Construction of Parallel Machines for Engineering Courses with Retired Personal Computers

Yili Tseng[1] and Claude M. Hargrove[2]

Abstract – As the parallel processing techniques have matured over the years, parallel computing becomes more and more powerful. As most of engineering problems and courses are based on mathematical modeling which appreciates ample computing power, parallel machines make the best platform to carry out the computations. Thanks to the development of open source parallel processing library, it is possible to build a powerful cluster out of commodity personal computers. Almost all institutions have plenty of retired personal computers. Supplemented with free operating systems and parallel processing library, they can be built into a parallel machine with minimal cost. The performance of the cluster is sufficient for education purpose. Offering parallel processing courses is no longer restricted by the resource issue. Procedures with practical details to build a cluster using PCs and free parallel processing library, MPICH, are introduced.

Keywords: engineering course, parallel processing, cluster, MPI, MPICH

INTRODUCTION

As the computing power of computers is strengthened, the digital or numerical methodologies have emerged for a variety of disciplines. In turn, as the digital methodologies evolve, there is a higher demand for computational power [Quinn, 1][Kaniadakis, 2][Pacheco, 5]. This cycle will stop when the computers hit the physical limitation. Unfortunately, the physical limitation of computers made of silicon is almost reached. That is, the execution speed can not be pushed faster and no additional data can be transferred in the same time frame. The only solution is to apply parallel processing before computers built on technologies other than silicon appear. As the parallel processing techniques have matured over the years [Grama, 4], parallel computing becomes more and more powerful and affordable. In the past, only major research institutes could afford parallel machines because of their extremely expensive prices. Now parallel machines can even be built out of connected commodity personal computers which are powerful and inexpensive [Sloan, 10]. This development means every researcher can take advantage of the computing power brought by parallel computers. Researchers can improve the accuracy of their existing algorithms by incorporating more computation steps into original algorithms while maintaining same or less execution times with parallelization.

As almost all engineering disciplines apply mathematical modeling and numerical methodologies, parallel processing has been adopted by most of them [Quinn, 1][Kaniadakis, 2][Grama, 4][Pacheco, 5]. Parallel processing should be incorporated into engineering courses to prepare engineering students for parallel computing trend. Although commodity supercomputers can be built with clustered personal computers, they are always dedicated to research and cannot afforded being spared for teaching and training purposes. All institutions must have accumulated some retired personal computers over the years. Supplemented with free operating systems and parallel

[1] Dept. of Electronics, Computer, and Information Technology, North Carolina A & T State University, Greensboro, NC 27411, ytseng@ieee.org

[2] Dept. of Electronics, Computer, and Information Technology, North Carolina A & T State University, Greensboro, NC 27411, cmhargro@ieee.org

processing library, they can be assembled into a parallel machine with minimal costs. Despite these old computers are slow compared with the high-end personal computers available in the market, the performance of the constructed cluster is still sufficient for training purpose of parallel programming. One draw back with the free parallel processing software libraries is that they come with no support because they are offered by other research institutions. There are still a number of practical issues left to be solved before a parallel computer can be put to real use. This paper intends to address those undocumented practical issues.

In Section 2, we describe the hardware consideration and our implementation. Section 3 includes the details of setting up the MPICH library, an implementation of Message Passing Interface (MPI) standard for parallel programming by Argonne National Lab. Section 4 explains the importance and the configurations of NFS and NIS, which facilitate executing parallel programs on a cluster. Section 5 discusses alternative parallel software package available and compares them with MPICH. Results from a testing parallel program are reported in Section 6 to reveal the characteristics of parallel computers with message-passing architecture.

HARDWARE

Architecture

There are five popular parallel machine models based on the Multiple Instruction Multiple Data (MIMD) Architecture, the most capable parallel computer: Parallel Vector Processor (PVP), Symmetric Multiprocessor (SMP), Massively Parallel Processor (MPP), Cluster of Workstation (COW) and Distributed Shared Memory (DSM) [Quinn, 1][Dongarra, 3]][Grama, 4] [Pacheco, 5]. The COW Architecture as shown in Figure 1 is the only one which can connect generic personal computers to make a parallel machine and hence the cheapest solution among the five models. Therefore, it is the most popular architecture to build a parallel computer. Naturally, the COW architecture was chosen to implement our parallel machine.

