Crossroads
The ACM Student Magazine

The Texas Tech Tornado Cluster: A Linux/MPI Cluster For Parallel Programming Education And Research

by ***Per Andersen***

Introduction

This article describes the implementation of the first Cluster of Workstations (COWS) at Texas Tech University using the Linux operating system and off-the-shelf personal computers and workstations [7]. The article describes the evolution of the Texas Tech Tornado Cluster (t3c) from its initial configuration to its current state as a research and educational tool in support of parallel process design and implementation. In addition, the article includes a case study of twelve students and their efforts to create their first parallel program on t3c.

Initial Cluster Hardware Configuration

The initial t3c cluster was built in the summer of 1996 by two CS doctoral students, and consisted of three workstations, or nodes. The equipment acquired for this cluster were cast-off 386s with Intel EtherExpress/16 and 3com 503 Network Interface Cards. The interconnection network was 10base2 TCP/IP. Dr. Antonio, Associate Professor of Computer Science, initiated this work to investigate the potential of Linux as an operating system (OS) for a multi-computer cluster. In August 1996 the author joined the effort and took over the project as a part of his master¹s thesis. At that time we were aware of the work behind standardizing MPI [3] and it was our goal to implement an MPI-based multi-computer cluster to support both research and teaching. Previously, the only parallel system CS had access to was a Maspar MP-1 SIMD machine. In the past year and a half, t3c has gone through a number of upgrades. About $1,300 was spent on the cluster during this period primarily on network cards and hard drives. The emphasis through out the construction of the Linux cluster was on low cost. With limited funds, startup projects like the t3c would normally receive very little money, but this was not a problem since part of the motivation behind the construction of a t3c was to demonstrate how little money a parallel system of this type

could be built for. In hindsight, even less could have been spent on the cluster. With little practical experience in configuring clusters it was felt that the best investment that could be made was in purchasing the best network cards and hard drives the budget would allow. It turns out that for low end cluster workstations like the 386/40s and 486/40s, high performance network cards (3Com 509 cards versus a 3Com 503 cards) and the 10baseT Ethernet switch we obtained provide little if any advantage over a 10base2 interconnection network. One lesson that can be stated with confidence: never reject a 386 or better system. For teaching parallel programming it¹s more important t o first have numbers in terms of cluster nodes than speed (although speed should not be ignored). Without numbers, the opportunities to study and learn about the behavior of a multi-computer cluster diminishes. Although there are definite advantages to a homogeneous cluster, for teaching purposes it¹s more of an advantage to have a heterogeneous cluster. With a mix of workstations of differing capabilities the students have the opportunity to learn about and deal with issues like load balancing. Therefore, any workstation that can be obtained should be considered for inclusion into a multi-computer cluster.

The first working cluster consisted of six workstations a Pentium 133, a 486/100, a 486/66, a 486/40 and two 386/40s interconnected via 10base2. The Pentium 133 had 32 Mbytes of RAM, the 486/100 had 20 Mbytes of RAM and the rest of the workstations had 8Mbytes of RAM. A few months into the project the Pentium 133 was traded for two Pentium 100s with 32Mbytes of RAM on each. The two Pentium 100s were Intel ExpressServer systems donated to the CS department. These two systems were to have been departmental NT servers but each had serious stability problems, which appeared to be associated with their system BIOS. The fact that Linux makes very little use of the system BIOS, once a system has booted, made these two systems ideal candidates for inclusion into the Linux cluster. Except for the occasional problem of getting one of the systems past the boot self test during power-up, both have been excellent additions to the cluster. The Pentium 133 that was traded away was configured with Linux and serves as the web/email/workgroup server for Dr. Antonio¹s **High Performance Computing Lab**.

