;login:

THE MAGAZINE OF USENIX & SAGE

August 2001 • Volume 26 • Number 5

Special Focus Issue:Clustering

Guest Editor: Joseph L. Kaiser

inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

home clusters

by Andreas Boklund
Andreas Boklund is a lecturer at the Department of Informatics andMathematics at the University of Trollhattan-Uddevalla,Sweden. He has also constructed, and remodelled a cluster for thermal spray and welding simulation research for the University's Department of Technology.

andreas@boklund.nu

This article is a description of the cluster that I have assembled in my home and why I did it. One of the most interesting questions is: Why would anyone want to have a cluster of computers in their home?

I have been interested in computers since my parents bought their first IBM PS/2 and I started to learn how to write simple BASIC programs. I have been programming ever since. Nowadays, I do most of my development work in C, although I occasionally write small hacks in other languages.

Back in 1997 I read an article about the Beowulf project, where NASA researchers managed to create a cluster of workstations that was so powerful that it could compete with the "supercomputers" of that time, especially in terms of power to price. The idea of parallel computing and clusters of computers is not new – it had been practiced for decades. The thing that made Beowulf clusters interesting was that they used standard PC parts that could be bought from any computer vendor. Ordinary hardware and the (then) new operating system Linux were used. For me, a computer science student, it was a dream come true. Now I could finally harness the power of a "supercomputer" in my own home. The only question was what would I use such a beast for?

The most tedious tasks that I know are to sit and wait for code to compile or for a movie to be compressed. Compressing a movie is not really a problem; it takes a long time but it can always be run overnight. Compiling is another story; when I compile something, I do not want to contemplate my code for a few minutes or even for a few seconds before being able to see if it works; maybe I am just an impatient person. The basic idea behind my home cluster was to be able to shorten the execution times for compiling and rendering, although I would not mind if the execution times of other programs could be speeded up as well.

Creating a powerful cluster is not an unworkable task if you are skilled in UNIX/Linux administration, understand the basics of parallel computing, and have proper funding. A harder task is to create a cluster with the limited resources of your home. I basically have three limiting factors: my budget, the space that the computers require, and my girlfriend (time). Based on the given factors, I constructed a cluster.

My personal cluster consists of three computers: my workstation, Phoenix, which is equipped with two Celeron 500MHz processors and 128MB of RAM; my girlfriend's workstation, Sunrise, which has a Celeron 466MHz processor and 96MB RAM; and our file server and Internet router, Sabrina, which has a Pentium II 350MHz processor and 128MB RAM. Phoenix and Sunrise have two Fast Ethernet cards each and Sabrina has three.

The network topology used is both simple and cheap, at least as long as it is used in a small network. All computers are connected through a dedicated network interface card to the other (two) computers by a crossover of twisted pair wire. Theoretically this means that all computers can send and receive information to the other two at the same time, at the maximum speed of a Fast Ethernet network card. As an extra bonus, the latencies on the network are lower than in a switched or hubbed network. The reason for this is that the data does not have to be handled by a switch or a hub, it goes straight from network card to network card. The extra, third, network card in Sabrina is connected to the Internet through which Sabrina routes all incoming and outgoing traffic. Sabrina also masquerades the IP addresses of Phoenix and Sunrise, which are from one of the free IP ranges.

The operating systems used on all three computers are various RedHat-based Linux distributions all running a Linux 2.2.x series kernel. The kernels have been patched with the MOSIX kernel enhancements. The MOSIX project was founded in the early 1980s at a university in Jerusalem. MOSIX extends the functionality to the kernel and allows it to move (migrate) already running processes to other computers that are running a MOSIX-patched Linux kernel. MOSIX is a transparent and universal tool for moving running processes to other computers. The only items the user has to specify are the IP addresses of the computers that the operating system is allowed to move processes to. MOSIX does not know what the process is doing and it does not care; it uses a set of algorithms to decide if a process would benefit from being migrated or not. If a process is doing a lot of I/O it will not be moved since all I/O operations need to be performed on the computer that the processes were initiated on, but if the process is doing a matrix multiplication it might benefit a lot from being moved.

MOSIX is a transparent and universal tool for moving running processes to other computers

I use an extension to MOSIX called MPMake. It is a patch for GNUmake that allows several makes to be spawned over a MOSIX cluster. To make use of MPMake you have to place your source code on a shared volume and mount it on the computers that you want to compile it on. When you compile your code, you use the MOSIX-patched version of GNUmake and specify the number of processes that you want it to use. So how is the performance of MPMake compared to an ordinary make? As always when it comes to computer architectures, it depends on your application and the task that it tries to perform. The performance gain will be different depending on which program you are compiling. I did some measurements a while ago with MPMake on a MOSIX cluster of Intel Pentium III 500MHz computers, compiling a Linux 1.2.14 kernel. When using two processors we got a performance gain of 43% and with eight processors the performance gain was 79%; although this system had been hand-tuned to lower compilation time, that’s not too bad! MOSIX and MPMake can be downloaded from their home page, *www.MOSIX.org*. You will also need the corresponding Linux kernel source.

MOSIX is a good tool for many issues, but it has a high demand for bandwidth and does all its I/O through the node that initiated the process. Therefore, I also installed the message-passing libraries MPICH and PVM. They were developed at Argonne and Oak Ridge National Laboratories, respectively. MPICH is an implementation of the MPI standard, which is used on a wide range of different architectures, from two-way SMP machines to large clusters and Cray supercomputers. The message-passing libraries do not work in the same way as MOSIX. When you use MPI or PVM, you have to write the program against one of the libraries. This approach is more time-consuming, and you need the source code; on the other hand, the application will execute faster. Most scientific applications that can run on clusters make use of either MPI or PVM. Message-passing libraries are faster because they start processes on other computer nodes and the processes run there, communicating with the other processes. Nowadays, I don’t run many programs at home that use either MPICH or PVM. When I occasionally compress a movie sequence, I use MPICH and the Berkeley MPEG compressor.

So what does this cluster do for me? It lowers the time that is used for compiling by approximately one-third. It also speeds up the execution of all applications that can use more than two CPUs at a time, as long as they do not use a shared memory segment since my setup does not allow shared memory between computers. It also lowers the time that it takes to compress movies.

What did it cost me? I already had all the hardware except for the three extra network cards, and now I do not need a hub or a switch. It did not take me long to set it up correctly, although it took me a while to learn how to do it. But as long as you are learning, the time is well spent.