A Novel and Efficient Introduction to Clustering using a Classroom Laptop-based Computer Cluster

Irv Englander
ienglander@bentley.edu
Computer Information Systems Dept., Bentley University
Waltham, MA 02254, USA

Abstract

Fitting newly important topics into an already crowded IS/IT curriculum is an ongoing challenge. In this paper we present an innovative and simple way to introduce and demonstrate computer clustering concepts in less than a single class hour, at no cost, by constructing a high performance cluster in the classroom using student laptop computers, classroom Ethernet facilities, and free, downloadable software. Although we initially tested our approach in a Linux Operating System class, the technique is applicable to any appropriate IT class in which students have been previously exposed to the basics of computer hardware, operating system software, networking, and system architecture.

Keywords: computer clustering, high performance computing, Beowulf, PelicanHPC, classroom computing, supercomputing

1. INTRODUCTION

Undergraduate IT/IS curricula must offer a wide range of knowledge to their students, providing a broad understanding of business, as well as in-depth knowledge of various aspects of information technology, all this in a limited number of courses. As a result, the requirements of an IS or IT curriculum inevitably exceed the time available to present the range of technology that is important to the success of an IS or IT graduate in the business world. It is obviously impossible to cover every important topic in the curriculum, and the creators of an IT/IS curriculum must pick and choose among the myriad of topics to cover some topics at various levels of depth and to neglect others. When newly important topics arise, it becomes necessary to find a way to introduce them at an appropriate level, and to eliminate or downsize the coverage of other topics that may still be important.

Computer clustering is a topic of growing importance, due to its role in web services, cloud computing, massively-parallel supercomputing applications, and other applications where large amounts of data or large numbers of transactions must be managed and manipulated rapidly. Clustered systems offer easy scalability, effective processing parallelization, and support for applications that mandate high system availability. For these reasons, clustered systems are often referred to as high performance computing (HPC) systems. In essence, modern clustering technology offers the ability to provide massive processing capability at minimal cost. For example, the entire Google search capability is based on a large- scale application of clustering technology, built from low-cost computers. Business applications for HPC systems include the Web and cloud services mentioned above, plus econometrics, data mining, financial modeling, statistical analysis, simulations, and many other applications. IT/IS stu-

dents graduating into the business world today could be expected to have the capability to understand, use, develop, and configure clusters and their appropriate application.

Because clustering has taken on such importance, we believe that space in the curriculum needs to be found for at least a basic coverage of clustering concepts. This paper describes a successful attempt to explain and demonstrate the features of clustered computing in the most efficient way possible, with minimum time required for the presentation and discussion, and at little or no cost. To do so, we utilize available classroom and networking facilities, student laptop computers, and free downloadable software to create a working parallel-processing cluster in the classroom. The entire clustering presentation takes less than a single class hour. Once set up initially, the exercise can be executed, with minimal advanced preparation, in any appropriate IS/IT course, with a properly equipped classroom as the only critical requirement. As an additional benefit, the presentation utilizes and reinforces system infrastructure concepts previously learned.

There have been a small number of other documented attempts to integrate clustering concepts into the curriculum. Most of these have taken place within computer science or computer engineering curricula. Typically, these have been specialized courses that focus on various aspects of clustering and parallel processing (Hastings, 2003), (Montante, 2002) or laboratory-focused courses that require significant student laboratory efforts (Prins, 2004). A substantial general discussion of clustering in the computer science curriculum may be found in Apon et al., (2001). Kitchens et al. (2004) have attempted to introduce clustering in the IS/IT curriculum by using the design of a computer cluster as the core group project in a system analysis and design course sequence. These efforts all require a course slot in the curriculum, and present computer clustering at a depth not needed for the typical IS/IT student. All of these previous attempts also have required substantial faculty and student preparation and require the significant use of specialized laboratory facilities and the like. All of the methods described in these papers required creation of special software disks and installation on each of the node machines, a procedure that we wished to avoid.

Modern low-cost clustering methods have found another application in the University setting: the ability to build inexpensive small-scale "supercomputing" facilities for research, using spare, perhaps obsolete, computers that are available in the laboratory. These clusters offer researchers processing power that would not ordinarily be available in typical University facilities. Much of the work in this area is based on free, public domain software. The result of one of these efforts, PelicanHPC, (Creel, 2010a) provided the basis for our classroom clustering effort.

As a result of these various efforts, there is a substantial assemblage of readily available technology tools and know-how for building and operating clusters that is applicable to our requirements. We saw an opportunity to implement a cluster in a classroom setting that would explain and demonstrate clustering quickly and inexpensively.

