Small-College Supercomputing: Building A Beowulf Cluster At A Comprehensive College†

Joel Adams **David Vos**
Department of Computer Science
Calvin College
Grand Rapids, MI 49546
{adams, dvos12} @calvin.edu

Abstract

A Beowulf cluster is a MIMD multiprocessor built from commodity off-the-shelf personal computers connected via a dedicated network, running free open-source software. Such a cluster can provide a supercomputer's performance at a small fraction of one's cost. For small colleges and universities, the relatively low cost of a Beowulf cluster makes it an attractive alternative to a commercial supercomputer. This paper details our experience building a Beowulf cluster at a four-year comprehensive college.

1 Introduction

In the early 1990s, NASA Goddard Space Flight Center researchers Donald Becker and Thomas Sterling needed more computing power, but funding for their projects was decreasing. They saw themselves as needing to be "liberated" from supercomputer vendors, for three reasons:

- Commercial supercomputers were very expensive.
- The proprietary (binary) software that came with commercial supercomputers could not be customized.
- The small market for supercomputers combined with the high R&D costs to develop them was driving most supercomputer vendors bankrupt — voiding their maintenance contracts and making upgrades impossible.

To gain their "liberty," Becker and Sterling built their own supercomputer by turning "a pile of PCs" into a MIMD multiprocessor, using ethernet and free open-source software (e.g., Linux, MPI [6], PVM [5]). To reflect their "liberation" theme, they chose names for their machines from the medieval epic in which Beowulf liberated the Danes from the monster Grendel. Such multiprocessors have since come to be known as *Beowulf clusters* [3][8].

†This work was supported by NSF grant MRI-0079739.

Becker and Sterling's goal was to achieve at least 1 Gflop performance for roughly $50,000. Their 1994 cluster — *Wiglaf* — consisted of 16 100-MHz 486 DX-4 PCs, connected with 10-Mbit/sec ethernet in a triple-bus topology, and customized Linux ethernet drivers to spread the communications traffic evenly across the three networks. Wiglaf achieved a top speed of 42 Mflops.

Their 1995 attempt — *Hrothgar* — substituted 100-MHz Pentium PCs for the 486s, and 100-Mbit/sec ethernet for the 10 Mbit/sec ethernet. These modifications improved Hrothgar's performance to 280 Mflops.

In 1996, Mike Warren from Los Alamos National Labs built *Loki* [10] out of 16 200-MHz Pentium Pro PCs connected with 100 Mbit/sec ethernet using a hybrid star and hypercube topology. Loki cost $63,000 and achieved 1.2 Gflops. Within a year, the price for its components had dropped to $28,000, making it one of the first clusters to achieve 1 Gflop for less than $60,000.

Since that time, the decreasing prices of components have made it feasible for institutions, departments, and even individuals to build their own Beowulf clusters that achieve a supercomputer's performance for a fraction of one's cost.

Figure 1 shows the growth in cluster-numbers over time:

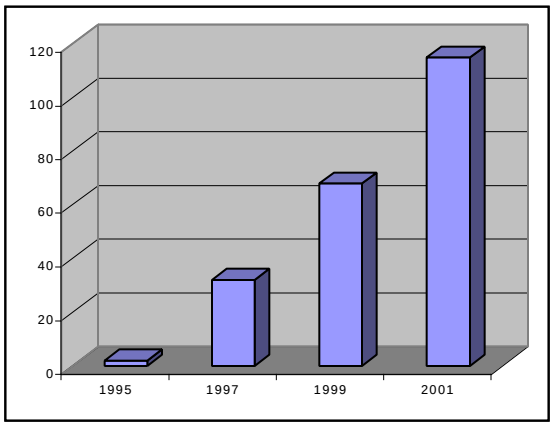

Figure 1. Clusters at www.beowulf.org by Year.

2 Preliminary Work

In 1998, we taught a *Parallel Computing* course using the only available facility — our department's network of workstations (NOW). On the NOW, student programs had to compete with system processes and each other for CPU and network bandwidth, making it difficult for them to experience any of the benefits of parallelism.

Unlike a network of workstations, a Beowulf cluster is dedicated to the computation it is running, which:

- Minimizes contention for CPU cycles and network bandwidth.
- Eliminates much of the system overhead (e.g., network services, protocol redundancies, security features) that is a necessity on a network of workstations.
- Permits special tuning of system services used by parallel processes (e.g., process migration, communication).

Early in 1998, we began to think about how we might build a Beowulf cluster.

