2006-1671: HIGH PERFORMANCE COMPUTING IN CLASSROOM ENVIRONMENT

Farid Farahmand, Central Connecticut State University

F. Farahmand is currently with the Computer Electronics and Graphics Technology department at Central Connecticut State University, New Britain, CT. He is a recent Ph.D. graduate from the University of Texas at Dallas. He has several years of teaching and industry experience combined with research background in optical and sensor networks.

Veeramuthu Rajaravivarma, Central Connecticut State University

V. Rajaravivarma is currently with the Computer Electronics and Graphics Technology department at Central Connecticut State University, New Britain, CT. Previously, he was with Tennessee State University, Morehead State University, and North Carolina A&T State University.

High Performance Computing in Classroom Environment

Abstract

The need for high performance computing continues to grow in the coming years. Such needs are no longer limited to highly advanced scientific organizations. Today, many educational and business communities also desire high computational power to expand their activities and maintain their services. However, the high cost of a single supercomputer with massive multi-processing power makes them infeasible for many such users. An alternative approach to high performance computing is computer clustering, which provides practical and feasible infrastructure to accommodate high computational power. The key concept in computer clustering is to unify available computing resources. Recently, many different organizations, including corporations and universities, have been implementing the cluster computing. Consequently, learning and understanding the basics of cluster computing can be considered as a valuable academic investment for IT and technology students.

The main purpose of this paper is to present a practical demonstration of computer clustering. We introduce a simple and easy-to-use Windows-based graphical software toolkit, called Paloma (Parallel Local Message-passing Adaptor), which was developed by Advanced Internet Technology Lab at CCSU. Paloma allows students to easily create a computer cluster using multiple PCs interconnected to each other via LAN or Ethernet. Through Paloma's integrated GUI and its performance monitoring capacity, the students will understand the motivation for high performance computing, advantages of computer clustering, as well as, the challenges and limitations of a cluster system. Flexibility of Paloma allows students to use it outside the classroom environment and develop their own what-if scenarios. In this paper we describe a number of test cases for laboratory experiments, which appeared to be very attractive to our students. Furthermore, we discuss possible extensions to these demonstrations and briefly outline our future plans for more sophisticated benchmarks and graphical features for Paloma to improve its educational aspect.

Introduction

Today's ever-growing demand for information and services has led academic and industrial communities to seek for high computing power at reasonable cost. The desire for higher computational power may be due to many reasons, such as real time constraints (completing a task within a certain time period), throughput (processing many related tasks together), or memory (delivering an efficient way to provide large amount of memory) [1]. Thus, the computer intensive applications are no longer limited to a few scientific communities. In fact, emerging commercial applications such as simulating mechanical devices, analyzing electronic circuits, investigating manufacturing process, modeling chemical reactions, collecting and processing financial transactions, offering

video conferencing, supporting advanced graphics and virtual reality, etc. happen to be the driving forces behind the development of faster computers.

As the technology enhances and the market demands more sophisticated applications and services, greater number of organizations realize that they can no longer satisfy their computational needs using a single commodity computer in order to maintain their competitiveness. On the other hand, high cost of high-performance computers capable of executing hundreds of thousands or millions of floating point operations per seconds (FLOPS), known as *supercomputers*, make them inaccessible to many organizations with limited budget.

In the past decade, cluster computing has been recognized as the feasible alternative to expensive high-end computers with multiple processors. In fact, cluster computing has become the enabling technology, providing affordable high-performance computing, high reliability, and easy upgradeability. A computer cluster consists of multiple off-the-shelf inexpensive (commodity) PCs that are connected together. Through networking and software, these independent computers can be combined into a unified system and collectively process requested tasks as a single machine. Hence, while the Internet is about getting computers to *talk* to each other, cluster computing is about getting computers to *work* together.

