The Development of a Tri-Use Cluster for General Computer Education, High Performance Computing Education, and Computationally Intensive Research

Andrew J. Pounds[*,†]
Dept. of Computer Science
Mercer University
1400 Coleman Avenue
Macon, GA 31207
pounds_aj@mercer.edu

Rajeev Nalluri[‡]
Dept. of Computer Science
Mercer University
1400 Coleman Avenue
Macon, GA 31207
nalluri_r@mercer.edu

Bennie L. Coleman[§]
Dept. of Computer Science
Mercer University
1400 Coleman Avenue
Macon, GA 31207
coleman_b@acadmn.mercer.edu

ABSTRACT

This paper describes the construction of a tri-use computer cluster for general computer science education, instruction in parallel high performance computing, and computationally intensive scientific research. While many schools today are starting to include parallel computing in their curriculum, a large number of smaller institutions lack the resources to devote a significant amount of their equipment budget exclusively to computers for parallel computing. Similarly, many schools would like to have computational facilities for scientific research, but cannot afford to designate machines solely for this purpose. Our system allows for one set of systems to be easily shared between pedagogical and research tasks. The system described herein utilizes standard hardware components and a combination of open-source Linux and commercial software. In addition, the development of a software program to selectively boot between operating systems is described.

Categories and Subject Descriptors

C.1.4 [**Parallel Architectures**]: Distributed architectures; K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Performance

*Corresponding author.
†Also affiliated with the Department of Chemistry
‡System and Network Administrator.
§Undergraduate research assistant.

43rd ACM Southeast Conference March 18-25, Kennesaw, GA, USA.

Keywords

High Performance Computing, Computational Science, Clusters

1. INTRODUCTION

Parallel computing is quickly becoming one of the standard topics in computer science education. Unfortunately, budgetary constraints prohibit many smaller colleges and universities from teaching courses in parallel programming because of the cost associated with building and maintaining a separate set of computers for parallel computing instruction. Furthermore, faculty that come to small colleges sometimes have to abandon their computationally intensive projects due to a lack of adequate high performance computing facilities. In short, to meet the most needs, a system needs to be designed which allows for (1) general computing instruction in a non-UNIX environment, for (2) general computing instruction in a UNIX environment, for (3) specific instruction in high performance parallel computing in a UNIX environment, and (4) for high performance computing cycles to be used by scientific researchers. Our system provides for all of these needs and does so in a manner to minimize system downtime and maximize use of computer systems on campus. Many current systems already implement (1) and (2) above. In these systems (1) pertains mainly to instruction in computer competency courses for business students and (2) pertains to mainly computer science core programming classes. It is the addition of (3) and (4) that make our system particularly attractive to smaller colleges and universities.

2. SYSTEM DESCRIPTION

Two of the teaching labs at Mercer contain a total of 42 Intel Pentium 4 computer systems with 512 MBytes of memory each. These machines are used during the daytime hours for either teaching computer competency classes or for teaching programming classes. Each time a system from one of these labs is booted, it becomes a member of one of two operating systems and one of three domains: *blackhawk* (Microsoft Windows XP Operating System, CSDept domain), *cobra* (RedHat Linux Operating System, Compsci domain), or *zeus* (RedHat Linux Operating System,Olympus domain).

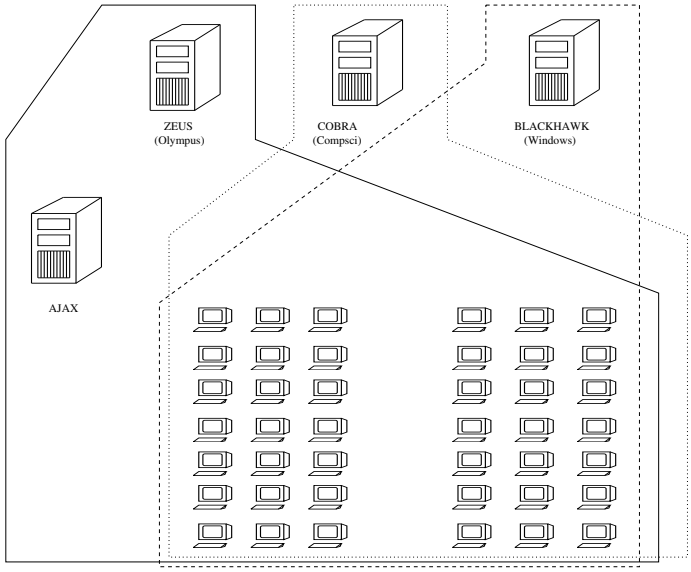

Figure 1: Diagram of the Tri-use cluster.