In mid-1998, one of our students received a grant of dozen cast-off 486 PCs and a 100 Mbit/sec ethernet network. He cannibalized the 486s to build an 8-node cluster with a hybrid star-hypercube topology for his senior project, which he named *MBH'99* [2]. The CPUs in his machine were too slow to be very useful, but his work taught us the basic steps one must follow to build a Beowulf cluster:

1. Design your cluster (based on the available funding).
2. Acquire the hardware for your cluster.
3. Install, configure, and tune the software for your cluster.

The remainder of this paper describes these steps in detail.

3 Designing And Funding Beowulf

Designing a cluster requires one to decide on its software and hardware. Since the hardware usually costs money, one's hardware decisions depend upon the level of funding that is available.

Even if one has little or no funding, it is still possible to build a Beowulf cluster if your institution (or a friendly local company) replaces its PCs on a regular basis. For example, the authors' institution replaces its machines every three years, and at this writing is replacing computers with 300MHz Pentium-II CPUs, 64 Mb RAM, and 3Gb disks. A Beowulf cluster consisting of 16 of these machines has a peak performance of 4.8 Gflops, and so might achieve about 3 Gflops measured performance.

3.1 Software

One of the reasons Beowulf clusters are so popular is because they use free, open-source software. Both the free and open-source features are necessary to be "liberated:"

- *Free* software permits you to stretch your budget, and spend it on items that are not free (i.e., hardware).
- *Open-source* software lets you modify the code (e.g., write custom drivers) if you need it to behave differently.

For these reasons, virtually every Beowulf cluster uses a free Unix-like operating system (e.g., Linux, OpenBSD, etc.) The vast majority of clusters use Redhat Linux, perhaps because of its relative ease of installation. We decided to follow suit, because most clustering software is written and distributed in Redhat's RPM format.

For parallel execution, most clusters use free, open-source implementations of MPI, the message passing interface [6]; and PVM, the parallel virtual machine [5].

3.2 Hardware

When it comes to hardware, there are two broad categories of decisions to be made:

1. What hardware you want for your cluster's *nodes*; and
2. How your nodes will communicate (their *interconnect*).

We will treat each of these separately.

Nodes. The vast majority of Beowulf cluster nodes have x86 CPUs (e.g., Intel, AMD) that were current when the clusters were built. The clock speeds of the CPUs have increased over the years, in keeping with CPU evolution. Main and secondary memory sizes have also increased over the years, as the price/bit has dropped.

Many clusters use the *processor farm model*, where a *master node* distributes the work of a computation to many *server nodes*. Each server does the job it is given, sends its results to the master, and awaits more work. To allow it to keep multiple servers busy, a master node may need a faster CPU, more main and secondary memory, and more network bandwidth than the server nodes.

Since it basically just serves CPU cycles, a server node can be much simpler than a master. Many clusters use diskless servers that boot from the network, and provide each server with enough RAM that it "never" has to swap. Similarly, servers do not need CD-ROM drives, as software installation can be performed via the network.

In order for both faculty researchers and students to use our cluster, we decided to build a cluster that could run in 'single-user mode' with 1 master and 16 servers (for faculty researchers); or run in 'two-user mode' with two masters each controlling eight non-overlapping servers (for student users). We thus planned for a cluster of 18 nodes.

Node Acquisition. There are two basic ways to acquire the nodes for a Beowulf system:

a. Acquire commodity off-the-shelf *preassembled systems* (e.g., Dell workstations, IBM servers, etc.) as nodes; or
b. Acquire commodity off-the-shelf *components* (CPUs, motherboards, disks, …) and build the nodes yourself.

If you can find a preassembled system that matches your needs, that route is the easier approach. The drawbacks to

it are (i) you are limited to the off-the-shelf configurations of a given vendor; and (ii) you get a bit less for your money, as you're effectively paying someone else to build your nodes for you. Those who choose this route often choose a high-end system for their master node(s) and a mid-level system for their server nodes.

Buying raw components lets you spend your money on the exact hardware you want, rather than a vendor's standard configuration. Its drawback is that it is *much* more work: you must (i) identify the precise components you want; (ii) find a vendor who will give you a good deal on a given component; and (iii) assemble your nodes.

Even though it was more work, we chose the latter approach because our students wanted to design and build the nodes. Their (free) labor let us buy better hardware than would otherwise have been possible.

For our proposal's budget, we listed a representative high-end configuration for each master costing roughly $5500, and a representative mid-level configuration for each server costing roughly $2200, with the understanding that the systems we would actually purchase would likely be different due to hardware evolution.