With the advent of powerful and inexpensive PCs, cluster computing is assuming an important role in future competitive market. Consequently, an increasing number of businesses and educational institutions are considering either building or purchasing cluster computing systems. Looking at such opportunities, Microsoft has introduced Beta version for Windows Compute Cluster Server 2003 [2], promising secure and affordable high-performance computing solution. On the other hand, IBM and many other leading computer industries have been offering complete computer clustering solution [3].

Facing such opportunities, IT and technology students, who have been introduced to cluster computing and understand its advantages and limitations can be very valuable in the future competitive job market. Therefore, a simple and effective approach to demonstrate the basic concept and capabilities of cluster computing can be highly beneficial to students.

In this paper we present a simple and easy-to-use Windows-based graphical software toolkit, called Paloma (Parallel Local Message-passing Adaptor), which allows students to easily create a cluster on multiple PCs interconnected to each other via LAN or Ethernet. Paloma is free, open-source, and can be executed using any PC with Windows operating system (i.e., Windows XP/2003/2000). A major advantage of Paloma is that it requires no additional installation or modification of system parameters. In spite of its development complexity (using massage-passing libraries and C-based sockets to provide connection points between the computer nodes), Paloma is designed to be user-friendly. In fact, it is intended for students with no programming skills, who prefer to learn from concrete and hands-on examples, as is typical of many IT and engineering technology students. Using easy-to-understand graphical interfaces, students can run different

benchmarks and visually observe the speedup in the cluster environment as a sample application, namely the Mandelbrot Set[1], is executed. Students can also examine the cluster performance under different system conditions and constraints.

Flexibility and ease-of-use of Paloma allows students to use it inside and out of classroom environment and, for example, test it at their convenience on their own home network. In fact, this was the case with our students and they played a very active role in developing and improving many of the forthcoming reported test cases. It is worth mentioning that most of participating students had no prior knowledge about computer clustering or distributed computing. Yet, they were able to develop their own what-if scenarios and learn from them. Furthermore, Paloma's open-source availability, allows students with more advanced programming skills to change the source code and add more sophisticated benchmarks and graphical features.

Motivation

There are many readily available literatures describing how to build a cluster computer, depending on the operating system of interest [2], [5]-[6]. In general, a cluster computer consists of several nodes (PCs or workstations), each containing one or more processors, memory and additional peripheral devices connected by a network that allows data to be transferred between the nodes. Depending on its cost, performance, and reliability, different network technologies can be selected, including Fast Ethernet or Gigabit Ethernet [7]. A network of PCs does not constitute a cluster until all PCs are configured to work together and act as a team to process a single task. This requires initializing the software environment on the cluster, which allows running applications using parallel programming. There are a number of software tools supporting cluster computing such as PVM (Parallel Virtual Machine) [8] and more recently MPI (Message Passing Interface) [9]. Using standard benchmarks, such as High-performance LINPACK, efficiency and performance of different cluster systems can also be measured and compared [10]-[11].

The main concern that led us to develop Paloma was constant difficulties that students were experiencing during the course of loading and installing different software tools to configure the cluster. Even some of the step-by-step procedures proved to be cumbersome and time consuming to implement for many students. Furthermore, the examples included in most existing software toolkits often provide no insight as to what is happening and how each node is contributing to the overall execution of an application, where the bottlenecks are, and how system constraints and modifications can impact the performance. Considering all these shortcomings, although there are no data comparing the performance of Paloma and other existing MPI-based software toolkits, we believe, for the purpose of teaching, Paloma serves as an excellent tool.

[1] Although other applications can be used for demonstration purpose, we chose the Mandelbrot Set due to it graphical nature, making it ideal for examining the performance of the computer cluster. Upon completion of the Mandelbrot application, students can observe very impressive fractal (self-similar) images.