These systems and domains are schematically shown in Fig. 1. Password authentication is done by each domain server and different user databases and authentication files are present on the respective domain servers. During the daytime the default boot OS is *Microsoft Windows* under the control of *blackhawk*. If a student reboots the computer during the daytime, they have the choice of either booting into either Windows or Linux. The default daytime Linux domain is "Compsci", with all information for the programming classes being stored on *Cobra*. Students working in a class on parallel computing, however, have the ability during the boot process, to respond to prompts that will place the system within the Olympus parallel computing domain. At night, the systems by default boot into the Linux OS within the Olympus domain. Only machines in the Olympus domain can operate within the parallel environment.

Each of the servers shown in Fig. 1 can be logged into at any time of day. Because users can log directly into *Zeus* and *Ajax*, parallel code can be developed and tested at any time. Once tested, production runs can be run across the entire cluster.

2.1 Hardware

No special hardware was used in this lab; all pieces are off the shelf components. As can be seen in Fig 1, there are three domain servers (*Blackhawk*, *Cobra*, and *Zeus*). These are all dual processor systems. *Zeus* is a dual PIII Xeon system while both *Blackhawk* and *Cobra* are PIV Xeon systems. Each system has 1 GB of RAM with varying amounts of disk space available on each system. The smallest amount of disk space (18 GB) is located on *Zeus*. The other server in the diagram is *Ajax*. It is a dual processor PIV Xeon System with 4 GB of RAM and a Terabyte of disk space. This disk space is available to anyone using the Olympus cluster via NFS. The primary role of *Ajax* is to run jobs that require a significant amount of memory and scratch disk space. At present all machines have 10/100/1000 mb/sec network cards and run at 100 mb/sec to Catalyst switches. A gigabit channel connects the two labs.

2.2 Software

There are four main issues that have to be addressed with respect to the development of the system software: operating system switching, authentication, filesystems sharing, parallel libraries, and job schedulers.

2.2.1 OS Switching

Thirty minutes after the lab doors are locked, either a Linux cron job or a Windows scheduled event runs which overwrites the `grub.conf` file on the Linux partition to set the default boot OS and domain to Linux and Olympus. Overwriting the `grub.conf` file from the Windows environment was accomplished by using a slightly modified form of the *LTOOLS* [7] utility. The *LTOOLS* code had to be modified and recompiled so that it would not prompt for a response before overwriting the boot loader.

Once the boot process starts, the `/etc/sysconfig/network` file is run. This file has been modified to boot into different Linux NIS domains based on the time of day. For added flexibility, the script was also modified to prompt users at the console for the domain into which they want to boot. This modified file must reside on each of the cluster machines. In addition, the `/etc/yp.conf` file must be modified on each cluster machine. Both domain servers must appear in the `/etc/yp.conf` file as two separate entries.

2.2.2 Authentication

While in the Linux OS, all user authentication is done through the Network Information Service (NIS). Because separate NIS user password files are maintained on *Zeus* and *Cobra*, access can be limited by and to a specific domain. By taking this approach, student access to the parallel cluster can be limited to only students taking classes in parallel computing. Similarly, researchers will only have accounts the parallel environment so that they do not interfere with class instruction taking place on departmental teaching systems. *Zeus* is the login server for the parallel environment and the NIS master.

There is another positive consequence of designing the sys-

tem this way. Before this system was designed, the systems in the laboratory were turned off at night for security purposes. There was a concern that having so many computers up and available to a large fraction of the student population presented a security risk. In the new configuration, where domains of all of the lab computers are switched to a different NIS domain at night, the number of users that can access the systems is dramatically reduced.

