5-2001

Cluster Computing in the Classroom: Topics, Guidelines, and Experiences

Amy Apon
Clemson University, aapon@clemson.edu

Rajkumar Buyya

Hai Jin

Jens Mache

Cluster Computing in the Classroom: Topics, Guidelines, and Experiences

Amy Apon[α], Rajkumar Buyya[β], Hai Jin[δ], and Jens Mache [φ]

Computer Science and Computer Engineering[α]
University of Arkansas, Fayetteville, Arkansas, USA
Email: aapon@comp.uark.edu

School of Computer Science and Software Engineering[β]
Monash University, Melbourne, Australia
Email: rajkumar@csse.monash.edu.au

Department of Electrical Engineering - System[δ]
University of Southern California, Los Angeles, USA
Email: hjin@ceng.usc.edu

Department of Mathematical and Computer Science[φ]
Lewis & Clark College, Portland, Oregon, USA
Email: jmache@lclark.edu

Abstract - With the progress of research on cluster computing, more and more universities have begun to offer various courses covering duster computing. A wide variety of content can be taught in these courses. Because of this, a difficulty that arises is the selection of appropriate course material. The selection is complicated by the fact that some content in cluster computing is also covered by other courses such as operating systems, networking, or computer architecture. In addition, the background of students enrolled in cluster computing courses varies. These aspects of cluster computing make the development of good course material difficult. Combining our experiences in teaching cluster computing in several universities in the USA and Australia and conducting tutorials at many international conferences all over the world, we present prospective topics in cluster computing along with a wide variety of information sources (books, software, and materials on the web) from which instructors can choose. The course material described includes system architecture, parallel programming, algorithms, and applications. Instructors are advised to choose selected units in each of the topical areas and develop their own syllabus to meet course objectives. For example, a full course can be taught on system architecture for core computer science students. Or, a course on parallel programming could contain a brief coverage of system architecture and then devote the majority of time to programming methods. Other combinations are also possible. We share our experiences in teaching cluster computing and the topics we have chosen depending on course objectives.

1. Introduction

Clusters, built using commodity-off-the-shelf (COTS) hardware components and free, or commonly used, software, are playing a major role in solving large-scale science, engineering, and commercial applications. Cluster computing has emerged as a result of the convergence of several trends, including the availability of inexpensive high performance microprocessors and high speed networks, the development of standard software tools for high performance distributed computing, and the increasing need of computing power for computational science and commercial applications [6][7]. Clusters have evolved to support applications ranging from supercomputing and mission-critical software, through web server and e-commerce, to high-performance database applications. Educators have an opportunity to teach many types of topics related to cluster computing in universities at various levels, from upper-division undergraduate to graduate.

Cluster computing provides an inexpensive computing resource to educational institutions. Colleges and universities need not invest millions of dollars to buy parallel computers for the purpose of teaching "parallel computing". A single faculty member can build a small cluster from student lab computers, obtain free software from the web, and use the cluster to teach parallel computing. Many universities all over the world, including those in developing countries, have used clusters as a platform for high perfo rmance computing.

Many resources are available for teaching cluster computing. For example, the IEEE Computer Society Task Force on Cluster Computing (TFCC) [16] provides online educational resources. It promotes the inclusion of cluster-related technologies in the core curriculum of educational institutions around the world through its book donation program in collaboration with international authors and publishers.

Even with all of the available resources for cluster education, it is difficult to design a good course that covers a reasonable subset of topics of cluster computing. The first difficulty has to do with the diverse set of topics that cluster computing entails. Many typical undergraduate or graduate courses have significant overlap with the topics that may also be covered in a cluster computing course. For example, undergraduate

courses in operating systems, networks, computer architecture, algorithms, or Java computing may cover topics such as threads and synchronization, network protocols and communication, or issues related to symmetric multiprocessing. Since all of these courses may or may not be required in the curriculum, students enroll in the cluster computing course with various backgrounds, depending on whether or not they have had such courses as prerequisites. It is impossible to assume all the above courses as prerequisites. If so, many students would not be able to take the cluster computing course because they would not have time to fit it in before graduation.

