Cluster Administration

Processes and files for administering the RFPK computational cluster are presented.

Table of Contents

Introduction **1**
Hardware **1**
Software **3**
Operations **10**

Introduction

Among RFPK's resources is a computational cluster which was purchased from Linux Labs at the end of the year 2001. With the OpenMosix enhancement to the Linux operating system, the cluster behaves towards software very much as would a distributed memory multiprocessor system, except that interprocess communication is accomplished via Ethernet rather than memory and, therefore, is many orders of magnitude slower.

The cluster consists of five individual computers, interconnected by Ethernet. The fact that each node is an individual machine increases the burden of system administration. This document describes the configuration both of hardware and software so that it can be restored in case it is destroyed or degraded. It also provides some processes for operating the cluster.

Hardware

Node Description

Each node of the cluster contains a Matsonic MS7308E mainboard, with a single 1GHz Pentium III processor, Ethernet, video, keyboard, mouse and IDE disk interfaces.

One of the nodes has a label on the front of its case that identifies it as master. This node differs from the others in several ways:

- There is a combination drive, which contains both a floppy drive and a CDROM drive. The other nodes have DVD drives capable of reading CDROMs, but no floppy.
- There are three PCI sockets, rather than the two which are found on the other boards. In the socket not found on the other boards, there is a second Ethernet interface. This interface is used to communicate with the other nodes, while the interface on the mainboard is used to communicate with the Internet.

  Because it can communicate with the Internet and because, with the OpenMosix software currently in use, no node is dominant over the others, *gateway* would be a better name than *master*. Nevertheless, we will continue to refer to this node as master, in order to avoid confusion.

The other nodes are named node0,...,node3. They are all the same, with the exception of node1, in which a defective Ethernet interface on the mainboard has been replaced with an Ethernet card with a bracket customized so that it will fit into one of the IDE sockets. These nodes have DVD drives, capable of reading CDROMs.

Node Interconnection

The five nodes are interconnected through an Ethernet hub, using category-5 Ethernet patch cables rated for at least 200 Mhz. The hub has exactly five cables connected to it, one for each node. For ease of management, connect master to socket-1, node0 to socket-2, etc.

The cable to master and the cable to node1 are connected to sockets on the back of their respective cases which are located to the far left when viewed from the front. The other nodes are connected to sockets that are two thirds of the way to the right, as viewed from the front.

Network Connection

The cluster is connected to the network via master. The cable to the network hub (not the hub described in the previous section) connects to the mainboard Ethernet interface, which is on the back of the case, two thirds of the way to the right, as viewed from the front.

Keyboard, Video and Mouse

On each node, the video connector is near the center of the back of the case. The keyboard and mouse connectors are just to the right of the mainboard Ethernet socket, as viewed from the front.

BIOS

The BIOS settings are the same for all nodes. One of the quirks of these systems is that it can be difficult to get to the BIOS configuration screens. At the beginning of the boot sequence, the BIOS will instruct you to press **DEL** if you want to enter BIOS configuration. Often you will be taken directly to the Grub boot screen even though you have pressed **DEL**. If this happens, press **CTRL-ALT-DEL** to reboot, then continue to press **DEL** several times a second. This usually gets the attention of the BIOS.

Small configuration changes need to be made at each of the following BIOS menu categories on all nodes:

- Standard CMOS Setup

  Use the *Auto* discovery capability to set the IDE devices correctly.

  For *Floppy Drive A*, select the value: *Not Installed*.

- Advanced Setup

  For *1st Boot Device*, select the value: *CDROM*.

- Power Management Setup

  For *Power Management/APM*, select the value: *Disabled*.

Software

Linux Installation

Initial Installation

Using CDROMs install on each node the same version of RedHat Linux that is currently installed on the Software Team workstations. Select the *workstation* configuration, rather than desk-top or server.

You will need to connect a keyboard, video monitor and mouse to each node in turn, as you install the software.

The interactive installation wizard will ask you to specify several options:

- Time Zone

  Select the time zone of Los Angeles, USA

  Check the *System Time Uses UTC* box.

- Network Configuration

  Skip this for now.

- Package Selection

  Just take the standard set of packages.

- Security

  On master, select the highest level of security, but customize the firewall to allow **ssh**.

  On the other nodes, select *No Firewall*.

- Boot Options

  Boot the Grub boot loader from the boot segment of the hda disk volume.

- Boot Diskette

  Write a boot diskette for master. The other nodes do not have diskette drives, so it is not possible for them to write boot diskettes.

Boot Configuration

On each node, edit `/etc/inittab` so that the specification for initial run level after boot looks like this:

```
id:3:initdefault:
```

Network Configuration

Everything in this section must be performed by the root user.

