for each compute node, and 2 for your master node. Instructions to make these yourself as an add-on project are in the appendix. Lastly, a network switch will be necessary to facilitate the local network between the master node and the compute nodes. Beyond that you will need, a monitor, keyboard, mouse, and a flashdrive to install the OS. Once all of these components have been acquired, you're good to go!

1.1.1 Building your own Nodes

If you would like to build your computer to act as the master node or compute nodes, we would recommend doing research on matching your parts before purchase to make sure they are all compatible. We also recommend researching compatibility of older Linux distributions (like the one we will be using) with the newer hardware you may choose to use in your build. For example, we are using an AMD Ryzen 2200G in our build, and its temperature sensors cannot be read by the 3.xx kernel in Debian 7. To enable the compatibility to read these temps, we would have to swap our kernel to a much more recent version. This is not hard, but it is an additional step once you have the cluster up and running. You likely will not have to experience breaking incompatibilities with new hardware, but just smaller things like not being able to read temperature sensors might occur. Checking the internet for these kinds of issues before hardware is purchased is a highly recommended step.

1.2 Software

We recommend vi, vim, or nano as a text editor when editing configuration files during this entire process. You may choose a different text editor, but be sure to validate that the text editor does not insert weird Windows characters or other characters which you cannot see. These will blow up your config files and make for a bad time.

Once you create your chroot environment, do not assume any packages you did not install are there. If the command is not recognized, try to install the package. For example, if the `make` command is not recognized, install `make` by typing `apt-get install make` in the terminal.

1.3 Configuration File Editing

Be sure to backup ALL configuration files prior to ANY alteration. There is no particular way you must do this, but the following method is good practice, as well as convenient. If yaes.conf is the file to alter, PRIOR to changing it, run the following command to copy it into the same directory with a timestamp as the extension:

```
cp /example/file/path/yaes.conf /examlple/file/path/yaes.conf.MMDDYYYY
```

You can add time or extra letters to this extension as well if you are going through many iterations of stable builds on the same day. With this practice, if you blow-up/break a config file, you have a backup to revert to. Use the Linux `diff` command to look for the difference between the backup file and new file to try and locate the differences in the file to debug.

2 Installing Debian

In this section, we will show the step-by-step process of installing Debian 7.11 onto the master node. If you have made the decision of choosing a different OS, these steps will be similar to what one would see in other installs.

2.1 Downloading/Installing from a Flash Drive

To make a bootable flashdrive on Linux, first download the Debian 7.11 .iso file from the website (debian.org/releases/wheezy/debian-installer/). You can download either the Netinst CD image (amd64) or full DVD set (amd64). In our case, we chose the full DVD set. If you choose to do the full DVD version, only download the first part (debian-7.11.0-amd64-DVD-1.iso).

Now we need to write the ISO to a flash drive. On Mac or Linux, you must find where your USB drive is located. For Linux, use a terminal and type the command `lsblk`. This will allow you to locate which port your flash drive is plugged into. Once you have found it (it should be /dev/sdxx where the x's are a letter then a number; it could also only be just a letter), remember this directory. You are going to need it to type the following two commands:

```
sudo umount /dev/sdxx
sudo dd bs=4096 if=path/to/your/ISO/file of=/dev/sdxx && sync
```

After several minutes, you will have a bootable flash drive.

For Mac, to find the mount point of your flash drive, type the following commands into your terminal:

```
ls /Volumes
diskutil list
```

This will provide a list of all mounted storage. Search this list until you find your flash drive (it will be a path similar to /dev/disk2, for example). Once you have found where your flashdrive is mounted, type the following commands into your terminal:

```
diskutil unmountDisk /dev/[your flashdrive mount point]
sudo dd bs=1m if=path/to/your/ISO/file of=/dev/[your flashdrive mount point]
```

This will take a while to complete, but you will have a bootable flashdrive once it is over.

If you are doing this on windows, you will need third-party software such as Rufus, PowerISO, or LinuxLive. We are going to use Rufus for this purpose since it is the most "plug-and-play" of the three. Once you download the .exe file from the website (rufus.ie/en_IE.html), plug in your flashdrive, and select it from the top drop down menu. Hit the SELECT button and it will open a file explorer, where you will select the .iso file you just downloaded. Then click START and Rufus will begin writing the .iso to the flashdrive.

Once you have installed the image of Debian 7.11 to your flash drive, insert the drive into the USB port of the computer you would like to be your master node and turn it on. As the master node is turning on, repeatedly tap the DELETE key (or whatever key the BIOS screen says) to get to the settings screen. Once here, TAB over to the boot order screen and choose the first boot option to be the USB. At this point, you can exit the BIOS screen, and the GUI for the Debian installer should boot up.

2.2 Steps for Installing

1. Select regular install.
2. Select your language of choice for keyboard and spelling.
3. Select the PCIe card as the primary network port. This is usually eth0 but it might say that it is eth1. When in doubt, pick eth0.
4. Choose not to configure the network during installation. This will be configured later.
5. Create the hostname for your computer.
6. Create a root password for the computer. Write this down and keep for later.
7. Create a new user for the computer by inputting the full name, username, and choosing a password for the user.
8. Select a time zone.
9. Select partition disks, then manual partition.

   Now we will be partitioning the memory in the drives of the master node. If you are using new hardware, you do not have to worry about emptying the drives. However, if you are, select the main drive and all other drives one by one and clear them of any partitions. DO NOT select the flash drive when deleting these partitions.

   The next few points will be the names and suggested sizes of the partitions to be made in memory. To make a partition, select free space on the drive that you want to partition. Next, select the partition size, primary, the beginning of the free space. For all except boot, swap, and scratch, select a partition name and select ”use as”, then ext4.

   (a) /boot
      i. 16GB ext4
      ii. In the Use As menu, pick boot as opposed to ext4.

   (b) swap space
      i. 2x Physical Memory
      ii. In the Use As menu, pick swap as opposed to ext4.

   (c) /
      i. 200GB ext4

   (d) /usr
      i. 200GB ext4

   Semi Optional:

   (e) /home
      i. 500GB ext4

   (f) /usr/local or /opt

        i. Size will depend on the amount of custom/source installation you'll have.
    (g) /tmp
        i. 200GB ext4
    (h) /scratch
        i. This partition needs to be specifically named, since there is no pre-existing option for /scratch.
        ii. This is common computing space for the cluster. It can take up the rest of the space.

10. Select to not use Network Mirror
11. Select to not participate in the survey
12. Software Selection: (Use SPACE to select the options, NOT ENTER. Be sure to read the instructions at the bottom of the screen for how to navigate this menu.)
    (a) Debian Desktop Environment
    (b) SSH Server
    (c) Standard System Utilities
13. Select to install Grub
14. With the next window you can select to remove the installation media and press continue.