A second difficulty with designing a good course in cluster computing is the illusion that many students (and instructors) have about cluster computing. Building a cluster, such as a Linux cluster, is so easy that even a person without much experience can handle this with the guidance of a brief brochure. Such information can easily be found on the web. This ease of constructing clusters may give students a misunderstanding of the difficulties involved in cluster computing. The next challenge, which is how to make the individual computers operate as an integrated system for the purpose of solving a single problem, is more difficult. The main challenges lie in developing applications that exploit the cluster infrastructure, and in understanding the design tradeoffs for the cluster architecture. Applications need to be parallelized and developed using application programming. Students may be unprepared for the difficulties involved in these tasks.

To address these difficulties in teaching cluster computing, we present a set of sample syllabi of cluster computing for instructors to use in major universities. In this paper, we focus on how to design a good syllabus for a cluster computing course, considering various student backgrounds, and without repeating most of topics covered by other courses. In section 2, we first list possible topics of cluster computing. We provide a set of sample syllabi in section 3 for use in teaching cluster computing at both the senior undergraduate and graduate level. These syllabi cover the necessary topics related to cluster computing, including system architecture, parallel programming, algorithms, and applications. For the convenience of instructors, several related books and references are also listed. We discuss our experiences in teaching cluster computing in several universities in the USA and Australia in Section 4. Finally, we end with conclusions.

2. Prospective Topics for Teaching

A cluster is a type of parallel or distributed processing system that consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource [6]. A generic architecture of a cluster computer is shown in Figure 1. A node of the cluster can be a single or multiprocessor computer, such as a PC, workstation, or symmetric multiprocessor (SMP). Each node has its own memory, I/O devices and operating system. A cluster can be in a single cabinet, or the nodes can be physically separated and connected via a LAN. Typically, a cluster will appear as a single system to users and applications.

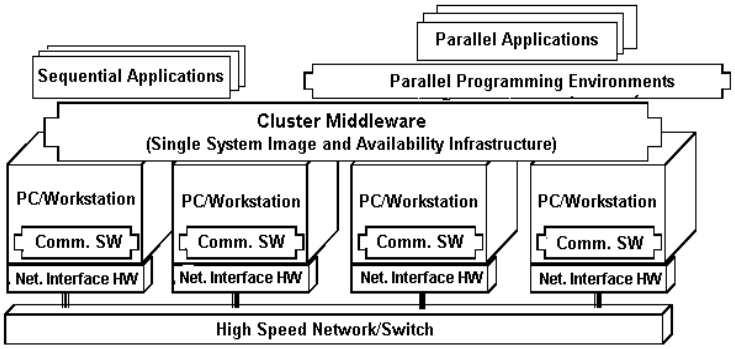

Figure 1: Typical Cluster Architecture. (Source [6])

In addition to describing these general cluster characteristics, topics that may be covered in a course on cluster computing include:

- System architecture
  - SMP and CC-NUMA multiprocessor architectures
  - Network interface
  - Network topology
  - Network communication protocols, including TCP/IP programming and low-latency protocols such as VIA (virtual interface architecture) [32].
  - Cluster run-time support, including
    - Hardware level support for communication such as Digital Memory Channel and hardware distributed shared memory
    - Operating systems such as Mosix [22]
    - Application level support, including runtime system support such as software DSM (e.g., TreadMarks) and parallel file systems (e.g., PVFS), and resource management and scheduling systems such as LSF and PBS.
  - Case studies, such as Linux Clusters, IBM SP2, Digital TruCluster, and Berkeley NOW [31].
- Parallel programming
  - Shared memory programming and tools such as POSIX and Java threads.
  - Distributed memory programming using message passing programming tools such as MPI and PVM
  - Middleware programming tools such as CORBA, Remote Procedure Call (RPC), Java Remote Method Invocation (RMI), Java Servlets, Java Database Connectivity (JDBC), and Jini.
- Parallel algorithms and applications
  - High performance algorithms and applications

- Techniques of algorithm design
- Performance evaluation and tuning, including optimization, visualization, high availability, network security, and benchmark experiments, such as NAS parallel benchmark (NPB) and Linpack benchmark.