Figure 1: Initial Linux Cluster

For the initial research conducted on the cluster, the cluster was installed in a Sun 4/280 cabinet as shown in Figure 1. The network hardware configuration consisted of 3Com 509 Combo cards that can be configured either in a 10base2 configuration or in a 10baseT configuration, and a 24port 3Com Superstack II Switch 1000 Ethernet switch [**1**] for the 10baseT configuration. The Ethernet switch was purchased by the CS department at half price and then donated to the Linux cluster project. The Superstack II switch was to have been an important part of the initial study of the cluster. The switch itself acts as a crossbar when the cluster is in the 10baseT configuration and it can be configured for several different packet forwarding modes: store-and-forward, Fast Forward (cut-through), Intelligent and Fragment Free. Only store-and-forward and cut-through were investigated during the cluster research phase. It was hoped that a significant improvement in cluster performance would be witnessed when the 10baseT network was compared to the 10base2 network, but this never occurred. Without the high performance workstations to drive the network, or better yet a larger cluster, the network overhead due to contention was negligible in practice. Larger message sizes might have made a difference, but the typical message sizes for the cluster tended to be around 200bytes or less, at least for the problem domain that we tested the cluster under.

The Linux OS was installed on each cluster workstation as if each system was an

independent multi-user system [9]. The distribution used was Slackware version 3.0 from Infomagic (April 1996 release). Without any prior experience on how to setup a cluster, it was felt that a standard Linux installation would be the best starting point. If any special configuration requirements were needed they could be implemented on each system after the OS was installed. The initial Parallel Software Development tool was LAM/MPI version 6.0 from the Ohio Supercomputer center [6]. The LAM/MPI development team is now located at Notre Dame University, **http://www.mpi.nd.edu/lam**. For further information on the initial research done on the cluster go to **http://hpcl.cs.ttu.edu/~andersen**.

Current Cluster Hardware Configuration

Between the summer of 1997 and the spring semester 1999, the cluster has undergone a number of configuration upgrades. The current t3c configuration is as follows; one dual Pentium 166 with 64 Mbytes RAM, two Pentium 133s with 16 Mbytes of RAM, four Pentium 100s with 32 Mbytes of RAM, one Pentium 90 with 16 Mbytes of RAM, one 486/66 with 20 Mbytes of RAM and four 486/40s with 8 Mbytes of RAM each. The interconnection network is the 3Com Superstack II switch. In addition LAM/MPI has been installed on eight Sun Sparc workstations, students are now able to complete projects on a heterogeneous cluster consisting of 22 nodes. The Sun systems are running Solaris 2.6.

The Slackware 3.0 distribution on the Linux cluster was replaced with Redhat Linux Extreme, which is now being replaced with Redhat 5.2.

Cluster System Administration

The configuration of the cluster with respect to the OS setup and system administration has also evolved. In the initial cluster configuration NIS [2] was avoided, because we felt that NIS for Linux was not very mature. An NFS directory server [2] was also not implemented because it was believed that NFS would create a bottleneck on the cluster. In fact, many standard system daemons were designated as unnecessary and therefore were never loaded on the Linux cluster. This included system daemons such as lpd, sendmail, httpd, atd, crond, syslogd, etc... Because the cluster was originally configured for users doing parallel processing research, these features were deemed unnecessary. During some of the early testing, even update was disabled for a period of time during which cluster benchmarking took place. This policy - keeping the number of system daemons to a minimum - continued on the cluster when the first group of students made use of it. Unfortunately this was only partially a good idea. Certainly the disabling of sendmail and httpd was justified but disabling syslogd and crond was not. Therefore, these daemons were enabled for the second group of students that used the cluster. Syslogd and crond are important tools in the account management and trouble shooting of the cluster.

The Control Workstation

A number of Linux cluster implementations designate one system as a control workstation (CWS). The designated system is responsible for account management, security management and software revision level management. The idea of setting up one system to act as a CWS was reviewed but deemed unnecessary for the initial cluster configuration. One of the primary advantages of the CWS is to serve as an NFS directory server but for reasons already stated, NFS was not implemented. The result of not having a CWS was that an account and home directory had to be created on each system for each student. This was done by copying the password file around the cluster and then by manually creating the home directories. A script was written that created the directories and set ownership and permissions on each cluster workstation, account names were not personalized in order to make account creation and management a little easier. This whole concept became a management nightmare as the number of cluster nodes and accounts increased. The time required to maintain this setup became unreasonable.