2. A BRIEF OVERVIEW OF CLUSTER COMPUTING

A cluster consists of a number of loosely-coupled, interconnected computers. In many respects, the configuration of a cluster appears similar to that of a network, however, the cluster differs in that all of the computers in the cluster are committed essentially to share the workload for the same task. Each individual computer serves as a node in the cluster. A node may represent a complete system, with individual disk storage, plus, perhaps, a display monitor, a keyboard, and other peripherals, or it may simply contain one or more CPUs, primary memory, and a means to support the interconnection, with a network card or other interconnection device. The interconnection itself may also provide a storage device; such a cluster is known as a "shared disk" cluster. A cluster with no shared storage is called a "shared-nothing" cluster. (Englander, 2010)

Clusters tend to be classified by their primary usage. There are clusters intended for large-scale transaction processing, with an emphasis on rapid access and easy scalability, clusters designed to provide "fail-safe" and high-availability processing, and clusters that provide massive amounts of distributed, parallel processing power for large-scale applications amenable to this capability (often referred to as "supercomputing"). This categorization is somewhat artificial, since the technology is similar in most cases, differing mostly in configuration and choice of software.

Although early clusters were built from specialized components, with dedicated specially-designed interconnection circuits, most modern clusters are designed around **c**heap (some prefer the word "**c**ommodity" or "**c**ommercial") **o**ff-**t**he-**s**helf (COTS) computers, with a dedicated, isolated Ethernet providing the interconnection. This design is often referred to as a Beowulf cluster. It has the advantages of low cost, simple creation, and easy upgrade. The cluster may be built from available personal computers, or may use specialized circuit boards known as blades. Most supercomputers today are built using some variation on this design. Depending on the type of application that the cluster is designed for, each node in the cluster may operate semi-independently; more commonly the nodes may be set up to operate in a master-slave configuration, with a master node delegating work to the slave nodes and organizing and reporting the results. The master node also provides the primary user interface and connection to the outside world.

Clustering software applications can be written in a number of programming and scripting languages, including Fortran, C, Python, and others. The required functions that allow the nodes to communicate are usually provided by the Message Passing Interface (MPI) application programming interface, a de facto, language-independent set of standards for communicating between cluster nodes. MPI functions initialize and determine the parameters of the cluster, send and receive requests and data between nodes, manage the data communication between nodes, and many other tasks. MPI libraries are readily available for nearly every hardware platform. Information and references about MPI may be found at --"Message Passing Interface", (2010) and in many of the books referenced above.

A good, though slightly dated, review of the hardware, software, and networking components used in Beowulf clustering is found in Sterling, et al.(1999).

General coverage of clusters may be found in Bookman (2003), Sterling et al. (1999), and Vrenios (2002).

3. ASSUMPTIONS AND REQUIREMENTS

Our plan was to create a high performance parallel-processing cluster in the classroom, using the laptop computers that students bring to class, the wired network available in the classroom, and free clustering software; and to explain and demonstrate the features of the cluster during operation. Time during a single class period was allocated for this purpose. To achieve our goal, we made a number of assumptions:

(1) Computer clustering is a derivative topic, as opposed to a fundamental one. A good, basic understanding of system architecture, computer hardware, operating system software, and networking is sufficient to allow a rapid grasp of the principles of clustering. Indeed, the integration and reinforcement of these concepts is a secondary goal of this project. If students have this background, we believed that it should be possible to teach the basics of clustering, complete with a demonstration of the basic features using only part of the time available in a single class period. Understanding clusters at this level is appropriate and sufficient for IS/IT students; a deeper understanding of clusters would be more time consuming and is a specialization that is more within the province of a computer science curriculum.

(2) It is highly desirable that the classroom computer cluster should require no advance preparation or installation of software on the part of students prior to the class or during class. Establishing this as a requirement eliminates the problem of students showing up unprepared to participate in the exercise. It also eliminates the need to prepare installation disks in advance for use during class. Furthermore, we preferred that the classroom demonstration should have no permanent impact on the student computers: there should be no reconfiguration of any computers or their operating systems, and no modification of program or data files. When the cluster is shut down, all evidence of the cluster's existence should disappear without a trace.

To meet these requirements, we elected to utilize "push" technology to create the cluster and "live" technology to support its existence on the student laptops. "Push" technology boots the nodes and installs the software remotely from a master node, using the net-

work. "Live" technology is volatile; each node computer operates completely from its primary memory (or "RAMdisk") and the network, without access to local disk storage. Therefore, the permanent state of a node machine is unaffected by any operations that take place while the node is active.

