LA-UR- 09-05117

Approved for public release;
distribution is unlimited.

Title:	Creating and Administering a Basic Diskless/Diskfull Cluster
Author(s):	Christian Romano, INST-OFF Dane Gardner, INST-OFF Jonathan Welters, INST-OFF
Intended for:	Academic Distribution

Form 836 (7/06)

CREATING AND ADMINISTERING A BASIC DISKLESS/DISKFULL CLUSTER

Christian Romano, Dane Gardner, Jon Welters

07/15/2009

LAUR#

Information Science and Technology Institute

ISTI/IAS Summer Institute
Computer System, Cluster, and Networking

http://institutes.lanl.gov

A collection of highly successful, strategic partnerships with leading research universities focused on IS&T topic areas of critical importance to the future of LANL

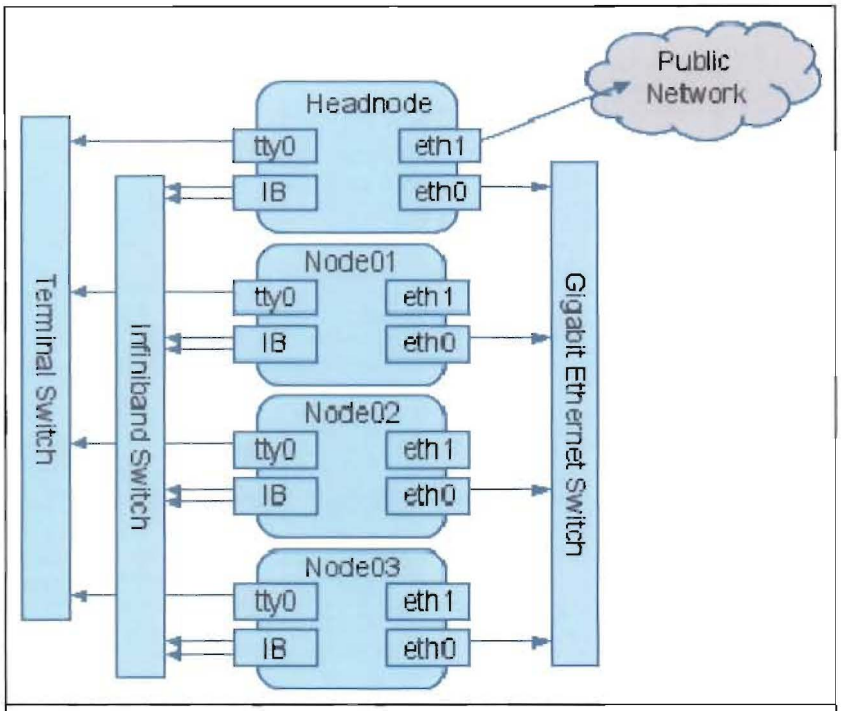
Table of Contents

Introduction

All of the subjects detailed here are vast and have manuals written solely for their installation and operation. This document solely follows the steps that our group took through the short institute boot camp and does not profess to be a manual of any kind. Most of the information was taken from the mind and slides of our instructor Andree Jacobson.

Code snippets and file snippets may exceed the length of the page and should therefore be interpreted to determine if it should be entered differently from copy and paste.

Physical Layout of the Cluster



General wiring layout of an individual cluster

Figure 1. Cluster Layout and Connections

The nodes of this cluster are connected in three separate methods.

- Serial
- Gigabit Ethernet
- SDR Infiniband

The serial connection allows the user to view the bios remotely through the terminal server. This makes last minute changes and troubleshooting easier because the administrator is not required to physically be present to configure and restart the machine.

Most clusters have both Gigabit Ethernet as well as the Infiniband connection. It is difficult to boot from an Infiniband connection because the Infiniband card firmware does not have an integrated booting utility such as gPXE and DHCP does not support long hardware addresses by default. It is much easier to simply provide the various management functions through Ethernet. The OS image and associated tools provided by pxeboot can be requested by a compute node through a variety of tools such as DHCP, NFS, web server, and tftp which easily integrate with Ethernet.

Infiniband provides a high-speed low-latency connection between the compute nodes which is generally required for MPI jobs.

Cabling the Cluster

Needless to say, cabling standards are important. All labels on the cluster should be consistent. Furthermore, the designations provided on the inside/outside of the system should be consistent with the designations provided on the cabling. When setting up a cluster, one should check with their organization or facility to see if there are any pre-existing standards.

- Each cable should have its source and destination clearly labeled on each end.
- All cables should be labeled consistently
- Labels should be updated if circumstances change
- Cabling should be consistent with facility standards
- Cables should be neat, fastened securely, not be a fire hazard, and not block airflow
- Cables should be well organized in tightly secured bundles

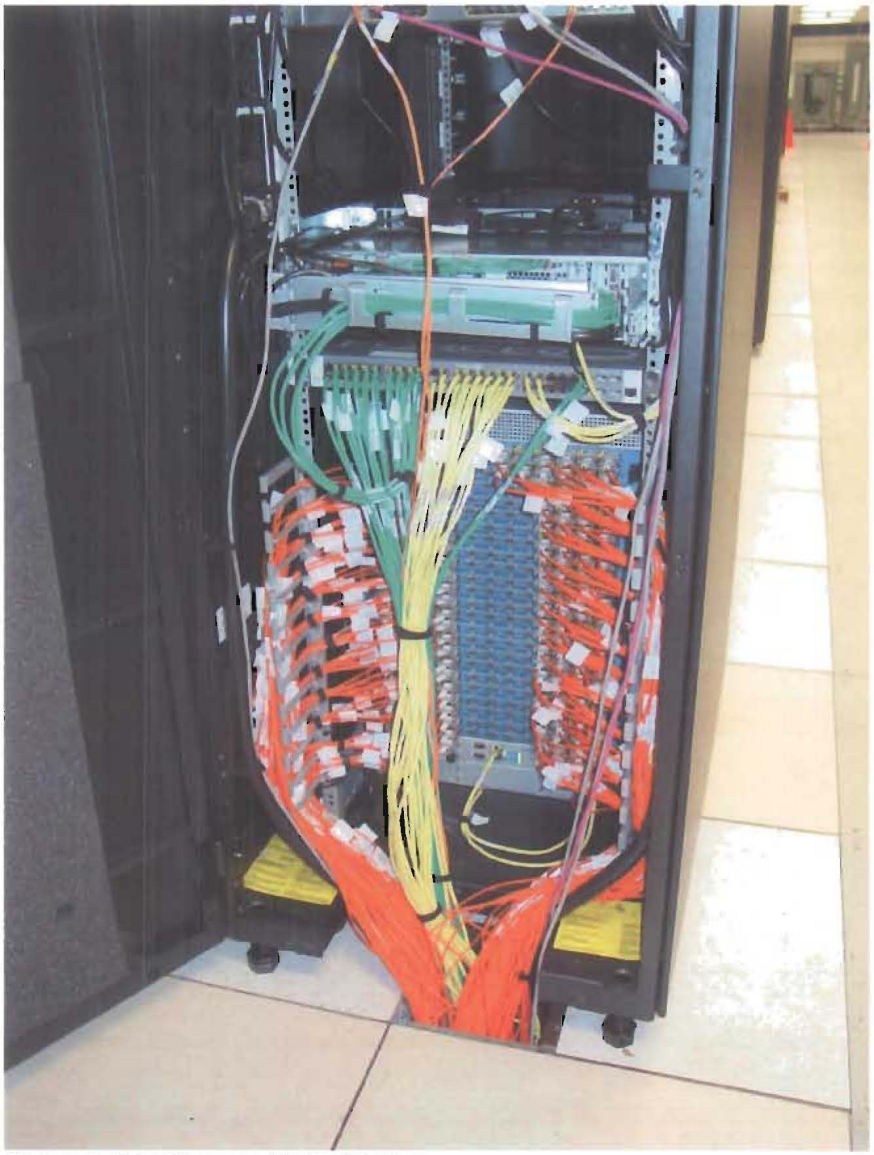


Figure 2. RoadRunner Node Cabling

Each end of the cable should be labeled with its intended port. It is also possible, but not required to label both ends with both designations having the designation closest to the end of the cord being that socket.

An example label would look something like this:

Server1 at 3Com 3970 =[]----------[]= Extreme 6816, blade 8, port 4

Serv1/3C-3970/P1 Ext6816/B8/P4

Serv1/3C-3970/P1 ; Ext6816/B8/P4 Serv1/3C-3970/P1 ; Ext6816/B8/P4

The reason that cabling standards are stressed so thoroughly is that a cluster without consistent cabling standards will be difficult to scale and makes it nearly impossible to convey instructions to fellow faculty.

Each machine, rack, and switch should be labeled consistently with the cabling for easy identification.

The roadrunner system at Los Alamos has 1728 main nodes each connected with Infiniband 4x cables. Each of these nodes is connected to every other node through a switch hierarchy. This adds up to a lot of cabling running side-by-side and each is indistinguishable.

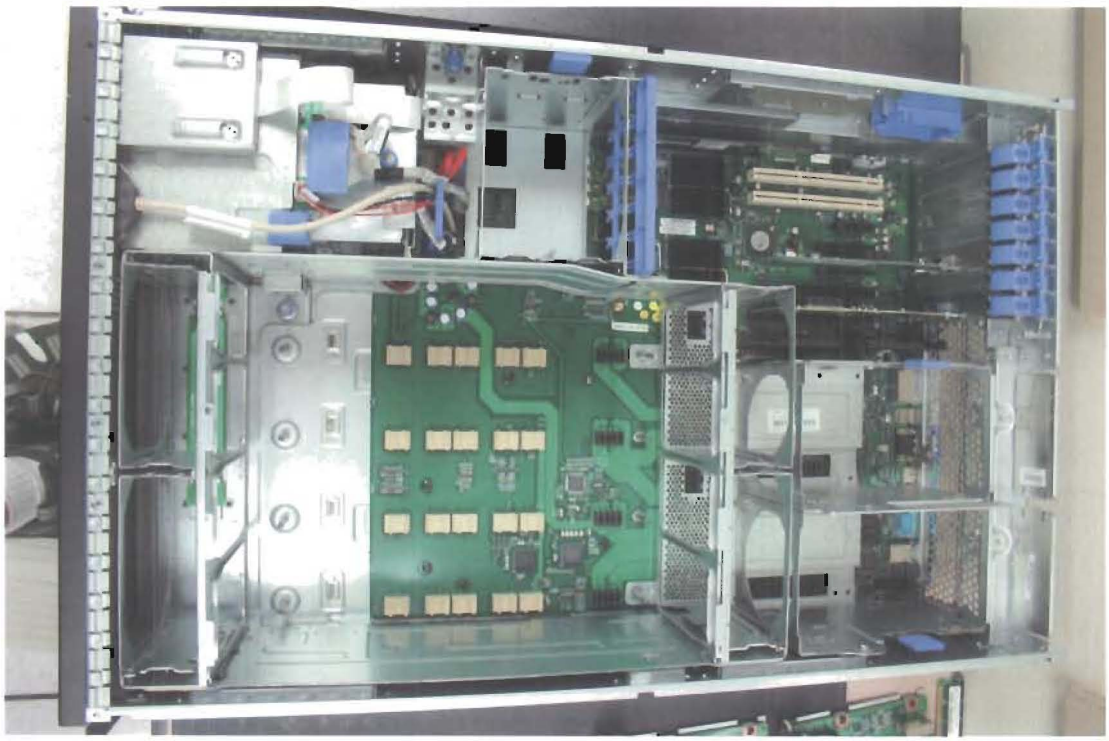


Figure 3. Intel x3755 stripped of parts displaying airflow

Server Room Layout

It goes well beyond the scope of this article to discuss the various dynamics of server room layout, cooling considerations, and power consumption. These are each complicated subjects that require thought and planning when building a server room.