System architecture topics relate the responsibility of the network interface hardware for transmitting and receiving packets of data between nodes and the responsibility of the communication software to offer an efficient and reliable means of data communication between nodes and potentially outside the cluster. For example, clusters with a high-performance network such as Myrinet generally use a user-level communication protocol such as VIA for fast communication between nodes. This interface bypasses the operating system and provides direct user-level access to the network interface, thus removing critical communication overhead. System-level middleware is responsible for offering the illusion of a unified system image (single system image) from a collection of independent but interconnected computers.

Programming topics include a discussion of portable, efficient, and easy-to-use tools for developing applications. Such environments include tools and utilities such as compilers, message passing libraries, debuggers, and profilers. A cluster can execute both sequential and parallel applications. A comprehensive discussion of cluster-based architectures, programming, and applications can be found in [6][7][13][25][29][30][33]. In the following sections, we discuss the organization of courses mainly based on these references and other sources.

3. Suggested Course Components

A cluster computing course can focus on a number of topics, including system architecture, programming environments and languages, the design of algorithms, and applications. One option is to conduct a separate course on each of these topics. For example, a full course on "Advanced Network-based Computer Architecture" or "Parallel Programming" can be taught as a single course. An alternate option is teaching a single course on cluster computing that comprises selected units from different course components. For example, instructors can pick complementary topics from system architecture and parallel programming to develop a course.

3.1 System Architecture

Junior undergraduate level courses in computer science teach computer organization, networking, basics of operating systems, and programming subjects. A student having studied these courses meets the prerequisite for a course on network-based advanced computer architecture. As cluster-based systems are developed using standard, COTS hardware and software components, a great exercise would be learning how to build one's own cluster-based high-performance and/or high-availability computer systems. Such a project can be coupled with a number of projects that focus on developing software that provides an illusion of a single system image. Course projects can also focus on developing scientific and business applications.

Two complementary textbooks on cluster-based system architecture issues that can provide the foundation for a course that focuses on system architecture are Buyya[6] and Pfister[25]. The system architecture course can be divided into four units: introduction, cluster building blocks, system-level cluster middleware (focusing on single system image and high availability infrastructure), and projects. Among these units, it is advisable to dedicate the largest amount of time (more than 50%) to system-level middleware.

Unit 1: Introduction

The computing industry is one of the fastest growing industries and it is fueled by the rapid technological developments in the areas of computer hardware and software. Many different computer architectures supporting high performance computing have emerged. These include: vector processors, Massively Parallel Processors (MPP), Symmetric Multiprocessors (SMP), Cache-Coherent Non-Uniform Memory Access (CC-NUMA), distributed systems, and clusters. The success of these systems in the marketplace depends on their price-performance ratio. This unit discusses these competing computer architectures and their characteristics. An important question to be addressed is, "What exactly is cluster computing, and why is it a good idea?" A primary goal is to understand the key reasons for the development of cluster technology that supports low-cost high performance and high availability computing. A suitable textbook for this unit is [13].

Unit 2: Cluster Building Blocks

Clusters are composed of commodity hardware and software components. The hardware components include standalone computers (i.e., nodes) and networks. Cluster nodes can be PCs, workstations, and SMP's. Networks used for interconnecting cluster nodes can be local area networks such as Ethernet and Fast Ethernet, system area networks such as Myrinet and Quadrics switches, or upcoming InfiniBand communication fabric [26]. Various operating systems, including Linux, Solaris, and Windows, can be used for managing node resources. The communication software can be based on standard TCP/IP or user-level messaging layers such as VIA. Suitable references for this unit are [6] (chapters 1, 9, 10) and [25].