Interconnect. When considering how to connect a cluster's nodes together, there are two decisions to be made:

a. What *kind of network* you will use; and

b. Your network's *topology*.

The goal is to purchase the fastest network you can afford.

Kind of Network. Most Beowulf clusters use a switched ethernet fabric, though a few use ATM or Myrinet networks. Fast ethernet (100Base-T) is most common, as Gigabit ethernet (1000Base-T) is just becoming affordable.

In many parallel computations, network latency is the primary bottleneck in the system, so a fast network is desirable. How fast a network you buy (e.g., Fast ethernet vs. Gigabit ethernet) is largely determined by your budget, as a Gigabit switch and NICs can be expensive.

Because we were designing a cluster for multidisciplinary use, at least some of our users' computations would require high bandwidth. In our proposal, we budgeted for Gigabit ethernet to accommodate the needs of these users.

Topology. Choosing a topology depends on (i) the kinds of computations your cluster will be running, (ii) the kind of network in your cluster, and (iii) how much money you have to spend.

A cluster's bandwidth-needs depend on its computations. To illustrate, the Stone Soupercomputer [7] is a 133-node Beowulf cluster used for various geographic mapping computations that require relatively little communication. It uses ordinary (10Mbit/sec) ethernet and a simple bus topology as its interconnect. However such computations are the exception rather than the norm; most computations require far more bandwidth than this.

Instead of a bus, the vast majority of Beowulf clusters use a star topology, like that shown in Figure 2:

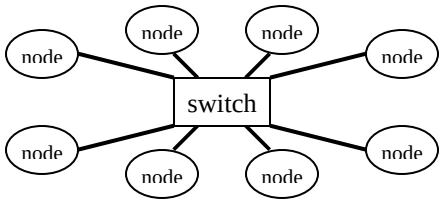

Figure 2: Star Topology Cluster

However, since network latency is often the bottleneck in a computation, some clusters augment this star with a ring, producing a star-ring hybrid as shown in Figure 3.

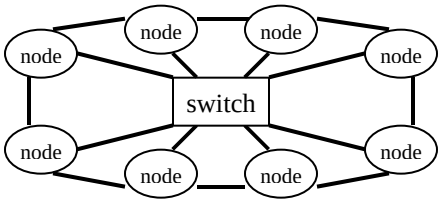

Figure 3: Star-Ring Hybrid Topology Cluster

In theory, the additional bandwidth of these extra links allows each node to communicate with two others directly, reducing traffic through the switch, reducing latency. Some clusters add even more bandwidth by augmenting the ring with a hypercube, as shown in Figure 4:

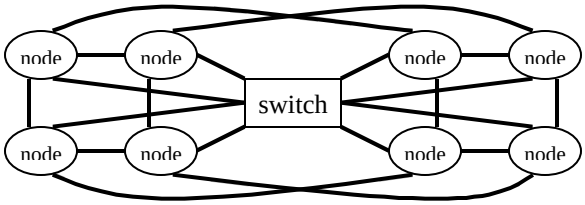

Figure 4: Star-Hypercube Hybrid Topology Cluster

Other, more elaborate topologies also exist, especially for clusters of more than 48 nodes.

The issue is complicated by manufacturers, who claim that their "smart switches" allow *n/2* of a cluster's *n* nodes to communicate with the other *n/2* simultaneously. If this is true, it is unclear whether the extra connectivity of a ring or hypercube is worth the expense.

We decided that this was a question worth exploring. During the first year of operation, our cluster will use a star topology; during its second year, we will supplement the star with a ring; and during the third year, we will supplement the star with a hypercube. Our project is to study the effects of different topologies on different kinds of computations, to try to determine which topology provides the best price/performance ratio for a multidisciplinary cluster [1]. The results of this study will be the subject of a future report.

In 1999, we submitted a proposal to the National Science Foundation's Major Research Instrumentation (NSF-MRI) program to build the cluster described above. That proposal was not funded, but was judged to be competitive and we were encouraged to revise and resubmit in 2000. We did so, and our 2000 proposal was funded.

4 Acquiring The Hardware

About eight months elapsed between when we submitted our revised proposal and when we learned it had been funded. Because of the drop in hardware prices during that time, we were able to purchase hardware that was superior to that of the representative systems in our proposal.