These are all of the steps needed to complete the installation setup for Debian. It will likely take a while to complete the installation process; be patient.

3 Updating the System

After the system reboots from the install, we will download some initial updates for the system. This process will be done multiple times through out the initializing of the cluster. Make sure that your `/etc/apt/sources.list` file looks like this before proceeding:

```
deb http://deb.debian.org/debian/ wheezy main contrib non-free
deb-src http://deb.debian.org/debian/ wheezy main contrib non-free

deb http://security.debian.org/ wheezy/updates main contrib non-free
deb-src http://security.debian.org/ wheezy/updates main contrib non-free

deb http://deb.debian.org/debian/ wheezy-updates main contrib non-free
deb-src http://deb.debian.org/debian/ wheezy-updates main contrib non-free
```

Now, type these lines into the root user command prompt.

```
apt-get update
apt-get upgrade
```

These commands will read from the `sources.list` file in order to see what is different from what the system has installed at the current time.

4 Setting Up Networking

4.1 What is Networking

A network is a way for different computers or nodes to communicate and share resources with each other over some type of connection. There are two types of networking that we will be concerned with while creating our own cluster: dynamic and static. **Dynamic networking** will be used for the cluster's connection to the internet so that it is able to pull any needed information from, and send information to, the internet through an Ethernet cable. **Static networking** will be used for communication between computers within the same cluster. This is usually done through a network switch, which manages the flow of the data coming from all of the different nodes in the cluster.

4.2 Initializing the Network

In this section, we will be configuring some files in order to get networking to work properly within the cluster. First, if the file `/etc/network/interfaces` does not exist, create this file. The file should look as follows:

```
#The loopback network interface
auto lo
iface lo inet loopback

# Use DHCP for public interface
auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet staticne
address 10.253.1.254
netmask 255.255.255.0
gateway 10.253.1.254
```

After this has been done, restart the master node. Once restarted, using the root user console, type in the command `ip route`. The output should resemble the following:

```
default via 136.160.116.1 dev eth0  proto static
10.253.1.0/24 dev eth1  proto kernel scope link  src 10.253.1.254
136.160.116.0/22 dev eth0  proto kernel scope link src 136.160.119.40
```

5 Important Packages

Now we need to install several packages that will be critical for our cluster. Type the following into your terminal:

```
apt-get install build-essential
apt-get install mysql-server mysql-client
```

This next block is all one command. It will install the rest of the essential packages you will need for the cluster. You can, if you wish, install these packages one at a time, or in chunks, if you do not want to all of this at once:

```
apt-get install ssh ntp qt-sdk pkg-config ncurses-dev nfs-server libselinux1-dev pdsh
↪  tftp gfortran libxml2-dev libboost-dev tk-dev apache2 libapache2-mod-perl2
↪  tftpd-hpadebootstrap tcpdump isc-dhcp-server curl libterm-readline-gnu-perl
```

At this point, after installing all of these important packages, reboot the master node. Note: If the file `/etc/selinux/config` exists, then change the file to set the following flag:

```
SELINUX=permissive
```

6 Warewulf

At this point we have a functional master node. However, it isn't ready to talk to any compute nodes yet; that's what Warewulf is for. Warewulf is a cluster implementation toolkit for high-performance computing systems.

6.1 Installing Warewulf

Whenever we install any packages from source, we always want to do so as the root user. Once you are root, make the directory `~/src` and navigate into it:

```
mkdir ~/src
cd ~/src
```

As of the time of this documentation's writing, Warewulf has halted support for the Debain OS. Knowing this, we have collected the packages for Warewulf versions 3.6 and 3.7 and posted them on GitHub at github.com/camerk/wwMpiGanglia. Once you have pulled all of the files you will need (Warewulf cluster, provsion, vnfs, and common, plus install-wwdebsystem), move all of them to your `~/src` directory (`mv [name of the file] ~/src`). Once this step is complete, simply add executable permissions to the install file and run it using the two commands below. For the second of the two commands, you must specify which version of Warewulf you are installing, i.e. if you chose version 3.6 for your cluster, the command will be `./install-wwdebsystem 3.6`.

```
chmod +x install-wwdebsystem
./install-wwdebsystem [Warewulf version]
```

Now that Warewulf has begun the installation process, there are a couple points we must keep in mind. Warewulf will fail during the installation with an error similar to "Invalid chroot path", this is normal since we have not done this yet. Second, make sure to say NO to wanting toolchains for other architectures. This will cause a failure from one of our packages, selinux.

6.2 Configuring Warewulf

Since Warewulf is now up and running at this point, we must begin the process of configuring it to serve our needs. You should do these steps as the root user. We will be editing a lot of configuration file at this point. So, make sure you remember to MAKE BACKUPS! Edit the file `/etc/exports` to contain the following lines (the spacing is VERY important):

```
/home 10.253.1.0/255.255.255.0(rw,no_root_squash,no_subtree_check)
/srv/chroots 10.253.1.0/255.255.255.0(ro,no_root_squash,no_subtree_check)
/srv/chroots/debian7/vnfs 10.253.1.0/255.255.255.0(ro,no_root_squash,no_subtree_check)
/usr/local 10.253.1.0/255.255.255.0(rw,no_root_squash,no_subtree_check)
/opt 10.253.1.0/255.255.255.0(rw,no_root_squash,no_subtree_check)
```

Now, edit the file `/usr/local/libexec/warewulf/wwmkchroot/include-deb` so that, instead of `/root/...`, the file has these lines:

```
DEB_CONF="/etc/apt-ww.conf"
DEB_CONF_FULL="$CHROOTDIR/etc/apt-ww.conf"
DEB_SRC="/etc/apt/sources.list"
bash}{DEB_SRC_FULL="$CHROOTDIR/etc/apt/sources.list"
```

Then edit the `hybridpath` field in `/usr/local/etc/warewulf/vnfs.conf` to read:

```
hybridpath = /srv/chroots/%{name}/
```

Also, there will be a line in this file that reads `exclude /var/log`. Comment this line out by adding # in front of the line; this will be important later when we install a package called Munge. Now we are going to make a file called `/usr/local/libexec/warewulf/wwmkchroot/debian7.tmpl`. This file will be the template for the operating system that Warewulf will serve to our compute nodes. It will read as follows (again, the spacing is VERY important; especially in PKGLIST, NO SPACES):

```
#DESC: A base 64 bit Debian wheezy chroot
# The general Deb include has all of the necessary functions, but requires
# some basic variables specific to each chroot type to be defined.

. include-deb   # The space here is intentional

ARCH="amd64"
SUITE="wheezy"
DEB_MIRROR="http://http.us.debian.org/debian/"

# Install additional packages with debootstrap.
PKGLIST="openssh-server,openssh-client,isc-dhcp-client,pciutils,strace,nfs-common,ethtool,
iproute,iputils-ping,iputils-arping,net-tools,linux-image-amd64,kmod,ntp,python-dev,libxml2,
gfortran"
```