Unit 3: System-Level Middleware

System-level middleware offers Single System Image (SSI) and high availability infrastructure for processes, memory, storage, I/O, and networking. The single system image illusion can be implemented using the hardware or software infrastructure. This unit focuses on SSI at the operating system or subsystems level. A modular architecture for SSI allows the use of services provided by lower level layers to be used for the implementation of higher-level services. This unit discusses design issues, architecture, and representative systems for job/resource management, network RAM, software RAID, single I/O space, and virtual networking. Books by Buyya [6] and Pfister [25] cover each of these topics in depth. A number of operating systems have proposed SSI solutions, including MOSIX, Unixware, and Solaris-MC. It is important to discuss one or more such systems as they help students to understand architecture and implementation issues.

Unit 4: Course Projects

Absorbing all the course's conceptual material is impossible without hands-on experience of some aspect of cluster systems. Fortunately, several cluster-based software systems are freely available for download (with source code), including Linux, VIA, PBS, Condor, MPI, PVM, GFS, GLunix, and MOSIX [8]. Students can explore these components by changing some of the policies used in these systems. For example, students can develop a new scheduling policy and program it as modification of the PBS cluster management source code.

Some of the projects that can be explored include:

- Build a low-cost cluster using PC's, Linux and Ethernet
- Install PVM and MPI on a cluster
- Evaluate various job management systems
- Develop tools for system administration
- Develop a simple job scheduler
- Implement a standard user level communication layer based on VIA
- Develop cluster monitoring tools
- Develop share-based scheduling policy and implement in systems such as PBS.
- Develop a computational economy-based policy for scheduling and implement it for PBS.
- Develop a web based job submission system
- Develop parallel Unix tools
- Students are advised to play with a number of clustering software systems listed in [8].

One of the best course projects for students is to develop web-based access mechanisms for clusters. Students can also identify deficiencies and limitations of existing systems and develop new solutions and policies to overcome them. Deeper explorations of new methods, mechanisms, and policies for single system image can also serve as nice thesis topics.

3.2 Programming

A course in cluster computing that focuses on programming can provide students with an abundance of hands-on experience with clusters. The tools that are required to teach these topics are generally available on most campuses, computer science students generally have the background required at the senior level to study cluster programming, and much tutorial material is available on-line in various locations. Thus, the course can cover a range of topics without requiring the students to purchase a large number of expensive textbooks.

It may be possible to teach a course on cluster programming with fewer prerequisites than is required for a course covering more advanced architectural topics. At a minimum, students should have been exposed to data structures, basic algorithms, and computer organization prior to the units in a course on cluster programming. A course on programming can be divided into four major units. The four units presented in this section are largely independent of one another, although the prerequisites for each unit may vary.

Unit 1: Shared Memory Programming

Given that many clusters are composed of symmetric multiprocessors, background in shared memory programming is a good entry into cluster programming for advanced undergraduates and beginning graduate students. A prior operating systems course is helpful, since an operating systems course usually covers topics such as mutual exclusion, synchronization, and the solution of classic problems. However, this material can be taught without prior operating systems exposure if some time is planned at the beginning to cover some basic operating systems concepts on processes.

Several languages are available to teach shared memory programming. The most accessible include:

- C or C++ using the pthreads library on Linux or Unix, or the threads library on Solaris. The pthreads library is a POSIX-compliant version of threads, and offers named condition variables. The monitor itself must be coded using mutex variables.
- Java threads. Java threads do not support named condition variables, but rather support wait sets on an object. An instructor using Java for shared memory programming may want to spend some time discussing programming techniques for Java threads. Depending on how much time the instructor wants to spend on language skills, the students do not have to have prior experience in Java to teach this unit in Java.

Students do not have to have access to a symmetric

multiprocessing computer to gain experience programming with threads. However, some programs that examine the performance of thread programs work best if the students have access to at least a dual-processor computer.

Typical topics that may be covered in a unit on shared memory programming include:

- Processes versus threads, pthreads, Java Threads
- Race conditions, synchronization locks, and mutual exclusion
- Monitors, semaphores, and solutions to classic synchronization problems
- Design patterns for concurrency

Resources include [1][5][12][18]. A suitable follow-on material for Unit 1 is the hardware issues associated with symmetric multiprocessors. A follow-on coverage of symmetric multiprocessors for students with limited hardware background is [25] Chapter 6. A sample course that covers this unit can be found at [2].