Description of Paloma

Paloma, as a stand alone executable program and requires no installation. Once Paloma is executed on a number of computers on LAN, a single node can be selected as the *master* node controlling the distribution of the task to a set of operating *slave* nodes. Such master/slave paradigm is a commonly used approach for distributed applications. In this paradigm, the accepted task by the master (or server node) is divided task into sub-tasks, each of which is sent to one or more slave (or client) nodes for processing. Each slave node processes the task and returns its partial or full response to the master via the LAN connection. The advantage of this approach is that the master node is often free to receive new requests and communicate with slave nodes. Figure 1 shows the physical interface between the master and slave nodes using LAN or Ethernet required by Paloma. We note that in this configuration we assume all computer nodes on the LAN are homogenous and fully trusted by each other. Hence, we ignore issues such as security and resource selection.

Experiments

The computer cluster experiment was performed in a classroom with 20 students each having a PC connected to the LAN. We ensured that no computer is behind firewall. All PCs were identical and configured as follows: Windows XP operating system, 1.13GHz Pentium III CPU, 512MB RAM, 10GB hard drive, a 1.44MB Floppy, and a CD ROM.

We divided the class into five groups and asked students to run Paloma on four computers in the group with one node designated as master and the rest as slave nodes. Upon execution of Paloma, two windows appeared: the Connection and the Control, as shown in Figure 2(a) and Figure 3(a), respectively. The description of each parameter on the Control and Connection window is shown Figure 2(b) and Figure 3(b), respectively. The Connection window is used to assign the master and slave node. On the other hand, the Control window is used to perform three different functions:
(a) Lists all the clients (slave nodes) connected to the server (master) node. For example, for Figure 1, the Control window on the master node will be listing three client nodes. (b) Determines the resolution and position of the fractal images of the Mandelbrot set executed by Paloma. The values of these parameters can be left to students. In fact, this happened to be an interesting part of the experiment for many of our students as they changed the zoom parameters and obtained different images. (3) Plots a graph identifying the contribution of each node in the cluster in executing the task (completing the fractal image generated by be Mandelbrot set).

In order to operate Paloma and set up the cluster, students were instructed to perform the following steps:

a) Determine the master (server) node and find its IP address (in Figure 1, the master node address is 192.168.0.1).
b) Type in the IP address of the master node in the Connection window, check the *Be a Server* box and click *Go*.

c) For each slave (client) node, type in the IP address of the master node in the Connection window, check the *Connect to Server* box and click *Go*.
d) Set the parameters on the Control window of the master node. You can also use the default setting.
e) Click *Go* on the Control window of the master node.
f) Observe the plot in the *Graph* both of the Control window. Also, observe the fractal image generated on each node.

When the cluster is constructed and Paloma is executed, the master node splits the workload, which is displaying the fractal image, and sends individual sub-tasks to slave (client) nodes for evaluation. Hence, each node displays the portion of the fractal image that it processed, indicating its processing contribution to the overall workload. The master node on the other hand, will be displaying the combined image generated by each slave node.

As the workload is being completed and the fractal image is constructed, the *Graph* box in the Control window displays the contribution of each client node in completing the overall task. The plot in the *Graph* box can be displayed in two modes: (a) iterations vs. time, which is in terms of millions of iterations (tasks) performed per second; (b) rate vs. time, which is simply the derivative of the former. Figure 4 shows typical examples of such plots for a cluster with a single slave node. Note that, as Figure 4(a) indicates, it took 8.3 million iterations in total time of 6.2 seconds to complete the task, before the entire image is displayed. Figure 4(b) show the rate at which the processing is performed as the time increases.

After becoming familiar with basic feature of Paloma and its control parameters, we asked our students to perform more advanced tests. In the following paragraphs we detail two specific experiments, which we asked each group of student to perform and note the results, a more complete list of tests, including more advanced cases, is provided in [13].

Extended Test Cases

Cluster performance with multi client nodes:
In this test we asked students to experience and compare the performance of the cluster as more client nodes are added to the cluster. Hence, they first setup a cluster with *three* slave node and then run Paloma. Upon completion of the program, by referring to the results on the graph box, the students can observe that the total time it took for the image to be completed and the total number of iterations (in million) performed by slave nodes. This is shown in Figure 5. Note that the time of execution in this case is smaller than the one shown in Figure 4(a), using a single client node in the cluster. Comparing Figure 5(b) and Figure 4(b) shows that the total rate of processing over time is significantly different from when a single client is used in the cluster.