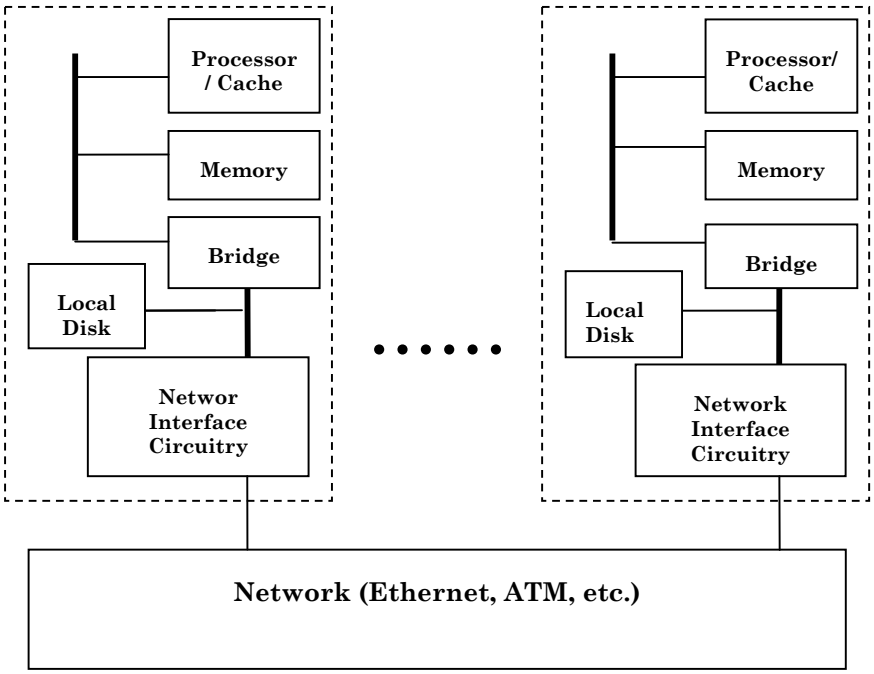

Figure 1: COW Architecture Model

Implementation

We built a 4-node Commodity Supercomputer as in Figure 2, using the COW architecture. The required hardware is as follows:

- Four Compaq PC's (Intel Pentium III 700 MHz CPU, 512 MB Memory, 40GB hard disk) with 100BaseT Ethernet network interface cards
- 1 network hub and one RJ45 cat. 5 cable for each PC

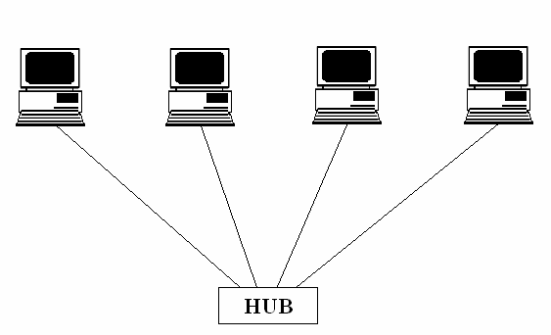

Figure 2: a PC cluster consist of 4 nodes

The four PC's were connected to the hub through a RJ45 cat. 5 cable to constitute a local network. Every PC works as a workstation. We installed the Fedora 6 Linux operating system (freely downloadable from http://fedoraproject.org/) on each workstation. The NFS, NIS, and rsh servers are required for the parallel processing operations but are not in the default installation. Hence, they have to be explicitly selected during the Linux installation process. The network settings were configured as soon as the operating system had been installed. For TCP/IP configuration, each computer was given an IP address, 10.0.0.1, 10.0.0.2, etc. The names, "Aggie1" through "Aggie4", were assigned as hostnames to each computer. After all the hardware and software was completely set up, the machines were ready to be installed the parallel processing libraries.

PARALLEL AND UTILITY SOFTWARE

MPI

Message-Passing Interface (MPI) is a standard specification for message-passing libraries and is the most suitable to COW architecture. MPICH is an available implementation of the MPI standard that runs on a wide variety of parallel and distributed computing environments. MPICH is attainable by anyone at no cost. MPICH contains, along with the MPI library itself, a programming environment for working with MPI programs. The programming environment includes a portable startup mechanism, several profiling libraries for studying the performance of MPI programs, and an X interface to all of the tools.