Without CWS the installation of a complete parallel development environment on each cluster node was unneccesarily difficult. The Pentium 100s and 486/66 were able to handle the load during parallel software development and testing, however, the 386s in the initial cluster could not. At one point, one of the 386's crashed. Unfortunately, the lack of system logs (no syslogd daemon etc.) prevented us from identifying the cause of the crash. The current cluster utilizes a control workstation in an NFS configuration. NIS is still not used, instead an automated file collection mechanism has been implemented which copies the key configuration files, such as the password file, to each cluster node once an hour.Cluster nodes are now built as bare bones networked workstations. To discourage users from logging directly into computational nodes, no development tools or other programs have been installed. Cluster administration has improved a great deal with these simple changes in configuration.

Students are able to log into the CWS using either telnet or XDMCP. The CWS, a dual Pentium 166 with 64 Mbytes of RAM, has never had any problems handling the load placed on it by the students, nor has it had any stability problems even though an early release of the Linux SMP kernel was installed.

Parallel Programming with a Linux Cluster

One of the goals for the cluster was as a tool to assist in teaching parallel processing and programming. A review of the current literature on COWS reveals that teaching parallel programming skills to groups of students is not a primary motivation for their construction [**8**]. Many clusters tend to evolve as single problem domain or "single task" parallel systems.

The motivation for building a cluster usually is derived from an individual or small group's need for more computational power. A simple way to derive this power with low expense is to collect or purchase a number of off-the-shelf low cost personal workstations. In addition to Linux, small cluster administrators often install freely available parallel software development systems or sets of parallel libraries. Typical cluster implementations include research in the following problem domains: N-Body simulations, parallel mpeg, Landscape Analysis, etc. Additionally, many of the cluster builders and programmers are experienced parallel systems programmers and administrators. A certain level of customization of the cluster takes place in order to meet the needs of the cluster users. These experienced users tend to become very familiar with any idiosyncrasies of the cluster they have built and are successfully able to accomplish their research on the cluster.

An important question is, can this type of cluster implementation provide an appropriate parallel environment in which the users or programmers who have no parallel programming background gain this experience? In order to gain a better understanding of how to configure and manage a cluster for teaching parallel programming a study was carried out on one group of students. This group consisted of twelve students with no prior parallel programming experience. The next section reports on the results and finding of this study.

Case Study On Cluster Utilization

A short one hour overview of the cluster was given and the MPI Primer/ Developing LAM was distributed to enable students to use the cluster rapidly. It was felt that the average student would be able to pickup message passing software development skills without any formal instruction and for the most part this was true. What was not anticipated was the difficulty a small number of students would have going from the MS Windows/Integrated Development Environment Programming Tools to the UNIX command line development environment.