4.1 Hardware: The Nodes

For each server node, we purchased these components:

Item	Price: 1/01
Tyan Trinity S2390 Motherboard	$112
AMD Athlon 1 GHz CPU	$199
Mushkin 1Gb PC-133 ECC SDRAM	$1,186
IBM 20Gb 7200 RPM Hard Disk	$123
Trident 975 4 Mb AGP Video Card	$22
Teac Floppy Drive	$11
Rackmount case and hardware	$380

Our master nodes were identical, except that they added:

48-X CD-ROM Drive	$32
Jaton 3Dforce 32Mb AGP Video Card (in place of the Trident card)	$60
Viewsonic GS815 Monitor	$825
Logitech Optical Mouse and Keyboard	$38

Our research indicated that AMD's Athlon CPU offered the best price/performance value. To build a star-hypercube hybrid topology, our motherboard needed six PCI slots. For reliability, we also wanted it to support error-correcting (ECC) RAM. Our Tyan was the most reputable Athlon motherboard with six PCI slots and ECC RAM support.

We chose Mushkin memory because of its good price and reputation for reliability. We chose the IBM disk drives for much the same reasons, and the video cards for their degree of Linux support.

We added two Belkin OmniView Pro keyboard-mouse-video switches (KVMs) so that we could control any node from a single monitor, keyboard, and mouse ($1,112).

Our IT department donated antistatic equipment to facilitate assembling the cluster's nodes. Bribed by pizza, 20 students, faculty, alumni, and administrators spent two enjoyable snowy winter evenings together, building our 18 nodes, and then racking them and the switch.

4.2 Hardware: The Network

For our switch, we chose the Extreme Networks Summit 7i 32-port Gigabit Ethernet switch ($16,796), due to its relatively low price/port ratio.

Each node also needed a network interface card (NIC) to connect to the switch. For this component, we chose the Syskonnect SK-9821 Gigabit Ethernet NIC ($482). Figure 5 shows our cluster's structure during year 1 of our project:

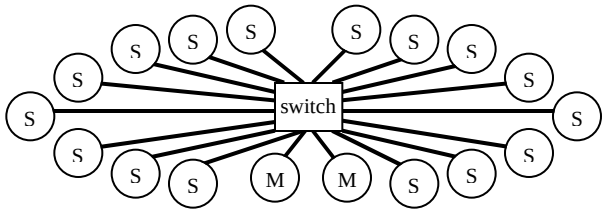

Figure 5: Our Cluster Year 1

Tripp-Lite Corporation generously donated the category-5 enhanced (CAT-5e) cabling for our Gigabit ethernet.

To guard our cluster against miscreants, we recycled a 200-MHz Pentium PC ($0) as a firewall. We purchased Intel EtherExpress Fast Ethernet NICs ($25) to link our master nodes to our firewall, and our firewall to our campus.

4.3 Hardware: Miscellaneous Items

To enhance our cluster's useability, performance, and reliability, we also purchased these items:

3 Mid-Atlantic Racks	$1635
3 Tripp-Lite Smart 2200 RMXL UPS units	$2,877
On-Stream Tape Drive (backups) + tapes	$582

A small room was available to house our cluster. This room's size was the major factor in our decision to rackmount our nodes, since less expensive casing options require considerably more space. Our budget included $10,000 to upgrade the power and climate-controls in this room to accomodate our cluster.

We plan to add a redundant array of inexpensive disks (RAID) to our masters, to avoid data loss on a disk failure.

5 Installation, Configuration, and Tuning

Some of NASA's original Beowulf researchers have started selling *ScyldOS*, the Scyld Beowulf Cluster Operating System [4]. ScyldOS is Redhat Linux preconfigured for a cluster, including a custom version of MPI. ScyldOS is commercial, open-source software (i.e., the source code is free, but you must pay for the documentation).

We experimented with ScyldOS at the outset, but decided to go with plain Redhat Linux (RHL) because the ScyldOS customizations presupposes that every cluster will have (i) a single master (i.e., no "two-user mode"), and (ii) use a star topology. RHL does not have such presuppositions.

5.1 Installation and Configuration

We began by building our firewall. We installed Redhat Linux 7.1 (RHL), used *iptables* to deny all incoming packets except those for ports 22 (*ssh*) and 80 (*www*), and disabled all unneeded network services.

Behind the firewall, we used a serial connection to configure our switch as a class C network switch; using the static IP addresses 192.168.1.1–18 for our nodes.

We then installed RHL on a master, using a custom installation to install only needed packages. We then configured our master as an NFS server.

