The TinyHPC Cluster

Mukarram Ahmad

Abstract

TinyHPC is a beowulf class high performance computing cluster with a minor physical footprint yet significant computational capacity. The system is of the shared memory category, and uses the Message Passing Interface (MPI) API to allow communication between nodes. The primary intended usage of TinyHPC is to run fluid dynamics simulations, as well as to test a new data compression technique that uses neural networks.

1 Introduction

This section discusses the basic definition of beowulf clustering, as well as advantages of constructing a cluster of such a configuration.

1.1 Definition

A beowulf system is a high performance computing infrastructure constructed from commidity hardware and running open source software and operating system. Such a setup typical consists of a server that is connected to a number of headless, diskless nodes over some form of a network [3]. Generally, in such a system, the server node assigns diskless nodes their IP addresses, provides support to allow each node to boot over the network, and hosts networked storage to which all nodes have access [5]. Beowulf systems are sometimes classified as being Class I or Class II, with the former being entirely composed of commidity hardware and thus less expensive and easily reproducible, while the latter using more specialized components to acheive better performance at a higher cost [2].

1.2 Rationale

Aside from the general speedup advantages of using multiple CPUs, the primary motivation to construct a beowulf computing cluster is the extremely low construction cost relative to traditional high performance computing systems. Also, since the speeds of memory modules is not necessarily at pace with that of processors, usage of multiple sets of these executing tasks in parallel provides a way to circumvent this gap.

2 Related Work

Mention a listing and brief descriptions of related small footprint beowulf systems.

3 System Design

Appendix A shows the hardware configuration of TinyHPC. The following two sections describe the system in more detail.

3.1 Hardware Setup

The TinyHPC cluster has a hybrid hardware composition, with two sets of nodes. Table 1 describes the specifications of each set of compute nodes.

	Node Set 1	Node Set 2
Motherboard	Toshiba A135-S2356	Intel 815GEW
Processor Architecture	Intel Pentium 4	Intel Celeron
Processor Clock Speed	1.5 GHz	1.3 GHz
Processor Cache (Level 1)	64 KB	32 KB
Processor Cache (Level 2)	1024 KB	256 KB
Main Memory	256 MB	256 MB
Network Interconnect	100 MB/s Ethernet	100 MB/s Ethernet

3.2 Server Disk Partitioning

The head node of the cluster contains three hard drives, with the sizes of 40 GB, 20 GB, and 10 GB respectively. For use in TinyHPC, they were set up as follows:

Disk / Partition	Size	Mount Point	Type	Usage
Disk 1 / Partition 1	40 GB	/home	Ext3	Storage space for home directory
Disk 2 / Partition 1	10 GB	/	Ext3	Root file system
Disk 2 / Partition 2	1 GB	swap		Linux swap space
Disk 3 / Partition 1	10 GB	/nodes	Ext3	Node root file system

3.3 Operating System Installation

The first step was to install the base operating system for the server. Ubuntu Server was chosen due to its lean nature in terms of initial software inclusion. The installation disk was used to customize the disks according the table listed in the previous section. The operating system installation then proceeded as follows:

- Ubuntu Server was installed with the root filesystem in the first partition of the second disk, home directory in the first disk, and swap space in the second partition of the second disk. The third disk was assigned a manual mount point of /nodes. All partitions except swap were Ext3 journaling filesystems.
- The installation was repeated (disk partitioning skipped this time, since this was already done), and this time, the entire filesystem was installed on the third disk.

3.4 Operating System Customization

The first step was to install the required packages to the rather basic Ubuntu Server installation. A connection to the internet was hence required, and the second network card of the head node was used for this purpose. The following lines were added to the /etc/network/interfaces file:

```
auto eth1
iface eth1 inet dhcp
```

This loads the interface eth1 on startup, and instructs it to gets its configuration information via DHCP. The next step was to update the package lists. All the sources (except the CD) were uncommented from the /etc/apt/-sources.list file, and the following command was issued to update the package lists:

```
sudo apt-get update
```

The packages were then downloaded and installed; **it is important to note that the process was repeated for the nodes root file system also, by chrooting to the /nodes directory as follows**:

```
sudo chroot /nodes /bin/bash
```

The packages were then installed; the commands are broken down to ease later explanation.

```
sudo apt-get install ssh rsh-client rsh-server
sudo apt-get install dhcp3-server tftpd-hpa nfs-
   kernel-server
sudo apt-get install python-central python-dev
   python-gd python-numpy python-scipy python-
   matplotlib
sudo apt-get install make gcc g++ gfortran g77 libc6
   -dev
sudo apt-get install xorg fluxbox
sudo apt-get install libatlas-headers libatlas3gf-
   base
```

In the above package installation script, the ssh and rsh software were installed in order to improve control over the nodes, and satisfy the requirement of the MPI packages to be later installed. The DHCP, TFTP, and NFS servers were installed in order to provide network booting capabilities for the client nodes. The python packages were installed for later installation of pyMPI. Packages xorg and the Fluxbox window manager were optional, but were installed for ease of use in occassional situations when performance was not an issue. The make package, and the C, C++, and Fortran compilers were installed to allow building software from source. Finally, the ATLAS library was installed to later build the HPL Linpack benchmark. The installation was repeated after chrooting the /nodes directory.