The version of MPICH we used is 1.2.7. It is downloaded from http://www-unix.mcs.anl.gov/mpi/mpich1/. Because MPICH is free software, full support documentation is not available. Since we had no support, we were faced with some difficulties during the installation process. The following are step by step instructions on the configuration process compiled from information scattered in the installation manuals [Gropp, 6] and from our experiences through trials. They are executed with the root account unless otherwise stated.

1) Unpack the '*mpich.tar.gz*' file into a build directory. (Directory '*/mpich-1.2.7*' will do.)

2) Invoke configure with the appropriate prefix:

   a. cd /mpich-1.2.7

b. *./configure - -prefix=/usr/local/mpich-1.2.7*

3) Make MPICH

make install

4) Add '*/bin*' to your path:

export PATH=/usr/local/mpich-1.2.7/bin:$PATH

5) Ensure '*/bin*' was added to your path:

echo $PATH

6) Identifying other computers (hosts and DNS)

a. Open the Network Configuration window. Click the Hosts tab.

b. Add the IP address and host name of all machines on you network.

c. Edit the file /etc/hosts to ensure all IP addresses and host names just added are there and delete unrelated entries, for example, localhost.localdomain.

7) Create a user account. We used "user" as the account name.

8) Create a '*.rhosts*' file in the user home directory /home/user with the user account Change the protection on it to user read/write only: *chmod og-rwx .rhosts*. Add one line to the '*.rhosts*' file for each node. An example of what should be in the file is as follows:

Aggie1 user

Aggie2 user

Aggie3 user

Aggie4 user

rsh

Remote shell (rsh) is used to run a text-mode program on another computer without logging into it with telnet, rlogin, or the like. The use of the *rsh* service is necessary in starting MPICH processes. It should be configured with the following steps with root account:

1) Enabling *rsh* [Gropp, 6] [Sobell, 7]: By default, the *rsh* server is not installed. The *xinetd* server controls the availability of the *rsh* and *rlogin* services. This server is installed by default, but by default *rsh* and *rlogin* services are disabled. In order to enable these services, complete the following:
   a. Open the file '*/etc/xinetd.d/rsh*'. Go to the last line of the file and change "disable=yes" to "disable=no". Save the changes that you have made and close the file.
   b. Repeat the above steps for the file '*/etc/xinetd.d/rlogin*'.
   c. At this point, the *xinetd* daemon must be restarted to register these changes by completing the following:

   /etc/rc.d/init.d/xinetd restart

2) Setup of the *rsh* [Gropp, 6]:
   a. Create a file called '*/etc/hosts.equiv*' and add the names of all the nodes. All the nodes listed will not be asked for passwords when they use rsh to start the MPI process on the local node.
   b. Check the files '*/etc/hosts.allow*' and '*/etc/hosts.deny*' and make sure that they are empty.

Extend to Multiple Nodes

Every node should be ready to run MPI programs. Log in the user account to compile and run the example or your own MPI program to ensure every node is correctly configured. If not, troubleshoot each node individually. Do not attempt to run MPI programs on multiple nodes before you are sure that all nodes are ready as it will compound the troubleshooting process. Once all nodes are correctly configured, log in as root to edit a machine file, /usr/mpich-1.2.7/util/machines/machines.LINUX, to include the names of all nodes. When mpirun is executed, MPI processes will be dispatched to the nodes listed in this file.

NFS AND NIS

What are NFS and NIS?

- **NFS (Network File System)** is a file-sharing protocol for Linux and other UNIX systems. It was developed to allow machines to mount a disk partition on a remote machine as if it were on a local hard drive.
- **NIS (Network Information System)** was created by Sun Microsystems as a way of managing information that is shared among a group of host computers on a network. NIS allows computers can share a common set of users' accounts, user groups, TCP/IP host names, password, as well as other information [Smith, 8].

Need for NFS and NIS

- NFS: Before we run a parallel program on our cluster, we need to dispatch a copy of the program onto every node. For example, if our Supercomputer has 1000 nodes, copying the program to all nodes is impractical. NFS is a great solution to help us to complete this mission. With NFS the programs only need to be saved on the file server and a copy of the program is automatically copied to the other computers.
- NIS: While we use NFS to solve the copying problem, there are some potential security problems. Each user has access to the shared directory, meaning any user can run, correct, save, and delete others' programs. To remedy this problem, NIS was used to manage the shared directory.