(3) We assumed that tools exist that meet the above requirements and that allow the rapid creation of a cluster using standard computer components and networking facilities with minimum (or no) expense. The concept of Beowulf clustering is well established, and it was safe to assume that a variety of tools to support such clusters were available and in the public domain. In addition, we sought tools that would include pre-built parallel applications to demonstrate various features of the cluster, as well as a benchmark program to measure the performance of the cluster as we varied the number of nodes in the cluster.

(4) We selected parallel-processing applications for our classroom cluster because they are the easiest to demonstrate. As we previously noted, the underlying technology is similar for most clustering applications.

The classroom cluster was presented as a topic within an undergraduate IT elective course entitled "Operating Systems Concepts with Linux." This class considers many different aspects of using the Linux operating system, including command-line operation, scripting, system configuration, tools, and system administration. The students in this class had all previously completed a prerequisite class in computer system infrastructure that included the basic concepts of system architecture, computer hardware, operating systems, and networking.

4. FACILITIES, SOFTWARE, AND ORGANIZATION FOR THE CLASSROOM CLUSTER

Like many (perhaps most) colleges and universities today, Bentley University provides a standard laptop computer to every full-time undergraduate student. The 2009-2010 laptop computer was an HP model 6930p, supplied with an Intel T9600 processor, 4 GB of primary memory, built-in Gigabit wired Ethernet and 802.11draft-n Wi-Fi networking, and many other features. The supplied software included Windows Vista and a variety of application software. The use of a standard computer model simplified somewhat (but less than one might imagine) the requirements for the cluster. The students in the IT Linux elective modify their computers at the beginning of the course to support dual-booting, so that Linux is already available on their machines, most recently, with Fedora 11. This might have simplified creation of a cluster in this course, but at the sacrifice of generality. We made a decision not to utilize this capability, but instead to create a common system that would act identically on every node of the cluster, regardless of the native operating system present on a node's computer. The student laptops are configured by the University to allow only a single network connection, either wired or Wi-Fi, but not both simultaneously. This feature proved to be useful in our setup, as it allowed us to configure the cluster with minimal concern for security.

Bentley University provides per-seat wired Ethernet in every classroom, as well as Wi-Fi services throughout the campus. The Ethernet switch in each classroom allows a classroom network to be isolated from the remainder of the campus network; this capability is controlled by software at the instructor's podium console. Since Wi-Fi classroom isolation is impossible to achieve, we chose to use wired Ethernet as the cluster interconnection medium and Wi-Fi as a network connection from the instructor's master node computer to the outside world. Isolation of the cluster interconnection network allowed us to eliminate firewalling and packet analysis, for faster operation.

Our research verified that there is indeed a wide range of clustering software available for free download from the Web. The software we selected for use, PelicanHPC (Creel, 2010a), offered several advantages over other possible choices. The features that made PelicanHPC our preferred choice included:

(1) The PelicanHPC software is based on push technology. The entire cluster is created from a single master node. Once the master node is booted, the remaining nodes are created by booting them from the network and installing the required software from the master node. The software even includes a

DHCP server for assigning network addresses to each node to set up network communication.

(2) PelicanHPC runs live from memory, utilizing Linux as its base. It offers an optional GUI or can be controlled from a command line prompt. Once a node is booted from the network, it will also run a live Linux operating system, along with the MPI application and other support functions.

(3) The PelicanHPC software includes a variety of pre-built distributed parallel-processing applications; these applications were written in GNU Octave. The provided software also includes an Octave interactive interpreter to run these applications. Octave is a high-level programming language designed for the creation and execution of high-performance parallel programs.

(4) Commands are included to add and remove nodes from the cluster dynamically. This simplifies our ability to demonstrate the effects of adding nodes to the cluster.

(5) The Pelican HPC package includes the Linpack HPL benchmark software to measure the performance of our classroom cluster. The Linpack HPL benchmark "measures the floating point rate of execution for solving a linear system of equations." (Petitet et al., 2008). It is one of seven benchmarks used together to evaluate the performance of the world's most powerful supercomputers (-- "HighPerformance..").

(6) PelicanHPC is easy to set up and easy to use. There is reasonable documentation (4), and the provided scripts are easily extended and modified to offer different or additional capabilities.