/etc/hosts

On all nodes, the file looks like this (where riso.rfpk.washington.edu is the domain name and host name of master):

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1               localhost.localdomain localhost
192.168.1.1             master riso.rfpk.washington.edu
192.168.1.2             node0
192.168.1.3             node1
192.168.1.4             node2
192.168.1.5             node3
```

/etc/resolv.conf

On master:

```
search rfpk.washington.edu u.washington.edu
nameserver 128.95.19.1
nameserver 128.95.120.1
```

On the other nodes:

```
# The cluster nodes do not need to resolv domain names
```

/etc/sysconfig/network

On master:

```
NETWORKING=yes
HOSTNAME=riso.rfpk.washington.edu
```

On the other nodes, set the HOSTNAME variable appropriately, and the GATEWAY variable to be the IP address of master. For example, the file should look as follows on node0:

```
NETWORKING=yes
HOSTNAME=node0
GATEWAY=192.168.1.1
```

/etc/sysconfig/network-scripts/ifcfg-eth0

On master:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
IPADDR=128.95.35.150
NETMASK=255.255.255.0
GATEWAY=128.95.35.100
NETWORK=128.95.35.0
BROADCAST=128.95.35.255
```

On node0:

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
```

On node1:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=no
```

On node2:

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
IPADDR=192.168.1.4
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
```

On node3:

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
IPADDR=192.168.1.5
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
```

/etc/sysconfig/network-scripts/ifcfg-eth1

On master:

```
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
```

On node1:

```
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
TYPE=Ethernet
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
IPADDR=192.168.1.3
NETMASK=255.255.255.0
```

On the other nodes, this file does not exist.

After making the file changes, reboot all of the nodes so that the changes can take effect.

Firewall Security

The network security strategy for the cluster consists of having high security on the Internet interface to master coupled with no Internet visibility for the other nodes. In RedHat Linux, the **lokkit** tool is used to generate the necessary *iptables* filtering rules. The following specifications should be provided to **lokkit**:

- Security Level: High
- Customize
- Trusted devices: eth1
- Allow incoming: SSH

In RedHat 8.0, unfortunately, the **lokkit** gui, normally started with the main menu sequence:

```
Main => System Settings => Security Level
```

does not present **eth1** in the list of trusted devices. Use the terminal version, instead, by executing **/usr/sbin/lokkit** as root.

Time Server Configuration

It is important that the system clocks of the nodes be closely synchronized, since the nodes are supposed to work together as a single computer system. The following implements the recommendations of the RFPK *Clock Synchronization* Howto.

/etc/ntp.conf

On all nodes, edit the line at the beginning of the file that reads

```
restrict default ignore
```

to read

```
restrict default nomodify
```

After the line that reads

```
# server mytrustedtimeserverip
```

on master insert the lines

```
restrict whitechuck.rfpk.washington.edu nomodify notrap noquery
restrict time.nist.gov nomodify notrap noquery
restrict tick.uh.edu nomodify notrap noquery
restrict tick.usno.navy.mil nomodify notrap noquery
server whitechuck.rfpk.washington.edu
server time.nist.gov
server tick.uh.edu
server tick.usno.navy.mil
```

and on the other nodes insert the lines:

```
restrict 192.168.1.1 mask 255.255.255.255 nomodify notrap noquery
server 192.168.1.1
```

All nodes should include the lines:

```
server 127.127.1.0
fudge   127.127.1.0 stratum 10
```

after the comment titled "GENERAL CONFIGURATION".

/etc/ntp/step-tickers

On master, this file should read:

```
whitechuck.rfpk.washington.edu
time.nist.gov
```

On the other nodes, it should read:

```
192.168.1.1
```

Enable the Network Time Service

On all nodes, configure **ntpd** to be started automatically whenever the system boots:

```
        /sbin/chkconfig --level 345 ntpd on
```

SSH Configuration

To simplify administration of multiple nodes, set up **ssh** so that the root user on master can access any of the nodes via **ssh** and **scp**, without providing the root password.

Generate and Install Public Keys

Follow the RPFK *SSH Configuration* Howto to set up public key authentication for root on master just as you would for your ordinary login on your workstation, except that root's public key should be installed on each of the other nodes rather than on whitechuck.

Setup Gnome and SSH

As the root user, start Gnome on master. Follow the instuctions in the *SSH Configuration* Howto in the section titled *Configure Gnome*.

Install Keychain

Much of the time you will be administering the cluster not from the console on master but remotely via **ssh** from your own workstation. An additional package, known as *keychain*, makes this convenient. It sets up authentication properly, even when you come in through a terminal, or a terminal emulation, such as **ssh**.

Download the keychain rpm from gentoo [1] and install it. Then add the following lines to `/root/.bash_profile` on master:

```
keychain ~/.ssh/id_dsa
. ~/.keychain/${HOSTNAME}-sh
```

Open Mosix Installation

Install the Open Mosix RPMs on all Nodes

Download the latest stable openmosix-kernel rpm and the latest openmosix-tools rpm from SourceForge[2] and install them first on master.