Application of NFS and NIS

The "Aggie1" was set up as both the NFS file server and the NIS master server with the steps described below [Sobell, 7].

a) User accounts were created on the NIS master server ("Aggie1"), for example, "user1", "user2", "user3", etc. The user accounts, their passwords and relevant information were shared with the other three computers. This allows each user to logon to the system from any of the other three computers using the same username and password.
b) After the user account is created, every user should have a directory on "Aggie1". For example, user1's directory should be '*/home/user1/*', user2'directory should be '/home/user2/', and so on. The whole '*/home*' directory of the NFS file server ("Aggie1") is mounted to the same directory ('*/home*') of other three computers.
c) At this point each user would need to save the programs he/she wrote to the '*/home/username/*' directory on the "Aggie1", and then their programs will automatically be copied to the same directory of other three computers. When any user logs in to the system on any computer, they only can access their own directory ('*/home/username/*'), and have the ability to operate their own programs.

The combination of NFS and NIS solves not only the problem with distributing the program to all the nodes but also the security problem.

Configuration of NFS and NIS

Set up the NFS server [Sobell, 7]:

1) From the ***NFS Server Configuration*** window, click ***File → Add Share***. The ***Add NFS Share*** window appears. In the ***Add NFS Share*** window Basic tab, type the following information:
    - **Directory** – Type the name of the directory you want to share. (The directory must exist before you can add it.)

- **Host(s)** – Enter one or more host names to indicate which hosts can access the shared directory. Host names, domain names, and IP addresses that are allowed here. Separate each name with a space.
- **Basic permissions** – Click ***Read-only*** or ***Read/Write*** to let remote computers mount the shared directory with read access or read/write access, respectively.

2) To permanently turn on the NFS service, type:

   chkconfig nfs on

   chkconfig nfslock on

Set up the NFS client [Sobell, 7]:

To set up an NFS file system to mount automatically each time you start your Linux system, you need to add an entry for that NFS file system to the /etc/fstab file. The /etc/fstab file contains information about all different kinds of mounted (and available to be mounted) file systems for your Linux system.

The format for adding an NFS file system to your local system is the following:

host:directory mountpoint options 0 0

The first item (host:directory) identifies the NFS server computer and shared directory. Mountpoint is the local mount point on which the NFS directory is mounted, followed by the file system type (nfs). Any options related to the mount appear next in a comma0separated list.

For our system just need to add the following NFS entries to /etc/fstab:

Aggie1:/home /home nfs rsize=8192,wsize=8192 0

Set up the NIS client [Sobell, 7][9]:

1) Defining an NIS domain name

   Our NIS domain name were **Aggie**, we can set it by typing the following as the root user from the shell:

   domainname Aggie

   To make the NIS domain name permanently, you need to have the domainname command run automatically each time your system boots. We can do it by adding the command line (domainname Aggie) to a run-level script that runs before the ypbind daemon is started. We edited the /etc/init.d/network file and added the following lines just after the first set of comment lines.

   # Set the NIS domain name.

   domainname Aggie

2) Setting up the /etc/yp.conf file

   We had an NIS domain called **Aggie** and a master server called **Aggie1**, we should have the following entries in our /etc/yp.conf file:

   domain Aggie server Aggie1

   domain Aggie broadcast

   ypserv Aggie1

3) Configuring NIS client daemons

   We need set up an existing run-level script called ypbind to start automatically at boot time. To do this, you can run the following command (as root user from a Terminal window):

   chkconfig ypbind on

4) Using NIS maps

   For the information being distributed by the NIS server to be used by the NIS client, you must configure the /etc/nsswitch.conf file to include nis in the search path for each file you want to use. In most cases, the local files are checked first (files), followed by nisplus. The following are examples of how some entries appear:

   passwd: files nisplus

   shadow: files nisplus

   group: files nisplus

   hosts: fiels nisplus dns

   For our purposes, we need change above entries into the following entries:

   passwd: files nis

   shadow: files nis

   group: files nis

   hosts: fiels nis dns

   As soon as the /etc/nsswitch file is changed, the data from NIS maps are accessible. No need to restart the NIS service.