PelicanHPC is downloaded from its Web site as an .iso file, which can be burned directly to a CD-ROM or copied to a USB flash drive. Once transferred to a local device, the software can be booted live directly on the computer that serves as the master, or frontend, node in the cluster. User interaction with the cluster also takes place at the master node.

PelicanHPC was built as a research tool for running applications that are amenable to parallelization. The use of parallel processing as a classroom demonstration tool suits our needs well, since it is easier to demonstrate the increased processing capacity offered by the classroom cluster as we add nodes than it would be to demonstrate the scalability of a web server or cloud server, for example. The parallel processes and benchmark supplied with PelicanHPC eliminated the need to develop our own applications, although the support to build our own applications is available in the package, should we wish to do so in the future.

The cluster organization that we implemented is shown in Figure 1. Each computer in the cluster is connected to the classroom network switch through a wired network port at a classroom seat. The classroom switch is isolated from the campus network, using software at the podium for this purpose. The master node is connected to the classroom projection display and may be connected to the campus network, if desired. The PelicanHPC software is loaded onto each node PC.

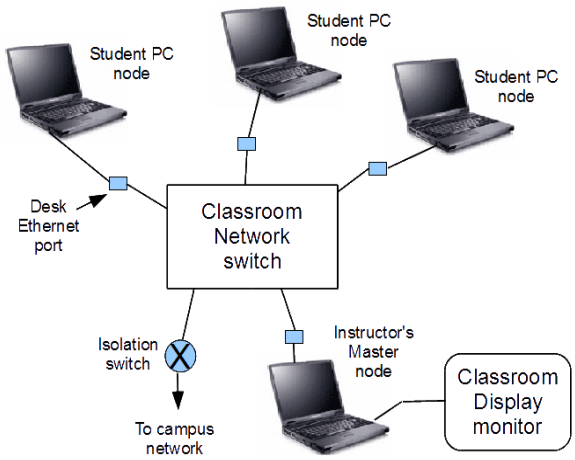

Figure 1
Classroom Cluster Layout

Each node in the cluster consists of a live Linux operating system, PelicanHPC control software, the MPI software, various applications, the HPL benchmark software, and a GNU Octave interactive interpreter, all running directly in memory. When booted, every node is initially configured identically. The software is loaded onto the master node from the live CD; slave nodes are loaded from the master node using the network.

Figure 2 shows a simplified view of the PelicanHPC software configuration. Each node boots into a basic Linux operating system. The operating system supports the node hardware, including I/O and networking services, offers command line and graphical user interfaces, and provides the usual services to the software layers above it. DHCP and a network boot loader are also included.

The Pelican software layer consists of Linux command shell scripts that allow a user to set up a cluster, customize the configuration, add and remove nodes, and operate the cluster. The Message Passing Interface manages the movement of data and program information between the master node and the slave nodes. Finally, the application layer provides a number of parallel econometrics applications, the Linpack HPL measurement suite, and the GNU Octave program interpreter.

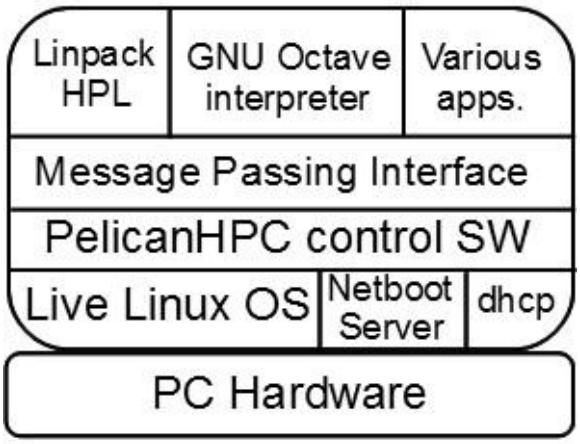

Figure 2
PelicanHPC Software

5. CLASSROOM PRESENTATION PLAN

Our classroom presentation plan included introductory discussion of the principles of clustering, followed by the setting up and activation of the cluster and demonstration of its various features and capabilities. The classroom plan which we ultimately implemented and demonstrated successfully is described next; however our initial attempt to offer the demonstration of the cluster as described was unsuccessful, due to a number of glitches that had to be resolved. These are described in the following section. (The attempts to solve these problems during the presentation actually led to a useful classroom discussion about the cause of the problems that occurred, and deepened student understanding of the cluster technology.) The second attempt, performed in class a week later, was successful.