Now that we have our template, we need to create a directory structure for our "magic land": the chroot environment. Create the following directories:

```
mkdir /srv/chroots
mkdir /srv/chroots/debian7
mkdir /srv/chroots/debian7/vnfs
mkdir /srv/chroots/debian7/srv
mkdir /srv/chroots/debian7/srv/chroots
```

Now it's time to create our magic land. The chroot environment is what gives our compute nodes a purpose. When we make a chroot, we are making the operating system for our compute node. Whenever we want to add something to the compute nodes, we need to step into this magic land and tell them what to do. In order to make this magic land, run the command `wwmkchroot debian7 /srv/chroots/debian7`. This tells Warewulf to make our magic land, called `debian7`, and put it in the directory `/srv/chroots/debian7`. Next, we have some more config files to edit. There are two files we are going to work with now: `/etc/idmapd.conf` and `/srv/chroots/debian7/etc/idmapd.conf`. We want these files to be identical. They should both contain the following:

```
[General]

Verbosity = 0
Pipefs-Directory = /var/lib/nfs/rpc_pipefs

Domain = tier1.cluster

[Mapping]

Nobody-User = nobody
Nobody-Group = nogroup
```

Where it says `Domain = tier1.cluster`, what this means is that we are giving our domain a name of cluster, and our cluster a name of tier1. You can name these however you would like. Just adjust the future steps to how you named yours. Now we need to edit the `/etc/defaults/nfs-common` file to read:

```
# Do you want to start the statd daemon? It is not needed for NFSv4.
NEED_STATD=no

# Options for rpc.statd.
#   Should rpc.statd listen on a specific port? This is especially useful
#   when you have a port-based firewall. To use a fixed port, set this
#   this variable to a statd argument like: "--port 4000 --outgoing-port 4001".
#   For more information, see rpc.statd(8) or http://wiki.debian.org/SecuringNFS
STATDOPTS=

# Do you want to start the idmapd daemon? It is only needed for NFSv4.
NEED_IDMAPD=yes

# Do you want to start the gssd daemon? It is required for Kerberos mounts.
NEED_GSSD=
```

Next, edit the `/usr/local/etc/warewulf/defaults/node.conf` so that it contains the following fields and values:

```
#groups = maingroup
cluster = tier1
domain = cluster
unique hwaddrs = yes
```

The values you entered into this file are dependant on how you named your domain. Then, edit the file `/usr/local/etc/warewulf/defaults/provision.conf` to have it include:

```
vnfs = debian7
files = dynamic_hosts, passwd, group, shadow
```

This will make sure that the dynamic_hosts, passwd, group, and shadow files get synced across the cluster. Our next step is to comment out the infiniband drivers in the `/usr/local/etc/warewulf/bootstrap.conf` file. After this, we need to configure how our magic land's partitions are mounted. We need to edit the file `/srv/chroots/debian7/etc/fstab` so that it reads:

```
tmpfs /dev/shm tmpfs defaults 0 0
devpts /dev/pts devpts gid=5,mode=620 0 0
sysfs /sys sysfs defaults 0 0
proc /proc proc defaults 0 0
10.253.1.254:/home /home nfs defaults 0 0
10.253.1.254:/srv/chroots /srv/chroots nfs defaults 0 0
10.253.1.254:/srv/chroots/debian7/vnfs /vnfs nfs defaults 0 0
10.253.1.254:/usr/local /usr/local nfs defaults 0 0
10.253.1.254:/opt /opt nfs defaults 0 0
```

Now, edit the file `/srv/chroots/debian7/etc/rc.local` to contain the following:

```
/bin/mount -a
exit 0
```

This will mount all of the shared file systems to your compute nodes after the cluster's networking has come up. After this is finishes, we need to restart the NFS server on the master node. We should already be the root user at this time, but make sure you are the root user before you type the following commands into your terminal:

```
/etc/init.d/nfs-kernel-server restart
/etc/init.d/nfs-common restart
```

In order to check that all of the file systems are being exported by master node, type the command `showmount -e 10.253.1.254`. The output should resemble the following:

```
Export list for 10.253.1.254:
/opt                         10.253.1.0/255.255.255.0
/usr/local                   10.253.1.0/255.255.255.0
/srv/chroots/debian7/vnfs 10.253.1.0/255.255.255.0
/srv/chroots                 10.253.1.0/255.255.255.0
/home                        10.253.1.0/255.255.255.0
```

All of your export directories should print on the screen at this time. Now, check the file `/etc/default/tftpd-hpa` to make sure the proper IP address is being used. It should look similar to this:

```
TFTP_ADDRESS="10.253.1.254:69"
```

Once this has been confirmed, restart the TFTP server by typing `/etc/init.d/tftpd-hpa restart`. After all of this configuration has been completed, we can now build our chroot environment. Run the following commands to do so:

```
wwvnfs --chroot /srv/chroots/debian7 --hybridpath=/vnfs
wwsh dhcp update
```

7 Building the Cluster

Now we're ready to add the first compute node! First, plug your compute node into your network switch with an Ethernet cable. Next, we need to know the MAC address for the computer that we are using as the compute node. This can be done by PXE booting the compute node or looking in the node's BIOS. We must also come up with a name and IP address for the node. The following terminal command is an example of how to add a compute node into a cluster:

```
wwsh node new n0001 --hwaddr=b8:ac:6f:32:37:08 --ipaddr=10.253.1.1
```

For the rest of the examples in this documentation, we will be using this convention for our nodes' IP addresses and names. You can choose to name the nodes whatever you want. However, you should use the same subnet mask as your master node (meaning all of the compute nodes' IP addresses should begin with 10.253.1, or whatever you have chosen). If your nodes are having trouble booting over TFTPD, restart your switch. It's possible it has bad ARP information. Once your node has successfully booted, check that all NFS partitions are mounted and preserving the permissions for files and directories. Run the following commands as root:

```
ssh n0001
df -k

# Output should resemble:
Filesystem                            1K-blocks    Used Available Use% Mounted on
rootfs                                  6163324 1503188   4660136  25% /
none                                    6163324 1503188   4660136  25% /
none                                    2465320       0   2465320   0% /run/shm
tmpfs                                   1232668     100   1232568   1% /run
tmpfs                                      5120       0      5120   0% /run/lock
tmpfs                                     10240       0     10240   0% /dev
tmpfs                                   2465320       0   2465320   0% /dev/shm
10.253.1.254:/home                    480618496 4602880 451601408   2% /home
10.253.1.254:/srv/chroots             209606656 6934528 192024576   4% /srv/chroots
10.253.1.254:/srv/chroots/debian7/vnfs 209606656 6934528 192024576   4% /vnfs
```

```
10.253.1.254:/usr/local                192245760 4526080 177953792   3% /usr/local
10.253.1.254:/opt                      288370688 4959232 268763136   2% /opt
# End output

ls -ltra /home

# Output should resemble:
drwx------  2 root        root        16384 Oct 16 01:57 lost+found
drwxr-xr-x  5 root        root         4096 Nov 28 18:39 .
drwxr-xr-x 28 [YourUser]  [YourUser]   4096 Dec 10 17:58 [YourUser]
drwxr-xr-x 24 root        root          520 Dec 10 17:59 ..
# End output
```