Set up the NIS server [Sobell, 7][9]:

1) To configure your Linux system as an NIS master server, you should first configure it as an NIS client (That is, set the NIS domain name, set up /etc/yp.conf, and configure client daemons as described earlier.)
2) Creating NIS maps

   To create NIS maps so that your Linux system can be an NIS master server, start from the /var/yp directory from a Terminal window as root user. In that directory, a Makefile enables you to configure which files are being shared with NIS. All default configurations in Makefile are ok for our purposes, so we don't need change them.

3) Configuring access to maps

   In the /etc/ypserv.conf file, you can define rules regarding which client host computers have access to which maps. For our purposes we just need add the following line into /etc/ypserv.conf to allow all hosts access to all maps:

   * : * : * : none

4) Configuring NIS serve daemons

We can use the following chkconfig command to set ypserv and yppasswdd scripts to start automatically at boot time.

chkconfig ypserv on

chkconfig yppasswdd on

5) Updating the NIS maps

If you modify the sources for NIS maps (for example if you create a new user by adding the account to the passwd file), you need to regenerate the NIS maps. This is done by a simple

make –C /var/yp

This command will check which sources have changed, creates the maps new and tell yperv that the maps have changed.

ALTERNATIVE CHOICES

There are alternative software packages available for building commodity clusters, namely OSCAR (Open Source Cluster Application Resources) and NPACI (National Partnership for Advanced Computational Infrastructure) Rocks [Sloan, 10]. Both are free and downloadable at http://oscar.openclustergroup.org/ and http://www.rocksclusters.org/wordpress/, respectively. Both include MPICH package and more open source packages for clusters. Although both intend to facilitate the installation of clusters, they also suffer from poor documentation and the lack of support. Their collection of several utilities for parallel processing further compounds the installation issues. The problems are more confusing and frustrating than those of installing MPICH. Therefore, installation and configuration of MPICH is the best lesson to start the parallel programming and computing with. OSCAR and Rocks will be easier to handle once you are familiar with MPICH.

TESTING THE SYSTEM

Testing Program

This program's purpose is to calculate a definite integral using the trapezoidal rule. This integral is the area of the regions under the curve of a nonnegative function *f(x)* where the base is bound by the x-axis from lower bound to the upper bound on an x-y axis chart. The approach is to divide the area under the curve into equal-distanced regions and apply a trapezoid to each partition to simulate the area under *f(x)* curve. In the sequential version program, the area of each trapezoid will be calculated one by one and summed up. In the parallel version program, the calculation of each trapezoid's area is distributed to different node and executed concurrently. The individual area is then passed back to a root node to be added up. The parallel version testing program was run on 2, 3, and 4 nodes respectively and compared against the sequential version program.

Results

The result of execution time against number of nodes is plotted in Figure 3 while the result of speedup against number of node is displayed in Figure 4. Both execution time and speedup scale to the number of nodes very well as the number of nodes increases. The performance of the "Commodity Supercomputer" is as expected. While this parallel computer is not suitable for intense computing required for research due to slow processors, it owns all the features for parallel processing. That means the parallel computer constructed with retired personal computer is sufficient to support the purpose of teaching and training parallel processing. As the software required is free and the only possible hardware cost for a hub and cables are minimal, every institution should be encouraged to recycle their decommissioned PC's to build parallel machines for engineering courses.