The introductory discussion of clustering principles begins with a brief overview of the cluster concept, its benefits, its various applications, and its basic organization. Review of the infrastructure technologies that form the basis for modern clusters follows. This includes short reviews of basic TCP/IP/Ethernet networking, computer hardware architecture, operating systems, and system architecture; this discussion is focused particularly on the relationships between these technologies and the cluster's implementation, integration, and use of these technologies. An introduction to distributed parallel processing, parallel programming, and basic algorithms is next. Although brief and relatively shallow, this introduction at least places the concepts and possibilities of parallel processing into the students' consciousness, and allows us to move directly to the creation and demonstration of the classroom cluster.

To begin the cluster creation procedure, we instruct the students to connect their laptop computers to the wired Ethernet, but not to turn their computers on. The instructor boots her/his computer using the PelicanHPC Live CD. The instructor's computer will serve as master node; its display is connected to the classroom projection monitor. Its Wi-Fi may be connected to the wireless campus network for remote access or use of network services, if desired, although we did not use this capability. The complete, step-by-step instructions for making the Pelican CD operational as a cluster are available on the Website (Creel, 2010b).

Once the master node is operational, we configure the cluster, following the steps in the tutorial instructions. When this step is completed, we have a working cluster, with only a single node. This step creates the directories and files that will be needed later for the benchmark commands. As we later discovered (see the next section), it is also necessary to do some additional setup for the HPL benchmark.

Once this is done, we first demonstrate a built-in PelicanHPC kernel regression analysis application example and the Linpack HPL benchmark. The regression application offers a graphical output that is best displayed using the optional GUI, however, the classroom projection system in the classroom we used did

not support the high resolution provided by the master node computer, so we actually ran the application twice, once with the GUI display, holding up the computer for the students to see the result, and once with an artificial text-based output that results from use of a command-line display that worked with the projector. In both cases, the results also include the time required to complete the application execution. These runs provide ground-level values that would allow the students to compare the single-machine results with those generated by the cluster.

Next, students are instructed to turn their computers on, one or two students at a time, and, if necessary, to reset their BIOS startup to boot from the network as first option. Two commands at the Linux prompt in the master node, pelican_setup and pelican_restarthpc, are used to control the cluster. These commands are used to boot each machine and to add or remove nodes in the cluster. Each time we add machines to the cluster we run the regression analysis and the HPL benchmark again and compare the results to the previous run. Details of the steps necessary to boot the slave nodes and to run the applications and benchmark are also provided in the PelicanHPC tutorial.

6. INITIAL TESTING AND CLASSROOM PROBLEMS AND SOLUTIONS

We initially tested our plan in the author's office, using the author's available netbook as the master node, two borrowed student laptop computers as the slave nodes, and a borrowed network switch to provide Ethernet interconnectivity. There were a number of minor issues that had to be resolved, but the entire cluster configuration and demonstration was complete and ready to run in the classroom in less than an hour of setup time and testing.

The major problem that occurred during initial testing involved configuration of the Linpack HPL software. As downloaded from the Web, a script called *Make.Pelican* prepares the Linpack software for execution on the cluster. The *Make.Pelican* file is located in the /home/user/hpl-2.0 directory. At the time that this paper is being written, this file contains several lines of script code that point to incorrect locations in the directory. The correct documentation for this script is located at http://pelicanhpc.788819.n4.nabble.com/file/n1839478/. We used a standard Linux editor to correct the errors in the file to match the documentation. This consisted of fixing lines that are labeled Mpdir, CC, and LINKER. Each of these lines contains a pointer to a file that is located differently from that in the script code.

We also struggled a bit to determine the sequence of commands required to run HPL. Once we corrected the *Make.Pelican* file, and changed our prompt to the ~/hpl-2.0 directory, the following sequence of commands did the job (all case sensitive, by the way):

```
sh SetupForPelican
cd ./bin/Pelican
mpirun --hostfile /home/user/tmp
/bhosts/ -np 3 xhpl (all on one line; no space)
```

Documentation for the mpirun command also proved to be useful; the documentation is found in in the *man* command: man mpirun. The value in the -np command option determines the number of processors to use for the program being run. We discovered that a minimum of 3 is required for the HPL benchmark. If the number exceeds the number of nodes in the cluster, mpirun creates multiple processes on the nodes available. This makes it possible to run the benchmark on a single-node cluster, before any slave nodes are activated.

After we made all of the required changes and figured out the correct commands, everything in the cluster demonstration worked as expected in our office environment.