Overloading a client node in the cluster:
Using this test the students can observe the impact of overloading on the cluster performance. In this case, one of the clients in the cluster is heavily loaded. This can be

performed by opening multiple programs simultaneously on the machine or executing a single C program, which results is infinite loops and consumes keeps the processor busy. As Figure 6 indicates, performance drop can be observed when the client is busy performing other tasks, simultaneously.

Cluster performance with hyper-threading:
Investigating the impact of having a machine which supports hyperthreading is also interesting and very educational. In this experiment, we asked two groups to share a single client node. Hence, a single client with be shared by two independent clusters. Figure 7 compares the performance of the client node with and without hyperthreading capability. This figure shows that there is a slight benefit in running two instances of the job on a single machine if the machine supports hyperthreading. Then, we asked three or more groups of students share the same client node, with hyperthreading capacity. In this case, the students observe that sharing the client node, with hyperthreading, between more than two clusters is no longer beneficial and the performance, in fact degrades.

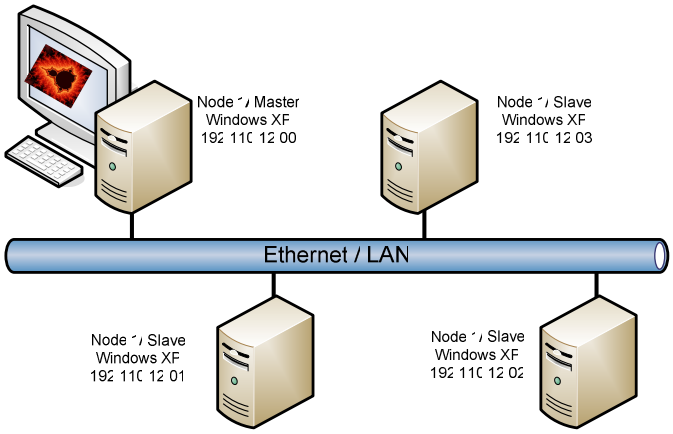
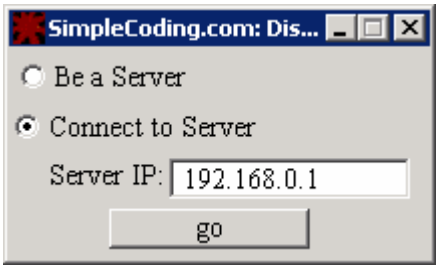


Figure 1: Laboratory cluster setup using Palma software toolkit. ?? ADDRESS CHANGE

