Microwulf Software and Network Configuration Notes

Tim Brom

May 16, 2008

Contents

1 Installing the Operating Systems

1.1 Installing the Head Node Operating System

1.1.1 Installation

At this point, the cluster should be physically assembled. There is a head node with a CD/DVD drive and a hard drive installed. The next step is to install the operating system. Put an Ubuntu Desktop CD in the drive and boot the head node. Follow the installation instructions (installation instructions can be found on the Ubuntu website `http://www.ubuntu.com`) until you get to the point where you partition the drives.

1.1.2 Partitioning the drive

How the drives are partitioned is a matter of personal preference and available drive space. On Microwulf, we had a 250 GB hard drive, and it was partitioned as follows.

(1)	50 GB	For the root (/) partition
(1)	150 GB	For /home. This will be exported to the nodes via NFS
(1)	1 GB	Swap
(3)	10 GB	These partitions are the root partitions for the diskless nodes

These sizes can be played with a bit. Currently, we have about 40 GB free in / on our head node, so that partition size could be pared down a bit. /home is where most of the disk usage is probably going to come from, so you should make that as big as possible. Our nodes are currently using about 1.5 GB of their partitions, so that could also be reduced somewhat, but I'd suggest not going below 5 GB.

Once the partitions have been created and formatted (I would suggest using ext3), finish the installation process. You are done with the base OS install on the head node.

1.1.3 Head node networking

The head node has three network cards. Two of them are for use in computations and connect to the gigabit switch. One of the cards connects to the outside world. On Microwulf, eth0 and eth1 are the gigabit cards and

eth2 is the 10/100 card used to connect to the outside world. eth2 is configured to get an address automatically via DHCP, eth0 is set to 192.168.2.1 netmask 255.255.255.0, and eth1 is set to 192.168.3.1 netmask 255.255.255.0. The reasons for this are explained in section 3 (Networking). Most traffic (such as NFS, or if a node is trying to reach the outside world) goes over the 192.168.2.0/24 network, as does MPI traffic. The 192.168.3.0/24 network is strictly for MPI traffic.

1.2 Installing the Diskless Node Operating System

Replace the Ubuntu Desktop CD with the Ubuntu Server CD and reboot the head node. Go through the installation process, using one of the partitions you created for the diskless nodes root partition as the root partition for this installation. Make sure you do not install a boot loader. You can either repeat this three times, once for each node, or once you have the first node installed reboot the head node (after removing the CD) and use dd(1) to copy the contents of the partition to the other two partitions. That puts the operating system into place for the diskless nodes to boot once you get remote booting configured.

2 Remote Booting

This section describes how to set up the head node so that the diskless nodes can do a remote PXE boot, getting their kernel and mounting their disk from the head node.

2.1 NFS

Before you can export the diskless nodes root partitions over NFS, you must mount them somewhere on the head node. I mounted them to /nodes/nfs/node1, /nodes/nfs/node2 and /nodes/nfs/node3. Feel free to change this, but those are the locations I will reference for the rest of this howto. Run the following commands to create the necessary directories:

```
sudo mkdir -p /nodes/nfs/node1
sudo mkdir -p /nodes/nfs/node2
sudo mkdir -p /nodes/nfs/node3
```

Put appropriate entries in /etc/fstab so these are mounted automatically at boot. These are the entries I added to our /etc/fstab:

```
/dev/sda6 /nodes/nfs/node1 ext3 noatime 0 0
/dev/sda7 /nodes/nfs/node2 ext3 noatime 0 0
/dev/sda8 /nodes/nfs/node3 ext3 noatime 0 0
```

After these entries are added to /etc/fstab, run the following command to actually mount the partitions:

```
sudo mount -a
```

Install the NFS utilities

```
sudo apt-get install nfs-common nfs-kernel-server
```

and configure NFS. Edit /etc/exports, and export the diskless nodes root partitions. Our /etc/exports is as follows:

```
/nodes/nfs/node1  192.168.2.0/24(rw,no_root_squash,sync,no_subtree_check)
/nodes/nfs/node2  192.168.2.0/24(rw,no_root_squash,sync,no_subtree_check)
/nodes/nfs/node3  192.168.2.0/24(rw,no_root_squash,sync,no_subtree_check)
/nodes            192.168.2.0/24(rw,no_root_squash,sync,no_subtree_check)
/home             192.168.2.0/24(rw,no_root_squash,sync,no_subtree_check)
```

This exports /nodes/nfs/node{1,2,3}, /home and /nodes to any machine in the 192.168.2.0/24 range. Restart nfs (/etc/init.d/nfs restart) on the head node. Note: once the diskless nodes are running, add entries to their /etc/fstab files so that /home is mounted at boot. These are the entries added to our /etc/fstab file:

```
192.168.2.1:/home      /home  nfs     defaults        0       0
```

