CSC 205 – Cluster Computing Project Overview

In the Cluster Computing Project, the CSC 205 class will study and then implement the first level of a Beowulf cluster computer. The students will learn by doing information related to networking, parallel architecture and the use of a cluster. If time permits, additional networking topics may be covered which are not essential to the cluster operation but make the networking portion of it more efficient. The goal of the project is to implement the cluster and run a simple parallel program on it.

What is a cluster computer?

A twist on the idea of multiprocessor machines is the cluster computer. The cluster computer uses separate machines connected with a high speed switch to combine the processing power. It is a variation of a multiprocessor system. The cluster depends on the networking features to implement parallelism, that is the capability of a computer system to run more than one process at a time.

The Beowulf Cluster

The Beowulf cluster (www.beowulf.org) is a popular version of this architecture that uses commodity off the shelf systems (COTS) that have outlived their usefulness for stand-alone functions. Of course, using new machines in a cluster computer can result in great performance and high computation power. The cluster architecture starts with a network design and then adds special software to share the processing among the several machines. This can be accomplished by the use of message passing protocols to help the machines share the computational load.

Setting up a network with Linux

Setting up a good network begins with a basic machine install. The examples discussed here will use a Linux machine. Other platforms may be possible but are not considered here. The setup task may be greatly enhanced if tools such as PVM, MPI and the required compilers, as well as the networking components described below, are included in the initial setup so they don’t have to be loaded at a later time.

Identifying machines to the network should be done as a private network and follow the addressing conventions of RFC 1918 (http://www.faqs.org/rfcs/rfc1918.html). This sets up specific IP addresses that can be used for a network not connected to the Internet. These will be typically connected together using Ethernet and a fast switch (100 megabits per second or 1 gigabit per second). It is not recommended that a hub be used in place of the switch. Hubs result in collisions in the message passing and, since a cluster is built on message passing, performance on complex problems that require a great deal of message passing will be diminished.

Sharing folders on the network is accomplished by use of the Network File System called NFS (http://nfs.sourceforge.net/). There are two types of folders to set up. First, it is useful to have a read-only folder on the server that is shared to all other machines. This makes passing software and important documents to the other machines easier. This can be used to contain
implementation notes and histories, which will be valuable as the project progresses. Second, there will be user folders associated with network accounts. These should follow the user from machine to machine as they log in. Both of these will require a look at FSTAB, the file system table, to make certain access is properly set on the machine from the NFS server.

While the home folders are done with NFS, creating network accounts to link those folders is done with Network Information Server, NIS (http://www.linux-nis.org/). Client machines get the list of available accounts from the NIS server to allow login to the network. The FSTAB link will bring up the user’s home folder on the server when setup is proper and login complete. With the machines set up and the network running, there may be issues related to firewalls. Especially in a cluster, firewalls may limit some traffic that is necessary. However, if machines are connecting to the Internet the firewalls will be required in some way. One possible solution is to keep the private addressing scheme (RFC 1918) and connect the individual machines to the Internet through another machine that is acting as an edge router. This machine would have two network cards in it, one that connects to the inside private network and the other that connects to the Internet. Machines get to the Internet by using the IP address of the edge router as their gateway. This edge router uses IP masquerading (in Cisco terms network address translation) to convert inside requests to Internet access. This edge routing machine can have a full firewall and since it is not used for computation or serving on the node, neither the firewall nor Internet access will affect performance on the cluster. Information on IP masquerading is available at http://www.faqs.org/docs/linux_network/x-087-2-ipmasq.html

Turning the network into a cluster computer
Turning the network into a cluster is the process of making data and access sharing automatic and a message passing system to operate the independent machines in a coordinated fashion. The sharing is done by using secure shell for remote access. The message passing is accomplished through a variety of programs. Two examples in a great deal of clusters are Parallel Virtual Machine (PVM, http://www.csm.ornl.gov/pvm/) and Message Passing Interface (MPI, http://www-unix.mcs.anl.gov/mpi/) for message passing and processing. These programs help the server program initiate other tasks and pass data to them, as well as the collection of results. The work of the cluster may also be dependent upon shared files that all the nodes can access.

PVM, MPI and message passing
The choice of PVM versus MPI for the messaging is not clear cut, although recent benchmarking and applications seem to be using MPI. Many clusters actually have both installed and developers can choose the one that suits there need or matches their technical skills.