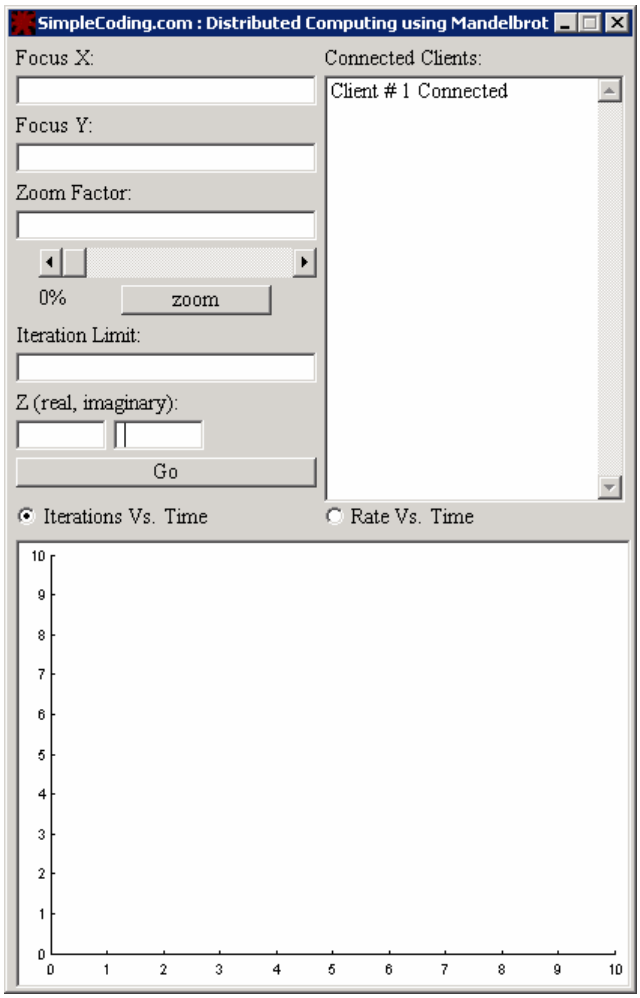


(a)

Box	Description
Be a server	Sets the program as a server to wait for incoming connections.
Connect to Server	Sets the program as a client to connect to a remote server instance of the program.
Server IP	IP address of the remote instance of the program that has been set as the server.
Go	Activates the program to either connect or start waiting for connections.

(b)

Figure 2: (a) Connection Window in Paloma; (b) description of each parameter in eh window.



(a)

Box	Description
Focus X	Real number to horizontally position the image to be generated. (click the display window to set this field)
Focus Y	Real number to vertically position the image to be generated. (click the display window to set this field)
Zoom Factor	Ratio between the image size and the resolution of the screen. The lower the number, the closer the zoom. (Can use the zoom button to set this field)
Zoom Percentage	Linear Percentage of the image area. This basically determines how much to divide the "zoom factor".
Zoom	Divides the "Zoom Factor" by the percentage determined by the Scroll Bar.
Iteration Limit	Integer that determines how many iterations a pixel must last to pass the Mandelbrot recursive test.
Z (Real,Imaginary)	Real and Imaginary components of the number "Z" to be used in the Mandelbrot test.
Button: "Go"	Starts the demonstration.
Iterations Vs. Time	Causes graph to display Millions of iterations vs Seconds.
Rate Vs. Time	Causes graph to display Millions of iterations per second vs sec. (derivative of "Iterations Vs. Time")
Connected Clients	Shows how many clients are connected and numbers them based on order of first connected.
Graph	Displays each client's contribution using a different color. The sum of the clients is the black line.

(b)

Figure 3: (a) Control Window in Paloma; (b) description of each parameter in the window.

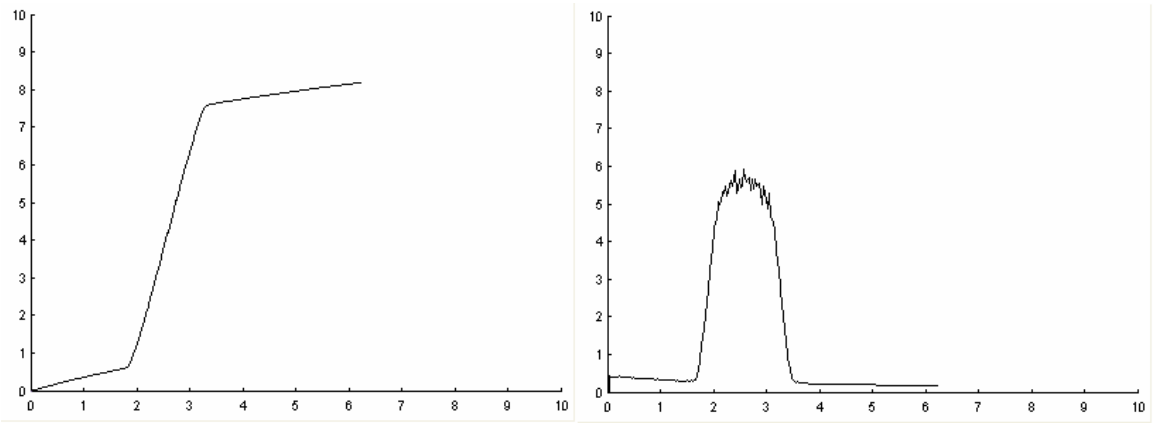


Figure 4: (a) Numbers of iterations (in millions) per second and (b) rate of processing over time using a single client in the cluster.