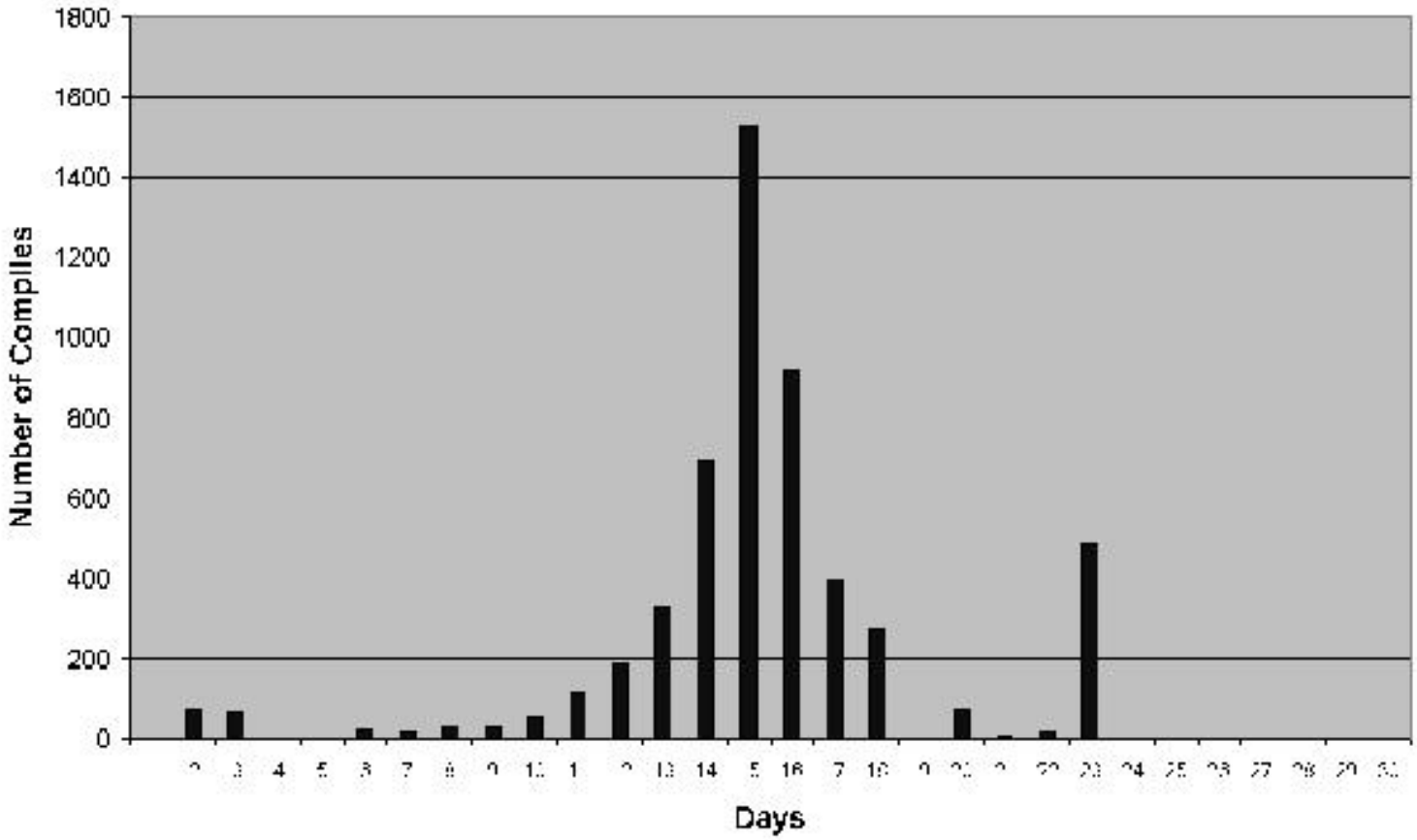

Figure 2: Compiler Activity

Figure 2 (showing compiler activity) clearly illustrate this lag between experienced and inexperienced UNIX programmers. The small peak on the 23rd is an inexperienced student completing their assignment late. An interview with the student confirms that the reasoning behind this peak was correct. An optional UNIX tutorial was offered to the group of students that followed the twelve students in this study on the cluster.

The cluster, as a teaching tool, has two distinct phases in its use. In phase one the cluster is used for parallel program development. In phase two the students benchmark the parallel programs they have written. The result of the two distinct phases is two different requirements in term of access to the cluster. During development the number of students on the cluster concurrently is not an issue but during benchmarking the timing of parallel and serial programs can be seriously compromised if more than one student is running tests on the cluster at a time. On one day during the case study between 18:00pm and 19:00pm all but one student are logged into the system at the same time. In addition, the study revealed that students tend to put off assignments until the pressure of a deadline becomes significant, which therefore increases the likelihood that a lot of them will be logged in at the same time. Figure 2, shows a peak on Sunday Nov. 15th, just a few days before the deadline of Nov 18th. An honors system and signup schedule was setup as a short-term solution to the concurrency problem. The signup schedule was setup in two hour intervals from 8:00am

to 6:00pm. The result was almost no load during the signup period and very heavy load from 6:00pm through to 8:00am the next day. To reduce the effects of this concurrency problem a parallel job scheduling package will be implemented on the cluster.

The data used to generate the daily login graph and many of the other graphs in this study was obtained from the process account management system for the Linux OS. The package logs every process that runs on a system, it includes the process name, the process owner, the CPU time utilized and the wall clock start time of a process. By using the process account system it was possible to exclude idle time from the data collected. Therefore it was possible to include a reasonably accurate analysis of student productivity. The total time each student was active on the system was determined and the lines-of-code (LOC) developed was measured. The standard for measuring the lines of code was taken from Watts Humphrey¹s text "A Discipline for Software Engineering" [4].