The layout of the server room used for the basic cluster is pretty standard practice.

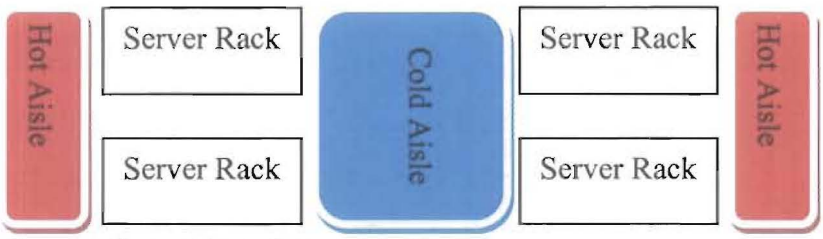


Figure 4. Server Room Layout Display

Understanding the airflow is necessary for proper cooling. Each server basically has cool air intake in the front and hot air exhaust at the rear. A series of fans within the server drive the airflow.

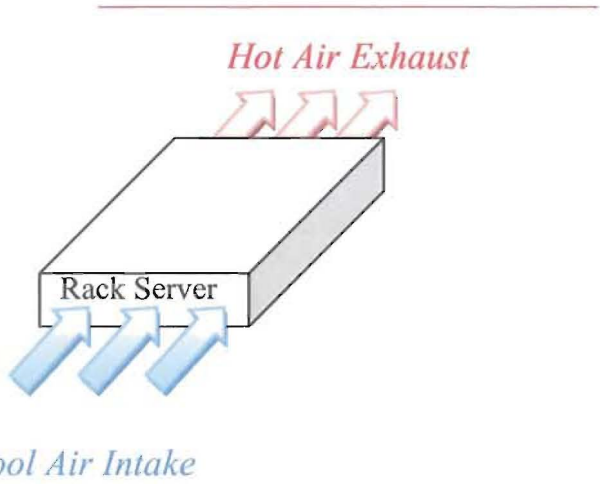


Figure 5. Server Airflow

Server racks can hold 25-30 servers and each rack may consume 10kW. As clusters grow, hundreds of racks will consume Megawatts of power. Furthermore, you will have to power the thousands of tons of cooling capacity required to keep the room temperate. A rough calculation would be:

4 x 11 servers = 44 x 350w = 15.4 kW

This does not include the switches and people.

Operating Systems

fedora

Fedora and the Infinity design logo are trademarks of Red Hat, Inc.

www. **fedoraproject**.org/

ubuntu

www.**ubuntu**.com/

CentOS

www.**centos**.org/

There are a wide variety of free operating systems that are available in addition to those above such as OpenBSD or Gentoo. The decision of which to choose is primarily motivated by each of use, hardware compatibility, experience, and purpose. In this cluster, Centos was used due to its kernel and current compatibility with the hardware.

In a basic cluster, the head node/mother superior will contain all of the necessary services for compute/backend nodes. In a larger installation, there will be multiple sister mothers as well as service nodes, io-nodes, and front-end servers. These are not necessary in our small cluster. A single beefy head node can handle the load required.

Essentially, everything that the compute nodes need to function without communicating with the outside world will be prepared and installed upon the head node. This includes yum repositories, resource manager, scheduler, file shares, remote boot tools, monitoring, and account management.

Services installed:

- ssh (remote access)
- http (remote file serving/kickstart)
- tftp (needed for pxe boot)
- dhcp (ip assignment)
- dns (domain name)
- rsync (needed for yum repository updating)
- nfs (file share)
- ldap (account management)
- openmpi (interprocess communication)
- ganglia (cluster monitoring)

Diskless vs Diskfull

Diskfull – Each node in the cluster contains at least one hard drive upon which the OS image is installed. For redundancy, RAID may be used.

Benefits:

- Can store many small files without thrashing the network
- Can accumulate large logs locally
- Can boot without head

Costs:

- Hard drives fail often
- More heat
- More maintenance

Diskless – There are no hard drives in the cluster. The entirety of the OS is stored within memory and is reinstalled each reboot.

Benefits:

- No hard drive to fail
- Less energy consumption
- Faster deployment
- Faster boot
- No disk interaction

Costs:

- Must be remotely booted each restart
- Requires small portion of RAM for system image
- Heavier network traffic at boot time
- Heavier load on head node at boot time
- Compute nodes unbootable without head

Modern large clusters systems are going diskless. It is much simpler to use Storage Area Networks (SANs).

Comparison

Disk-full (e.g. kickstart)

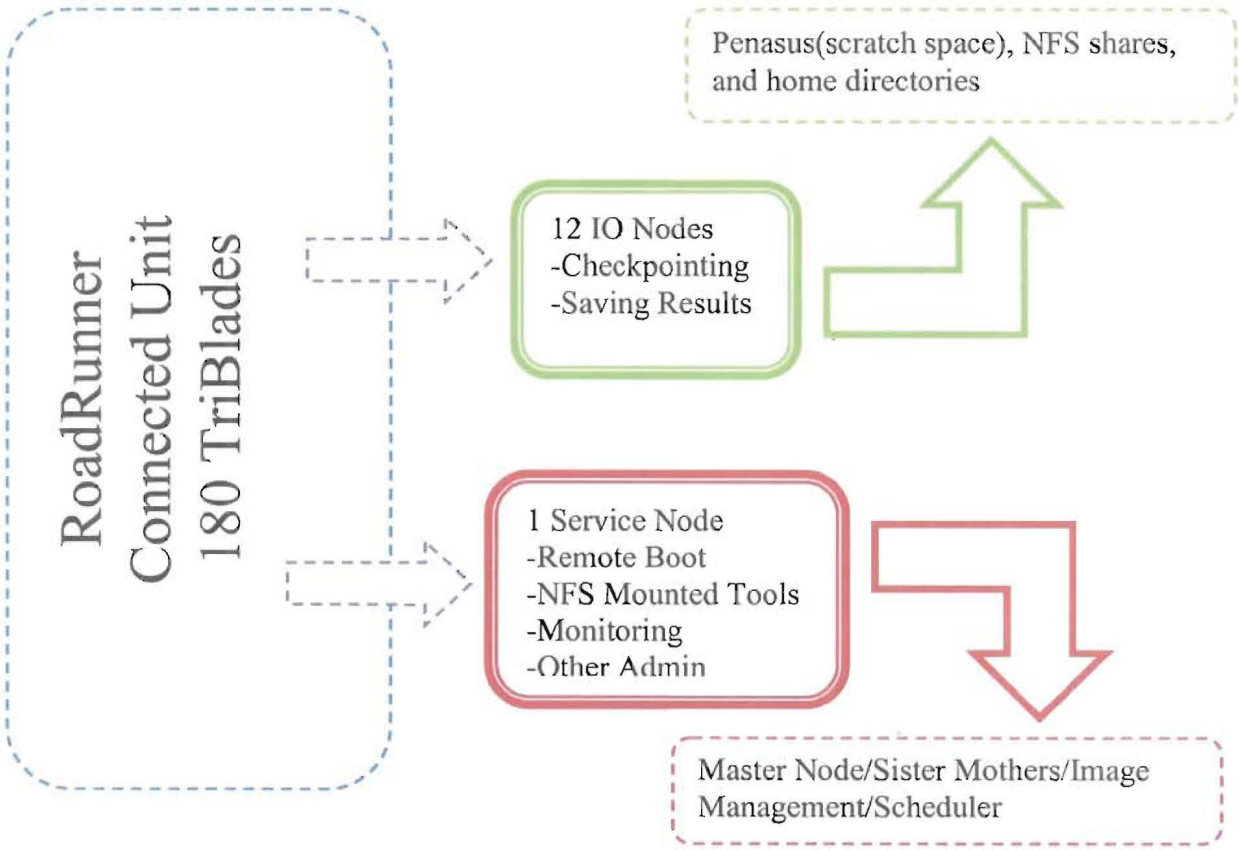
- Construct kickstart file
- Boot node to install to HD
- Changes require reinstall or cron job
- Requires new installation upon a slow HD
- Updates must be maintained in kickstart file as well as live node hd

Disk-less (e.g. Perceus)

- Construct vnfs image
- Boot node pushes image to RAM
- Changes can be handled by quick reboot or hotsync
- Updates to vnfs pushed to live nodes
- Requires 2 min reboot for reintegration

Modern Diskless

Modern cluster computers use a variety of solutions to solve the problems of scaling. There is not a single head node and the head node's services are further subdivided to a variety of servers.

