A Beowulf Cluster for Ray-tracing and Stereographic Visualization: Issues and Implementation

David Backeberg
Department of Math/Computer Science/Physics
Wartburg College
Waverly, IA 50677
backebergd@wartburg.edu

Abstract

A Beowulf-class cluster consists of multiple commodity processing units linked by a high-speed network. Using a protocol such as Parallel Virtual Machine (PVM), a programmer can specify a solution in parallel across theoretical machines, without regard for the cluster-specific hardware configuration.

Stereographic (3D) imaging mimics the viewing process performed by the human eye. Using a specialized language to describe a scene, the camera view is offset to the left or right by a distance proportional to the distance between the human eyes. The scene is then rendered for both "eyes." Various techniques composite the two images into one final scene the human views as a stereographic image.

Ray-tracing is the technology used to render highest-quality graphics and requires significant computer power and time. With parallel-capable ray-tracing software the render time is limited only by physical network bandwidth and hardware budget constraints. Initial experimentation showed an 80% reduction in wait time.

Introduction

The entire computer industry was founded on the principle of developing a machine able to perform a complicated task as quickly as possible. High performance computing (HPC) was performed on very large, expensive machines because there were no other options available. As personal computer hardware became progressively cheaper, faster, and more reliable, HPC researchers reviewed the feasibility of assembling an array of nodes made from readily available PC hardware.

In 1994, Donald Becker and Thomas Sterling constructed a 16-node cluster constructed with 486DX4 processors [1]. The cluster sustained a 60 Mflops calculation and demonstrated that commodity components and creative network setup can achieve a reasonable throughput. Subsequent hardware upgrades achieved sustained 1+ Gflops calculations at a price below $50,000 in 1996 and cemented the term Beowulf as referring to a cluster of commodity processing nodes linked by a high-speed network [2].

Beowulf and Stereographic Visualization at Wartburg College

Anthony Betz created a Beowulf cluster from discarded 486 machines as a senior project in Winter 2000. The cluster was operative and demonstrated time gains in calculations, but suffered from several shortcomings related to the minimal budget and outdated hardware. Seniors who attempted to perform research on the cluster in Winter 2001 suffered from hardware failures and a fundamental lack of documentation. It became glaringly obvious that for a subsequent attempt at a Beowulf cluster to have a lasting impact the project would need to have a reasonable equipment budget and should be accompanied by thorough documentation.

A senior project by David Lindner in Winter 2001 researched the possibility of a portable station for viewing stereographic images. Lindner's final station used dual LCD display projectors to display stereo pairs on a screen. After careful setting of viewing angles and distances, students wore polarized glasses to view the 3D action on the screen. Lindner's equipment was purchased through a Maytag Innovation grant and his final report detailed setting up the equipment for displays.

Latest Beowulf Project

Dr. Josef Breutzmann and I co-wrote a proposal and we applied for a Maytag Innovation grant in Spring 2001. A review of computer hardware prices in September 2001 showed that AMD Duron processors yielded the greatest performance for our budget of about $2000. At this point, several design decisions remained. With a limited equipment budget, we knew this could not be a cluster with dozens of nodes. Reviewing prices showed that it would be best to buy an Ethernet switch with spare capacity for future additions. Cluster nodes require special considerations to ease remote logins and are therefore less secure. The decision was made to select one node as a network gateway

and as a master for the other nodes. Most accesses of slave nodes will be automated network logins, so it was decided to exclude video cards from the initial equipment list. A summary of the equipment purchased for the present project:

Quantity	Item	Unit Price	Total
1	Nortel 10BT/100BTX 24-Port Baystack 70-24T unmanaged switch	$275.00	$275.00
6	3com 10BT/100BTX OfficeConnect 3CS0H0100TX PCI 1-PK	$25.00	$150.00
5	IBM 10.0GB Ultra-ATA/100 07N7401 7200rpm 8.5ms 2mb buffer	$72.00	$360.00
5	Duron (OEM) Bundles (Tyan S2390B; AMD Duron 800(200mhz); Mwave 32x64 256mb PC 133; Assemble/Test Bundle)	$183.00	$915.00
1	Aopen CD950E 50x EIDE internal cdrom	$31.00	$31.00
5	mWave CI-6306 mini-mid tower w/250W ATX power supply 3x5.25" 2x3.5" (hidden)	$46.00	$230.00
			$1,961.00
	Shipping		$128.34
			$2,089.34