Unit 2: Message Passing Primitives

Although new high-performance protocols are available for cluster computing, some instructors may want to provide students with a brief introduction to message-passing programs using the BSD Sockets interface to Transmission Control Protocol/Internet Protocol (TCP/IP) before introducing more complicated parallel programming with distributed memory programming tools. If students have already had a course in data communications or computer networks then this unit should be skipped.

Students should have access to a networked computer lab with the Sockets libraries enabled. Sockets usually come installed on Linux workstations. Typical topics covered in this unit include:

- Basic networking, the Internet Protocol (IP) stack, client/server programming
- TCP and UDP sockets library
- Internet programming issues: stream messaging vs. datagram messaging, endian issues

A nice project that combines Unit 1 and Unit 2 is for the students to develop a multithreaded server application and the corresponding client using sockets and threads. A good follow-on for Unit 2 is some coverage of the Virtual Interface Architecture (VIA) [32], and discussion of the overhead in TCP/IP.

A convenient on-line resource for instructors is a *Guide to Sockets Programming*, available at http://www.ecst.csuchico.edu/~beej/guide/net/ and a sample course that covers this unit is [3].

Unit 3: Parallel Programming Using MPI

An introduction to distributed memory programming using a standard tool such as Message Passing Interface (MPI)[23] is basic to cluster computing. Current versions of MPI generally assume that programs will be written in C, C++, or Fortran. However, Java-based versions of MPI are becoming available. The resources for students for this unit include a networked cluster of computers with MPI installed. Setting up a cluster the first time can require some effort, but once the cluster is set up little to no maintenance is required on it throughout the semester.

Typical topics that may be covered include:

- Introduction to parallel computing
- I/O on parallel systems
- Tree communication, broadcast, tags, safety
- Collective communication: reduce, dot product, allreduce, gather/scatter, allgather
- Grouping data, derived types, type matching, pack/unpack
- MPI communicators and topologies
- Data-parallel vs. control-parallel
- Algorithm development and MPI programming

A good co-unit for this unit is some coverage of parallel algorithms and applications (see section 3.3.) Resources for instructors include [7][23][24][28][33] and sample courses that cover this unit are [3] and [9].

Unit 4: Application-Level Middleware

Application-level middleware is the layer of software between the operating system and applications. Middleware provides various services required by an application to function correctly. A course in cluster programming can include some coverage of middleware tools such as CORBA, Remote Procedure Call, Java Remote Method Invocation (RMI), or Jini. Sun Microsystems has produced a number of Java-based technologies that can become units in a cluster programming course, including the Java Development Kit (JDK) product family that consists of the essential tools and APIs for all developers writing in the Java programming language through to APIs such as for telephony (JTAPI), database connectivity (JDBC), 2D and 3D graphics, security as well as electronic commerce. These technologies enable Java to interoperate with many other devices, technologies, and software standards.

Advanced middleware products such as CORBA are often taught as an entire course of their own, often forming the basis for topics courses at the advanced undergraduate or beginning graduate level. Resources available to instructors include Jini [10] and Javasoft (http://www.javasoft.com/products/). A sample course for this unit is: http://csce.uark.edu/~rdeaton/cscomp/

3.3 Algorithms and Applications

In a course on cluster computing, it makes sense to study algorithms and applications, since "killer-applications" are one building block of cluster computing (together with killer-microprocessors, killer-networks, and killer-

tools). Moreover, covering algorithms and applications provides (1) the opportunity and context for programming projects and (2) examples of how clusters are put to work. The algorithms and application topic can be divided into four units: overview of applications, techniques of algorithm design, evaluation & tuning, and specific algorithms & applications.

Unit 1: Overview of Applications

Clusters have infiltrated not only the traditional science and engineering marketplaces for research and development, but also the huge commercial marketplaces of commerce and industry. It should be noted that clusters are not only being used for high-performance computation, but increasingly as a platform to provide highly available services, for applications such as web and database servers.