To install RHL on the remaining nodes, we used the Redhat Kickstart utility for booting nodes across a network. Using *mkkickstart*, we created a file (*ks.cfg*) of the configuration information common to each node — partitions, packages, and so on — which we then copied onto a boot floppy containing the *netboot* image.

We then booted each node with this floppy. The *netboot* image prompted us for various network-install parameters, including the IP address of the NFS server, and the location of the Redhat installation CDs on the NFS server. The netboot image then installed RHL from the master to that server, in roughly 7.5 minutes (faster than via CD-ROM).

To configure each node, we gave it its static IP address, and then set up the Berkeley *rsh* service (needed by MPI).

With the nodes configured, we then installed MPI and PVM, plus commercial software purchased by faculty researchers for their specific computations.

5.2 Testing and Tuning

To get a rough idea of our cluster's performance, we used the popular *Linpack* benchmark [9]. After installation, we achieved 3.8 Gflops using 1 master and 16 servers. This was disappointing, given our theoretical peak of 17 Gflops.

We then began tuning the cluster: reconfiguring *gcc* to compile for the 686; recompiling our kernel and MPI: enabling direct memory access (DMA) via *hdparm*; setting Linux to use 100% of RAM, and so on. After three weeks of tuning, we achieved a respectable 10.4 Gflops.

6 The Future: Using the Cluster

Our computer science faculty plans to use the cluster for research on object-oriented parallel languages, recursive matrix algorithms, network protocol optimization, and graphical rendering. Several non-CS faculty also plan to use the cluster for computational modeling, including a physicist modeling electron behavior in high-energy laser fields, a chemist modeling complex inorganic molecules, and an astronomer modeling the interaction of Saturn's rings and troposphere. Computer science students will receive extensive experience using the cluster in our *High Performance Computing* course, and we expect that several seniors will want to use it for their senior projects.

7 Final Thoughts

Building our cluster required a great deal of time. Thankfully, NSF's MRI program supports such startup costs and provided funding for the authors. We are very grateful to the NSF for their support in our project.

Roughly 30 different individuals — students, faculty, administrators, and alumni — have contributed to the construction of our cluster. The project of building a supercomputer has captured the imagination of our campus, and has provided an enjoyable focus this past year.

As hardware continues to evolve, cluster technology is increasingly affordable. We hope that this report and the documentation available at [1] and [8] will encourage other institutions to build their own clusters.

References

[1] Adams, J., *The Ohm Project*, Calvin College, 2001. Online. Internet. [Aug. 15, 2001]. Available WWW: http://cs.calvin.edu/research/ohm/.

[2] Adams, J., Laverell, D., Ryken, M. MBH'99: A Beowulf Cluster Capstone Project, *Proceedings of the 14th Annual Midwest Computer Conference*, Whitewater, WI, March 2000.

[3] Becker, D., Sterling, T., et al. Beowulf: A Parallel Workstation for Scientific Computation, *Proceedings, International Conference on Parallel Processing*, Oconomowoc, Wisconsin, Aug. 1995, p. 11-14.

[4] Becker, D., et al. *Scyld Beowulf Cluster Operating System*, Scyld Computing Corporation, 2001. Online. Internet. [Aug. 15, 2001]. Available WWW: http://www.scyld.com/.

[5] Geist, A. *PVM: Parallel Virtual Machine*, Oak Ridge National Laboratories, 2001. Internet. [Aug. 15, 2001]. Available WWW: http://www.csm.ornl.gov/pvm/.

[6] Grop, W., Lusk, E. *The Message Passing Interface (MPI) Standard*, Argonne National Laboratories, 2001. Online. Internet. [Aug. 15, 2001]. Available WWW: http://www-unix.mcs.anl.gov/mpi/.

[7] Hoffman, F., Hargrove, W. *The Stone Soupercomputer*, Oak Ridge National Laboratories, 2001. Online. Internet. [Aug. 15, 2001]. Available WWW: http://stonesoup.esd.ornl.gov/.

[8] Merkey, P. *The Beowulf Project*, Scyld Computing Corporation, 2001. Online. Internet. [Aug. 15, 2001]. Available WWW: http://www.beowulf.org/.

[9] *The Netlib Repository at UTK and ORNL*, Oak Ridge National Labs, 2001. Online. Internet. [Aug. 15, 2001]. Available WWW: http://www.netlib.org/linpack/.

[10] Warren, M. *Loki — Commodity Parallel Processing*, Los Alamos National Laboratories, 2001. Online. Internet. [Aug. 15, 2001]. Available WWW: http://loki-www.lanl.gov/.