Hardware Setup and Configuration

In the initial setup, the network was configured with the Ethernet switch uplink connected to the building LAN and all nodes connected to the switch. Redhat Linux 7.0 was chosen as the operating system due to its stability and low cost (free). The master node was setup first, which included dual network cards and a CD-ROM. After the master was set with the necessary software, the shell command *# dd if=/dev/hda of=/dev/hdb* was used to clone the hard drive for each node [3]. The cloning method avoided the repetitive installation process, and only machine name and Internet Protocol (IP) settings needed to be changed for the slaves.

Network Drivers

Research prior to purchase indicated that the 3c59x kernel module supported the 3Com 3CSOHO100 network card. In practice, the driver loaded with no errors, but the 10/100 cards negotiated the connection to the Ethernet switch at 10 Mbit/sec rather than the expected 100 Mbit/sec. Hours of web searching uncovered a document describing a driver tweak. Modifying the driver code and rebuilding the module solved the problem, and the new build replaced the modules throughout the cluster.

Services Offered by the Master

NFS (Network File System)

NFS allows files needed on each machine to be stored in one central location. Any parallel job operating on the cluster needs access both to the program running and any file used for I/O. The cluster uses NFS to share three directories:

1) */cshare* This directory serves as a conduit to move system files around to the machines with a remote shell. Commands spawn faster if they are stored on a local drive. Programs added after the drives had been cloned were installed from this directory.

2) */mnt/cdrom* This is the directory pointing to the root of the CD-ROM file system on the master. This share enables slaves to mount the install CD or any other CD placed in the drive on the master.

3) */home* This directory is the root directory for files owned by each user. Sharing home enables remote login shells to "look and feel" the same regardless of which node the user is using. Most parallel jobs depend on i/o relating to files stored in a user's home directory. Sharing home only cause minimal network lag and offer several conveniences.

The master's */etc/exports* describes the file permissions for the NFS shares and lists which nodes should be permitted to connect to the server. Each slave node uses the NFS client and sustains a full-time connection to the master. The slaves must have a modified */etc/fstab* pointing to the master and an empty directory on the local file system for each share listed in */etc/fstab* [4].

NIS (The Network Information System)

When a user logs in, the login program verifies the user credentials with a database of username/password pairs. The Root account may add or remove user accounts as necessary, but such a change would only affect the machine where Root logged in. NIS seeks to solve this problem and similar subtleties by sharing vital system files across the network. Like NFS, NIS works using a client-server setup.

The NIS server requires setting a NIS domain name. The NIS domain name is a unique name for the NIS daemon and should not be a word used in the fully qualified domain name. The NIS maps are really just database files. To make the database, edit */var/yp/Makefile* to include the appropriate files and then run # *make*. Each NIS client must be setup with the appropriate domain listed with the *NISDOMAIN* directive in */etc/sysconfig/network* and an */etc/yp.conf* giving the address of the NIS server it should bind to [4].

It's common to hit a few snags while setting up NIS. First, the master also needs to be able to attach to the NIS server, meaning that the master needs to run both the NIS server and the NIS client. Without this configuration, making the database throws several strange errors. The *Makefile* on the master was configured to share */etc/groups* and */etc/passwd*, which means that group and username information was exported. These two files are vital for working with NFS because the file system needs to be able to check file ownership against the user and group files to determine effective permissions for users.