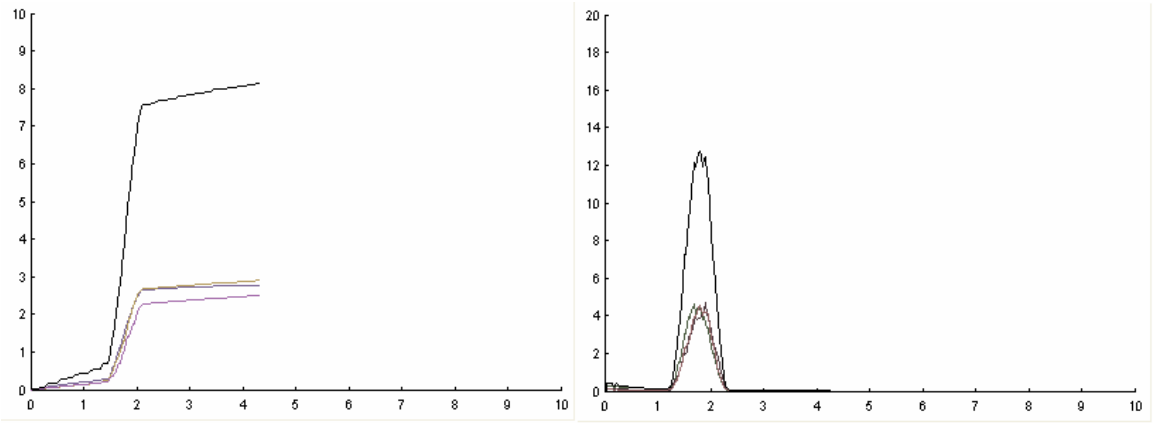


Figure 5: (a) Numbers of iterations (in millions) per second and (b) rate of processing over time using three client nodes in the cluster.

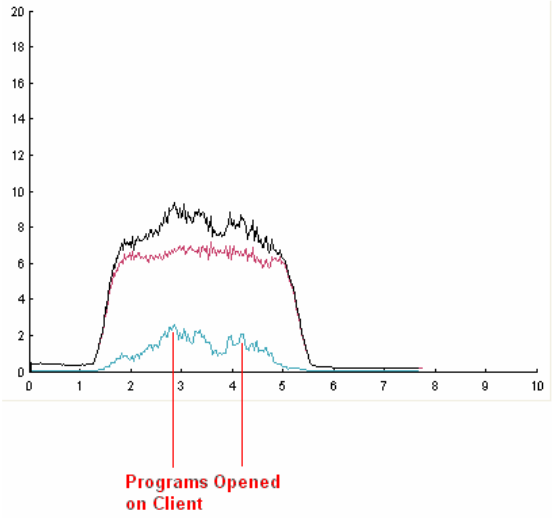


Figure 6: Rate of processing over time using two client nodes in the cluster. Note that performance drops as the rate decreases when programs are opened on the client side machine.