Next, the operating system of the nodes was to be configured. Three things were done here; first, the /etc/fstab file was edited to remove all lines except that which mounts /proc, and to add:

```
10.11.12.1:/home /home nfs auto defaults 0 0
```

This basically mounts the shared /home of the server as a replacement of its own home directory. This is desired so that software built from source, as well as other work files, can be easily shared among the nodes of the cluster.

After that, the inital ramdisk was customized so that the root file system was done mounted over NFS. This was done by modifying the BOOT=local line to

```
BOOT=nfs
```

and then regenerating the ramdisk by entering

```
update-initramfs -u
```

The names of the kernel and the initial ramdisk were then noted from the /nodes/boot directory. Finally, the /etc/rc.local file was modified to customize the /etc/hosts file of the nodes on boot with appropriate IP and hostname, and display that and a welcome screen at startup.

```
clear
sleep 1
echo "===================================="
echo "              TinyHPC               "
echo "===================================="
echo ""
echo ""
sleep 2
echo "Welcome to the TinyHPC Cluster!"
echo ""
echo ""
```

```
sleep 1
echo "Copyright (C) 2008 Mukarram Ahmad."
echo ""
echo "----------------------------------"
DATEVAL=`date`
MYIP=`ifconfig | grep -i "Ethernet" -A 1|grep "inet
   addr"|cut -d " " -f 12|cut -d ":" -f 2`
echo "Today, the date is: $DATEVAL"
echo ""
echo "Also, the IP of this node is $MYIP"
echo "----------------------------------"
echo "The Boot Process is Now Complete. Please Login
    Below."
echo "----------------------------------"
echo ""
rm -f /etc/hosts
touch /etc/hosts
MYHOSTNAME=`/bin/hostname`
echo "$MYIP $MYHOSTNAME" > /etc/hosts
exit 0
```

Once the required packages were installed, and the node operating system was modified to a satisfactory level, configuration of the server node was proceeded with. The first step was to set up network booting, so that the client nodes could boot over PXE. To do this, the configuration files for the DHCP server, the TFTP server, and the Network File System server had to be modified. The DHCP configuration file, located as the /etc/dhcp3/dhcpd.conf, was modified to have the following content:

```
allow booting;
allow bootp;
default-lease-time 600;
max-lease-time 7200;

subnet 10.11.12.0 netmask 255.255.255.0 {
next-server 10.11.12.1;
filename "pxelinux.0";
option subnet-mask 255.255.255.0;
range 10.11.12.2 10.11.12.50;
}
```

This basically defines the network on which the DHCP server is to run on, and the name of the file from which the nodes boot. The TFTP server's configuration file, with the path /etc/default/tftpd-hpa, had the following, which basically made sure that the TFTP daemon ran on the server, and specified the location of the TFTP root, where the files for booting were to be later placed.

```
RUN_DAEMON="yes"
OPTIONS="-l -s /tftpboot"
```

Finally, the /etc/exports file was set up so that the right directories were exported via NFS - namely the /nodes directory, where the client node root file system is held, and the /home directory, which is to be shared among the cluster nodes, as previously described.

```
/nodes 10.11.12.0/24(rw,no_root_squash,no_subtree_check)
/home 10.11.12.0/24(rw,no_root_squash,no_subtree_check)
```

Having set up the configuration files, the three servers were restarted, **after issuing a command that configures the network interface over which the booting occurs**, all this being done as root:

```
ifconfig eth0 10.11.12.1 netmask 255.255.255.0 broadcast 10.11.12.25

/etc/init.d/dhcp3-server restart
/etc/init.d/tftpd-hpa restart
/etc/init.d/nfs-kernel-server restart
```

After testing this setup by booting one client node to make sure it works, SSH was set up so that it did not prompt for passwords:

```
ssh-keygen  -q -t rsa -N "" -f "$HOME/.ssh/id_rsa"
cp $HOME/.ssh/id_rsa.pub $HOME/.ssh/authorized_keys
chmod 600 $HOME/.ssh/authorized_keys
```

3.5 MPI Installation

MPICH2 was chosen as the MPI implementation for the cluster and was installed in the shared /home directory with shared libraries enabled by issuing as root:

```
mkdir $HOME/MPICH2
./configure --prefix=$HOME/MPICH2 --enable-sharedlibs=gcc
make
make install
```

4 Performance Figures

The system has a theoritical peak of **28 GFLOPS** ((1.5 GHz x 5 Cores x 2) + (1.3 Ghz x 5 Cores x 2))

TBD - Coming Soon

5 Applications

TBD - Coming Soon

6 Conclusion

TBD - Coming Soon

References

[1] Beowulf.org

[2] Beowulf.org class i and ii page

[3] Beowulf book

[4] Microwulf

[5] TBD

[6] TBD