7. CLASSROOM RESULTS

Building the cluster in the actual classroom revealed two problems, one major, one minor, that did not occur during our initial testing. Most importantly, our first attempt to run the cluster demonstration in the classroom was unsuccessful, due to a problem with the classroom network switch setup. Although the connection between the classroom network and the campus network was in fact disabled, the network switch itself was configured to reject DHCP packets even when the switch was isolated from the campus network. Therefore, the master node DHCP server was unable to issue IP addresses to the student nodes, making the network, and therefore the cluster, inoperable. Within a few days, a university network administrator reconfigured the classroom switch, and our second attempt to create the cluster, a week later, was successful.

Once we were able to create the cluster successfully in the classroom, we became aware of a second, less critical, problem: the instructor's low power netbook, which served as the master node, had insufficient processing capability to boot all of the student laptops quickly enough; as a result, we were limited to six nodes in the cluster. Attempts to boot more nodes resulted in network timeouts. We could have increased the number of nodes somewhat by adjusting the network timeout parameters, but there was no compelling reason to do so: the demonstration was sufficiently impressive, even with only six nodes. In the future we plan to use a more powerful laptop computer for the master node.

Given a six-node cluster, both the regression analysis example and the HPL benchmark worked as expected. The measured time dropped by a factor of more than 2 when the first student node was added, (this was true because the netbook had significantly less processing power than the student laptop), and continued to drop proportionally as we added nodes to the cluster. The change of speed with which the graphic output from the regression analysis forms was particularly impressive to students.

8. CONCLUSIONS

We were extremely pleased with the results of this effort. The classroom cluster proved to be a useful exercise for introducing cluster concepts to the students. We were able to explain and demonstrate clustering technology using classroom facilities and software that were readily available at no cost, with a minimum of preparation time, classroom time, or effort. The method we used was sufficiently interesting to keep student attention at a high level during the exercise. Later testing indicated that we succeeded in achieving the understanding of clustering that we wanted the students to have. The use of push technology made it easy to implement and operate in the classroom and live technology assured that we did not have to rely on advanced preparation on the part of the students to implement the cluster, nor have concern for any lasting effect on the students' computers. Although the exercise was initially confined to the elective Linux operating system course, it is being extended this fall into the basic undergraduate and graduate computer infrastructure courses. For students with more interest in the subject, we will make copies of the live CD available to students for further experimentation. There are also plans to make the technology available for use in the networking lab that is associated with our elective advanced networking course.

9. REFERENCES

Apon, A., Buyya, R., Jin, H., & Mache, J. (2001). Cluster Computing in the Classroom: Topics, Guidelines, and Experiences. *1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001),* Brisbane, Australia, May 15-18, 2001. At www.buyya.com/papers/cc-edu.pdf

Bookman, Charles. (2003). Linux Clustering: Building and Maintaining Linux Clusters. New Riders, Indianapolis

Creel, M. (2010a). PelicanHPC Software. At http://pelicanhpc.org.

Creel, M. (2010b). PelicanHPC Tutorial. At http://pelicanhpc.org/Tutorial/PelicanTutorial.html .

Englander, I. S. (2010), The Architecture of Computer Hardware, Systems Software, & Networking: An Information Technology Approach, 4th Edition. J Wiley, Secaucus, NJ.

Hastings, D. A. (2003). Experience Teaching Hands-on Parallel Computing at a Small College. *J. of Computing Sciences in Colleges, 18*(3), 62-67.

---, High Performance Cluster Computing website. At http://icl.cs.utk.edu/hpcc/.

Kitchens, F. L., Sharma,S. K., & Harris, T. (2004). Intgrating IS Curriculm Knowledge through a Cluster Computing Project- a Successful Experiment. *J. of Information Technology Education,* 3. 264-278.

---. "Message Passing Interface". (2010, June, 15) At http://wikipedia.org/wiki/Message_Passing_Interface.

Montante, R., (2002). Beowulf and Linux: an Integrated Project Course. *J. of Computing Sciences in Colleges*, *17*(6), 10-18.

Petitet, A., Whaley, R. C., Dongarra, J. & Cleary, A. (2008). HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-

Memory Computers. At http://www.netlib.org /benchmark/hpl/.

Prins, P. (2004). Teaching Parallel Computing using Beowulf Clusters: a Laboratory Approach. *J. of Computing Sciences in Colleges, 20*(2), 55-61.

Sterling, T., Salmon,J., Becker, D. J., & Savarese, D. F. (1999). How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters. MIT Press, Cambridge, MA.

Vrenios, A. (2002). Linux Cluster Architecture. Sams, Indianapolis.