Now, create a regular (non-root) user account. We are going to set up SSH for a password-less log in. Type the following commands onto your terminal:

```
su [username of the non-root user you created]
cd
ssh-keygen -t rsa       # Enter no password for this step
cd ~/.ssh
cp id_rsa.pub authorized_keys
cd
```

This will allow you to assume the role of your compute node while using the master node. In order to do this, type `ssh n0001`, and answer Yes when prompted. You should not have to answer this prompt for this compute node ever again. To test this, exit your compute node by typing `exit`, and then `ssh n0001` again. Now, to make sure the VNFS systems that we make stay updated, make sure that the contents of your debain7 chroot's `sources.list` match those of the master node's `sources.list`. If they don't match, add the following 2 lines to the beginning of the compute node's `/srv/chroots/debian7/etc/apt/sources.list`:

```
deb http://security.debian.org/ wheezy/updates main
deb-src http://security.debian.org/ wheezy/updates main
```

Now we need to update our chroot environment. To do this, we change chroot and run the update and upgrade commands as follows:

```
chroot /srv/chroots/debian7
mount -t proc proc proc/
apt-get update
apt-get upgrade
exit        # When we're done, we exit the chroot environment
```

Whenever we update or add something to our magic land, we need to rebuild our VNFS. We do this by running the following command and restarting the compute nodes:

```
wwvnfs --chroot /srv/chroots/debian7  --hybridpath=/vnfs
```

Now we're going to sync all of the nodes' clocks by editing the file `/srv/chroots/debian7/etc/ntp.conf` to look like this:

```
# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5),ntp_mon(5).

#driftfile /var/lib/ntp/drift

# Permit all access over the loopback interface.  This could
# be tightened as well, but to do so would affect some of
# the administrative functions.
#restrict 127.0.0.1
#restrict default ignore
server 10.253.1.254
#restrict 10.253.1.254 nomodify
```

Now we need to rebuild the VNFS to incorporate the changes, then reboot the compute node. Once the node comes back up, SSH into it and run the `ntpq` command. Once the `ntpq` command line comes up (it should resemble `ntpq>`), type `peers` and hit ENTER.

The output should look like this:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*[Master node]    108.61.73.243    3 u   59   64  377    0.288   -3.667   3.297
```

Now that we have one node functional, we can add more! All that needs to be done is to run the new node command that we used for the first one (`wwsh node new ...`). Remember, you need the MAC address for the computer to do this. Once you have added all the nodes that you plan on adding, sync and update everything! The following commands do this:

```
wwsh file sync
wwsh dhcp update
wwsh pxe update
```

Now, from the master node, SSH into all of the nodes (including the master node) to add the address to the SSH config files. At this point, we are going to test out running a command in parallel across the cluster. To do this, we're going to use a parallel shell (pdsh). Simply type the following commands into your cluster as root and a regular user:

```
pdsh -R ssh -w masterName,computeName1,computeName2 hostname
pdsh -R ssh -w masterName,computeName1,computeName2 uname -a
```

These are just simple test commands used to make sure the cluster can run commands in parallel. If something isn't working properly, check if you can SSH into all nodes (including the master) again as root and a regular user. As an optional step, you could install `rsyslogd` on your cluster as a way to log your cluster's processes. If you wish to do this, enter the root environment and install it, rebuild the VNFS, update DHCP, and restart the compute nodes. These are the commands to do so:

```
chroot /srv/chroots/debian7
mount -t proc proc proc/
apt-get install rsyslog
exit

wwvnfs --chroot /srv/chroots/debian7  --hybridpath=/vnfs
wwsh dhcp update

pdsh -R ssh -w computeName1,computeName2 reboot
```

At this point, you should have a functional high-performance cluster! You should also have the knowledge necessary to add packages to your cluster, add nodes, and run commands in parallel across the cluster. Now we're going to get into running parallel, multi-process programs using MPI.

8 MPI: Message Passing Interface

Since we have a high-performance cluster, we'd ideally like to use it to solve some really complex problems. Normally, we'd write a serial (single process) program to solve these problems; however, this can take a really long time to run to completion. That's where MPI comes in.

8.1 What is MPI?

MPI is a way to program for parallel computers like our cluster. It allows for point-to-point communication between processes, as well as collective communication. This way, if we're working with a lot of data, we don't have to work on it in a serial way. We can split up the work by having multiple processes on multiple hosts work on portions of the problem.

Lets assume we're working with an array that contains 1 billion elements. Even if we only planned on printing out the data stored in this array, it would take a long time to complete. With MPI, we can send this array to all processes in our "MPI world" and each one of them can print a section of the array. Depending on how many processes this program would be run with, the time to complete this job could be increased immensely! This is why we are going to install it on our cluster.

8.2 Installing MPI

First thing's first, we need to choose which MPI libray we want. The two most popular ones are OpenMPI and MPICH. For this documentation, we are going to choose MPICH. Since we are going to be building this from source, make sure you are the root user. Go to www.mpich.org/downloads and pick a release version to download. Just download an "mpich" package as opposed to "hydra". Once the package has finished downloading, move it to the `~/src` directory:

```
cd ~/src
mv /home/[your non-root user]/Downloads/mpich-[version number].tar.gz ~/src
```

To un-tar (decompress) this file, type `tar zxvf mpich-[version number].tar.gz`. Now `cd`, into your mpich folder. To install this package, type the following commands:

```
./configure --enable-fc --enable-f77 --enable-romio --enable-mpe --with-pm=hydra
make
make install
```

Each of these commands will take a while to complete; so be patient. After the installation process has completed, reboot your compute nodes.

8.3 Your First MPI Program

To test that your installation of MPI is successful, create a fill called `hellompi.c` with the following contents:

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char ** argv) {

    int size,rank;
    int length;
    char name[80];
    MPI_Status status;
    int i;

    MPI_Init(&argc, &argv);
    // note that argc and argv are passed by address

    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    MPI_Get_processor_name(name,&length);

    if (rank==0) {
        // server commands
        printf("Hello MPI from the server process!\n");
        for (i=1;i<size;i++) {
            MPI_Recv(name,80,MPI_CHAR,i,999,MPI_COMM_WORLD,&status);
            printf("Hello MPI!\n");
            printf(" mesg from %d of %d on %s\n",i,size,name);
        }
    }
    else {
        // client commands
        MPI_Send(name,80,MPI_CHAR,0,999,MPI_COMM_WORLD);
    }

    MPI_Finalize();
}
```

MPI has special commands to compile and run programs. The compiler is called `mpicc`, it works the same as `gcc` or `g++` if you have ever used these before. If you haven't, it's not a big deal, so don't worry. To compile and run your program, run the following commands:

```
mpicc -o hello hello.c
mpirun -hosts master,compute1,compute2 -n 12 ./hello
```

In the `mpirun` command, the `-hosts` flag specifies which of your nodes on which you'd like to run your program. The `-n` flag specifies the number of overall processes you would like to be spawned. These processes will be split as evenly as possible across the hosts you have chosen. Now you know how to run an MPI program!