Furthermore, the NIS maps are static, meaning they don't get updated unless Root manually updates them. Every time a group is added or removed the NIS maps must be remade with a # *make* in */var/yp*. NIS provides an alternate version of *passwd* to change user passwords. y*ppasswd* attaches to the NIS server running the *yppasswdd* daemon and automatically updates the NIS maps with the new password. Renaming the standard *passwd* and providing a symbolic link from *yppasswd* to *passwd* to makes the difference seamless to the user [4].

Parallel Software

Foremost among the software packages that assist programmers to write parallel applications are MPI (Message Passing Interface) and PVM (Parallel Virtual Machine). People who have programmed with one or the other usually have a strong preference for "their" package with fervor bordering on religion. The cluster resolves this conflict by offering both software packages. Both packages require remote logins to machines without prompting for a password.

Remote Login Without Prompting for a Password

When using *rsh* or *rlogin*, the computer receiving the login request checks */etc/hosts.equiv* for an entry that matches the current user on the requesting machine and the machine name. For instance, a user named *beowart* on machine *sephora* could request a remote shell on machine *jethro*. If the *rshd* daemon on *jethro* finds an entry *sephora beowart* in */etc/hosts.equiv*, the login will be granted without prompting for a password. If each */etc/hosts.equiv* on each slave must explicitly include a machine-name/username pair, the file could become very large. Luckily, *rshd* allows */etc/hosts.equiv* to use wildcards to describe connections from specific machines or usernames, or */etc/hosts.equiv* can explicitly permit all connections from all workstations. This configuration is blatantly insecure and should be avoided.

Many people feel that *rsh* and *rlogin* should be avoided and replaced with secure shell services *ssh* and *slogin*. The secure services encrypt the password before sending it over an insecure network. The cluster would not gain any advantage by using secure services because all inter-connections take place over a subnet separate from the normal building LAN.

PVM

PVM comes packaged in a RPM, which makes installation much easier. After a # *rpm –ivh pvm-3.4.3-4.i386.rpm*, PVM needs to know where to find itself and users should add the following to their shell init files [5]:

```
export PVM_ROOT=/usr/share/pvm3
export PVM_ARCH=LINUX
```

An advantage of PVM is the concept of virtual machines. A single parallel job can run on nodes of different architectures simultaneously. PVM provides a text-based console for configuring the virtual machine. Commands include *add, delete,* and *conf* for changing the machines included in the virtual machine and *help* for any commands not used very often. Programs dependent on PVM are compiled and linked against the library. These programs are launched while the PVM instance remains running according to the configuration set at the console.

MPICH

MPI is intended for an array of homogenous machines, but is based on an open standard and has multiple implementations. Development to the MPI standard benefits all implementations simultaneously, and it's possible that MPI may overtake all other software methods of easing parallel computing. MPICH is developed at Argonne National Laboratory.

The MPICH source comes packaged with a very thorough *configure* script. A subsequent *make* built the proper modules. Editing */usr/share/mpich/machines.LINUX* on the master provided a list of available workstations. The installs were repeated on the slaves. MPICH provides *mpirun –np <number of processors> <program name and arguments>*, which spawns the appropriate services according to the optional flag *–np*. Omitting the flag uses only the local machine. Five is the current maximum of processors in the cluster.

Converting the Master to a Gateway

The project started with the minimum necessary investment of effort to get networking running properly. With parallel operations running, there was time to revisit the idea of running the master as a gateway. With this approach, the master would use both the network cards installed earlier. Negotiating two networks adds complexity to the configuration, but allows reconfiguration of the wiring so that the slaves attach to the master through the uplink jack on the Ethernet switch. Such a configuration also shields the slaves from outside network traffic.

Driver Support for Dual Network Cards

The network module built earlier was reused here, but the second network card requires special parameters. The first issue was distinguishing which network card was which [6]. Listing PCI information with *$ cat /proc/pci* yielded the interrupts and the base i/o addresses for the two cards. Masking tape on the back of the computer identifies the devices by their aliases. The following information was set in */etc/modules.conf*:

alias eth0 3c59x
alias eth1 3c59x
options 3c59x io=0xdc00,0xe000 irq=10,11

Configuring Each Device

Device *eth0* was assigned to a static IP address on the building LAN. Device *eth1* was assigned to a static IP address on a different private subnet subsequently used by the slaves. Each slave was moved to a new IP address in the private subnet and */etc/hosts* files were updated to reflect the changes. Each slave was also modified to use *eth1* of the master as a gateway rather than the building gateway.

