# 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.

### 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:



120

100

80

60

40

20

0

1995

1997

1999

2001

### Figure 1. Clusters at [www.beowulf.org](http://www.beowulf.org/) by Year.

1. **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.

### 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.

* 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].

* 1. **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:

* + - 1. Acquire commodity off-the-shelf *preassembled systems*

(e.g., Dell workstations, IBM servers, etc.) as nodes; or

* + - 1. 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

Thank you for using [www.freepdfconvert.com](http://www.freepdfconvert.com/) service!

Only two pages are converted. Please Sign Up to convert all pages. <https://www.freepdfconvert.com/membership>