[HPC Cluster Install: Admin and Kickstart Server](http://idolinux.blogspot.com/2010/05/hpc-cluster-install-admin-and-kickstart.html)

Welcome to the third installment of the series on installing an HPC cluster. Here we will dive into the build log, referring mostly to software systems configuration. Check out the prior two posts for some background on hardware and prerequisites:

* [HPC Cluster Install: Intro](http://idolinux.blogspot.com/2009/10/hpc-cluster-install-intro.html)
* [HPC Cluster Install: Planning and Purchasing](http://idolinux.blogspot.com/2010/04/hpc-cluster-install-planning-and.html)

First, a note on documentation... I have to expect my reader to be reasonably proficient with Linux systems to find this series of any use. If something is unclear, please comment bellow or email me. Most of the commands here can be clarified by reading the relevant man page, or diving into the [RHEL](http://www.redhat.com/docs/manuals/enterprise/#RHEL5) or [CentOS](http://www.centos.org/docs/5/) documentation. I frequently refer to my copies. In particular, the [Installation Guide](http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.5/html/Installation_Guide/index.html), and the section on [Kickstart Installations](http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.5/html/Installation_Guide/ch-kickstart2.html), should be considered required reading.

A note on style... There are two types of code in my blog entries with two different box styles:

# this is a command at the root prompt

this is output from the command

# this is a comment in a config file

this is an active line in a config file

So, let's get to it. At this point, there is a full rack of equipment before us, neatly wired and plugged into power, housed in our hosting facility. We'll call the whole system Fiji, a short and pleasant name to type. The first step, before we go to the server room, is to download and burn [a CentOS install DVD for 64-bit](http://isoredirect.centos.org/centos/5/isos/x86_64/), currently version 5.4. Once the install media is in-hand, it is time to install the admin node, which will act as a kickstart server to install all the compute/exec nodes.

On this project, I decided to have a separate head node for the users to log into, and I combine the storage and admin node. I believe this is a reasonable configuration for a small to medium cluster. Large clusters, beyond a hundred nodes, may require segmenting systems into logical groups or blocks, to eliminate hot spots. This walk-through could be extended, as a guide to one building block that could be replicated N times, for a large cluster. Here, we are looking at only one full rack of equipment.

The admin/stor node and the head/login node both have multiple ethernet interfaces. The first interface (eth0) connects to the private, internal gigabit switch and 192.168.1.0 network, where all internal communication between nodes and management interfaces will occur. The second ethernet interface (eth1) will connect to the public network, for user and admin login. This is important to note, because the DRAC/BMC management board interface is configured to share the first ethernet port, meaning you will see TWO different MAC addresses and TWO different IPs on this port, both we want facing internally.

Hook up the KVM to the admin node and power on. Check BIOS settings, setting a password and choosing the boot order. Then configure the DRAC/BMC, setting an admin username with password, and configuring an internal IP for the device.

Everything is redundant on the admin and head node, including the root drive which will hold the OS. We configure RAID 1 mirror for the internal root drive through the RAID BIOS. There is also a large external data storage enclosure of 10TB, which we configure as RAID 6 plus one hot spare. I recommend letting the storage fully initialize before continuing. After full RAID initialization, power off the node and unplug the external RAID enclosure before the OS install, as the partitioner doesn't like very big disks.

Now insert the CentOS installation DVD. Boot the installer and select mostly all defaults. The only things we will do differently here are setting a bootloader password, configuring the two network interfaces, and switching selinux to permissive mode. We call the admin/storage node "fijistor" with internal IP 192.168.1.2. For the package selection, we do not change the defaults. Note that the "cluster" packages do not refer to HPC clustering in this context.

Once the Anaconda installer completes, shutdown the system, hook the external RAID back up, remove booting from DVD in the BIOS and power on. Admire your fresh login screen. Please note that most of this build log is accomplished via remote login through SSH. The only time we need to be in the server room is when messing with the BIOS/BMC and powering the nodes for the first time. Additionally, I use the *screen* package for persistent shells with ssh, but the awesomeness of *screen* is beyond this article…

Since ssh is our main entry to this system, which we have named Fiji, we need to lock sshd down. I always install and test my personal ssh key for the root user, and then limit an admin group to login on the admin node with keys only.