Routing

Modifying */etc/networks* provided alias and subnet pairs:

bhs-net 172.31.0.0 # BHS = Becker Hall of Science
beo-net 192.168.1.0 # Beo = first three letters of Beowart, my project pet name

The routing table now appeared as:

```
Kernel IP routing table
Destination  Gateway   Genmask     Flags Mtrc Ref Use Iface
beo-net          *     255.255.255.0 U    0    0   0  eth1
bhs-net          *     255.255.0.0   U    0    0   0  eth0
127.0.0.0        *     255.0.0.0     U    0    0   0   lo
default 172.31.254.254 0.0.0.0       UG   0    0   0  eth0
```

This output reads quite nicely. Any traffic bound for the building LAN will be routed out *eth0* and any traffic bound for the slaves will route through *eth1*. 127.0.0.1 is the same as saying loopback, which is TCP/IP-speak for only connecting to itself. This setup is sufficient for establishing a wall of security between the slaves and the building LAN. However, there are instances where it will be helpful for the slaves to access outside networks.

IPChains

IPChains uses a table full of rules for processing network traffic. Most Linux firewalls are based on a complex set of IPChains rules. IPChains is very complex and the only essential issue for the cluster is that packets from the slaves get forwarded and masqueraded through the master. While the following solution is far from enterprise-level security, *ipfwadm –F –a m –S 192.168.1.0/24 –D 0.0.0.0/0* served the purpose of forwarding and masquerading traffic [7]. Rebooting at this point would lose the IPChains rules. The command # *ipchains-save -v > /etc/sysconfig/ipchains* dumped the rules and

saved them in a place where the init.d script would read on startup. Restarting IPChains with the init.d script ensured that the configuration was correct. Slaves were then able to access machines both inside and outside of the building LAN.

POV-Ray

POV-Ray is the established standard in ray-tracing. Since POV-Ray is an open-source product, it has been ported to many platforms and many people have written plug-ins or other modifications to the code. POV-Ray parses a C-like language describing a scene and produces a high quality image of the scene. For advanced scenes, this process can stretch to more than twenty-five minutes on a node equipped with a Duron 800Mhz processor.

POV-Ray can be compiled as a command-line program or alternatively can be compiled to use a graphical front-end that displays the image while the render progresses. Since the command line can seem intimidating at first, the graphical Windows version is a useful tool to gain experience with POV-Ray.

PVMPOV

PVMPOV is an unofficial POV-Ray modification that cross-links the render engine against the PVM library. PVMPOV is distributed as a patch used on the standard POV-Ray source [8]. PVMPOV provides even more command-line options than standard POV-Ray, including options to slice the render jobs into smaller pieces for other nodes to process. Like other PVM applications, PVMPOV depends upon prior configuration of the user's virtual machine.

PVMPOV defaults to running one slice of the render on each machine added in the virtual machine. With a virtual machine of all five machines and the command *pvmpov +Iskyvase.pov +Oskyvase.png +h768 +w1024*, PVMPOV finishes in sixteen seconds and leaves the following output:

PVM Task Distribution Statistics:

host name	*[done]*	*[late]*	*host name*	*[done]*	*[late]*
lilia	*[21.35%]*	*[0.23%]*	*miriam*	*[21.74%]*	*[0.24%]*
sephora	*[21.61%]*	*[0.32%]*	*jethro*	*[20.31%]*	*[0.27%]*
rameses	*[14.97%]*	*[0.00%]*			