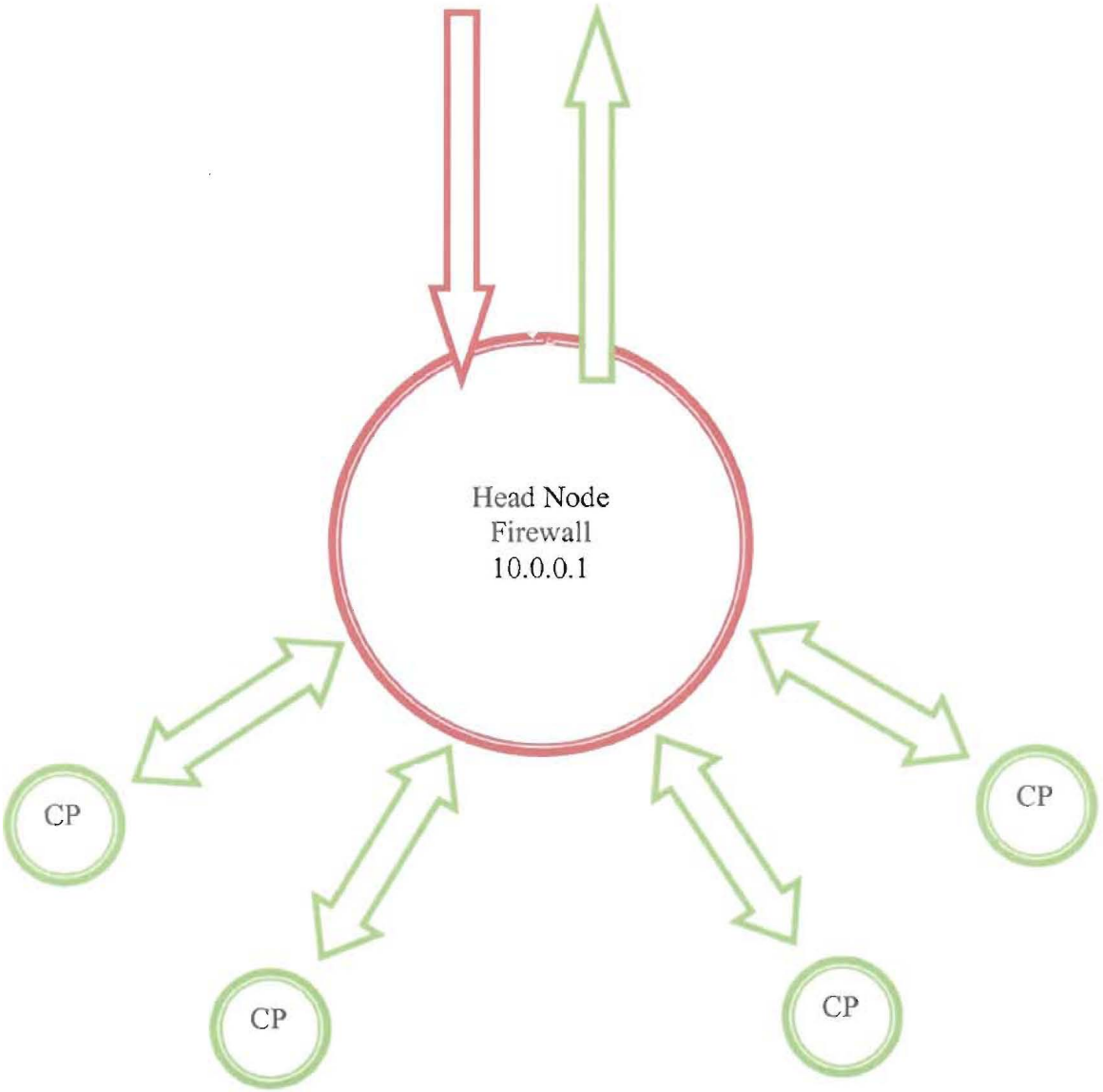


The Internal Network

Internal networks usually have an address in the range of:

10.0.0.0	-	10.255.255.255
172.16.0.0	-	172.31.255.255
192.168.0.0	-	192.168.255.255

The iptables rule settings given to the cluster are show below. These rules allows the compute nodes to forward traffic through the head node, prevents all traffic to the head node except on ports 22 and 80, and masquerades outgoing compute node packets as coming from the head node.



```
[root@head ~]# cat /root/nat.sh
echo "dumping old rules"

iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

echo "Setting up new rules"
#allow basic services
iptables -A INPUT -d 10.201.3.80 -p tcp --dport 22 -j ACCEPT


## ALLOW remote access to cyclades
iptables -t nat -A PREROUTING -p tcp -i eth1 -d 10.201.3.80 --dport 8888
-j DNAT --to 10.0.0.254:80
iptables -A FORWARD -p tcp -i eth1 -d 10.0.0.254 --dport 80 -j ACCEPT



#--------------------------------------------------------------------
# Allow previously established connections
# - Interface eth0 is the internet interface
#--------------------------------------------------------------------
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth1 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT


#--------------------------------------------------------------------
# Allow port 80 (www) and 22 (SSH) connections to the firewall
#--------------------------------------------------------------------

iptables -A INPUT -p tcp -i eth1 --dport 22 --sport 1024:65535 \
  -m state --state NEW -j ACCEPT
iptables -A INPUT -p tcp -i eth1 --dport 80 --sport 1024:65535 \
  -m state --state NEW -j ACCEPT



iptables --table nat --append POSTROUTING --out-interface eth1 -j
MASQUERADE
iptables --append FORWARD --in-interface eth0 -j ACCEPT
# Enables packet forwarding by kernel

#global reject
iptables -A INPUT -d 10.201.3.80 -j REJECT

echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
root@head ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             10.201.3.80          tcp dpt:ssh
ACCEPT     all  --  anywhere             anywhere             state
RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp
spts:1024:65535 dpt:ssh state NEW
ACCEPT     tcp  --  anywhere             anywhere             tcp
spts:1024:65535 dpt:http state NEW
REJECT     all  --  anywhere             10.201.3.80          reject-with
icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             ts                   tcp dpt:http
ACCEPT     all  --  anywhere             anywhere             state
RELATED,ESTABLISHED
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state
NEW,RELATED,ESTABLISHED
```

Security First

Security and performance are often counter intuitive. However, it is possible to take minimal security precautions to prevent malicious users from taking down your cluster and removing data.

Configuring the Public Firewall

Configure iptables to block all incoming track on the public Ethernet interface (eth1) except for port (22). This port is reserved for ssh traffic and will allow users to access the cluster remotely. By default, CentOS will block everything out. To add port 22 to exceptions, modify `/etc/sysconfig/iptables` after the line that reads `COMMIT`.

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp
      --dport 22 -j ACCEPT
```

Save the file and restart the iptables server.

```
/etc/init.d/iptables restart
```

Remote Traffic Control

Modify the /etc/hosts.allow file to only allow local and trusted hosts on the outside ethernet device and allow all traffic on the internal ethernet device.

```
#10.201.3.* is the local net, 10.0.0.* is the internal net
      ALL: 10.201.3.* 10.0.0.*
      ALL: *.mydomain.com          #Any named domains help too!
      ALL: *.myvzw.com             #My remote internet system
```

Modify the /etc/hosts.deny file to block all hosts on the publicly connected ethernet device and to allow all traffic on the internal ethernet device.

```
ALL: ALL
```

SSH and Encrypted Key Pairs

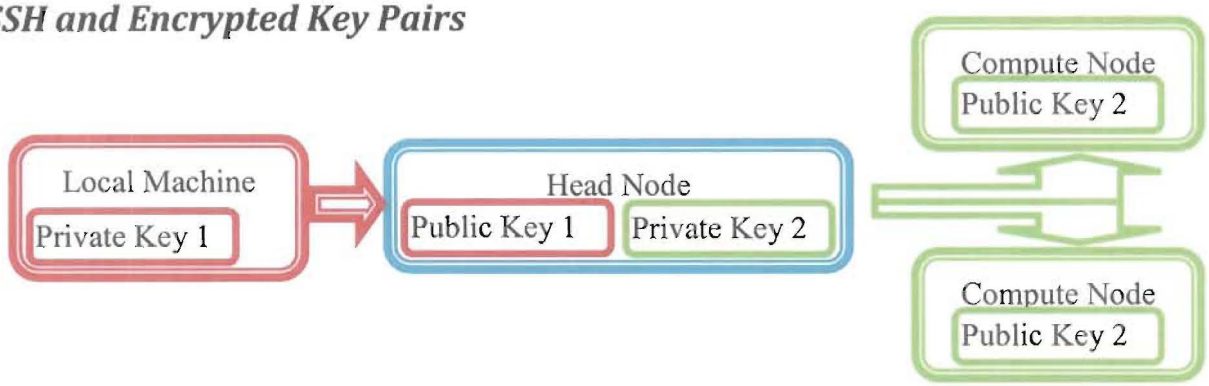


Figure 6. Key Use

Password authentication is not as strong and convenient as having key pairs. By generating an encrypted key pair (public/private) and using those exclusively for logging in to the headnode, one can increase the security of their system drastically. To do this, we'll generate the key pair, keeping the private key on the local machine, and placing the private and public key on the headnode to authenticate against. Then we'll completely disable the use of passwords on the remote machines, removing the possibility of general password brute force attacks from the outside world.

Generate a key pair to the `~/.ssh` directory (`id_rsa & id_rsa.pub`):

```
ssh-keygen -t rsa -b2048
```

Copy the public key (`~/.ssh/id_rsa.pub`) to the server and concatenate it to the `~/.ssh/authorized_keys` file on the server:

```
#Copies files to server
scp -rp ~/.ssh/id_rsa.pub exampleuser@10.201.3.80:~/.ssh/
scp -rp ~/.ssh/id_rsa exampleuser@10.201.3.80:~/.ssh/

# ... LOGIN TO SERVER ... #
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Create the file `~/.ssh/config` and enter:

```
Host head
  Hostname 10.201.3.80                    #External IP of headnode
        User exampleuser
  IdentityFile ~/.ssh/id_rsa
        ForwardX11 yes
```

Create the `~/.ssh/config` file on the headnode and enter:

```
Host head
  IdentityFile ~/.ssh/nopass_rsa
  ForwardX11 no
Host node*
  IdentityFile ~/.ssh/nopass_rsa
  ForwardX11 no
Host desktop
  Hostname 10.201.3.45
  User exampleuser
  IdentityFile ~/.ssh/id_desktop_rsa
  ForwardX11 no
```

Change the settings in the `/etc/ssh/sshd_config` file to disallow password logins and only allow SSH2 connections. *Note: Ensure that you stay logged in while you test it to ensure that it works before you lose the connection to the server!*

```
Protocol 2                          #SSH2 only
PermitRootLogin no
PubkeyAuthentication yes            #RSA only
PasswordAuthentication no           #Disallow passwords
X11Forwarding yes                   #Allow GUI apps to forward
```

Using LDAP, we can propagate the keys to the compute nodes and increase security within. It is recommended to attach a password to your RSA key; any user using your identity on the head node has automatic access to all compute nodes.

No Root Static SSH Domain Name

If the cluster admin does not have root access on the local machine in which he is working, then it is impossible to change the hosts file and have manual domain entries for easy access to other servers. However, these entries can be made for the ssh client by creating `/home/<user>/.ssh/config` and adding entries for the necessary hosts.

Using Putty

Download the putty client and puttygen executables from the putty download page. Use the puttygen executable to generate your key pair. Save the public and private keys locally. On the head node, open the file `/home/<user>/.ssh/authorized_keys` and paste the new key after the other keys. An example is given below with the public key prepended with ssh-rsa. Ensure that the entire key has no newline characters placed within it and only a space between the key type and the actual key text.

```
ssh-dss
AAAAB3NzaC1kc3MAAACBAIVVvLm^JUHGRFYPS07lq+6whrrlH8ASXjLq+w3WE4JaMJZLHiTbP
pz3M6+hN9jBC/XJ3cOdWf/hujX8sI8f6UYen4U39rjmt4dqzvn/wziGAOB+gv4kf3Mrhmxnnl
WzE2uIDouXEvxPz3380UF7gPFqUdMs23470XObKLZ9dqy5DAAAAFQCTLxNGooBZazpC+PY8tT
y5Ir6D0QAAAIAwhV5jg9LdLbU3PmEcQK0336aW3aviJ2koc+7ULPyMPtqCDpT78shfIhqbieA
eK+V/usEVDdFpF6+5p0xwwfXKPlEZTt91V9F/f8TgsJ8Ibu/Yu4BYPqEyza3ZuYriGtBKKNCk
sbW3ITirPNCNeTncqisPp1IP3rpzDd2IzNL37QAAAIA3jwPTc+A/eXlAu42Ca95ayvRJTvLay
vKiehT4Vud8ifl9Vf5pctr+XRm8qaE4H0yMnShsmYjSZxu7XGnfbM7tAirEDEdTECx+a5D6XN
z/mVgPr/Lr+RdOZtL6nuDMq05FZhkcoxpW4E7TFCCIGAbNkQe3bJkVPUyb1YFXyBoMdw==
exampleuser@head

ssh-rsa
AAAAB3NzaC1yc2EHDDUFIWKDIXOCU%iuS8VO+H1JeXJovyvuUA0DJW37UEHj7wjEarWpFybSG
mtvK2VAbt2UAb4HGz/t4kliSW9YJlBAf5brRE4xEDTdLUNycOQCkZsOwM+NjOErCTPcZngYmF
hFh0THaMG3/My117tO1c3T3lM/ih7MrZdomn/f2HA1Fjyq4WmUfWeC0=
```

To login, simply use the putty client on the local machine to enter the username and server address on the first line. Navigate to the Auth option using the left pane. Select the Private key file for authentication option.