Clusters are used in many scientific disciplines, including biology (genome mapping, protein folding), engineering (turbo-fan design, automobile design), high-energy physics (nuclear-weapons simulation), astrophysics (galaxy simulation) and meteorology (climate simulation, earth/ocean modeling).

Typical topics that may be covered include [4]:

- *Internet applications*: Systems like Linux Virtual Server directs clients' network connection requests to multiple servers that share their workload.
- *Compression*: Systems like Evoke Communications' speech-to-email service uses clusters to perform transcoding that meets real-time requirements.
- *Data mining*: Efforts like Terabyte Challenge uses clusters at multiple sites to manage, mine, and model large distributed data sets for high-energy physics, health care or web crawlers.
- *Parameter study*: Tools like Nimrod uses a cluster to execute task farming and parameter study applications (the same program repeatedly executed with different initial conditions) as a means of exploring the behavior of complex systems like aircrafts or ad-hoc networks.
- *Image rendering*. A ray tracer can distribute the rendering among different nodes.

Unit 2: Techniques of Algorithm Design

It is important to show by example how to design and implement programs that make use of the computational power provided by clusters. Typical topics that may be covered include:

- Process-level parallelism
- Partitioning and divide-and-conquer
- Communication and synchronization
- Agglomeration and mapping
- Load balancing and termination detection

Unit 3: Evaluation and Tuning

Once the fundamental issues of the existence of sufficient parallelism have been addressed, there often are several algorithms or strategies available, and tradeoffs must be weighed to determine which is most appropriate. How to choose and develop appropriate algorithms? How to evaluate resulting implementations? How to optimize overall performance?

Parallel algorithms can be categorized according to a number of criteria, including regular or irregular, synchronous or asynchronous, coarse or fine grained, bandwidth greedy or frugal, latency tolerant or intolerant, distributed or shared address space. Typical topics that may be covered include:

- How to model, measure and analyze
- Visualization and debugging
- Optimization by minimizing communication and tolerating latency

Unit 4: Specific Algorithms and Applications

Typical topics that may be covered include:

- Sorting
- Numerical algorithms
- Image processing
- Graph algorithms
- Computational geometry
- Searching and optimization
- Genetic algorithms
- Parallel simulation
- Molecular modeling
- Climate ocean modeling
- Computational fluid dynamics

Teaching resources include [7][11][17] [19][27] [33].

4. Discussions and Experience

Generally, the topics chosen for a course on cluster computing depends on the course objective. In this section, we discuss our experiences with teaching courses that centered on cluster computing in our universities.

University of Arkansas

Cluster computing has been taught three times and the two different versions of the course have been offered at the advanced undergraduate level [2][3]. Students coming into the course have a varied background, but which includes at least computer organization and generally also operating systems. Serious students who have not had operating systems prior to cluster computing can be successful in the course. The best-prepared students either had good background in computer architecture, computer networks, or significant experience with programming or setting up computers.

The first part of the course covers shared memory parallel programming. Programming exercises have focused on the solution to the classic problems that are typically found in an operating systems course. The programs have not generally included concepts such as GUI interfaces, although these could be incorporated if desired. Since many students have not had a course in computer architecture, about a week of lecture is spent on hardware issues such as cache, symmetric multiprocessing, and snoopy bus architectures. If time allows, distributed shared memory and memory consistency are also covered.

The second part of the course covers distributed memory parallel programming using sockets and MPI. Programming exercises include a client and concurrent server program, a matrix/vector multiply program in MPI, and at least one advanced MPI program. The advanced program is sometimes beyond the capability of weaker students in the course, so it is generally at most ten percent of the total grade. MPI features such as collective communication, communicators, and efficient message passing are covered. Along with MPI, the course also covers programming issues such as control parallel versus data parallel programming and systems issues such as network protocol stacks, cost of network communication, Amdahl's Law, and a comparison of clusters to symmetric multiprocessors.

The third part of the course covers miscellaneous topics, as time allows, including high availability, single system image, tools for cluster setup and maintenance, and performance testing.