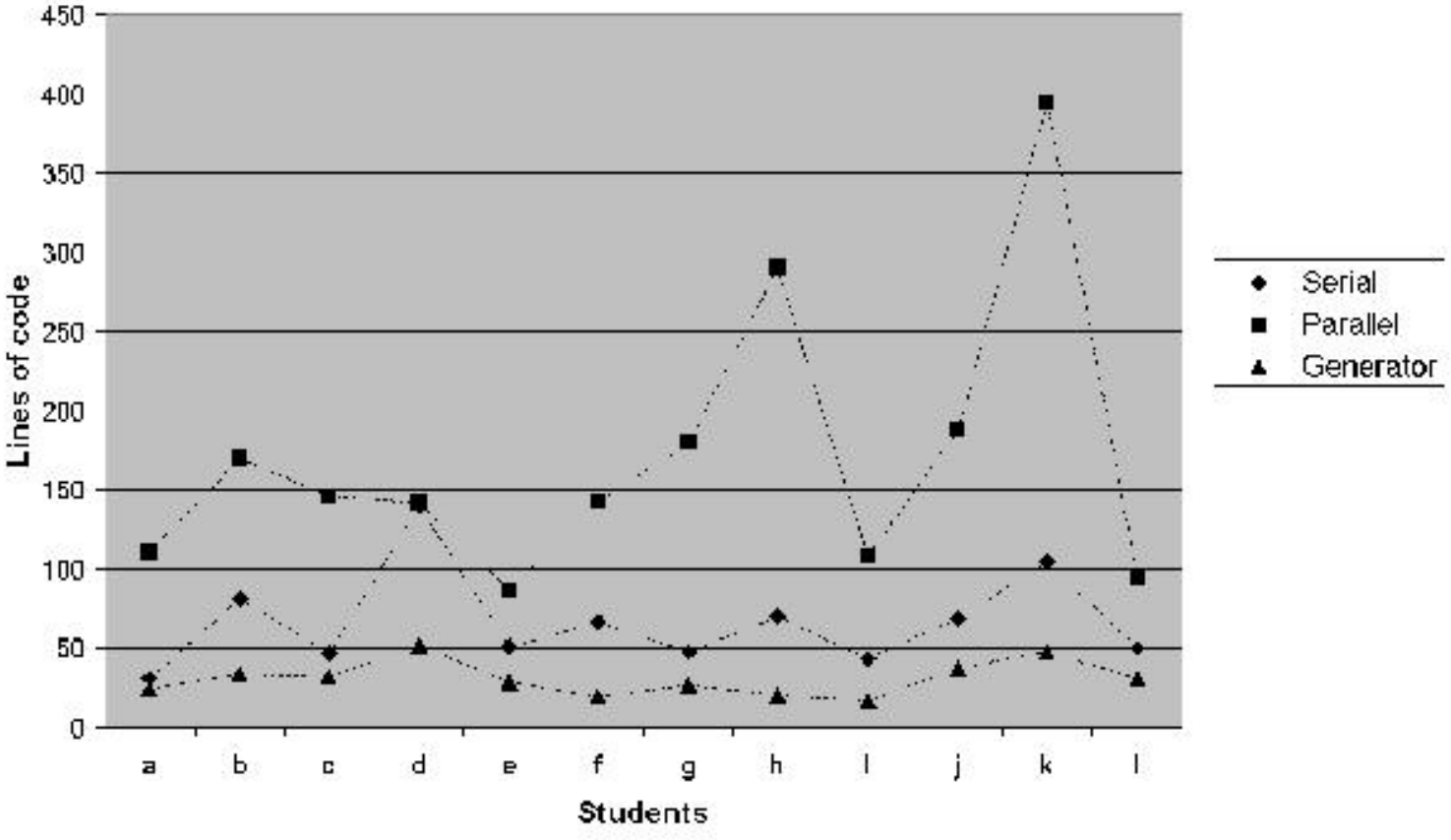

Figure 3: Lines of Source Code

Figure 3 shows the number of lines of code each student developed for their final program versions. The generator program generated two matrices of random numbers, the serial program multiplied the two matrices and stored the result in a binary file, and the parallel program implemented the matrix multiplication in parallel [5]. It is clear from Figure 3 that the generator and serial program for all students was about the same size. The largest peak