Browse to the private key you had previously saved and select it.

You can save a profile of the hostname and private key file settings by choosing Session on the left pane. Enter a name for the new profile and choose Save.

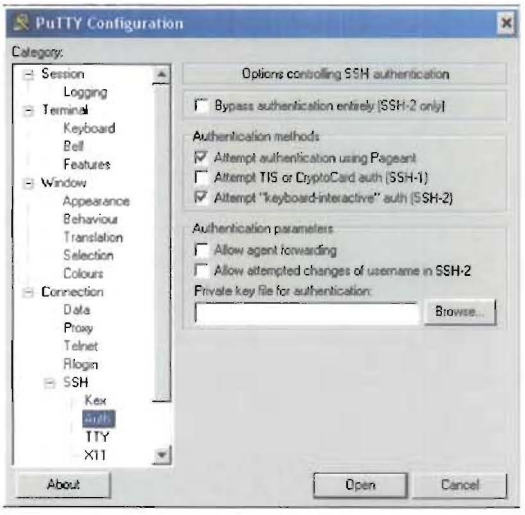


Figure 7. Putty Auth Option

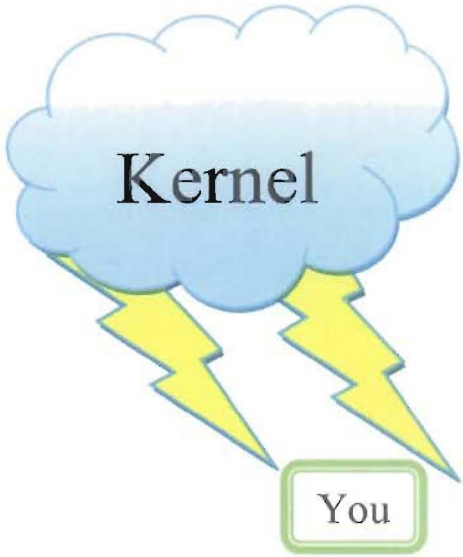
OS Images and Repositories

It is best to have a local OS and repository for cluster computing. This will relieve excessive bandwidth from your gateway connection and speed up each installation.

```
[root@head /var/www/html/centos]# ls
CentOS5ISO  comps.xml  extras  os  rpmforge  updates
```

The folder `CentOS5ISO/` contains the operating system retrieved from CD. The other four folders contain various packages/rpms. The file `comps.xml` adds metadata to your repositories.

OS Image



The OS image used on the compute nodes can be retrieved from a variety of sources. All files required can be retrieved from a CD, local network repository, or internet repository.

Regardless of the method in which an image was retrieved, the image must be available to PXEBoot using NTP, web server, or other. For this example, an Apache web server was used to serve the OS image as well as the packages available for installation.

Updating/Installing the Repositories

First choose a mirror that's closest to you from this list (http://mirror-status.centos.org/)

Make folders for each repository you wish to mirror (ex: `/var/www/html/centos/os`)

To retrieve a complete list of available repositories :

```
rsync -vr rsync://`mirror ip`/
```

In order to link and index the repositories on your computer run createrepo on the folder you have installed your repository.

Setup a cron job to update them on a reasonably regular basis (typically this is done daily) . Below is an example of both a cron update and initial installation:

```
#!/bin/bash
rsync -vr rsync://10.201.3.82/centos5.3-os-x86_64/
/var/www/html/centos/os/
rsync -vr rsync://10.201.3.82/centos5.3-updates-x86_64/
/var/www/html/centos/updates/
rsync -vr rsync://10.201.3.82/centos5.3-extras-x86_64/
      /var/www/html/centos/extras/
rsync -vr rsync://10.201.3.82/centos5.3-rpmforge-x86_64/
      /var/www/html/centos/rpmforge/
createrepo /var/www/html/centos/os/
createrepo /var/www/html/centos/extras/
createrepo /var/www/html/centos/rpmforge/
cp -r rpm-gpg/ /configs/
createrepo -g /var/www/html/centos/os/repodata/comps.xml
      /var/www/html/centos/os
```

+
Note the `createrepo -g`, this was needed to provide package group linking, it appears to only be needed for the OS repository.

Yum repositories can be modified by navigating to `/etc/yum.repos.d/` and adding or changing the files to include the desired repositories. The existing repositories can serve as excellent examples.

Anaconda's Kickstart Script

Figure 8. Anaconda Logo

Anaconda is an automated installer. Using a kickstart script, the installation can be handled easily, quickly, and consistently. When the PXEBoot client retrieves the kickstart script from the web server, the Anaconda installer will run it. A kickstart script can be configured manually or automatically through a GUI. Visit the website for additional details and available options.

For the GUI, install the Kickstart GUI (`system-config-kickstart`) or copy the `/root/anaconda-ks.cfg` kickstart file which is automatically generated from your head node's installation parameters.

The GUI can be executed by the command `system-config-kickstart`.

When designing the kickstart script for this basic cluster, it was noticed that the order of the script can have adverse affects when it comes time to install. It can cause installation errors that are often cryptic.

Web Server Configuration

Create a directory listing to include the yum repository location.

Create a kickstart script and store it in a directory accessible from the Apache server.

Services

Most of the modifications to services listed below can be automatically executed by the kickstart script at installation. Furthermore, using Perceus makes most of these services useless on the head node. In fact, DHCP, DNS, and the tftp daemon must all be disabled for the proper execution of Perceus in this cluster. This is due to the fact that Perceus is an all in one tool which contains varieties of these services that work closely with its main service of image management.

If you're designing your kickstart file to handle these services, then choose your configuration carefully.

DHCP - Dynamic Host Configuration Protocol

View configuring DHCPD in Network Booting the subnodes

DNS – Domain Name Service

Point the headnode's DNS service at itself from the `/etc/resolv.conf` file.

```
search exampleuser.net      #What are we calling our internal network?
nameserver 10.0.0.1         #This server's IP internal network address
nameserver 205.105.7.62     #External NS provided by provider
```

Add /etc/dhclient-enter-hooks which prevents the OS from reconfiguring resolv.conf on restart or each dhcp request:

```
make_resolv_conf( ) {
    logger -t "dhclient-enter-hooks" "Doing nothing to /etc/resolv.conf"
}
```

The Named/Bind daemon processes can be configured using the GUI `system-config-bind` or it can be configured manually using the changes below. The particulars of what each of these files mean and the domain specific to one's cluster need to be understood before modification. It is recommendable to use the GUI first and make any necessary changes later. Also, the administrator should plan for expansion of the cluster. Organizations may wish to expand the capabilities of the cluster at a later time.

Within the `/var/named/chroot/etc/named.conf` file add the following:

```
zone "0.0.10.IN-ADDR.ARPA." IN {
        type master;
        file "10.0.0.db";
};
zone "teamyellow.com." IN {
        type master;
        file "danegardner.net.db";
};
```

Create file /var/named/chroot/var/named/10.0.0.db

```
$TTL 1H
@       SOA     head.exampleuser.        root.head.exampleuser. ( 4
                                                 3H
                                                 1H
                                                 1W
                                                 1H )
                        NS      head.danegardner.
101     PTR     node01.danegardner.net.
102     PTR     node02.danegardner.net.
103     PTR     node03.danegardner.net.
104     PTR     node04.danegardner.net.
105     PTR     node05.danegardner.net.
106     PTR     node06.danegardner.net.
107     PTR     node07.danegardner.net.
108     PTR     node08.danegardner.net.
109     PTR     node09.danegardner.net.
1       PTR     head.danegardner.net.
254     PTR     ts.danegardner.net.
```

Create file /var/named/chroot/var/named/exampleuser.net

```
$TTL 1H
@       SOA     head.exampleuser.        root. ( 3
                                                 3H
                                                 1H
                                                 1W
                                                 1H )
                NS      head.exampleuser.
node01  IN      1H      A       10.0.0.101
node02  IN      1H      A       10.0.0.102
node03  IN      1H      A       10.0.0.103
node04  IN      1H      A       10.0.0.104
node05  IN      1H      A       10.0.0.105
node06  IN      1H      A       10.0.0.106
node07  IN      1H      A       10.0.0.107
node08  IN      1H      A       10.0.0.108
node09  IN      1H      A       10.0.0.109
ts      IN      1H      A       10.0.0.254
```

NTP - Network Time Protocol

NTP is a daemon service that synchronizes the system time with a time server. Furthermore, other systems can synchronize themselves against this computer as well if properly configured. In a cluster computing environment, the compute nodes must have up-to-date time synchronized with the head of the cluster.

The time on the compute nodes needs to be hard reset at the time of the original boot. The large time difference between the compute nodes and the head node will take forever for the NTP daemon service to correct. This is due to the fact that NTP makes small incremental changes to the time.

It should be noted that some scientific clusters have modified or used alternatives to NTP due to reliance on a more exact time standard and/or due to some performance degrading interrupts generated by the NTP daemon.

Head

Add this line to `/etc/ntp.conf` in the head node to allow broadcasting of the time service.

```
# -- CLIENT NETWORK -------
restrict 10.0.0.0 mask 255.255.255.0 nomodify notrap
```

Modify `/etc/dhcpd.conf` to include this line to have the DHCP daemon automaticly distribute the location of the NTP servers to DHCP clients.

```
option ntp-servers <ip or domain name of ntp server>;
```

Compute Node

Add this line to the compute node's `/etc/ntp.conf` file to have them query the correct server. It may be necessary to remove the default Centos public NTP servers.

```
server <ntp server domain>
```

NFS - Network Filesystem

NFS is a trivial system of sharing files in a unix enviornement. For the most part all configuration for NFS servers can be done with a single file (`/etc/exports`). Below you'll find an example exports file. In this example we use ip addresses however it's also possible to use domain names assuming that reverse lookup is reliable on your system. You can use wildcards for subdomains (ex: *.teamyellow.com).

In the basic cluster, the NFS service will be installed and run solely from the head node. In a larger cluster, the load placed on a single server will be too great. Instead, dedicated NFS servers will be used for user's files. Furthermore, the LANL RoadRunner uses a Penasus scratch space. This is less reliable then the NFS shares, but is faster. Both Penasus and NFS will share IO bandwidth in a large cluster. For a diskless cluster, administrators will maximize the parts of the OS that can be shared over the network. Tools such as compilers and special interpreters can be provided at the last minute by the service node. This saves space in the OS RAM.