9 Ganglia

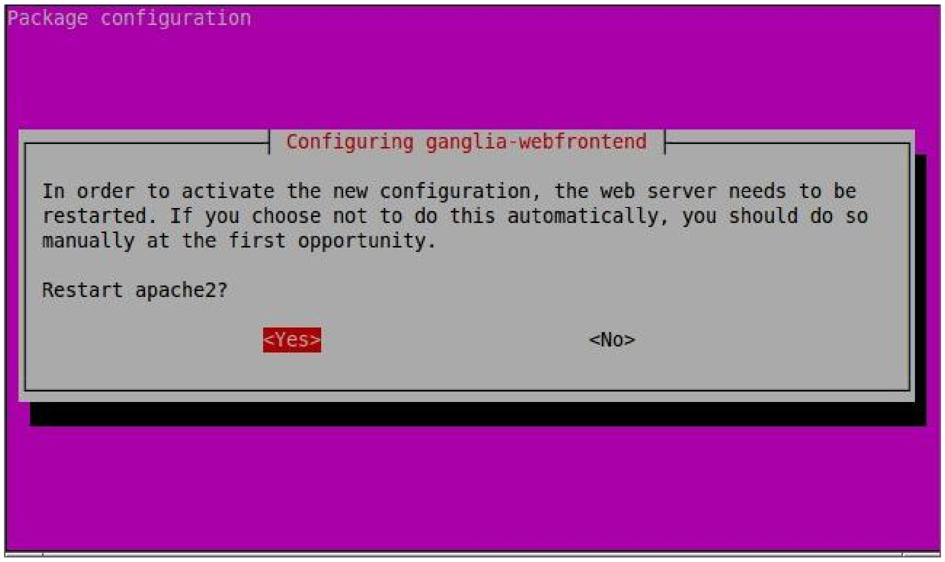
Now that we can run large-scale programs across our cluster, we want to be able to check on our clusters utilization statistics while these programs are running. Ganglia is a scalable monitoring system for high-performance computing structures. It has a very low overhead due to it using carefully-engineered data structures. The whole system consists of two daemons: gmetad (Ganglia Meta Daemon) and gmond (Ganglia Monitor Daemon). The Ganglia Meta Daemon runs only on the master node and collects information from the compute nodes. The Ganglia Monitor Daemon runs on all nodes which you want to monitor. It monitors the host state, announces any relevant changes, and listens to the state of every node in the cluster.

9.1 Installing Ganglia

Installing Ganglia is quite straightforward and requires very minimal configuration file edits. To begin, install all of the necessary packages (as root):

```
apt-get install ganglia-monitor rrdtool gmetad ganglia-webfrontend
```

During this installation, you will be asked to restart your Apache server to activate the new configuration. Click YES to continue.



```
Package configuration

           ┤ Configuring ganglia-webfrontend ├

 In order to activate the new configuration, the web server needs to be
 restarted. If you choose not to do this automatically, you should do so
 manually at the first opportunity.

 Restart apache2?

                 <Yes>                        <No>
```

9.2 Configuring the Master Node

To configure the master node, copy the Ganglia config file `/etc/ganglia-webfrontend/apache.conf` to the location `/etc/apache2/sites-enabled/ganglia.conf`:

```
cp /etc/ganglia-webfrontend/apache.conf /etc/apache2/sites-enabled/ganglia.conf
```

Now we need to edit the file `/etc/ganglia/gmetad.conf`. In this file, find the line that looks similar to this: `data_source "my cluster" 50 10.254.1.253:8649`. This line, as we have shown it to you, means that the logs will be collected every 50 seconds. The `my cluster` is your cluster's name. You will need to change the IP address (not the port number) to your master node's IP. Once the changes have been made, edit the file `/etc/ganglia/gmond.conf`. Do so by finding the following sections and modifying them as shown:

```
[...]
cluster {
  name = "my cluster"   ## Name assigned to the client groups
  owner = "unspecified"
  latlong = "unspecified"
  url = "unspecified"
}

[...]

udp_send_channel   {
#mcast_join = 239.2.11.71 ## Comment
  host = 10.254.1.253   ## Master node IP address
  port = 8649
  ttl = 1
}

[...]

udp_recv_channel {
  port = 8649
}

/* You can specify as many tcp_accept_channels as you like to share
   an xml description of the state of the cluster */
tcp_accept_channel {
  port = 8649
}

[...]
```

These changes allow your master node to collect the data from all nodes on the UDP port 8649. Now start the ganglia-monitor, gmetad, and apache2 services.

```
/etc/init.d/ganglia-monitor start
/etc/init.d/gmetad start
/etc/init.d/apache2 restart
```

9.3 Configuring the Compute Nodes

Now that we have Ganglia running on the master node, it's time to get it running on our compute nodes. Enter your magic land, the chroot environment, and install the `ganglia-monitor` package as shown below:

```
chroot /srv/chroots/debian7
mount -t proc proc proc/
apt-get install ganglia-monitor
```

Now, we need to edit the file `/etc/ganglia/gmond.conf`. Find the following sections and make the changes shown:

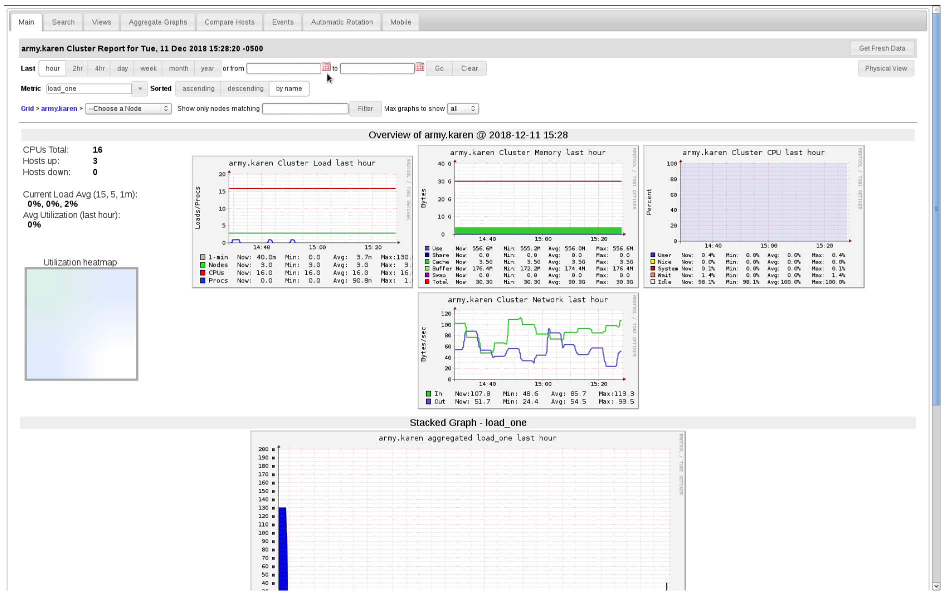
```
[...]

cluster {
  name = "my cluster"     ## Cluster name
  owner = "unspecified"
  latlong = "unspecified"
  url = "unspecified"
```

```
[...]

udp_send_channel {
  #mcast_join = 239.2.11.71   ## Comment
  host = 192.168.1.104   ## IP address of master node
  port = 8649
  ttl = 1
}
## Comment the whole section
/* You can specify as many udp_recv_channels as you like as well.
udp_recv_channel {
  mcast_join = 239.2.11.71
  port = 8649
  bind = 239.2.11.71
}
*/

tcp_accept_channel {
  port = 8649
}

[...]
```

Now we need to restart the ganglia-monitor service by running `/etc/init.d/ganglia-monitor restart`. Once this is finished, exit the chroot environment, rebuild the VNFS, sync files, run a DHCP update, and reboot your compute nodes.

```
exit
wwvnfs --chroot /srv/chroots/debian7  --hybridpath=/vnfs
wwsh  file sync
wwsh dhcp update
pdsh -R ssh -w computeName1,computeName2 reboot
```

At this point, you should have Ganglia up and running! To test it out, open a web browser and go to http://localhost/ganglia. You should see something similar to this:



army.karen Cluster Report for Tue, 11 Dec 2018 15:28:20 -0500

Metric: load_one
Max graphs to show: all

Overview of army.karen @ 2018-12-11 15:28

CPUs Total: 16
Hosts up: 3
Hosts down: 0

Current Load Avg (15, 5, 1m):
0%, 0%, 2%
Avg Utilization (last hour):
0%

```
army.karen Cluster Load last hour
1-min   Now: 40.0m   Min:  0.0   Avg:  3.7m   Max:130.
Nodes   Now:  3.0    Min:  3.0   Avg:  3.0    Max:  3.
CPUs    Now: 16.0    Min: 16.0   Avg: 16.0    Max: 16.
Procs   Now:  0.0    Min:  0.0   Avg: 90.8m   Max:  1.

army.karen Cluster Memory last hour
Use     Now: 556.6M  Min: 555.2M  Avg: 556.0M  Max: 556.6M
Share   Now:   0.0   Min:   0.0   Avg:   0.0   Max:   0.0
Cache   Now:   3.5G  Min:   3.5G  Avg:   3.5G  Max:   3.5G
Buffer  Now: 176.4M  Min: 172.2M  Avg: 174.4M  Max: 176.4M
Swap    Now:   0.0   Min:   0.0   Avg:   0.0   Max:   0.0
Total   Now:  30.3G  Min:  30.3G  Avg:  30.3G  Max:  30.3G

army.karen Cluster CPU last hour
User    Now:  0.4%   Min:  0.0%   Avg:  0.0%   Max:  0.4%
Nice    Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
System  Now:  0.1%   Min:  0.0%   Avg:  0.0%   Max:  0.1%
Wait    Now:  1.4%   Min:  0.0%   Avg:  0.0%   Max:  1.4%
Idle    Now: 98.1%   Min: 98.1%   Avg:100.0%   Max:100.0%

army.karen Cluster Network last hour
In      Now:107.8    Min: 48.6    Avg: 85.7    Max:113.3
Out     Now: 51.7    Min: 24.4    Avg: 54.5    Max: 93.5
```

Stacked Graph - load_one

army.karen aggregated load_one last hour

10 Munge and Slurm

Now that we can view our stats and run large tasks, It's time to install a workload manager. Slurm is that workload manager. It can allocate access to resources to users for some period of time, provide a framework for executing and monitoring work on a set of allocated nodes, and arbitrates contention for resources by maintaining a queue of pending jobs. However, before we can install Slurm, we need to install Munge, an authentication service.

10.1 Installing Munge

We already began this process way back when we were installing Warewulf (remember that?). That will save you a lot of pain later. There are two ways to proceed: from packages or from source. For this documentation, we will be proceeding with package installs. If you would like to install from source, you can download Munge at dun.github.io/munge/. Instalation instructions can be found at github.com/dun/munge/wiki/Installation-Guide. For a source install, proceed as we did for MPICH. For a package install, we'll start by installing some necessary packages on the master node (as root): `munge`, `libmunge-dev`, and `libmunge2`.

```
apt-get install munge libmunge2 libmunge-dev
```

Now we need to generate the Munge key. This will be our authentication for Slurm. There are two ways to do this:

```
# This will take longer to generate, but is more secure
dd if=/dev/random bs=1 count=1024 > /etc/munge/munge.key

# This will generate quickly, but is not as secure
dd if=/dev/urandom bs=1 count=1024 > /etc/munge/munge.key
```

It's time to make a Munge user. Add the following line to the bottom of the `/etc/passwd` file:

```
munge:x:501:501::var/run/munge;/sbin/nologin
```

Now we need to change permissions and ownership of this Munge key. Run the following commands:

```
chown munge:munge /etc/munge/munge.key
chmod 400 /etc/munge/munge.key
```

After this, we start the Munge Daemon by running the command `/etc/init.d/munge start`. If you get any errors that resemble "INVALID PERMISSIONS", don't panic. All that needs to be done is running the following command:

```
chown munge:munge /directory/specified/in/error
```

Even if you don't have an error like this, run the previous command (as root) for the `/var/log/munge`, `/var/lib/munge`, and `/var/log/munge/munged.log` files and directories. Do this to the equivalent files and directories in your chroot path as well (add `/srv/chroots/debian7` before each of these paths. After this is taken care of, Munge should be functional on your master node. Test this out by running the following commands:

```
munge -n                #Should print out gibberish
munge -n | unmunge      #Should print out the previous gibberish decoded
```