2.2 DHCP

Install the DCHP server

```
sudo apt-get install dhcp
```

and edit the configuration file /etc/dhcpd.conf. Our config file follows:

```
option domain-name "calvin.edu";
option subnet-mask 255.255.255.0;
option domain-name-servers 153.106.4.99;
next-server 192.168.2.1; #TFTP server
filename "/tftpboot/pxelinux.0";

subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.150 192.168.2.200;
    option broadcast-address 192.168.2.255;
    option routers 192.168.2.1;
}

host node1 {
        hardware ethernet 00:16:17:ef:fa:38;
        fixed-address 192.168.2.2;
        option root-path "/nodes/nfs/node1";
}

host node2 {
        hardware ethernet 00:16:17:ef:f9:81;
        fixed-address 192.168.2.3;
        option root-path "/nodes/nfs/node2";
}

host node3 {
        hardware ethernet 00:16:17:ef:fb:68;
        fixed-address 192.168.2.4;
        option root-path "/nodes/nfs/node3";
}
```

Obviously, some of this will need to be edited. The domain-name and domain-name-servers should be set to whatever is appropriate for your institution, and the ethernet addresses will definitely be different for your nodes.

The host sections of the dhcpd.conf file specify options for certain hosts, identified by MAC address. This part assigns them a certain address and passes the root-path option, which is used to mount the root partition over NFS during the boot process.

Finally, edit /etc/defaults/dhcp so that dhcp only listens on the appropriate interface (eth0). It's a bad idea to accidentally run a DHCP server on your campus network.

2.3 TFTP

Install a tftp server, I like atftpd. Since atftpd is designed to start from xinetd you need to install that too.

```
sudo apt-get install atftpd xinetd
```

and save the following as the file atftpd in the folder /etc/xinetd.d (this allows xinetd to start atftpd)

```
service tftp
{
    disable     = no
    socket_type = dgram
    protocol    = udp
    wait        = yes
    user        = nobody
    server      = /usr/sbin/in.tftpd
    server_args = --tftpd-timeout 300 --retry-timeout 5
       --mcast-port 1758 --mcast-addr 239.239.239.0-255
       --mcast-ttl 1 --maxthread 100 --verbose=5  /tftpboot
}
```

create the /tftpboot folder and chmod it 777.

```
sudo mkdir /tftpboot
sudo chmod 777 /tftpboot
```

2.4 PXE

At the time of this writing, PXELinux can be obtained from `http://syslinux.zytor.com/pxe.php`. Download it, and find the file pxelinux.0. There are a lot of files in the tarball you download, but that is the only one you need to be concerned with. Put the pxelinux.0 file in /tftpboot.

Note, through the rest of this section, KERNEL_VERSION should be replaced with the currently running kernel version, typically the output of

```
uname -r
```

2.4.1 Getting the Kernel and initrd Into Place

The next step is to generate the correct initial ramdisk, and to put the initial ramdisk and the kernel image in the /tftpboot directory. To generate the initial ramdisk, chroot so that one of the diskless nodes root directory becomes your root directory.

```
sudo chroot /nodes/nfs/node1 /bin/bash
```

Edit the file /etc/initramfs-tools/initramfs.conf and set BOOT=nfs (originally, it would be BOOT=local) and save the file. Generate a new initial ramdisk by running the command

```
sudo update-initramfs -u
```

The program will output the filename of the newly created initial ramdisk (default: /boot/initrd.img-KERNEL_VERSION). Exit the chroot (type exit) and copy that new initial ramdisk and the kernel image to the /tftpboot directory.

```
sudo cp /nodes/nfs/node1/boot/initrd.img-KERNEL_VERSION /tftpboot/
sudo cp /nodes/nfs/node1/boot/vmlinuz-KERNEL_VERSION /tftpboot/
```

2.4.2 Configuring PXE

Create the directory /tftpboot/pxelinux.cfg

```
mkdir /tftpboot/pxelinux.cfg
```

The PXE config files go into this directory. When a node boots, it gets its config file by looking in this directory for a filename with the format 01-xx-xx-xx-xx-xx-xx, with the xx replaced by the nodes MAC address (so, for our first node the filename is 01-00-16-17-ef-fa-38 since this nodes MAC address is 00:16:17:ef:fa:38). This file contains the information needed to boot the system. Our config file is as follows:

```
default linux

label linux
kernel vmlinuz-KERNEL_VERSION
append initrd=initrd.img-KERNEL_VERSION \\
        nfsroot=192.168.2.1:/nodes/nfs/node1
1
```

As you can see, this file looks a lot like a typical grub or lilo config file. You specify the name of the kernel (the filename of the kernel that you put in /tftpboot), the initial ramdisk (the filename of the initial ramdisk that you put in /tftpboot) and the filesystem that is to be used as the nodes root filesystem. Make a config file for each of your diskless nodes.