in the serial program was the result of one student going beyond the project requirements. The number of lines of code in the parallel programs varied a great deal. The heterogeneous nature of the cluster and the requirements of the parallel programs to load balance resulted in this wider range. The lines-of-code per hour (LOC/hr) each student was able to achieve was calculated. Again there is wide range in productivity, the mean is 9.3 LOC/hr. The total time spent on the cluster by all the students during the study period was approximate 378 hours which is an average of approximately 31.5 hours for each student. One surprising statistic was the amount of time in one day a given student might spend on the cluster. Every student, except one, spent at least 7 hours or more online on at least one day during the study. The maximum logon time for a single day by one student was approximately 12 hours.The amount of disk space used by the students for such a small project was a surprise, but perhaps it shouldn¹t have been since one of the purposes of parallel system is to do calculations on large data sets. The disk utilization on this cluster would seem to confirm this. The total disk space used by all the students during this study was 182 Mbytes. Interviews with the students revealed that some students keep their directories purged of old files while some students kept every file and data set created. The average amount of disk space used be each student is approximated 15 Mbytes.

The performance of the student programs varied a great deal. The serial program execution times varied as much as the parallel programs. The students were expected but not required to use doubles as random numbers but only three did so. Five students used floats and four students used integers.

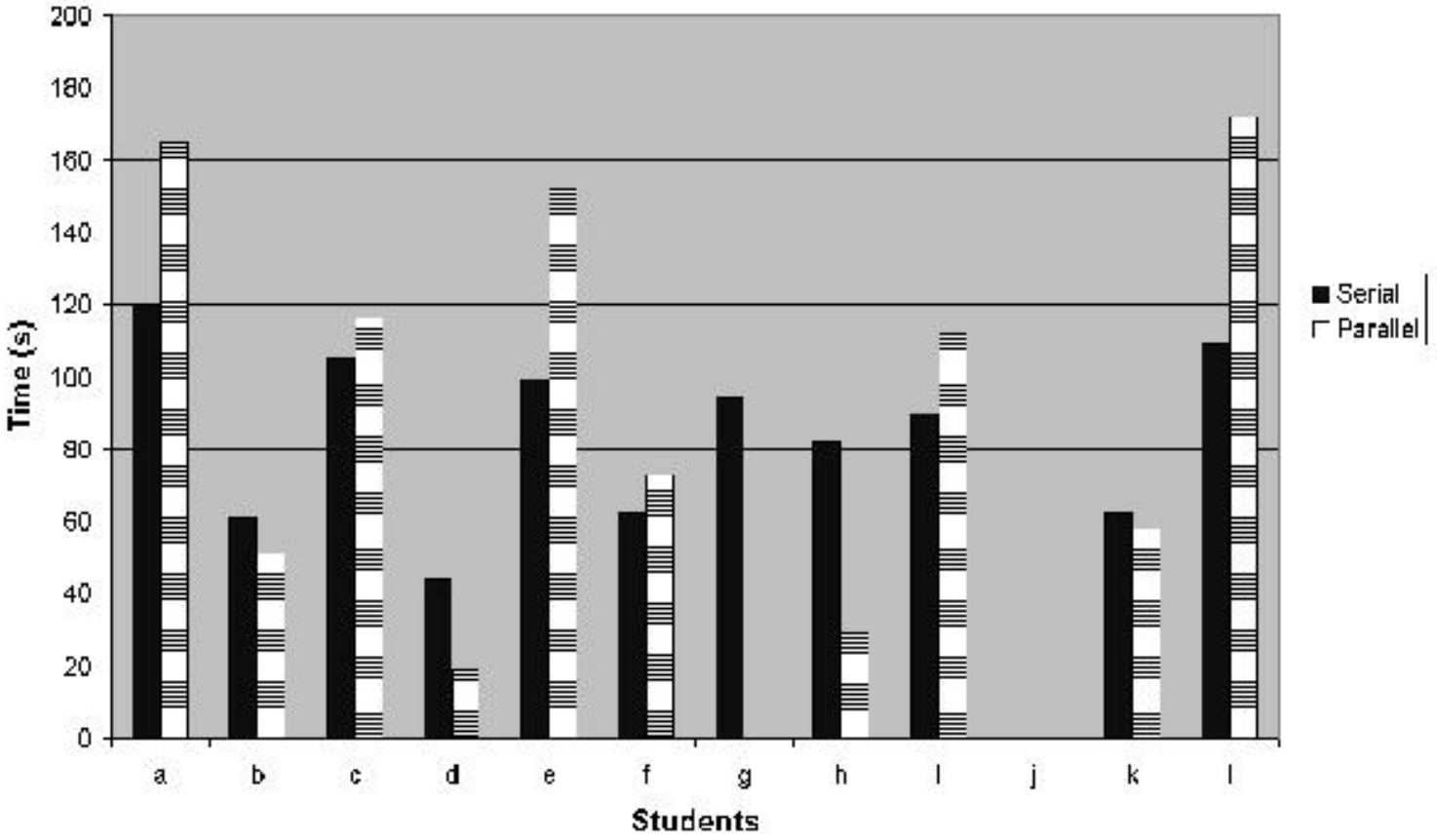

Figure 4: Execution Times

Six students dynamically allocated memory for the two-dimensional arrays required for the matrix multiplication, the other six students used one-dimensional arrays with a calculation for each index into a matrix. There was no single reason for the wide range in the execution time of the serial programs; some students paid a price for the index calculation other students had left diagnostic statements in their code. For the parallel program, the students that got a reduction in execution time going from their serial program to their parallel program just did a better job load balancing the cluster. For an overview of the serial execution times and parallel execution times see Figure 4. Obtaining a peak performance of the cluster (FLOPS) was not required for this project, the point of the project was for the students to learn about the benefits of parallel systems made from components that they work with every day.

Conclusions And The Future