Now it's time to take care of the compute nodes. Start by entering your chroot environment, mounting the `proc/` directory, and installing all of the same packages for Munge as you did on the master node. Now, exit chroots and copy the `munge.key` file to your chroots directory and change the permissions:

```
cp /etc/munge/munge.key /srv/chroots/debian7/etc/munge/munge.key
chown munge:munge /srv/chroots/debian7/etc/munge/munge.key
```

Now, rebuild VNFS and start the Munge daemon on your compute nodes.

```
wwvnfs --chroot /srv/chroots/debain7 --hybridpath=/vnfs
wwsh file sync
wwsh dhcp update
pdsh -R ssh -w master,compute1,compute2 reboot  #Reboot everything!
```

After this, start the Munge daemon in your chroots environment:

```
chroot /srv/chroots/debian7
mount -t proc proc proc/
/etc/init.d/munge start
exit
```

Now to run some tests to make sure everything is working:

```
munge -n | ssh compute1 unmunge      #Tests if Munge info can be remotely decoded
```

If this doesn't work, try starting your compute nodes manually:

```
pdsh -R ssh -w compute1,compute2 /etc/init.d/munge start
```

Once Munge is up and running, we can install Slurm!

10.2 Installing Slurm

You can download the Slurm source from www.schedmd.com/downloads.php. We chose to download Slurm verison 18.08.4. Download the .tar file and, as root, move it to your /src directory and un-tar the file.

```
cd ~/src
mv /home/[your non-root user]/Downloads/slurm-[version number].tar.bz2 ~/src
tar xvjf slurm-[version number].tar.bz2
```

Now we can begin the installation process. Change into the newly decompressed Slurm folder, run the configure file with the `--enable-multiple-slurmd` flag, run `make` and `make install`.

```
cd slurm-[version number]
./configure --enable-multiple-slurmd
make
make install
```

If during the `make` process, you get error exits (`Exit code 1`), check where the errors are coming from. You will more than likely need to comment out some dependencies in a Makefile. For example, during our install, our `make` errored out with a dependency issue with HDF5 in the directory `slurm-[version]/src/plugins/acct_gather`
Because of this, we commented out all mentions of HDF5 (non case-sensitive) in the Makefile from the `acct_gather_profile` directory. If you run into any issues like this, follow similar steps. Afterwards, your `make` should run to completion. At this point, Slurm is installed on the master node. Now we need to make a `slurm.conf` file. To do this, you can use the built-in slurm configurator, a web-based config file generator located in `slurm-[version]/doc/html/configurator.html`. This will generate the contents of a `slurm.conf` file within your browser to copy/paste into your own file. You can also use our `slurm.conf` file as an example of what one should look like:

```
#
# slurm.conf file generated by configurator.html.
#
# See the slurm.conf man page for more information.
```

```
#
ClusterName=army.karen
ControlMachine=hasselhoff
#ControlAddr=
#BackupController=
#BackupAddr=
#
SlurmUser=slurm
#SlurmdUser=root
SlurmctldPort=6817
SlurmdPort=6818
AuthType=auth/munge
#JobCredentialPrivateKey=
#JobCredentialPublicCertificate=
StateSaveLocation=/tmp
SlurmdSpoolDir=/var/spool/slurm/slurmd
SwitchType=switch/none
MpiDefault=none
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmdPidFile=/var/run/slurmd.pid
ProctrackType=proctrack/pgid
#PluginDir=
CacheGroups=0
#FirstJobId=
ReturnToService=1
#MaxJobCount=
#PlugStackConfig=
#PropagatePrioProcess=
#PropagateResourceLimits=
#PropagateResourceLimitsExcept=
#Prolog=
#Epilog=
#SrunProlog=
#SrunEpilog=
#TaskProlog=
#TaskEpilog=
#TaskPlugin=
#TrackWCKey=no
#TreeWidth=50
#TmpFS=
#UsePAM=
#
# TIMERS
SlurmctldTimeout=300
SlurmdTimeout=300
InactiveLimit=0
MinJobAge=300
KillWait=30
Waittime=0
#
# SCHEDULING
SchedulerType=sched/backfill
#SchedulerAuth=
#SchedulerPort=
#SchedulerRootFilter=
SelectType=select/linear
FastSchedule=1
#PriorityType=priority/multifactor
#PriorityDecayHalfLife=14-0
#PriorityUsageResetPeriod=14-0
#PriorityWeightFairshare=100000
#PriorityWeightAge=1000
#PriorityWeightPartition=10000
#PriorityWeightJobSize=1000
#PriorityMaxAge=1-0
#
# LOGGING
SlurmctldDebug=3
#SlurmctldLogFile=
SlurmdDebug=3
#SlurmdLogFile=
#JobCompType=jobcomp/none
#JobCompLoc=/tmp/slurm_job_completion.txt
#
# ACCOUNTING
JobAcctGatherType=jobacct_gather/linux
JobAcctGatherFrequency=30
#
#ccountingStorageEnforce=limits,qos
#AccountingStorageType=#accounting_storage/slurmdbd
AccountingStorageHost=slurm
#AccountingStorageLoc=/tmp/slurm_job_accounting.txt
#AccountingStoragePass=
#AccountingStorageUser=
#

# COMPUTE NODES
# control node
NodeName=spongebob NodeAddr=10.253.1.1 CPUs=4 Sockets=1 CoresPerSocket=4 RealMemory=12037 State=UNKNOWN
NodeName=patrick NodeAddr=10.253.1.2 CPUs=4 Sockets=1 CoresPerSocket=4 RealMemory=12037 State=UNKNOWN
NodeName=hasselhoff NodeAddr=10.253.1.254 CPUs=4 Sockets=1 CoresPerSocket=4 RealMemory=6945 State=UNKNOW
# PARTITIONS
# partition name is arbitrary
PartitionName=jellyfishers Nodes=hasselhoff,spongebob,patrick Default=YES MaxTime=8-00:00:00 State=UP
```

However you chose to make your file, it must find its home it /usr/local/etc/slurm.conf. Copy it there using the following command:

```
cp wherever/you/put/it/slurm.conf /usr/local/etc/slurm.conf
```

We need to make a Slurm user now. Run the following commands to do so:

```
echo "slurm:x:2000:2000:slurm admin:/home/slurm:/bin/bash" >> /etc/passwd
echo "slurm:x:2000:slurm" >> /etc/group
```

Now we can start the Slurm Control Daemon (slurmctld) to begin debugging. To start it in the foreground for debugging purposes, run the command slurmctld -Dv. If you see a big block on the screen that says something like ”AUTHORIZATION IS NOT SECURE”, this is normal; don't panic. If you encounter any errors, they will more than likely be similar to the ones encountered with Munge. If it says that the daemon has started successfully in the messages you are seeing, you have successfully installed Slurm on your master node! You can kill the foreground daemon by pressing CTRL+C. Now to start it in the background, simply type `slurmctld`.