2.4.3 Configuring the Diskless Nodes

Not much has to be done here. When the node is initially booted, go into the BIOS and set up PXE booting. On most motherboards, that's simply a matter of going into the boot order options and choosing the LAN adapter as the default boot device. Consult your motherboard manufacturers manual if that doesn't work.

As mentioned in section 2.1, you need to configure the slave nodes to mount the /home directories automatically at boot. You do so by adding the following line to the /etc/fstab file:

```
192.168.2.1:/home       /home   nfs     defaults        0       0
```

Here is the complete /etc/fstab file for our slave nodes:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
192.168.2.1:/home       /home   nfs     defaults        0       0
192.168.2.1:/nodes      /nodes  nfs     defaults        0       0
```

Once you have ths set up, you should be able to boot the nodes, and have it all work.

Congratulations, your cluster is running.

2.4.4 Upgrading the Kernel

Upgrading the kernel that the diskless nodes use is simple. Simply copy the new kernel and the new initrd from one of the diskless nodes into /tftpboot on the head node, update the config files in /tftpboot/pxelinux.cfg to point to the new kernel version, and reboot the diskless nodes.

3 Networking

Each of the nodes in this cluster has two network cards (well, the head node has three, but the third one is strictly used to communicate with the outside world). The method I decided to use to balance the network load between the two network cards is described here. This is by no means the only way, and I haven't had an opportunity to show that this is the best way, but it is fairly simple to implement and it will certainly result in some degree of load balancing for most workloads.

3.1 Topology

As described earlier, the onboard NICs are given addresses in the 192.168.2.0/24 range, and the PCI-Express NICs are given addresses in the 192.168.3.0/24 range. There are four nodes in this cluster, but eight cores, so logically from a networking standpoint I treat this cluster as if it were an eight-node cluster. Rather than setting up a DNS server, I modify the hosts file on each node so that we can use hostnames rather than IP addresses. The following table shows how the network is set up:

	Onboard NIC			PCI-Express NIC		
	IP Address	name	via	IP Address	Hostname	via
Head	192.168.2.1	pc0	static	192.168.3.1	pc1	static
Node 1	192.168.2.2	pc2	DHCP	192.168.3.2	pc3	static
Node 2	192.168.2.3	pc4	DHCP	192.168.3.3	pc5	static
Node 3	192.168.2.4	pc6	DHCP	192.168.3.4	pc7	static

3.2 Configuration

Each node needs a custom /etc/hosts file. If possible, you want communication to happen over the loopback interface, since this is the lowest latency, highest bandwidth interconnect, but obviously it only works between two cores on the same board. The /etc/hosts file on the head node looks like this

```
127.0.0.1       pc0
127.0.0.1       pc1
192.168.2.2     pc2
192.168.3.2     pc3
192.168.2.3     pc4
```

```
192.168.3.3     pc5
192.168.2.4     pc6
192.168.3.4     pc7
```

and the /etc/hosts file on Node 1 looks like this

```
192.168.2.1     pc0
192.168.3.1     pc1
127.0.0.1       pc2
127.0.0.1       pc3
192.168.2.3     pc4
192.168.3.3     pc5
192.168.2.4     pc6
192.168.3.4     pc7
```

and so on. Each node sees that three of the other nodes are on one network, and three of the other nodes are on the other network, with one node reachable over the loopback interface. Assuming that each node communicates with every other node approximately equally, this should provide good load balancing. Obviously, whether or not this assumption holds true depends on the workload.

If you are writing your own code, you may be able to exploit this topology for maximum performance. If two parts of your MPI program need to do a large amount of communication, code your program so that those two parts run on two different cores of the same node, so communication can occur over the loopback interface. You could also combine MPI and threads (either pthreads or OpenMP) and take advantage of the fact that each node is really an SMP machine. Another direction to go is bonding the two interfaces together into one logical etherchannel link, if your switch supports it.

3.3 Allowing Diskless Nodes Outside Access

This part is optional. I decided that I wanted our diskless nodes to be able to access the outside internet in order for them to more easily get software updates. I accomplished this by setting up iptables on the head node to do IP masquerading (NAT) on behalf of the diskless nodes. To accomplish this, I installed iptables

```
sudo apt-get install iptables
```

and added the following lines to /etc/rc.local, to enable ip forwarding every time the head node booted

```
## Enable IP Masquerading
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o eth0 -j MASQUERADE
```

The diskless nodes should be set up so that 192.168.2.1 is their default gateway.

4 Conclusion

The previous sections describe the software and network configuration of Microwulf. From here, it is a matter of installing whatever version of MPI you use, and whatever other tools you need on your cluster. Enjoy.