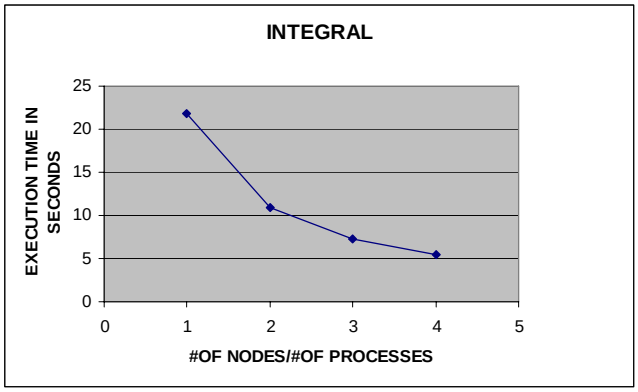

Figure 3: Execution time vs. number of nodes

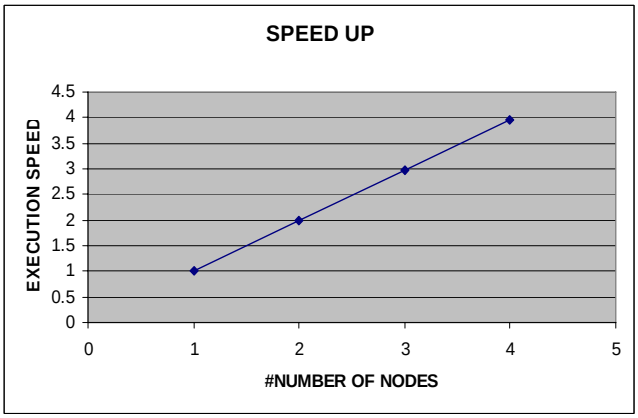

Figure 4: Speedup vs. number of nodes

CONCLUSIONS

As the digital and numerical methodologies in various disciplines continue to grow, the power and speed of the computer must also increase to keep up with the demand. A solution to the speed and throughput limitations of computers made of silicon processor chips is the acquisition of a parallel machine. Parallel processing is the only way to push the envelope of computing power. As the engineering disciples heavily utilize mathematical modeling and simulation, the facts make them the perfect candidate for parallel processing. In the past, extremely expensive costs of parallel machines practically prevented parallel processing from being incorporated into engineering courses. With the effective development of free Linux operating systems and parallel processing software over the years, it is now possible to build affordable commodity supercomputers. With our experiment, it proves that personal computers decommissioned from each institution can be effectively used to construct a parallel machine. The performance and features of the commodity cluster is adequate for education and training of parallel processing. As each institute must own plenty of retired PC's, several parallel machines can be assembled to serve several engineering courses. It is also the best way to recycle the retired PC's. Every institution should use their decommissioned PC's to construct parallel machines to enhance their engineering courses with parallel processing.

REFERENCES

[1] Quinn, Michael J., *Parallel Programming in C with MPI and OpenMP*, McGraw-Hill, 2004

[2] Karniadakis, George E. and Kirby, Robert M., *Parallel Scientific Computing in C++ and MPI*, Cambrige University Press, 2003

[3] Dongarra, Jacl et al., *Sourcebook of Parallel Computing*, Morgan Kaufmann Publishers, Inc. 2003

[4] Grama, Ananth et al., *Introduction to Parallel Computing*, Addison-Wesley, 2003

[5] Pacheco, Peter S., *Parallel Programming with MPI*, Morgan Kaufmann Publishers, Inc. 1997

[6] Gropp, William and Lusk, Ewing, *Installation and User's Guide to MPICH, a Portable Implementation of MPI version 1.2.7, The ch_p4 device for Workstation Networks*, ANL/MCS-TM-ANL-01/x Rev x

[7] Sobell, Mark, *A Practical Guide to Red Hat Linux*, Third Edition, Prentice-Hall, 2006

[8] Smith, Roderick W., *Advanced Linux Networking*, Addison-Wesley, 2001

[9] http://www.linux-nis.org/nis-howto/

[10] Sloan, Joseph D., *High Performance Linux Clusters*, O'Reilly, 2005

Yili Tseng

Yili Tseng received the BS degree in mechanical engineering from the National Taiwan University before serving as a second lieutenant of artillery in Taiwanese Army for two years. Later, he received the MS degree in engineering science from the University of Florida and the MS and PhD degrees in computer engineering from the University of Central Florida, respectively. He is currently an associate professor with Dept. of Electronics, Computer, and Information Technology, North Carolina A & T State University. He is the advisor of the computational technology concentration in the department. Prior to that, he has taught in Dept. of Computer and Information Sciences, Florida A & M University for four years. His research interests include high-performance computing, parallel and distributed computing, grid computing, and numerical methods.

Claude M. Hargrove

Claude M. Hargrove has been at North Carolina A&T State University at the rank of Assistant Professor since Fall 2004. He received his PhD at North Carolina State University in 1999 in Biological Engineering. Other degrees earned at North Carolina State University were MS in Computer Engineering, BS in Electrical Engineering and a BS in Computer Engineering. He has served as the Co-Advisor for the IEEE student chapter, Chair of Central Carolina IEEE section, member of Epsilon Pi Tau, and Graduate faculty for the Technology Management PhD consortium sponsored by Indiana State University. Interest includes biotechnology and STEM education. Submitted another paper to 2008 ASEE South East Section Annual Conference titled "A Method to Improve Course Instruction by Utilizing Teleconferencing Techniques".