2.2.3 *Filesystem Sharing*

Based on the domain chosen, home directories are NFS mounted from either *cobra* or *zeus*. *Zeus* contains all of the home directories for users in the parallel environment. In addition to this, a `usr2` filesystem was created on *zeus* and exported to all of the machines in the cluster. This filesystem contains all of the common code and libraries needed to develop, run, and test parallel applications. Finally, a terabyte of disk space located on *ajax* is exported to the cluster for users to write large amounts of data. While users are encouraged to use the local `tmp` directories on the cluster machines to enhance disk performance and cut down on NFS traffic, the NFS mounted directories on *ajax* are available for cases that are too big for the local temporary file space.

2.2.4 *Parallel Libraries*

Currently both the PVM [2] and MPI [5] libraries are available on the system. The MPI library is implemented via MPICH. At present there are no commercial high performance parallel compilers on the system. All code generation is done with the standard Gnu compiler suite that comes with the Linux distribution.

2.2.5 *Job Scheduler*

The Portable Batch System (PBS) [3] was selected as the batch scheduler for the system. *Zeus* serves as the job scheduler. At present three batch queues are available: one for serial jobs, one for weeknight parallel jobs, and one for weekend parallel jobs. Under the current configuration, up to 42 machines can be assigned to a single parallel job. Users are responsible for checkpointing computational results. Serial jobs can only be run on *ajax*.

3. RESULTS

To demonstrate that the parallel environment works, and to also get some idea about the general computational power of the cluster, two sets of tests were conducted. In the first case a direct matrix multiplication algorithm was run. In the second case, a distributed electrostatic potential grid calculation was run.

3.1 Direct Matrix Multiply

To test the usefulness of the cluster as a teaching tool in parallel computing, a textbook parallel matrix multiply method [6] was coded and run. Using the direct method of parallel matrix multiplication, the product of two square matrices dimensioned at 4000x4000 was computed using both PVM and MPI. All of the tests were run through the PBS job scheduler a minimum of five times to generate statistical data. A standard speedup curve was then plotted (Fig. 2) and these data fit to Amdahl's law [1] to determine the amount of code that had to be run serially.

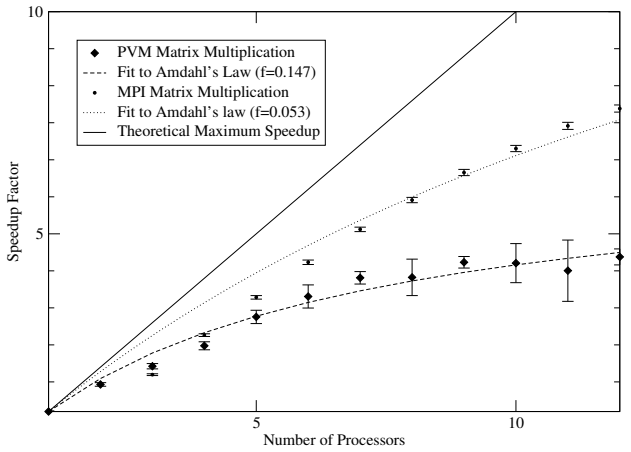

Figure 2: Parallel matrix multiplication speedup factor using two different message passing libraries

Based on the the data obtained in our matrix multiply tests, there is clearly a performance difference between PVM and MPI on our system. While the reason for this disparity is unknown, it has been postulated that a problem existed with one of our network switches when the PVM tests were being conducted. If this is the case, it would also account for the larger larger variability of results (notice the error bars in Fig. 2) present in the PVM runs. Further tests are necessary to determine if an actual problem exists.

3.2 Electrostatic Potential Generation

To test the usefulness of the cluster for research, a PVM code for the generation of a 3D molecular electrostatic potential [4] (ESP) was run across the cluster. A speedup

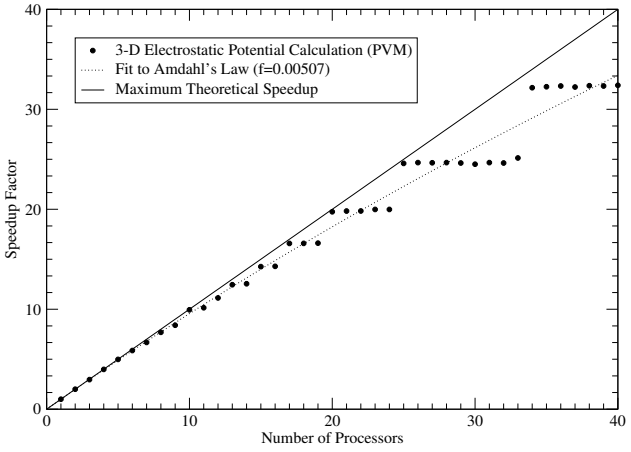

Figure 3: Parallel electrostatic potential computational speedup factor. Sustained average performance with 40 machines was 8.4 GFlops using double precision arithmetic.