It is recommendable to activate root-squash to prevent root users on remote machines from performing whatever actions they wish.

```
[root@head src]$ cat /etc/exports
/home 10.0.0.0/24(rw)
/scratch 10.0.0.0/24(rw)
/configs 10.0.0.0/24(ro)
/configs *(ro)
/var/lib/perceus/ 10.0.0.0/24(rw,no_root_squash,async)
/usr/local 10.0.0.0/24(rw,no_root_squash,async)
```

Notes:

1. It's important that the portmap service is running in order for NFS to properly function.
2. There are two ways to mount an NFS share
    1. mount server:/home /home
    2. In /etc/fstab (to automount on boot): 10.0.0.1:/home /home nfs
3. When you mount an NFS share by default it sits on top of any existing filesystem.

LDAP - Lightweight Directory Access Protocol

LDAP will serve as the authentication mechanism in our cluster. It provides directory listing for the compute nodes. This will allow all compute nodes to share identical user names and ids. The shared user ids are necessary for NFS to operate properly. If additional security precautions can be activated on your compute nodes without substantial detriment to performance, then it would be recommendable.

Head

1. The following packages need to be installed for NFS to function(yum install openldap openldap-clients openldap-devel nss_ldap openldap-servers).

2. Create a home for the LDAP database

```
mkdir /var/lib/ldap/example.com
chown ldap:ldap /var/lib/ldap/example.com
```

3. Generate a root LDAP password. Typically this should be different then your normal system root password. This can be done by simply typing slappasswd and entering your new PW. You'll want to make sure you save this hash somewhere we'll need it for later use.

4. Edit /etc/openldap/slapd.conf (change information according to your configuration)

```
suffix "dc=example,dc=com"
rootdn "cn=Manager,dc=example,dc=com"
rootpw {SSHA}v4qLq/qy01w9my60LLX9Bvv0mioeRhOjQZ
directory /var/lib/ldap/example.com
```

5. Within this file (/usr/share/openldap/migration/migrate_common.ph) you'll find several instances where LDAP developers have Padl as the domain. You'll want to replace all instances of Padl with your domain name (ex: example).

6. Copy the example LDAP database configuration file.

```
cp /etc/openldap/DB_CONFIG.example /var/lib/ldap/example.com/DB_CONFIG
```

7. Migrate the users currently present on your system.
/usr/share/openldap/migration/migrate_all_offline.sh Note: For the migrate_offline script to

function LDAP server has to be stopped. Otherwise you do have the option of using the migrate_all_online script). Both are functional.

8. Make sure the permissions on the LDAP database are correct.

```
chown -R ldap:ldap /var/lib/ldap/example.com
```

9. You should now be able to start LDAP successfully. It's also a good idea to enable it on boot (`chkconfig ldap on`).

Compute Node

1. The following packages are required for the LDAP client to function. (yum install Openldap openldap-clients openldap-devel nss_ldap)

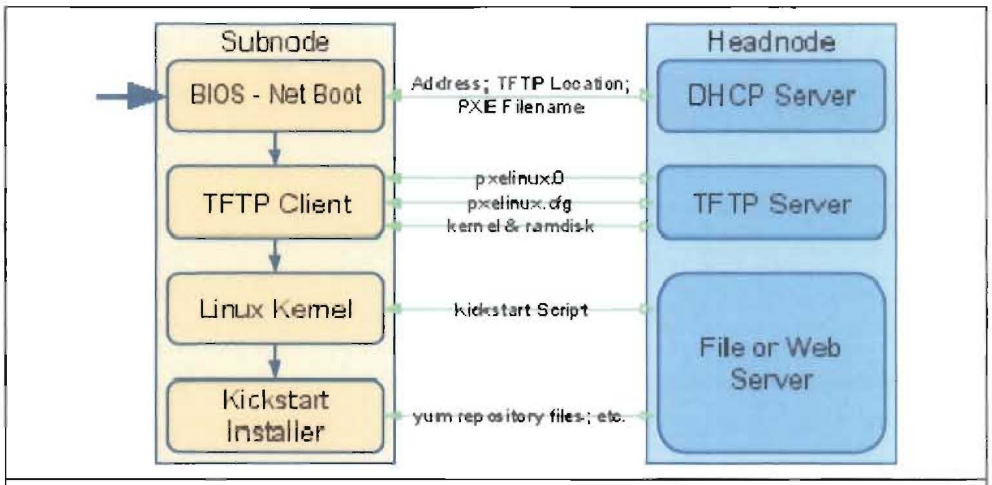
2. Edit /etc/ldap.conf to reflect your system. Typically all that needs changing is the HOST and BASE sections.

3. In order to enable authentication via ldap the /etc/nsswitch file needs to be modified. This can be accomplished with the command `authconfig-tui`. If desired you can also allow local authentication at this time.

Note: At this point LDAP authentication should function and for cluster authentication should be sufficient however by default LDAP communication is not encrypted. If this is the desired option you should be sure you've taken precautions against attackers (firewalls etc).

Diskfull Remote Booting

In order to boot from the network using PXE (pronounced 'pixie'), the subnode will ask for an IP address from the DHCP server, which will also provide the address to the TFTP server (via the `'next-server'` parameter) and the filename of the PXE executable. The subnode will then request the file via the TFTP protocol and execute it upon receipt. This executable (usually, `pxelinux.0`) will then begin looking for a configuration file, which will tell it what to kernel to load. Once the kernel is loaded into memory it will be executed and the boot procedure will continue on much like a CD or hard drive boot would.



How the subnode interacts with the headnode during the net boot process.

Figure 9. PXEBoot Diagram

Configuring DHCPD

Enable the DHCP server by specifying in the `/etc/sysconfig/dhcpd` file which ethernet device you'd like it to be active on:

```
DHCPDARGS=eth0
```

Modify the `/etc/dhcpd.conf` appropriately. Notice the underlined text:

```
ddns-update-style interim;     #Required by DHCPD

subnet 10.0.0.0 netmask 255.255.255.0 {
        option domain-name "[mydomain.com]";
        option ntp-servers [ntpIP];
        option domain-name-servers [dnsIP];
        range 10.0.0.200 10.0.0.254;
        authoratative;
}

#Group the nodes together for netbooting
group {

        next-server 10.0.0.1;                  #TFTP server
        filename "pxelinux.0";                 #Points to the PXE bootfile

        #Each node in the cluster needs an entry like this
        host node## {
                hardware ethernet ##:##:##:##:##:##;
                fixed-address 10.0.0.1##;
        }
}
```

Restart the DHCP server and ensure that it is loading at boot up.

```
/sbin/service dhcpd restart
/sbin/chkconfig dhcpd on
```

Configuring TFTP

Install TFTP and SysLinux:

```
yum install tftp tftp-server syslinux
```

Turn on TFTP by modifying the file `/etc/xinetd.d/tftpd`. Change the following lines to match:

```
server_args     = -s -v /tftpboot     #Verbose logging
disable         = no                  #Enable!
```

Copy the following files to the `/tftpboot/` directory. *Make sure that the kernel & initrd.img are the ones from the install disc, not the operating system -- or modules/drivers for the os are going to load and fail on different hardware.*

```
cp /usr/lib/syslinux/pxelinux.0 /tftpboot/    #PXE boot environment
cp /usr/lib/syslinux/menu.c32 /tftpboot/      #Advanced menus for PXE
cp /mnt/[bootcd]/isolinux/vmlinuz /tftpboot/ #Kernel image from boot disc

#Initial RamDisk from boot disc
cp /mnt/[bootcd]/isolinux/initrd.img /tftpboot/
```

Create the following directory and files:

```
mkdir /tftpboot/pxelinux.cfg/
touch /tftpboot/pxelinux.cfg/default
touch /tftpboot/pxelinux.cfg/install.menu
touch /tftpboot/pxelinux.cfg/utility.menu
```

PXE linux will, by default, check for configuration files in the pxelinux.cfg directory (using TFTP) in the following order:

```
##-##-##-##-##-##        #The hardware (MAC) address of the ethernet device
0A0000##                 #The IP address in hex format
0A0000#
0A0000
0A000
0A00
0A0
0A
0
default                  #The name of the default config file
```

Create /tftpboot/pxelinux.cfg/install.cfg and edit to contain the direction to boot from the install kernel and pass the perameters to load the kickstart file and use the console:

```
default linux
label linux
    kernel vmlinuz
    append initrd=initrd.img ramdisk_size=65536

    #Redirects output to serial console as opposed to a video device
    append console=tty0 console=ttyS0,115200n8

    #Point to the Kickstart script, accessible from the Apache server.
    append ksdevice=eth0 noipv6 ks=http://10.0.0.1/kickstart/pn-ks.cfg
```

Create /tftpboot/pxelinux.cfg/0A0000 and edit to contain the directions to boot from the local hard drive.

```
default bootlocal
label bootlocal
    localboot 0
```

Create a bash script to cycle through the possible IP addresses (in hex) to load the `install.cfg` contents:

```
for I in 101 102 103 104 105 106 107 109
do
    IPHEX=`gethostip -x 10.0.0.$I`
    cp /tftpboot/pxelinux.cfg/install.cfg /tftpboot/pxelinux.cfg/$IPHEX
done
```

After the new operating system is installed, we need to have it remove the configuration from the PXElinux's view, or it will try to reinstall after a reboot. When the file is set to empty the 0A0000 file will eventually be checked, and it will tell the PXE to boot the local hard drive. To do this add to the Anaconda Kickstart 'Post Install Script' the following lines. *Note: you may have to install the TFTP client into the Anaconda ramdisk to get this to work. Another method would be to have a PHP script do the modification to the PXE configuration file on the headnode after POSTing to Apache from the subnode.*

```
#Get the IP of this computer
IPADDR=`ifconfig eth0 |grep --only-matching 10.0.0.1[0-9]*`

#Translate the IP into hex
IPHEX=`gethostip -x $IPADDR`
touch emptyFile

#Copy over the config file for this machine
tftp [tftpServer] -c put emptyFile pxelinux.cfg/$IPHEX
rm --force emptyFile
```

Restart the XINET server and ensure that it is loading at boot up.

```
/sbin/service xinetd restart
/sbin/chkconfig xinetd on
```

After checking the bios to ensuring that your subnode will boot from it's connected network card, you should then be able to boot from the network. If it fails, try checking your Head node's `/var/log/messages` file to ensure that the `pxelinux.0`, the associated configuration, and the kernel and ramdisk files are being received by the remote device. Things that can go wrong include corrupted `pxelinux.0`, `vmlinuz` or `initrd.img` files or permissions on the files that restrict the TFTP server from accessing them. If they are being downloaded ensure that the appended kernel commands aren't wonkey and that the kickstart file can be accessed from the Apache server.

Note: One can test the TFTP server by running the following command. Be careful not to run this in the tftpboot directory, or the file itself will be corrupted as tftp tries to both read and write the file at the same time!

```
tftp [tftpServer] -c get pxelinux.0
```

Perceus Headless Installation

So, now that we know how to install everything manually, let us kill everything, and reinstall using Perceus. Install and configure NFS and LDAP again using the above methods. Ensure that xinetd (tftp), named (dns) and dhcpd are no longer running on your headnode -- Perceus uses it's own daemons to cover these services.