The changes made to the initial cluster in support of teaching proved to be successful. Two immediate changes are required before the next group of students starts using the Linux Cluster. One tool that needs to be implemented is a LAM/MPI process management tool or process watchdog tool. Many students did not properly startup and shutdown their MPI applications and the supporting LAM daemon. What this meant was that it was not unusual to log on and find a number of user processes running after the student had logged off the

cluster. As a result cluster resources were being wasted. This usually occurred on one or more of the computational nodes. Disk quota control will also be implemented.

By demonstrating and proving that a Linux multi-computer cluster has merit regardless of the computational capabilities of the workstations we were able to get support to expand and improve the Texas Tech, Computer Science Linux Cluster. The response to the cluster has been positive enough for it to be included in the first undergraduate course in parallel programming, which is being offered this spring 99 semester.

References

1. "SuperStack II Desktop Switch, High Performance, Dedicated Switching for End Stations," *3Com Buyers Guide*, Nov. 1996.
2. A. Frisch, " Essential System Administration," O'Reilley & Associates, Inc., Sebastopol, CA. 1995.
3. W. Gropp, E. Lusk, and A. Skjellum, " Using MPI, Portable Parallel Programming with the Message Passing Interface," The MIT Press, Cambridge, MA., 1994.
4. W. A. Humphrey, " A Discipline for Software Engineering," Addison-Wesley, Reading, MA., 1995.
5. V. Kumar, A. Grama, A. Gupta, and G. Karypis, "Introduction to Parallel Computing," The Benjamin/Cummings Publishing Co., Redwood City, CA., 1994.
6. "MPI Primer/Developing with LAM," Ohio Supercomputer Center, Ohio State University, Nov. 11 1996.
7. G. Pfister, " In Search of Clusters. The Ongoing Battle in Lowly parallel Computing," Prentice-Hall, Upper Saddle River, NJ, 1998.
8. Sterling et al., "Beowulf: A Parallel Workstation for Scientific Compuation," *Proceedings of the 1995 International Conference of Parallel Processing*," Vol. I, Aug. 1995, pp I-11 - I-14.
9. M. Welsh and L. Kaufmann, "Running Linux," O'Reilly & Associates, Inc., Sebastopol, CA, 1995.

Biography

Per Andersen is a Computer Science Doctoral Student. He received hisBachelors degree in Electrical Engineering in 1980 from Carleton University,Ottawa, Canada and his Masters of Science degree in Computer Science in 1997from Texas Tech University, Lubbock, Texas. He can be reached at p.andersen@ttu.edu. Texas Tech University is located in Lubbock, Texas. Lubbock (population194,000) is situated on the High Plains of West Texas at an elevation of3,200 feet, and is located halfway between Dallas and Albuquerque. The1,839-acre campus is one of the nations largest, the total studentenrollment is 25,000 who can choose from 150 undergraduate, 100 masters, and50 doctoral degree program. You can visit Texas Tech online at **http://www.texastech.edu**.