3D TECHNIQUES

Most people have scene a 3D visualization, whether it was in the using a View Master or perhaps watching an IMAX movie. Humans view the world with two eyes set at the same Y and Z coordinates, but X is separated by the width between the two eyes, or the ocular

distance. This distance is about 2” and in ray-tracing or stereo-photography a rule-of-thumb is to keep a 30:1 ratio between the main subject and the camera separation on the X axis [9]. In other words, when using a film camera (with a tripod!) to capture a stereo pair of a subject 30 feet away, the tripod should be moved one foot to the left or right for the second image of the pair. Other reputable sources justify using a 1:10 ratio [10]. With less separation, the objects will hardly “stand-out” and with more separation, the objects can “stand-out” too much. Sometimes this hyper-stereo effect can be used for creative effects.

POV-Ray Stereo Pair

To create a stereo pair, the settings for the camera in the scene description file must be adjusted relative to the object nearest the camera [11]. This demonstration modifies *skyvase.pov*, which is included with every POV-Ray distribution. The lines:

```
#declare WIDTH = 848;
#declare HEIGHT = 600;
#declare EYESEP = 0.2;
```

should be inserted above the camera description. [11] provides a mathematical formula for calculating the offset. In this instance, the formula is used to create an image of size 848 x 600. The file was saved as *pov_l.pov* and the *camera* description was modified to read [11]:

```
camera {
  location <-EYESEP/2,28.0,-200>
  up y
  right WIDTH*x/HEIGHT
  angle 60
  sky <0,1,0>
  look_at <-EYESEP/2,-12.0,0.0>
}
```

After saving the change to *pov_l.pov*, the file was saved again, as *pov_r.pov*. The only modification necessary to the right side is to remove the negative sign before the two *EYESEP* entries in the camera description. The render command for the left view was *pvmpov +Iskyvase_l.pov +Oskyvase_l.png +h600 +w848 +v*. The process was repeated for the right view. 848 x 600 is certainly an abnormal dimension for an image. Because the viewing perspectives are different, the left eye sees area to the left of the area seen by the right image and the right eye sees area to the right of the left image. The solution is to trim off the left 48 pixels of the left image and the right 48 pixels of the right image. The area common to both images is the 800 pixels of each image forming the stereo pair [11].

Making a Composite of the Stereo Pair

The completed stereo pair could be displayed on the portable visualization workstation. This approach is preferred when a large audience needs to see the visualization, but not all researchers have access to these resources. An affordable alternative is the use of shutter glasses (approximately $80) [12]. VRJoy™ 2000 glasses provide a pass-through interface between the video card and the monitor of a workstation to display a stereo pair. First, the stereo pair is combined using a specialized program that reads the two images one line at a time and produces a third composite file made from alternate lines of the pair. Viewing this file looks very strange without shutter glasses.

Shutter glasses alternatively block the vision of each eye several times each second. If the viewer closes their left eye, they'll see the left side image and the opposite for the right eye. While wearing the shutter glasses, the power button is pushed and line-alternate mode is enabled on the controller. The flicker is noticeable if looking at a light. Now when viewing a line-alternate image, the lines for each side separate according to the shutter glasses flipping. Each eye sees the proper image, but in half the vertical resolution as the original stereo pair. It's possible that the image is on an odd pixel on the screen, so the controller also includes a button to flip left and right views.

While free programs for creating line-alternate images may exist thorough searching only found commercial programs. The visualization example used a DOS program from [13] after the PNG images were converted to 8-bit Windows System color palette bitmaps with Adobe Photoshop. The loss in image quality due to the conversion to 8-bit bitmap was quite disappointing. A goal for the near future it to write a program to composite stereo pairs with no image quality loss other than the 50% reduction in vertical resolution.

Lasting Impact of the Project

The cluster is currently serving as the platform for a separate senior project exploring colliding galaxies with MPI. The portable visualization station can be used to show 2D or 3D images and animation to large groups. Using Moray or another 3D modeler with POV-Ray scene file output, an ambitious art student could batch the cluster to create a series of images that could then be animated to form a short 2D or 3D movie. Future Graphics and Parallel Programming classes will be able to use the cluster for fast ray-tracing and comparisons of PVM vs. MPI on like equipment. Future senior projects may seek to add nodes to the cluster or radically expand the services offered to serve the interests of a wider student audience like the PIRUN project at Kasetsart University of Bangkok [14].