Edit /etc/ssh/sshd\_config and insert:

PermitRootLogin without-password

AllowGroups fijiadmin

Add the admin group and restart sshd:

# groupadd fijiadmin

# usermod -G fijiadmin root

# service sshd restart

We want to generate an ssh key for the fijistor root account, which we will then allow to all exec/compute nodes.

# ssh-keygen # just hit enter, no passphrase

# cp /root/.ssh/id\_rsa.pub /root/.ssh/authorized\_keys.fijistor

# chmod -R go= /root/.ssh

Since we will not be using the console GUI, we can disable the graphical login by commenting out the relevant line in the /etc/inittab file. This will free up some memory.

# Run xdm in runlevel 5

#x:5:respawn:/etc/X11/prefdm -nodaemon

Do your first full system package update, install the development tools and then reboot into the new kernel.

# yum -y update

# yum -y install @development-tools

# reboot

It is important to keep logs, so let's extend that in /etc/logrotate.conf

# keep 4 weeks worth of backlogs

#rotate 4

rotate 999

We may enable network forwarding on the admin node, so that all the exec nodes we are about to install can resolve public services when needed. Add the following to /etc/sysctl.conf

net.ipv4.ip\_forward = 1

And we must tweak our iptables firewall settings, or else nothing will get through. This is mostly a default iptables config with a few tweaks. The speed at which ssh may be attempted is limited, to curtail ssh brute forcing. Everything from the internal cluster network is allowed. Dropped packets are log for debug purposes, which can be commented out and disabled later. And all outbound traffic from the internal network is put through NAT. /etc/sysconfig/iptables

# Firewall configuration written by system-config-securitylevel

# Manual customization of this file is not recommended.

\*filter

:INPUT ACCEPT [0:0]

:FORWARD ACCEPT [0:0]

:OUTPUT ACCEPT [0:0]

:RH-Firewall-1-INPUT - [0:0]

-A INPUT -j RH-Firewall-1-INPUT

-A FORWARD -j RH-Firewall-1-INPUT

-A RH-Firewall-1-INPUT -i lo -j ACCEPT

-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT

-A RH-Firewall-1-INPUT -p 50 -j ACCEPT

-A RH-Firewall-1-INPUT -p 51 -j ACCEPT

-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT

#-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT

#-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT

-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

### begin ssh

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

-A RH-Firewall-1-INPUT -p tcp --dport 22 --syn -s 192.168.1.0/255.255.255.0 -j ACCEPT

-A RH-Firewall-1-INPUT -p tcp --dport 22 --syn -m limit --limit 6/m --limit-burst 5 -j ACCEPT

### end ssh

### begin cluster

-A RH-Firewall-1-INPUT -i eth0 -s 192.168.1.0/255.255.255.0 -j ACCEPT

-A RH-Firewall-1-INPUT -i eth0 -d 255.255.255.255 -j ACCEPT

-A RH-Firewall-1-INPUT -i eth0 -p udp --dport 67:68 -j ACCEPT

-A RH-Firewall-1-INPUT -i eth0 -p tcp --dport 67:68 -j ACCEPT

### end cluster

### begin log

-A RH-Firewall-1-INPUT -m limit --limit 10/second -j LOG

### end log

-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited

COMMIT

### begin nat

\*nat

:PREROUTING ACCEPT [0:0]

:POSTROUTING ACCEPT [0:0]

:OUTPUT ACCEPT [0:0]

-A POSTROUTING -o eth1 -j MASQUERADE

#-A PREROUTING -i eth1 -p tcp --dport 2222 -j DNAT --to 192.168.1.200:22

COMMIT

### end nat

Now we reload our kernel sysctl parameters and restart the firewall:

# sysctl -p /etc/sysctl.conf

# service iptables restart

Now to format the external RAID storage on /dev/sda. Notice that we overwrite the pre-existing partition info on the device and then use LVM directly. I also specify the metadatasize to account for 128k alignment. The pvs command will reveal that requesting 250k will actually result in 256k being used.