←----------TODO, image management and more description and weblink

Compiling and Installing Perceus 1.5.2

Follow the directions in the Perceus manual -- there's no need to unpack the Perceus download, just follow the directions. In CentOS I had to use yum to install `rpm` and `rpm-devel` and download the four dependency files from http://www.perceus.org/downloads/perceus/v1.x/dependencies/.

```
wget http://www.perceus.org/portal/files/perceus-1.5.2.tar.gz
```

Install the required dependencies.

```
yum -y groupinstall "Development Tools"
yum -y install nasm
yum -y install bash-completion
yum -y install perl-IO-Interface.x86_64
yum -y install perl-Net-ARP.x86_6
yum -y install perl-Unix-Syslog.x86_64
yum -y install autoconf
yum -y install rpm-build
yum -y install elfutils-libelf-devel.x86_64
```

Create rpm from tarball.

```
export TAR_OPTIONS=--wildcards
rpmbuild -ta perceus-1.5.2.tar.gz
```

Install the new Perceus RPM.

```
rpm -Uvh /usr/src/redhat/RPMS/*/perceus-$PERCEUS_VERSION-*.rpm
```

Import DHCP settings to dns-masq

```
/usr/share/perceus/import-scripts/import-isc-dhcpd.pl /etc/dhcpd.conf
```

Create a vnfs image by modifying one of the genchroot.sh scripts in `/usr/share/perceus/vnfs-scripts` to fit your purposes. Initial testing of creating a Fedora image using these scripts was not successful.

Import the vnfs image as a Perceus capsule.

```
perceus vnfs import /root/<new_os_image.vnfs>
```

Kill previous services.

```
service dhcpd stop
service named stop
service xinetd stop
chkconfig dhcpd off
chkconfig named off
chkconfig xinetd off
```

Start Perceus services.

```
service perceus start
```

Set node groups to compute in order to assign a vnfs image.

```
perceus node set group compute node[01-99]
```

Set the vnfs image of the group compute.

```
perceus group set vnfs centos-5.1-1 compute
```

Mount the vnfs image in order to make the necessary changes.

```
perceus vnfs mount centos-5.1-1
```

Chroot into the directory in order to run yum. Do not start the services in the chrooted vnfs image or there may be conflicts.

```
chroot /mnt/centos-5.1-1
```

Unmount the vnfs image in order to save the image to capsule.

```
perceus vnfs umount centos-5.1-1
```

Restart all nodes to allow the changes made to the vnfs image to take effect.

Networking Perceus

The 'ipaddr' module script that comes with Perceus has an issue with creating static IP addresses for the ethernet device that was booted from. Our original project used static IPs to identify each node, but Perceus automates this process. We run into problems when we try to use a seperate subnet for the InfiniBand cards -- the DNS server isn't automatically set up to recognize them. Using the ipaddr module in Perceus to get them running with the right ip address isn't trivial.

The original ipaddr module script doesn't handle the hardware types appropriately, and modification is necessary for the InfiniBand cards. Modify the `/var/lib/perceus/modules/ipaddr/nodescripts/50-ipaddr.pl` file as below:

```
Original:
if ( $dev =~ /^([a-z]+\d)_(\d+)$/ ) {
    $dev = "$1:$2";
    $type = "alias";
} else {
    $type = "ethernet";
}
Modified:
if ( $dev =~ /^([a-z]+\d)_(\d+)$/ ) {
    $dev = "$1:$2";
    $type = "alias";
} else {
    if ( $dev =~ /^(ib)/ ) {
        $type = "infiniband";
    } else {
        $type = "ethernet";
    }
}
```

Now configure the module to assign the IP addresses on each node by modifying the `/etc/perceus/modules/ipaddr` file:

```
* eth0:[default]/[default] eth1:[default]/[default]/[default]
node01 eth0:[default]/[default]/[default]
ib0:10.0.1.101/[default]/[default]
node02 eth0:[default]/[default]/[default]
ib0:10.0.1.102/[default]/[default]
node03 eth0:[default]/[default]/[default]
ib0:10.0.1.103/[default]/[default]
node04 eth0:[default]/[default]/[default]
ib0:10.0.1.104/[default]/[default]
node05 eth0:[default]/[default]/[default]
ib0:10.0.1.105/[default]/[default]
node06 eth0:[default]/[default]/[default]
ib0:10.0.1.106/[default]/[default]
node07 eth0:[default]/[default]/[default]
ib0:10.0.1.107/[default]/[default]
node08 eth0:[default]/[default]/[default]
ib0:10.0.1.108/[default]/[default]
node09 eth0:[default]/[default]/[default]
ib0:10.0.1.109/[default]/[default]
```

In order to get DNSmasq to register the new IPs add them to the `/etc/hosts` file:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain localhost
::1             localhost6.localdomain6 localhost6
10.0.1.101      ibnode01.teamyellow.com
10.0.1.102      ibnode02.teamyellow.com
10.0.1.103      ibnode03.teamyellow.com
10.0.1.104      ibnode04.teamyellow.com
10.0.1.105      ibnode05.teamyellow.com
10.0.1.106      ibnode06.teamyellow.com
10.0.1.107      ibnode07.teamyellow.com
10.0.1.108      ibnode08.teamyellow.com
10.0.1.109      ibnode09.teamyellow.com
```

Monitoring the Cluster

Ganglia, Nagios, and Zenoss are just a few of the many options available for cluster monitoring. Ganglia is the monitoring program of choice for the small cluster. However, it does not scale well. Nagios would not be an ideal option either.

Ganglia

Ganglia is an open source cluster monitoring software suite. It is composed of three main components:

Gmeta - Retrieves node information from the Gmond daemons and compiles the node statistics for viewing

Gmond - Retrieves and stores statistics about the local machine as well as any nearby nodes that it has received multicast packets from

Web Frontend - PHP code used by Apache to display the statistics stored by Gmetad

Head Node

- Download Ganglia source in a tarball
- Compile Confuse as position independent code with ./configure --with-pic
- Compile Ganglia (may have errors because of using 32bit libs compared to 64bit)

```
/root/ganglia/configure
/root/ganglia/make
/root/ganglia/make install
```

Add Ganglia frontend php files to Apache directory for viewing

```
mkdir /var/www/html/ganglia
cp -fR /root/ganglia/web/* /var/www/html/ganglia
```

Start the gmond daemon from the ganglia folder

```
cp ./gmond/gmond.init /etc/rc.d/init.d/gmond
chkconfig --add gmond
chkconfig --list gmond
/etc/rc.d/init.d/gmond start
```

Start the gmetad daemon from the ganglia folder

```
mkdir -p /var/lib/ganglia/rrds
chown -R nobody /var/lib/ganglia/rrds
cp ./gmetad/gmetad.init /etc/rc.d/init.d/gmetad
chkconfig --add gmetad
chkconfig --list gmetad
/etc/rc.d/init.d/gmetad start
```

Compute Node

The gmond daemon is required on all compute nodes to broadcast their state to all other nodes using multicast. A static route is required on each system to tell them which interface to broadcast the signal upon.

Install the gmond daemon on all compute nodes using the rpmforge repository stored on the head node.

Add to kickstart or chroot into perceus vnfs and install

```
yum install ganglia-gmond
```

Start the gmond daemon and add to chkconfig for automatic startup.

InfiniBand

InfiniBand Hardware (wiring)

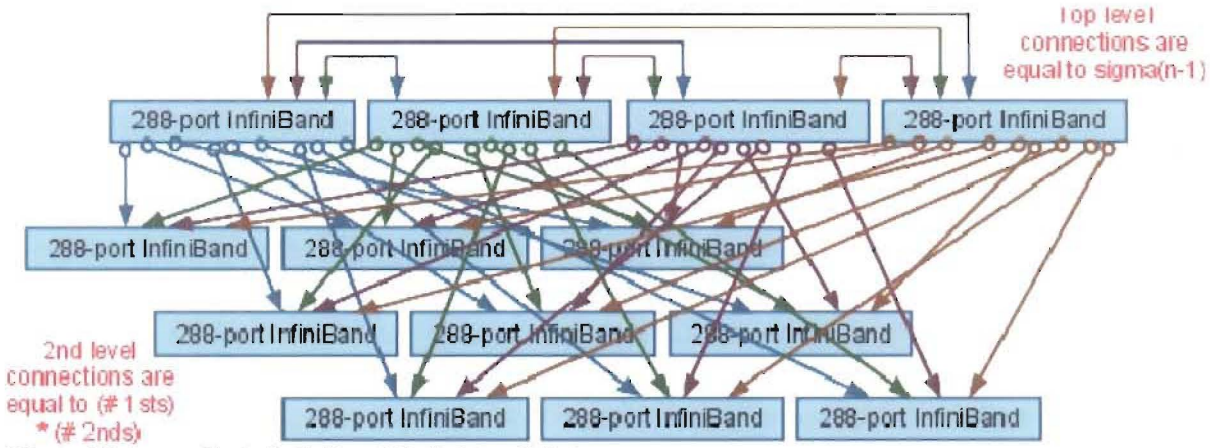


Figure 10. Large Scale Infiniband Switching Fabric

InfiniBand Software

Software consists of the drivers for the hardware and the protocols. In fact, Infiniband QDR has firmware on the cables. It can be important if the administrator must flash the bios on the cables, especially if there are hundreds of them.

Install the software, and ensure that the IB daemon is running:

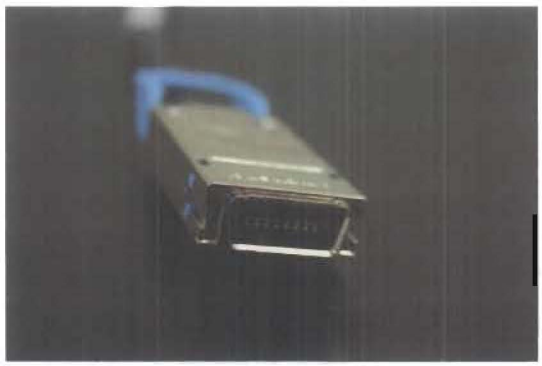


Figure 11. Infiniband 4x Cable

```
yum install openib libibverbs libmthca ibutils
/sbin/service openibd restart
/sbin/chkconfig openibd on
```

On the subnet manager (headnode) install OpenSM and start it:

```
yum install opensm
/sbin/service opensmd restart
/sbin/chkconfig opensmd on
```

Install the MPI protocol:

```
yum install mpich2
```

Config Files:

openib -- `/etc/ofed/openib.conf`

opensm -- `/etc/ofed/opensm.conf`

Infiniband hardware -- `/configs/etc/sysconfig/network-scripts/ifcfg-ib0`

Create the file /configs/etc/sysconfig/network-scripts/ifcfg-ib0 and fill with:

```
# Infiniband Device ib0
DEVICE=ib0
TYPE=Infiniband
BOOTPROTO=static
BROADCAST=10.0.1.255
NETMASK=255.255.255.0
NETWORK=10.0.0.0
ONBOOT=yes
USERCTL=no
IPV6INIT=no
PEERDNS=yes
# IPADDR=10.0.1.0    #Leave for the Kickstart post-install script to
append
```

Test the MPI interface once you've wired the system up:

OpenMPI

Install OpenMPI, MVAPICH2, and MVAPICH from repository

Add OpenMPI to executable path by modifying /home/<user>/.bashrc

```
PATH=$PATH:/sbin/:/usr/lib64/openmpi/1.2.7-
gcc/man/:/usr/lib64/openmpi/1.2.7-gcc/bin/:~:~/scripts/:~/..:/usr/sbin/
```

mpicc - Compile with the appropriate flags for mpi jobs

mpirun - Run the compiled program across the cluster (n - number of processes, -machinefile - list of machines capable of handling the job)

Install *mpitests** from the repository in order to have a variety of OSU created benchmarks for the infiniband interface

Create the file `ethmachines` with contents:

```
10.0.0.101
10.0.0.102
10.0.0.103
10.0.0.104
10.0.0.105
```

And the file `ibmachines` with contents:

```
10.0.1.101
10.0.1.102
10.0.1.103
10.0.1.104
10.0.1.105
```

To compile: `mpicc hello.c -o hello`
To run: `mpirun hello -pn 5 -machinefile machines`

Unlock the 32k memory limits by adding the following lines to `/etc/security/limits.conf`:

```
* soft memlock unlimited
* hard memlock unlimited
```

The asterisks are required.

Basic MPI Programs

Hello World

Create the file `hello.c` with the contents:

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
  int numprocs, rank, namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_size(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(processor_name, &namelen);

  printf("\n\nDane\'s Hello World.\nProcess %d on %s out of %d\n\n",
rank, processor_name, numprocs);

  MPI_Finalize();
}
```

Benchmarking

There are a number of valuable testing suites available to the user.