Remaining items for the project include a program to composite stereo pairs to be released under the GNU Public License and writing user's and administrator's manuals to be hosted on the departmental web server.

References

[1] Merky, P. (1998). "Introduction to the CESDIS Beowulf Project". Retrieved February 18, 2002 from the World Wide Web: http://www.beowulf.org/intro.html

[2] Becker, D., et. al. (1995). *Beowulf: A Parallel Workstation for Scientific Computing*. Retrieved February 18, 2002 from the World Wide Web: http://www.beowulf.org/papers/ICPP95/icpp95.html

[3] Craig, O. (1998). "Instructions for 'Cloning' a Linux/Win95 Dual Boot Machine". Retrieved September 24, 2001 from the World Wide Web: http://www.cs.umass.edu/~olc/linux_clone.txt

[4] Kirch, O. and T. Dawson. (2000). Linux Network Administrator's Guide, Second Edition. Sebastopol: O'Reilly.

[5] Geist, A., et. al. (1997). PVM: Parallel Virtual Machine. Cambridge: MIT Press.

[6] Ramsey, P. (2000). "Red Hat Linux 6.x as an Internet Gateway for a Home Network." Retrieved September 26, 2001 from the World Wide Web: http://www.coastnet.com/~pramsey/linux/homenet.html

[7] Muday, J. (2000). "A Simple Linux Router". Retrieved February 13, 2002 from the World Wide Web: http://www.wfu.edu/~mudayja/router.html

[8] Dilger, A., H. Deischinger and Brad Kline (2001). PVMPOV How-to. Retrieved October 24, 2001 from the World Wide Web: http://pvmpov.sourceforge.net/PVMPOV-HOWTO.pdf

[9] Rule, K. (1995). 3D Stereo Pairs with POV-Ray. Retrieved February 25, 2002 from the World Wide Web: http://www.povray.org/povzine/povzine2/3d.html

[10] Bourke, P. (1999). Calculating Stereo Pairs. Retrieved March 21, 2001 from the World Wide Web: http://astronomy.swin.edu.au/~pbourke/stereographics/stereorender/

[11] Bourke, P. (2001). "Creating Correct Stereo Pairs From Any Ray-tracer." Retrieved February 28, 2002 from the World Wide Web: http://astronomy.swin.edu.au/~pbourke/povray/raystereo/

[12] We used the VRJoy™ 2000 Set, including shutter glasses and a VGA pass-through port from VR Standard Corp. Purchase online at: http://www.vrstandard.com

[13] Akka, B. (1996). SimulEyes VR Stereo Combiner Version 1.0. Retrieved February 27, 2002 from the World Wide Web: http://www.stereographics.com/support/developers/comp32.zip

[14] Uthayopas, P., S. Sanguanpong and Y. Poovarawan (2000). *Building a Large Scalable Internet Superserver for Academic Services with Linux Cluster Technology*. Retrieved December 27, 2001 from the World Wide Web: http://pirun.ku.ac.th/newhtml/building-superserver.pdf

Acknowledgements

First, I want to thank to the Maytag Foundation for their research grant and the MCSP Department for covering non-budgeted equipment costs. I'd like to thank Dr. Josef Breutzmann for his pep talks, emails, and positive attitude and Dr. John Zelle for his assistance with the network driver module problems, grant writing, and miscellaneous Linux questions through the years.

The most helpful documentation came from Eugene Kuznetsov who (re)posted information about the 3CSOHO100-TX at http://lhd.datapower.com/db/dispreport.cgi?DISP?151. A very close second goes to Paul Bourke of Swinburne University of Technology in Melbourne, Australia whose documents about stereo pairs are accurate and accompanied by very helpful diagrams.