# cat /proc/partitions

# dd if=/dev/urandom of=/dev/sda bs=512 count=64

# pvcreate --metadatasize 250k /dev/sda

# pvs -o pe\_start

# vgcreate RaidVolGroup00 /dev/sda

# lvcreate --extents 100%VG --name RaidLogVol00 RaidVolGroup00

# mkfs -t ext3 -E stride=32 -m 0 -O dir\_index,filetype,has\_journal,sparse\_super /dev/RaidVolGroup00/RaidLogVol00

# echo "/dev/RaidVolGroup00/RaidLogVol00 /data0 ext3 noatime 0 0" >>/etc/fstab

# mkdir /data0 ; mount /data0 ; df -h

With our storage ready, we are ready to build a file repository which will feed all system yum updates and node kickstart installs. Here we create a directory structure with the base installer packages from the DVD, and then we pull all of the latest updates from a local file mirror. I strongly urge you to [find your own local mirror that provides rsync](http://www.centos.org/modules/tinycontent/index.php?id=30).

# mkdir -p /data0/repo/CentOS/5.4/iso/x86\_64

# ln -s /data0/repo /repo

# cd /repo/CentOS

# ln -s 5.4 5

# wget http://mirrors.gigenet.com/centos/RPM-GPG-KEY-CentOS-5

# cd /repo/CentOS/5.4/iso/x86\_64

# cat /dev/dvd > CentOS-5.4-x86\_64-bin-DVD.iso

# wget http://mirror.nic.uoregon.edu/centos/5.4/isos/x86\_64/sha1sum.txt

# sha1sum -c sha1sum.txt # re-download or torrent

# mount -o loop CentOS-5.4-x86\_64-bin-DVD.iso /mnt

# mkdir -p /repo/CentOS/5.4/os/x86\_64

# rsync -avP /mnt/CentOS /mnt/repodata /repo/CentOS/5.4/os/

# mkdir -p /repo/CentOS/5.4/updates/x86\_64

# rsync --exclude='debug' --exclude='\*debuginfo\*' --exclude='repoview' --exclude='headers' -irtCO --delete-excluded --delete rsync://rsync.gtlib.gatech.edu/centos/5.4/updates/x86\_64 /repo/CentOS/5.4/updates/

# mkdir -p /repo/epel/5/x86\_64

# cd /repo/epel

# wget http://download.fedora.redhat.com/pub/epel/RPM-GPG-KEY-EPEL

# rsync --exclude='debug' --exclude='\*debuginfo\*' --exclude='repoview' --exclude='headers' -irtCO --delete-excluded --delete rsync://archive.linux.duke.edu/fedora-epel/5/x86\_64 /repo/epel/5/

# mv /etc/yum.repos.d/\* /usr/src/

# cat /dev/null >/etc/yum.repos.d/CentOS-Base.repo

# cat /dev/null >/etc/yum.repos.d/CentOS-Media.repo

Create repo config for yum in /etc/yum.repos.d/fiji.repo

# CentOS base from installation media

[base]

name=CentOS-$releasever - Base

#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os

#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/

baseurl=file:///repo/CentOS/$releasever/os/$basearch/

gpgcheck=1

gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-5

protect=1

# CentOS updates via rsync mirror

[update]

name=CentOS-$releasever - Updates

#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=updates

#baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/

baseurl=file:///repo/CentOS/$releasever/updates/$basearch/

gpgcheck=1

gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-5

protect=1

# Extra Packages for Enterprise Linux (EPEL)

[epel]

name=Extra Packages for Enterprise Linux 5 - $basearch

#baseurl=http://download.fedora.redhat.com/pub/epel/5/$basearch

#mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=epel-5&arch=$basearch

baseurl=file:///repo/epel/$releasever/$basearch

enabled=1

protect=0

failovermethod=priority

gpgcheck=1

gpgkey=http://download.fedora.redhat.com/pub/epel/RPM-GPG-KEY-EPEL

Let's create a script that we can cron to update our file repo:

# mkdir /root/bin

# touch /root/bin/update\_repo.sh

# chmod +x /root/bin/update\_repo.sh

/root/bin/update\_repo.sh

#!/bin/bash

OPTS='-vrtCO --delete --delete-excluded --exclude=i386\* --exclude=debug --exclude=\*debuginfo\* --exclude=repoview --exclude=headers'

rsync $OPTS rsync://rsync.gtlib.gatech.edu/centos/5.4/updates/x86\_64 /repo/CentOS/5.4/updates/

rsync $OPTS rsync://archive.linux.duke.edu/fedora-epel/5/x86\_64 /repo/epel/5/

Now that we have a file repository for all our RPMs, let's share it out via NFS, along with /data0 and the /usr/global/ directory. On this system, user home dirs happen to be in the /data0/home/ directory. I use /usr/global/ just like /usr/local/ except it is mounted on all nodes via nfs. The global.sh script will be symlinked on all nodes to /etc/profile.d/global.sh to set user environment variables.

# mkdir -p /usr/global/etc/profile.d/

# touch /usr/global/etc/profile.d/global.sh

# chmod +x /usr/global/etc/profile.d/global.sh

Example /usr/global/etc/profile.d/global.sh file:

# Sun Grid Engine

export SGE\_ROOT=/usr/global/sge

. /usr/global/sge/default/common/settings.sh

alias rsh='ssh'

alias qstat='qstat -u "\*"'

# Intel compilers

. /usr/global/intel/Compiler/11.1/064/bin/iccvars.sh intel64

. /usr/global/intel/Compiler/11.1/064/bin/ifortvars.sh intel64

. /usr/global/intel/Compiler/11.1/064/mkl/tools/environment/mklvars64.sh

export INTEL\_LICENSE\_FILE=/usr/global/intel/licenses:$INTEL\_LICENSE\_FILE

/etc/exports

/data0 fiji(rw,async,no\_root\_squash) 192.168.1.0/255.255.255.0(rw,async,no\_root\_squash)

/usr/global 192.168.1.0/255.255.255.0(rw,async,no\_root\_squash)

/kickstart 192.168.1.0/255.255.255.0(ro)

/repo 192.168.1.0/255.255.255.0(ro)

# chkconfig nfs on

# service nfs start

Enable remote syslog logging from nodes by adding the following options in /etc/sysconfig/syslog

SYSLOGD\_OPTIONS="-m 0 -r -s fiji.baz.edu"

# service syslog restart

List all of your hosts in /etc/hosts

127.0.0.1 localhost.localdomain localhost

::1 localhost6.localdomain6 localhost6

192.168.1.1 fiji.baz.edu fiji

192.168.1.2 fiji.baz.edu fijistor

192.168.100.9 ib

192.168.1.101 node01

192.168.1.102 node02

192.168.1.103 node03

...snip

192.168.1.200 fiji-bmc

192.168.1.201 node01-bmc

192.168.1.202 node02-bmc

192.168.1.203 node03-bmc

...snip

Enable network time syncronization

# ntpdate -u -b -s 1.centos.pool.ntp.org

# hwclock --utc --systohc

# chkconfig ntpd on ; service ntpd start

Now let's install all the services we need for a kickstart server and build a file structure for network booting via PXE, TFTP and NFS. This will allow a new exec node to boot via the network in a similar manner that it would boot from CD/DVD. Build the TFTP file structure for network PXE boot:

# yum install dhcp xinetd tftp tftp-server syslinux

# mkdir -p /usr/global/tftpboot ; ln -s /usr/global/tftpboot /tftpboot

# mkdir -p /tftpboot/pxelinux.cfg /tftpboot/images/centos/x86\_64/5.4

# cd /tftpboot/images/centos/x86\_64/ ; ln -s 5.4 5

# rsync -avP /mnt/isolinux/initrd.img /mnt/isolinux/vmlinuz /tftpboot/images/centos/x86\_64/5.4/

# cd /usr/lib/syslinux

# rsync -avP chain.c32 mboot.c32 memdisk menu.c32 pxelinux.0 /tftpboot/

# mkdir -p /usr/global/kickstart ; ln -s /usr/global/kickstart /kickstart ; cd /kickstart

# mkdir -p /kickstart/fiji/etc ; cd /kickstart/fiji/etc

# mkdir -p rc.d/init.d profile.d ssh yum/pluginconf.d yum.repos.d

# touch rescue.cfg ks-fiji.cfg ; ln -s ks-fiji.cfg ks.cfg

Edit our PXE menu in the /tftpboot/pxelinux.cfg/default file:

DEFAULT menu.c32

PROMPT 0

TIMEOUT 100

#ONTIMEOUT local

ONTIMEOUT centos

NOESCAPE 1

ALLOWOPTIONS 0

MENU TITLE Fiji PXE Menu

LABEL local

MENU LABEL Boot local hard drive

LOCALBOOT 0

LABEL centos

MENU LABEL CentOS 5 Fiji Node Install

KERNEL images/centos/x86\_64/5/vmlinuz

APPEND ks=nfs:192.168.1.2:/kickstart/ks.cfg initrd=images/centos/x86\_64/5/initrd.img ramdisk\_size=100000 ksdevice=eth0 ip=dhcp

LABEL rescue

MENU PASSWD $4$XXXXXX

MENU LABEL CentOS 5 Rescue

KERNEL images/centos/x86\_64/5/vmlinuz

APPEND initrd=images/centos/x86\_64/5/initrd.img ramdisk\_size=10000 text rescue ks=nfs:192.168.1.2:/kickstart/rescue.cfg

Please generate your own password to replace the above stub. SHA-1 encrypted passwords start with "$4$". Here is an example of encrypting the password "password":

$ sha1pass password

$4$gS+7mITP$y3s1L4Z+5Udp2vlZHChNXd8lhAg$

Copy in some files we will want to sync to all nodes:

# cp /etc/hosts /kickstart/fiji/etc/

# cp /root/.ssh/id\_rsa.pub /kickstart/fiji/authorized\_keys

# cp /etc/yum.repos.d/\*.repo /kickstart/fiji/etc/yum.repos.d/

Edit /kickstart/fiji/etc/ntp.conf and change the server to the admin node:

#server 0.centos.pool.ntp.org

#server 1.centos.pool.ntp.org

#server 2.centos.pool.ntp.org

server 192.168.1.2

When you get to booting your exec nodes, you will be able to paste their MAC addresses into /etc/dhcpd.conf and then restart the dhcpd process. Then the node will boot off the network with your chosen IP. This is a bit of a tedious process, which I'm sure could be automated with more advanced management techniques. Any suggestions? /etc/dhcpd.conf example:

ddns-update-style interim;

ignore client-updates;

option option-128 code 128 = string;

option option-129 code 129 = text;

subnet 192.168.1.0 netmask 255.255.255.0 {

option routers 192.168.1.2;

option subnet-mask 255.255.255.0;

option nis-domain "fiji.baz.edu";

option domain-name "fiji.baz.edu";

option domain-name-servers 123.123.123.123;

option time-offset -18000; # Eastern

option ntp-servers 192.168.1.2;

default-lease-time 21600;

max-lease-time 43200;

allow booting;

allow bootp;

next-server 192.168.1.2;

filename "/pxelinux.0";

host node01 {

hardware ethernet 00:11:22:33:44:a0;

fixed-address 192.168.1.101;

}

host node01-bmc {

hardware ethernet 00:11:22:33:44:8d;

fixed-address 192.168.1.201;

}

host node02 {

hardware ethernet 00:11:22:33:44:83;

fixed-address 192.168.1.102;

}

host node02-bmc {

hardware ethernet 00:11:22:33:44:61;

fixed-address 192.168.1.202;

}

...snip

In /etc/xinetd.d/tftp we change disabled to no:

service tftp

{

socket\_type = dgram

protocol = udp

wait = yes

user = root

server = /usr/sbin/in.tftpd

server\_args = -s -v /tftpboot

disable = no

per\_source = 11

cps = 100 2

flags = IPv4

}

Now we load the newly configured services:

# service xinetd restart

# chkconfig dhcpd on

# service dhcpd restart

With a complete PXE file structure, we can get to the major step of preparing kickstart configuration files. Kickstart configs are flat text files that specify all of the options the installer can accept, along with all of the pre- and post-install scripting we care to add, to get our nodes installed exactly as we want. Here is a very simple kickstart config that only boots into the rescue mode of the installer. /kickstart/rescue.cfg

lang en\_US

keyboard us

mouse none

nfs --server=192.168.1.2 --dir=/repo/CentOS/5/iso/x86\_64

network --bootproto=dhcp

And here is the exec/compute node kickstart file. I have gathered this from multiple cluster installs over the years. I will typically paste in an old kickstart config I already have, and then test install the first exec node. Then I will edit the kickstart file again and again, making adjustments, completely re-install the first exec node until it is perfect. After the first exec node is perfect, all we have to do is power on each additional node and it will install exactly the same. /kickstart/ks.cfg

# boot: linux ks=nfs:192.168.1.2:/kickstart/ks.cfg

### Use NFS installation media

nfs --server=192.168.1.2 --dir=/repo/CentOS/5/iso/x86\_64

### Repository for current OS updates

repo --name=updates --baseurl=file:///repo/CentOS/5/updates/x86\_64/

repo --name=epel --baseurl=file:///repo/epel/5/x86\_64/

### System authorization information

auth --useshadow --enablemd5

rootpw --iscrypted $1$XXXXXXXX

### System bootloader configuration

bootloader --location=mbr --md5pass=$1$XXXXXXXX

### Use graphical install

#graphical

text

#vnc --port=10001 --password=kickme

### Firewall configuration

firewall --disabled

### Run the Setup Agent on first boot

firstboot --disable

### System keyboard

keyboard us

### System language

lang en\_US

### Installation logging level

logging --level=info --host=192.168.1.2

### Network information

network --bootproto=dhcp --device=eth0 --onboot=on

### Reboot after installation

reboot

### SELinux configuration

selinux --permissive

### Disable some services

#services --disabled bluetooth,firstboot

### System timezone

timezone --utc America/New\_York

### Install OS instead of upgrade

install

### X Window System configuration information

#xconfig --defaultdesktop=GNOME --depth=32 --resolution=1280x1024 --startxonboot

#xconfig --startxonboot

# partition

zerombr

clearpart --all --initlabel

part /boot --fstype ext3 --size=100

part pv.0 --size=1 --grow

volgroup VolGroup00 pv.0

logvol swap --fstype swap --name=swap --vgname=VolGroup00 --size=2048

logvol / --fstype ext3 --name=root --vgname=VolGroup00 --size=10240 --grow

%pre

#!/bin/sh

### Mount repo for current OS updates

mkdir /repo

mount -t nfs -o ro,nolock,hard,udp,vers=3,rsize=32768,wsize=32768 192.168.1.2:/repo /repo

%post

#!/bin/sh

### Mount configuration dir

mkdir /kickstart

mount -t nfs -o ro,nolock,hard,udp,vers=3,rsize=32768,wsize=32768 192.168.1.2:/kickstart /kickstart

### Mount repo for current OS updates

mkdir /repo

mount -t nfs -o ro,nolock,hard,udp,vers=3,rsize=32768,wsize=32768 192.168.1.2:/repo /repo

echo "fijistor:/repo /repo nfs noatime,rsize=8192,wsize=8192,intr 0 0" >>/etc/fstab

rm -f /etc/yum.repos.d/\*

cp -vf /kickstart/fiji/etc/yum.repos.d/fiji.repo /etc/yum.repos.d/

cp -vf /kickstart/fiji/etc/yum/pluginconf.d/fastestmirror.conf /etc/yum/pluginconf.d/