The parallel ESP code utilizes a divide and conquer parallelization strategy. When the results of this computational method were originally presented, the author claimed a near linear speedup. In that study the calculations were only run

across eight machines. Clearly, based on Fig. 3, the author was correct based on the limited number of machines used. Only by using an increased number of machines does one notice the scalability issues. Already this system has demonstrated its usefulness for research by identifying scalability problems in code that was thought to be devoid of these problems.

In comparison to the matrix multiplication codes, the ESP code does appear to scale much better. This is primarily due to network loads. In the matrix multiply code, an entire matrix was sent to each machine and then N/P columns of the other matrix were sent to each machine (where N is the number of machines and P is the number or processors used). For a 4000x4000 matrix of doubles, a minimum of 128 MBytes of data must therefore be sent to each system. The average message size for the ESP code is less than a megabyte.

The other issue for research grade calculations is sustained performance. By explicitly counting the floating point multiplies and adds, it was determined that the average performance on the cluster utilizing 40 machines was 8400 megaflops (8.4 GFlops) using double precision arithmetic. Clearly this is satisfactory for research calculations.

4. DISCUSSION

For the last four months the system described in this paper has been functioning flawlessly. Each night numerous calculations come online and run for several ours and each day the systems are used for their normal pedagogical purposes. While the construction of the system has been an marked as an unqualified success, it has also allowed us to see that the major area for improvement is in the networking infrastructure.

While grid computing has taken off in the last few years as a way for researchers to get their calculations done, faculty at smaller schools have often found themselves left out of this picture. In addition, all parallel calculations cannot fit into the "grid" model. Some calculations require significant amounts of memory and disk space. It is hoped that, assuming others adopt the model presented in this paper, smaller schools could combine their resources to make larger clusters that come online for research calculations after normal business hours. With such a model, faculty at even small schools would have computational resources to rival those at larger institutions. Also, as more computational science programs come online at colleges and universities around the country, there is going be an even greater need for these type of high performance cycles for student work. The system constructed in this paper is easy to build, cost effective, and once started, very easy to maintain and could easily meet the needs of these programs.

5. CONCLUSIONS

We have clearly demonstrated that it is possible to construct a system of computers in such a way that they can serve the needs of three academic groups: those needing computers for their general coursework, those needing access to computing resources to learn how to program, and those that need high performance cycles for research. We accomplish all of this by only making minor changes to two Unix system files and adding a batch scheduler to control the flow of job onto the system. The other significant contribution of this research is the introduction of a method to have the systems switch between operating systems automatically at specific times of day.

6. ACKNOWLEDGMENTS

We would like to thank the Dr. Richard Fallis, Dean of the College of Liberal Arts, and the Department of Computer Science for providing funding for this project. We would also like to thank Altair for granting us licenses to use the PBS batch scheduler.

7. REFERENCES

[1] G. Amdahl. Validity of the single-processor approach to achieving large-scale computing capabilities. In *Proc. 1967 AFIPS Conf.*, volume 30, page 483, 1967.

[2] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *Parallel Virtual Machine: A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, Massachusetts, 1994.

[3] J. P. Jones. *Portable Batch System Users Guide*. North America, February 2004.

[4] A. Pounds, M. Buttersworth, and B. Mantooth. A high performance web-based tool for molecular electronic structure visualization. *Journal of Computational Sciences in Colleges*, 19(3):238–248, January 2004.

[5] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI – The Complete Reference. Volume 1, The MPI Core, 2nd ed.* MIT Press, Cambridge, Massachusetts, 2000.

[6] B. Wilkinson and M. Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2nd ed.* Prentice Hall, Upper Saddle River, New Jersey, 2004.

[7] W. Zimmermann. Ltools: Access your linux files from windows 9x and windows nt. *Linux Journal*, pages 164–171, November 2000.