- OSU MPI tests - http://mvapich.cse.ohio-state.edu/benchmarks/ or these tests can be retrieved from the yum repository *mpitests** as mentioned earlier.
- Unix TestBench can be used for system profiling.
- Iperf TCP/IP network benchmarks – http://sourceforge.net/projects/iperf
- Stream Benchmark provides memory bandwidth benchmarks - http://www.cs.virginia.edu/stream/

It is important to benchmark each piece of the system to ensure that each piece of the cluster is performing as predicted. If one part of the system is dysfunctional, it will cause a bottleneck for the entire cluster.

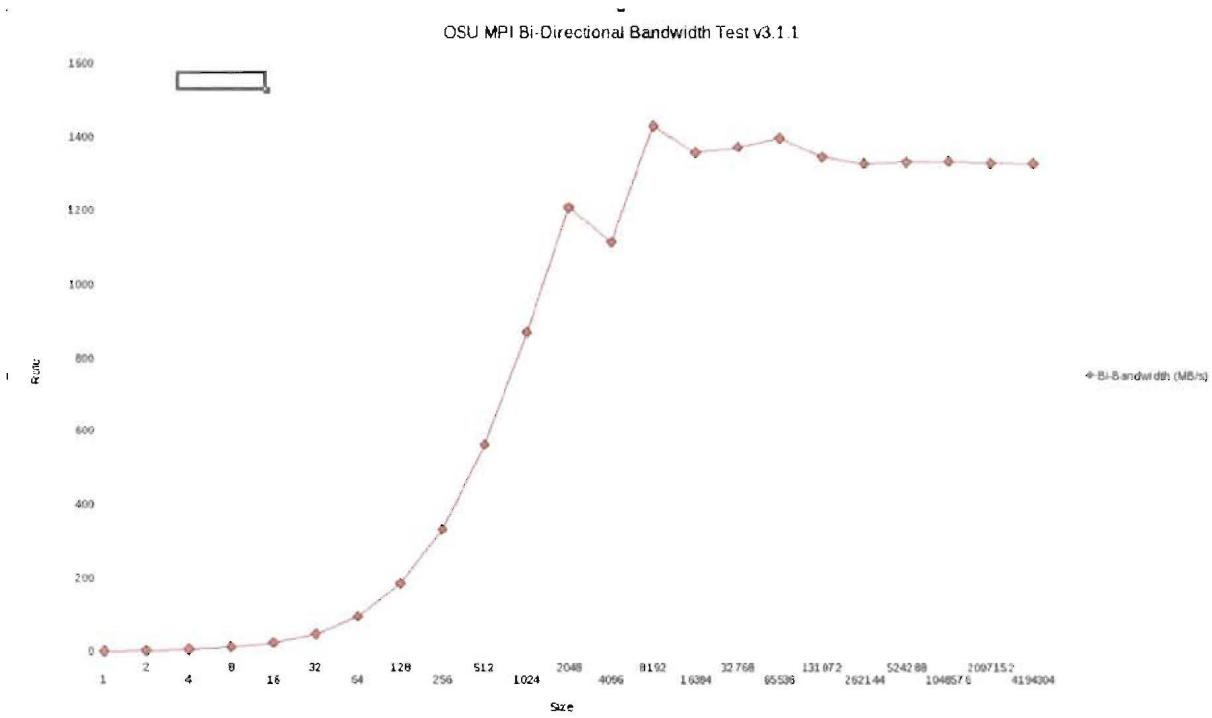


Figure 12. OSU Bidirectional Bandwidth Test

Torque/Maui Job Scheduler

←----------TODO, More description and options

Download and install Torque resource manager.

```
wget http://www.clusterresources.com/downloads/torque/torque-
2.4.0b1.tar.gz
tar xzf torque-2.4.0b1.tar.gz
cd torque-2.4.0b1/
./configure
make
make install
```

Ensure that the hostname for the server is properly configured by modifying the `/etc/hosts` file to include an entry for the head node attached to its ip other than the local 127.0.0.1 hostname.

Configure the resource manager

```
./torque/torque.setup root
```

Install the correct packages on the perceus vnfs image.

```
make packages
perceus vnfs mount centos2-5.3-1
cp ./torque-package* /mnt/centos2-5.3-1/scratch/
chroot /mnt/centos2-5.3-1/
./torque-package-clients-linux-x86_64.sh --install
./torque-package-devel-linux-x86_64.sh --install
./torque-package-doc-linux-x86_64.sh --install
./torque-package-mom-linux-x86_64.sh --install
./torque-package-server-linux-x86_64.sh --install
```

Create /var/spool/torque/mom_priv/config and add the line:

```
$usecp head.teamyellow.com:/home /home
```

Add the pbs_mom service to startup by modifying /etc/rc.d/rc.local to include the line.

```
/usr/local/sbin/pbs_mom
```

Restart the pbs_server daemon on the head node and check its status.

```
qterm
pbs_server
qstat -q
```

Verify the status of all nodes.

```
pbsnodes -a
```

Add the commands to the execution path of the head.

```
export PATH=$PATH:/usr/local/maui/bin
export PATH=$PATH:/usr/local/maui/sbin
```

Create the file pbs_mom. An example is given in the scripts section at the end of this document. Add the pbs_mom service to the compute nodes.

```
cp /scratch/pbs_mom /etc/init.d/
chkconfig --add pbs_mom
```

Save the Perceus capsule and restart all Nodes.

Download the Maui tarball from http://www.clusterresources.com/product/maui/.
Unpack and install.

```
tar xzf maui-3.2.6p21.tar.gz
cd maui-3.2.6p21/
./configure
make
make install
```

The Maui scheduler will replace the pbs_server provided by Torque so the pbs_server daemon must be exited. Start the Maui daemon and add it to services and startup and its ready to go.

Glossary

Compute Nodes - sub nodes of a cluster, being controlled by a head node. Usually a very lightweight OS install, allowing it to do a specialized job or task. Sometimes the OS is only loaded into memory from a network booting environment (PXE) eliminating the hard drive as a point of failure and speed bottlenecks.

Head Node - the controlling node of a cluster of nodes. Usually has everything installed needed to upkeep and run the compute nodes.

Infiniband - specialty networking hardware used to create a very high speed switched network 'fabric'. Current speeds are exceeding 40Gb/s -- as compared to 1Gb/s ethernet!

Infiniband Subnet - a set of ports and associated links with a common Subnet ID and managed by a common Subnet Manager.

Infiniband Subnet Manager - the entity that discovers all of the devices in a subnet at startup time, configures them, and then performs a periodic sweep to detect any changes to the subnet's topology.

Preboot Execution Environment (PXE) - Pronounced as 'pixie'. An environment to remotely boot computers using a network interface, without the use of physical data storage devices (CDs or hard drives) or an already installed operating system.

Remote Direct Memory Access (RDMA) - Allows data to be written into the memory of a remote computer without using the operating system. Very fast with low overhead, but can cause collisions.

References

Perceus - perceus.org

David Rowe, Lead Developer at ForensiT, Amateur cluster-builder's blog - onemansjourneyintolinux.blogspot.com

The XCPU project comprises of a suite of tools for cluster management. It includes utilities for spawning jobs, management of cluster resources, scalable distribution of boot images across a cluster as well as tools for creation and control of virtual machines in a cluster environment. xcpu.org

Robert G. Brown, Duke University Physics Dept.,
Introduction to the Beowulf Design - phy.duke.edu/~rgb/Beowulf/beowulf_intro_2003.php
Engineering Beowulf-style Compute Clusters - phy.duke.edu/~rgb/Beowulf/beowulf_book.php
The Beowulf Design - phy.duke.edu/~rgb/Beowulf/beowulf_advanced.php
Wulfware Cluster Monitoring Suite - phy.duke.edu/~rgb/Beowulf/wulfware.php

John Borwick, johnborwick.com/writing/system-automation.html

Syslinux documentation - syslinux.zytor.com/wiki/index.php/PXELINUX

Anaconda's Kickstart - fedoraproject.org/wiki/Anaconda/Kickstart

CentOS - centos.org

Numa, Memory, etc.
ftp://ftp.kernel.org/pub/linux/kernel/people/christoph/pmig/numamemory.pdf

What Every Programmer Should Know About Memory -
people.redhat.com/drepper/cpumemory.pdf

http://freshmeat.net/projects/numactl/

Scripts

Install Infiniband Drivers

```
#!/bin/bash
#installIB.sh
# --== Installs infiniband drivers and gets it all working ==-- #

if [ `whoami` != "root" ]; then #Ensure user is root
  echo '        !!! Use sudo !!!'
  exit 1
fi

#We're using the -sm flag to denote Subnet Manager nodes
[ $1 = "-sm" ] && SUBNET=1

if [ $1 = "--help" ]; then
  echo "        Installs the InfiniBand drivers and MPICH2"
  echo "        If you want to install as a Subnet Manager add the '-sm' flag"
  exit 1
fi

yum -y install openib libibverbs libmthca ibutils mvapich2 openmpi
if [ $STATUS -ne 0 ]; then                              #Make sure it worked
  echo "DOH!!! General install no workie."
  return $STATUS
fi

[ -z $SUBNET ] && yum -y install opensm                 #Install the Subnet Manager
if [ $STATUS -ne 0 ]; then                              #Make sure it worked
  echo "DOH!!!  Subnet Manager install no workie."
  return $STATUS
fi

/sbin/service openibd restart                           #Restart the IB service
/sbin/chkconfig openibd on                              #Ensure that IB is up when booted

[ -z $SUBNET ] && /sbin/service opensmd restart         #Restart the Subnet Manager
[ -z $SUBNET ] && /sbin/chkconfig openibd on            #Ensure the Subet Manager is up when
booted
# --== Setup the device ==-- #

#Grab the IP address from eth0 so we can reuse it for the IB
IPADDR=`/sbin/ifconfig eth0 | grep -Eo 10.0.0.1[0-9]{2} |grep -Eo 1[0-9]{2}`

#Create a /etc/sysconfig/network-scripts/ifcfg-ib0 file to setup the device
IFCFG="//etc/sysconfig/network-scripts/ifcfg-ib0"

if [ ! -f $IFCFG ]; then
  echo "# Infiniband Device ib0" >> $IFCFG
  echo "DEVICE=ib0" >> $IFCFG
  echo "TYPE=Infiniband" >> $IFCFG
  echo "BOOTPROTO=static" >> $IFCFG
  echo "BROADCAST=10.0.1.255" >> $IFCFG
  echo "IPADDR=10.0.1.$IPADDR" >> $IFCFG    # In reality, I copied a config file without this
  echo "NETMASK=255.255.255.0" >> $IFCFG    #   line, and merely appended this to it.
  echo "NETWORK=10.0.0.0" >> $IFCFG
  echo "ONBOOT=yes" >> $IFCFG
  echo "USERCTL=no" >> $IFCFG
  echo "IPV6INIT=no" >> $IFCFG
  echo "PEERDNS=yes" >> $IFCFG
fi

#Restart the InfiniBand device(s)
/sbin/ifdown ib0
/sbin/ifup ib0

exit 0
```