Lets install it on our compute nodes now! Start by copying your slurm folder into your chroots directory. We chose to put ours in the `home` directory of chroots.

```
cp -r ~/src/slurm-[version] /srv/chroots/debian7/home/
```

Change into the chroots environment, mount your proc/ directory, change into your `/home` directory, and follow the same steps you went through for installing Slurm on your master node (from configure through make install). At this point, you may need to install the `make` command. Run `apt-get install make` if you don't have it. Now we need to make some directories.

```
mkdir /var/spool/slurm
mkdir /var/spool/slurm/slurmd
```

After this step has been completed, exit chroots, rebuild your VNFS, run a `wwsh file sync` and `dhcp update`, and reboot your compute nodes. Once they come back to life, run `slurmd` on your compute nodes in the foreground (same flags as `slurmctld`) to debug. If you encounter any errors, they will be similar to the ones encountered with the `slurmctld`. Once `slurmd` any errors have been resolved, you can kill the foreground process with CTRL+C and run `slurmd` in the background (only type `slurmd`).

Once you have all of the daemons running, back on the master node, type the command `sinfo -Nl`. This will print out a status table for all of your nodes. If your compute nodes are reading that they are "idle", this is great! This means that Slurm is working exactly how it should. It they are reading "down", a full reboot of everything should fix this. If your system requires a reboot, make sure that you start the daemons again on startup.

```
# On the master as root

slurmctld
pdsh -R ssh -w compute1,compute2 slurmd
```

10.3 How to Use Slurm

Now you are ready to start running jobs with Slurm! For now, we are going to run through how to create a basic Slurm job. First, create a file called `hostname.sh` with the contents:

```
#!/bin/sh

hostname

exit0
```

To run this with Slurm, type the following command:

```
sbatch -N1 -n1 -t00:05:00 hostname.sh
```

This command requests one node (-N1) for one task (-n1) with a max run time of 5 minutes (-t00:05:00) to run the job `hostname.sh`. Once this has been done, you will receive a job number. If you want to check the status of your job, type the command `scontrol show job [job number]`. This shows a list of statistics for your job, including the running status and the node(s) it is being run on. If you are running multiple of these batch jobs on your cluster, you can see a list of these by typing `squeue`, or if you want to specifically see your jobs `squeue -u [username]`. If you would prefer not to use all of those flags in your terminal when you run a Slurm task, you can include them in your `hostname.sh` file as follows:

```
#!/bin/sh
#SBATCH -N1 -n1 -t00:05:00
#SBATCH -J hostname         # This gives the job a name

hostname

exit0
```

With these additions, you can run the Slurm job by simply typing `sbatch hostname.sh`. Finally, if you would like to view the output of your task, it will be stored in a file called `slurm-[job number].out` by default. Simply open it in your text editor of choice or `cat` it in the terminal to see what the output was. These are all of the basics for using Slurm! If you would like to learn more about how to use it, Slurm's website has a great set of tutorials on how to do just this. They can be found at slurm.schedmd.com/tutorials.html.

11 Final Notes

Now that you have a functional cluster, you may begin installing other packages and software for your specific use case (i.e. graphics card drivers, kernel updates, etc). We'd recommend doing these installs in directories offered up to the nodes over VNFS such as `/opt` to make finding them easier. Always remember the following commands to refresh the provisioned operating system so that your changes are reflected across the cluster.

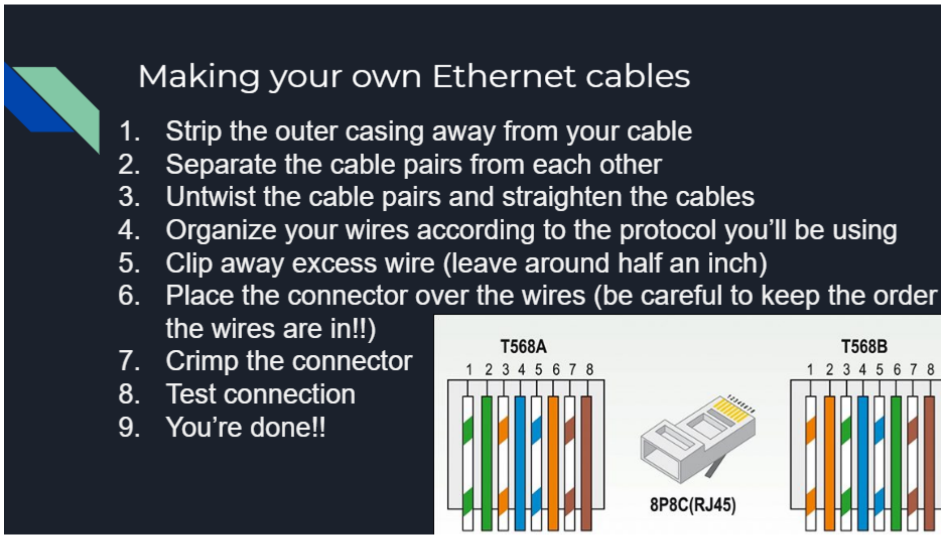
```
wwsh file sync
wwsh dhcp update
pdsh -R ssh -w nodes reboot
```

Congratulations on completing this incredible achievement!

Appendices

A Making an Ethernet Cable

In this section we will show how to construct an Ethernet cable from just the cable and the ends. This process is fairly simple but must be done precisely or the connection will not work properly. Our first suggestion is to super glue the ends of your thumb and pointer finger separately on your dominant hand. DO NOT GLUE THEM TOGETHER. This is not necessarily needed but will make the whole procedure hurt significantly less if making multiple cables. Follow the directions below:



Making your own Ethernet cables

1. Strip the outer casing away from your cable
2. Separate the cable pairs from each other
3. Untwist the cable pairs and straighten the cables
4. Organize your wires according to the protocol you'll be using
5. Clip away excess wire (leave around half an inch)
6. Place the connector over the wires (be careful to keep the order the wires are in!!)
7. Crimp the connector
8. Test connection
9. You're done!!

In case the picture has come out black and white, the order of the T568A is green stripe, green, orange stripe, blue, blue stripe, orange, brown stripe, brown. The T56B is orange stripe, orange, green stripe, blue, blue stripe, green, brown stripe, brown.

Once you have completed a small test wire, we suggest creating a 25ft and four 10ft wires for your cluster. The 25ft is for connecting to an internet port and the others are for connecting the nodes to the switch so that they can communicate with each other.

B Installing Anaconda Python

We built anaconda normally from source and once the install script prompted for an install directory we chose `/opt` so that it was in a directory offered up over VNFS for the compute nodes to use. Once it is installed, run the following command on each node and profile on master node:

```
export PATH="/opt/anaconda3/bin:\$PATH"
```

Check by running the `conda` in a few compute nodes and users to make sure it's a recognized command