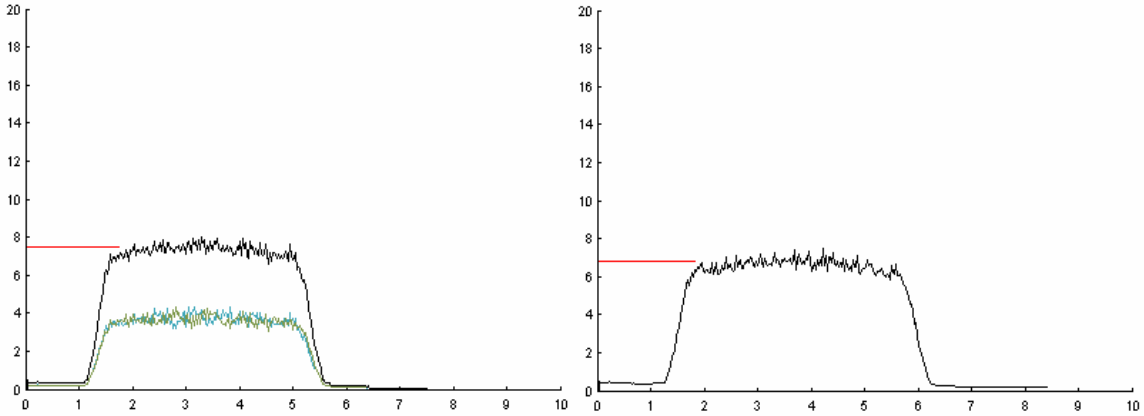


Figure 7: Rate of processing of a single client over time when it is shared by two clusters. (a) The client supports hyperthreading; (b) no hyperthreading is supported by the client node.

Conclusion

In this paper we presented a practical demonstration of computer clustering. We introduced a simple and easy-to-use Windows-based graphical software toolkit, called Paloma. Paloma allows students to easily create a computer cluster using multiple PCs interconnected to each other via LAN or Ethernet. Flexibility of Paloma allows students to use it outside the classroom environment and develop their own what-if scenarios. In this paper we describe a number of test cases for laboratory experiments. These test cases can easily be extended to examine more complicated and advanced scenarios.

Acknowledgement

The authors would like to thank Mr. Matt Conti and Mr. Jeremy Monteiro for their efforts in setting up and testing the experiment at the Advanced Internet Technology in the Interest of the Society (AITIS) laboratory at Central Connecticut State University.

Bibiliography

[1] Beowulf Cluster Computing With Linux, 2nd Edition, edited by William Gropp, Ewing Lusk, and Thomas Sterling, published by MIT Press, 2003; ISBN 0-262-69292-9.

[2] Windows Compute Cluster Server 2003 Product; updated November 15, 2005; available online at Overview http://www.microsoft.com/windowsserver2003/ccs/overview.mspx

[3] IBM Clustering Technology, http://www.research.ibm.com/topics/popups/deep/scalable/html/cresearchers.html

[4] Linux Cluster Architecture, Alex Vrenios, published by Sam's Publishing; ISBN 0-672-32368-0.

[5] Karl Kopper, "*The Linux Enterprise Cluster: Build a Highly Available Cluster with Commodity Hardware and Free Software*", Oreilly & Associates Inc., 2005.

[6] How to Build a Beowulf - by Thomas Sterling, Paul Angelino, Don Becker, Jan Lindheim, & John Salmon http://www.cacr.caltech.edu/beowulf/tutorial/tutorial.html

[7] Computer Networks and Internets, Dougles E. Comer Prentice Hall, Inc. 1999. 3-Designing and Building Parallel Programs (online), Ion Foster

[8] Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sundaram (1994), *PVM: Parallel Vir- tual Machine*, The MIT Press: Cambridge, Massachusetts

[9] Pacheco, P. (1997), *Parallel Programming with MPI*, Morgan Kaufmann Publishers Inc.: San Francisco, California.

[10] Jack J. Dongarra et al. *Java Linpack Benchmark*; available at `http://www.netlib.org/benchmark/linpackjava/`

[11] LAPACK -- Linear Algebra Package available at http://www.netlib.org/lapack/ *4-How to Build Cluster Computers* (online article); www.devbuilder.org/article/24

[12] The program is available at www.simplecoding.com/Downloads/Distributed Mandelbrot.exe

[13] Avaiable at www.ccsu.edu/~farahmandfar