yum -y update

cp -vf /kickstart/fiji/etc/hosts /etc/

### mount home & data & global

mkdir /data0 /data1 /usr/global

cat << xxEOFxx >> /etc/fstab

fijistor:/data0 /data0 nfs noatime,rsize=8192,wsize=8192,intr 0 0

fijistor:/data1 /data1 nfs noatime,rsize=8192,wsize=8192,intr 0 0

fijistor:/usr/global /usr/global nfs noatime,rsize=8192,wsize=8192,intr 0 0

xxEOFxx

### Disable some unused network services

/sbin/chkconfig --level 2345 firstboot off

### PC smart card daemon annoyance

/sbin/chkconfig pcscd off

touch /var/run/pcscd.pub

### Enable network time sync and set hardware clock

cp -vf /kickstart/fiji/etc/ntp.conf /etc/

/sbin/chkconfig --level 2345 ntpd on

/usr/sbin/ntpdate -u -b -s 192.168.1.2

/sbin/hwclock --utc --systohc

### Enable root SSH auto-login

mkdir /root/.ssh

cp -vf /kickstart/fiji/authorized\_keys /root/.ssh/

### Updates on boot & daily

chkconfig yum-updatesd off

#chkconfig yum-updateonboot off

chkconfig yum-cron on

# selinux

chkconfig setroubleshoot off

# disable some services

chkconfig cups off

chkconfig sendmail off

chkconfig bluetooth off

# SGE

ln -s /usr/global/etc/profile.d/global.sh /etc/profile.d/

adduser -u 186 sgeadmin

mkdir /var/spool/sge

chown sgeadmin.sgeadmin /var/spool/sge

# Ganglia

yum -y install ganglia-gmond

cp /kickstart/fiji/etc/gmond.conf /etc/

chkconfig gmond on

### Unmount conf dir

echo ".o0o.o0o.o0o. THATS ALL .o0o.o0o.o0o."

umount /kickstart

%packages

@editors

@core

@base

@legacy-software-support

@base-x

@development-tools

device-mapper-multipath

-sysreport

setroubleshoot

yum-protectbase

-yum-fastestmirror

yum-cron

compat-libstdc++-33

xterm

libXp

openmotif

openmotif22

ganglia-gmond

monit

libstdc++

libX11

Please generate and replace the above dummy password stubs with your own password. Encrypted passwords in MD5 format start with "$1$". Here is an example encryption for the password "password":

$ grub-md5-crypt

Password:

Retype password:

$1$d6oPa/$iUemCR50qSyvGSVTX9NrX1

Exec node install procedure:

* cable kvm & hit power button
* F12 for PXE boot
* ctrl-s get system ethernet MAC address
* ctrl-e to get BMC ethernet MAC address  
  + set LAN Parameters: IPv4 IP Address Source: DHCP
  + set LAN User Configiuration: enter & confirm password
  + hit Esc: Save Changes and Exit
* add MACs to fijistor /etc/dhcpd.conf and 'service dhcpd restart'
* hit enter on Fiji Node Install PXE menu
* wait for package Dependency Check
* next node

The head node install is just a standard install from DVD with mostly default options. This could also be made into a kickstart.

* config bmc ipmi
* vanilla install
* update to cluster yum repo config
* create user accounts
* edit /etc/sysconfig/iptables
* add and mount nfs shares in /etc/fstab

Here are a few required software packages I also install to obtain a fully functional cluster software stack. I typically install these after all nodes have been installed.

* [Deploying Sun Grid Engine on a Cluster](http://idolinux.blogspot.com/2008/09/deploying-sun-grid-engine-on-cluster.html)
* [Ganglia Cluster Monitoring Made Easy](http://idolinux.blogspot.com/2009/03/ganglia-cluster-monitoring-made-easy.html)
* clustered ssh with dsh dancers shell

You can use IPMI to control the DRAC/BMC, remotely power cycling locked nodes if needed. With Dell R410 nodes, something like the following will do the job:

# yum -y install OpenIPMI-tools

# ipmitool -H 192.168.1.2XX -U root -P PASSWORD -I lanplus chassis status

# ipmitool -H 192.168.1.2XX -U root -P PASSWORD -I lanplus chassis power cycle