Assessing how well the cluster performs is important, and it may be dependent upon the application being run. One possibility is to use a variation of Linpack (http://www.netlib.org/benchmark/hpl/), which helps identify how well the cluster is running by way of a benchmark test. Since it is built to run on MPI, the Linpack decision may force the message passing decision for the cluster to be MPI.

PVM does come with many Linux install sets and is a reasonable starting choice. There is a great deal of HowTo information on PVM available on the Internet.
Programming the Cluster

Programming the cluster is arguably the most important and most difficult part of the whole process. At the core of the programming is the idea of Parallel Programming. In this style of programming, parts of the program are written to be run at the same instant, possibly even on different processors or machines. Child processes can be initiated by a parent process and the necessary data is passed to that process. When the child process is complete or at other critical parts of the work, messages may be sent to other processes or the parent. This processing requires the programmer to think about what can be done at the same time and not in a linear style, which is the historical way programmers have been trained.

An introduction to programming with PVM can be found at [http://www.csm.ornl.gov/pvm/intro.html](http://www.csm.ornl.gov/pvm/intro.html). Information on MPI can be found at [http://www-unix.mcs.anl.gov/mpi/](http://www-unix.mcs.anl.gov/mpi/), which also includes links to papers discussing the differences it has with PVM.

The 2006 Cluster Computing Project at PDCCC

The Beowulf Cluster at PDCCC includes 14 different machines connected by a 100 megabit switch. The cluster has the machines setup to have 1 edge router, 2 servers, and 11 computer nodes.

The edge router runs Fedora Core 4 because that version is optimized for IP masquerading. It also runs the firewall and provides Internet access by being on both the private network (class B, 172.16.0.0) and the college network via DHCP. The edge router provides IP masquerading from the private network to the college network and the Internet.

The project did not originally contain two servers. The second one was the last element built applying all that had been learned during the project. The first one was actually the first machine set up, and was based on an earlier server on the initial Linux project. It ran NFS and NIS and was also the host for the server programs that were run through PVM.

PVM was the choice in the PDCCC cluster. PVM forced the group to assess the performance of the cluster by programming it, which in retrospect was a real advantage. The performance numbers meant more to the group because they were based on a program that was well understood. The disadvantage of this choice is that the speedup estimates are based on research and observation and not on a standard benchmark, although some individual machine benchmark information using Linpack was developed.

Finally, there are 11 compute nodes running PVM. Each of these is setup to allow secure shell login, which is automated to access alternate nodes running PVM. All machines run PVM, and all were similar machines.

The program that was developed used the C programming language and the PVM libraries. The process was a fully partitioned numerical method process to estimate a value for PI. The method uses an approach similar to the Monte-Carlo method, substituting a grid that can be varied to increase both the quality of the estimate and the number of calculations required. The
partitioning of the calculation is done on the server and then child tasks are initiated to do a specific part of the processes. The number of nodes to be used in the calculation is saved in a file, which allows the study of the relationship of number of processors to time required without the need to recompile programs or have separate versions of the program for each level of study.

The programs use libraries to record seconds and microseconds in the process. This is gathered during the run and tabulated to study the speedup that is gained in the parallel architecture. The study was done with a range of 1 to 11 processors (not including the server) and for problems of 100 million and 10 billion calculation cycles. The composite 11 child processor cluster showed sufficient speedup in the tests using a large problem to do the equivalent of a 13.95 gigahertz processing machine. This is based on the fact that the usual machine was a 1.7 gigahertz processor and the speedup in time required for 11 processors was 8.2 times faster than the same program ran on 1 processor. The results for a small problem were the equivalent power of a 13.17 gigahertz machine. The difference may be in that the smaller problem’s processing time requirement is closer to the speed of the communications where it is more likely to be a major factor in the overall result.

The 2010 Cluster Project

Students in the CSC 205 class will implement the new cluster. The new cluster project for the students is divided into the following labs:

- Overview (this document and the classroom discussion)
- Identifying resources and research (lab 1)
- Basic Linux operating system install (lab 2)
- Setup of required networking hardware (lab 3)
- Setup of required networking features (lab 4)
- Install of PVM (lab 5)
- An overview of a parallel program (lab 6)
- Configuring the cluster and running a simple program (lab 7)

If time permits in CSC 205, there will be additional labs on implementing the full set of networking features and on writing a parallel computer program to use the cluster. At the time of the writing those labs are under development and will be ready in the Spring semester of 2010.