Connect the keyboard, video monitor and mouse to master, and boot the machine. When the Grub boot screen appears, OpenMosix should be in the list of kernels, although it will not be the default kernel. Select it, and continue the boot process.

If the newly installed OpenMosix kernel on master seems to be working alright, install both rpm files on all of the other nodes.

Update the Grub Configuration

The configuration of the grub boot loader must be changed to boot the new OpenMosix kernel by default. On each node, in `/etc/grub.conf` you should see a list of kernels that are installed on the system. If the new kernel is the first in the list, edit the first line after the initial comment to read

```
default=0
```

Install OpenMosix View on Master

Down load the `openmosixview` rpm appropriate for your version of RedHat Linux from the OpenMosixView.com download page [3], and install it.

You may need the `glut` rpm as well. You can get that from the RedHat rhn site.

Install OpenMosix Stress Test on Master

Download the omtest [4] rpm and install it.

You will probably also need a `tk` rpm and an `expect` rpm, both of which can be obtained from the RedHat rhn site.

Mapping the Nodes

The auto-discovery process does not work properly on this cluster. In his OpenMosix Howto, Kris Buytaert says that this sometimes happens with PCI Ethernet adapters and describes a work-around that involves placing the Ethernet interfaces in promiscuous mode. This is a bad idea from the point of view of security. On a small cluster such as RFPK's, it is better to use static mapping.

On all nodes, `/etc/openmosix.map` should be the same. On master, append the following line to that file

```
1     192.168.1.1     5
```

then copy the file to the other nodes.

Configuring oMFS

OpenMosix has its own distributed file system, called oMFS. This option is compiled into the rpm kernel. The Direct File System Access (DFSA) is also compiled in.

In order to activate oMFS, make the following changes to *all nodes:*

- Create a directory on which to mount the file system:

  ```
  mkdir /mfs
  ```

- Add the following line to `/etc/fstab` on master

  ```
  mfs_mnt                  /mfs                     mfs     dfsa=1          0 0
  ```

- Copy `/etc/fstab` to each of the other nodes.

Configuring openMosixCollector

The `openmosix-tools` rpm installs a real-time performance data collection daemon. Its service management script is `/etc/rc.d/init.d/openmosixcollector`. Before it can be automatically stopped and started, the following lines must be added to the script, at the beginning:

```
# chkconfig: 2345 96 4
# description: openMosixCollector records real-time performance
#              data for openMosix
```

After editing the script, have **chkconfig** complete the complex task of setting up links in the subdirectories of `/etc/rc.d`:

```
cd /etc/rc.d/init.d
chkconfig --add openmosixcollector
```

Operations

Powering the Cluster Up

1. Toggle the power switch on each of the nodes.
2. After master comes up (this takes less than two minutes), as the root user open a terminal window.
3. Check that the clocks are all synchronized by entering the **ntpcheck** command. If the clocks are not in sync, use the **resync** command to attempt to synchronize them. If the cluster has been down for awhile, this may not work immediately. You may have to wait for several minutes after the reboot of master before **resync** will work. Alternate the running of **resync** and **ntpcheck** until you are sure that the clocks are all in step with each other.

Utility Shell Scripts

There are some useful shell scripts in the `/root/bin` directory on master:

- **datecheck:** runs the **date** command on each of the nodes and outputs the results.
- **ntpcheck**: runs the **ntpq -p** command on each of the nodes and outputs the results. This provides feedback on the degree to which the clocks are synchronized.

- **powerdown**: powers the cluster down.
- **reboot**: reinitializes all the nodes in the cluster. After the systems come back up (this takes about two minutes), run **ntpcheck** to determine whether or not the clocks are still in sync. If not, alternate running **resync** and **ntpcheck** until you see that they are in step. Sometimes it takes a couple of minutes after the reboot of master before synchronization succeeds.
- **resync**: restarts the network time protocol daemons on all the nodes except master, in order to resynchronize the clocks.

Backup

At present, only the `/etc` directory on each node and the `/root` directory on master are backed up. Initiation of backup is manual, and should be done every time changes are made to the configuration files and administrative scripts in these directories.

The backup procedure is the following:

1. As root on master, run the **backup** script that resides in `/root/bin`.
2. In the `/tmp` directory, the script will create a directory called `yyyy-mm-dd-tttt.config`, where *yyyy-mm-dd-tttt* represents the current date and time.
3. The script will then use **tar** to create an archive for the `/etc` directory of each node and `/root` on master.
4. Next, the script will create a compressed tar file of the directory.
5. Finally, the script will use **scp** to transfer the compressed archive to the `/home/cluster/backup` directory on whitechuck. During this process, you will be asked for the password for cluster on whitechuck.

Notes

1. http://www.gentoo.org/proj/en/keychain.xml
2. http://openmosix.sourceforge.net/
3. http://www.openmosixview.com/download.html
4. http://www.openmosixview.com/omtest/