Node Status

```
#!/bin/bash
#nodestatus
echo "--======--"
for N in `seq 1 9`
do
  NAME=`nslookup node0$N | grep Name |grep -Eo [a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\.[a-zA-Z0-9-]\{2,6\}`
  ADDRESS=`nslookup node0$N |grep -E "Address\: 10.0.0.[0-9]+" |grep -Eo "10.0.0.[0-9]+"`

  if [ -z "$NAME" ]; then
    echo -e "\e[31mThere is no DNS record for node0$N\e[00m"
  else
    echo "$NAME ($ADDRESS)"

    ETHERNET=`/usr/sbin/fping node0$N |grep ^node |grep -El "alive" |wc -l`
    INFINIBAND=`/usr/sbin/fping ibnode0$N |grep ^ibnode |grep -El "alive" |wc -l`

    if [ $ETHERNET == "1" ]; then
      echo -e "\e[32mEthernet is alive\e[00m"
    else
      echo -e "\e[31mEthernet is dead\e[00m"
    fi

    if [ $INFINIBAND == "1" ]; then
      echo -e "\e[32mInfiniband is alive\e[00m"
    else
      echo -e "\e[31mInfiniband is dead\e[00m"
    fi
  fi

  echo -e "\e[00m"
done
echo "--======--"
exit 0
```

Run Command on All Nodes

```
#!/bin/bash
#runallnodes
for N in `seq 1 9`; do
        echo -e "\e[34;1m--== Node0$N ==--\e[00m"
        ssh node0$N $*
        #ssh 10.0.0.10$N $*
done
exit 0
```

PBS_MOM Initialization Script

```
#!/bin/sh
#pbs_mom
# $Id: pbs_mom.init 180 2003-03-07 20:38:36Z romano $
#
# chkconfig: - 70 40
# description: pbs_mom startup script
#
PBS_MOM=/usr/local/sbin/pbs_mom

. /etc/rc.d/init.d/functions

RETVAL=0

case "$1" in
   start)
      echo -n "Starting PBS_MOM: "
      [ -f $PBS_MOM ] || exit 1

      daemon $PBS_MOM
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && touch /var/lock/subsys/pbs_mom
        ;;

  stop)
      echo -n "Shutting down PBS_MOM: "
      killproc pbs_mom
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/pbs_mom
        ;;

  restart|reload)
        $0 stop
        $0 start
        RETVAL=$?
        ;;
  status)
        status pbs_mom
        RETVAL=$?
        ;;
  *)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
esac

exit $RETVAL
```

Kickstart

```
# To make installation interactive, remove comment on next line

# System authorization information
authconfig --enableshadow --enablemd5

# System bootloader configuration
bootloader --location=mbr --driveorder=sda --append=" console=tty0 console=ttyS0,115200n8"

# Clear the Master Boot Record
zerombr yes

# Clear all partition information and format new partitions
# Note: The anaconda autogenerated VolGroups etc, doesn't allow yum to run in
#       chroot in the post portion of the script, therefore, just KISS
clearpart --all --initlabel
part / --fstype ext3 --size 16384 --label=ROOT

# Run installation in text mode (most nodes don't have video cards anyway)
text

# See all install output
cmdline

# Set up a somewhat restrictive firewall (or disable)
firewall --disabled

# Install keyboard
keyboard us

# System language
lang en_US
langsupport en_US

# Installation logging level
logging info

# Use network installation from head node web server
url --url http://10.0.0.1/centos/os

# Setup network interfaces so that we can get to repositories
network --device eth0 --bootproto dhcp --onboot=on

# Root password
rootpw --iscrypted $1$.2187FQU$ackhP.45BWkW3.yfehfxP.

# No one will poke around anyway
selinux --disabled

# System timezone
timezone --utc America/Denver

# Perform installation (rather than update)
install

# Don't reboot automatically after installation - require user input
# Uncomment next line if you want the automatic reboot
reboot

# X Window System configuration information (not applicable to compute node)
# xconfig  --driver "nv" --depth 24 --resolution 1280x1024 --startxonboot
skipx

##############################################################################
###                    Pre install section of install script                ###
##############################################################################
%pre
#!/bin/bash
%end
```

```
################################################################################
###                Package install section of install script                 ###
################################################################################
# Packages to install section
%packages --ignoremissing
@base
@core
@text-internet
@admin-tools
@system-tools
@clustering
@cluster-storage

################################################################################
###                   Post install section of install script                 ###
################################################################################
# Post script section
%post

# Create temp folders
echo "Making Directories" >> /root/ks.log
mkdir /scratch
mkdir /configs

# Mount the physical hardrive so that installation affects proper kernel
echo "Mounting FileSystems" >>/root/ks.log
service portmap status >> portmap.statusbefore
service portmap restart
service --status-all >> all.status
mount -t nfs 10.0.0.1:/configs /configs>>/root/ks.log>&/root/ks.er
df >> /root/df.output
echo "moving folders" >>/root/ks.log

# Move LDAP configuration to it's new home
cp -f /configs/ldap.conf /etc/ldap.conf>&/root/ks.er
cat  /configs/fstab >> /etc/fstab
cp -f /configs/sudoers /etc/sudoers
cp -f /configs/nsswitch.conf /etc/nsswitch.conf

# Copy centos repository information
cp -f /configs/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo

# Setup NTP service
cp -f /configs/ntp.conf /etc/ntp.conf
echo "dealing with php keys" >>/root/ks.log

# Add static route to multicast channel
cp -f /configs/route-eth0 /etc/sysconfig/network-scripts/route-eth0

# Setup PGP keys
rm -rf /etc/pki/rpm-gpg/
cp -rf /configs/rpm-gpg/ /etc/pki/
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-*
mkdir /root/.ssh
cp -rf /configs/root/.ssh /root/

# Yum the necessary packages
yum install -y tftp
yum install -y ntp
yum install -y htop
ntpdate 10.0.0.1

# Start and Restart all necessary services
echo "Dealing with Services" >>/root/ks.log
chkconfig ntpd on
service ntpd restart
chkconfig portmap on
service portmap restart
service nfs restart
chkconfig nfs on
service sshd restart
```

```
chkconfig sshd on
echo "installing tftp"
chkconfig yum-updatesd off
chkconfig bluetooth off

# Yum the necessary packages
rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
yum install -y ganglia-gmond
chkconfig gmond on
cp -f /configs/gmond.conf /etc/gmond.conf

# Update OS
yum update -y
updatedb
# Finish up and report to install.log
echo "All done" >> /root/ks.log
mount -a
echo "mounted" >> /root/ks.log
df >> /root/ks.log

############## KILL THE PXELINUX.CFG ##########
# Kill the PXElinux.cfg file for this IP address -- it's already been installed!!!
IPADDR=`ifconfig eth0 |grep -o 10.0.0.1[0-9]*`
IPHEX=`gethostip -x $IPADDR`
touch emptyFile
tftp 10.0.0.1 -c put emptyFile pxelinux.cfg/$IPHEX
rm -f emptyFile
############## END KILL THE PXELINUX.CFG ##########


#stuff from andreee that might be needed

# Set up serial port communication for compute nodes
#echo "Setting up serial port configurations for node" >> $LOGFILE
#echo 's0:2345:respawn:/sbin/agetty ttyS0 115200 vt100' >> /etc/inittab
#echo -n "Added to /etc/inittab: " >> $LOGFILE
#tail -1 /etc/inittab >> $LOGFILE 2>&1
echo "Removing X from inittab" >> $LOGFILE 2>&1
perl -pi.bak -e 's/^x:/\#x:/g' /etc/inittab >> $LOGFILE 2>&1

echo "Making sure grub works" >> $LOGFILE
perl -pi.bak -e 's/--speed=115200/--speed=115200 --word=8 --parity=no --stop=1/'
/boot/grub/grub.conf
perl -pi.bak -e 's/serial console/serial/' /boot/grub/grub.conf
perl -pi.bak -e 's/splashimage/\#splashimage/' /boot/grub/grub.conf
perl -pi.bak -e 's/ rhgb quiet/ console=tty0 console=ttyS0,115200n8/' /boot/grub/grub.conf

#######  INFINIBAND ###########
# Installs infiniband drivers and gets it all working
yum -y install openib libibverbs libmthca ibutils mvapich2 openmpi
/sbin/service openibd restart        #Restart the IB service
/sbin/chkconfig openibd on        #Ensure that IB is up when booted

# Create the /etc/sysconfig/network-scripts/ifcfg-ib0 file to setup the device
IPADDR=`/sbin/ifconfig eth0 | grep -Eo 10.0.0.1[0-9]{2} |grep -Eo 1[0-9]{2}`
IFCFG="etc/sysconfig/network-scripts/ifcfg-ib0"
cp /configs/$IFCFG /$IFCFG
if [ -f $IFCFG ]; then
  echo "IPADDR=10.0.1.$IPADDR" >> $IFCFG
fi

# Restart the InfiniBand device
/sbin/ifdown ib0
/sbin/ifup ib0
########## END INFINIBAND ##########

########## ROOT'S PASSWORDLESS SSH ##########
# We don't want /root/ on the Network File System
#   so we have to copy the authorized keys over manually
cp /configs/root/.ssh/authorized_keys /root/.ssh/
chmod 600 /root/.ssh/authorized_keys
```

```
# Tired of messing with the ~/.ssh/known_hosts file?
#   We're copying the host keys over during install,
#   so they always have the same one
cp /configs/etc/ssh/ssh_host_* /etc/ssh/
chmod 600 /etc/ssh/ssh_host_key
chmod 600 /etc/ssh/ssh_host_dsa_key
chmod 600 /etc/ssh/ssh_host_rsa_key
########## END ROOT'S PASSWORDLESS SSH ##########

%end
```