A significant portion of the course grade is based on a programming project that each student selects and which is due near the end of the semester. The project is a great way to allow students to focus more thoroughly on a topic that interests them, or to cover a topic for which class time did not allow.

Monash University

As part of the subject "CSC433: Parallel Systems" for BSc Honors degree, we dedicated half of the course to: (a) parallel systems and machine architectures and (b) various communication models and languages for parallel programming [9]. The topics covered in part (a) include: pipelined architectures, shared memory, distributed memory, SIMD, MIMD, MPP, and application specific parallel systems. The topics covered in part (b) include: early work on simple language extensions for concurrency; simple extensions for message passing; programming with tuples; message passing for parallel architectures; data parallel programming; mapping problems to parallel systems, and optimization of parallel programs to exploit architectural features.

The remaining half of the course is dedicated to: (a) cluster computer architecture, (b) message passing programming with MPI, and (c) development of parallel programs using MPI. The topics covered in cluster architecture include cluster building blocks, middleware and single system image. The teaching material is drawn from [6] (chapters 1, 2, 4, 26) and [25] (chapter 11). Topics covered in MPI programming include data types, process management, point-to-point and group communications, communication patterns, etc. Each of these topics is illustrated with example programs. Much public domain information on MPI is available on the web. The teaching material is drawn from books [7] (chapters 1, 2, and 3), [30] (selected sections from chapters 3, 4, 9, 10, and 12), and tutorial material [20].

As part of the laboratory experiments, we have used MPICH software [23] on our 32-node Linux cluster [21]. All 32 machines are dual-processor Pentium PCs running Linux operating systems. Among these, two nodes are acting as servers and are publicly accessible, whereas remaining client nodes located on a private fast network, which are only accessible through the servers. Students are given assignments to develop parallel programs for solving matrix manipulation, sorting, searching, data mining, and shortest path algorithms.

The second assignment focused on developing a survey report on selected topics in state-of-the-art cluster technologies such as cluster operating systems, resource management systems, cluster administration, new programming environments, genetic programming, commercial applications, and emerging cluster building blocks. For this, each student surveyed a different topic with a focus on recent advances and wrote a report. The outcome of both the laboratory experiments and the state-of-the-art-report writing experience was rated by students to be a good experience and helped in evaluating the students understanding of the course.

University of Southern California (USC)

Cluster computing has been taught as part of courses "EE557: Computer System Architecture" [14] and "EE657: Parallel Processing" [15]. EE557 focused on system architecture for parallel and distributed systems. They include SMP and CC-NUMA multiprocessors, clusters of servers and PC/workstations, and massively parallel processing (MPP) systems. Another cluster related topic covered is distributed software RAID and parallel I/O. As USC Trojans cluster research group has done very extensive research work on distributed software RAID, more design detail and benchmark experiments are taught in these courses. Students can learn very intensive techniques on how to implement single I/O space in the environment of Linux PC cluster.

Taking EE557 was a prerequisite for EE657 research-oriented course that covers scalable computers, network

security, concurrent programming, agent technology, and middleware support for cluster and Internet applications. Case studies of parallel computers and benchmark programming experiments are performed on SGI Origin 2000 superserver and on USC Trojans PC cluster. Again, based on extensive research experience on agent technology and cluster and network security, several units are devoted to explain the topics on multi-agent technologies, firewall security architecture, and e-commerce security applications. As this is a research-oriented course, only two projects are performed to finish this course. The mid-term project is a research report. Students can select one out of 20 different topics on cluster and network security. For the final project, students need to use MPI parallel programming to perform the Linpack benchmark experiments on SGI Origin 2000 and USC Trojans PC Linux cluster.

Lewis & Clark College

Cluster computing has twice been taught to upper-division undergraduate students as a 10-week summer course. The first part of the course was an introduction to cluster building blocks and MPI programming. Afterwards, most of the time was spent on building the cluster and parallelizing a ray tracer, including performance measurements and tuning. Experiments included different network technologies (including Gigabit Ethernet), different network topologies, and different file systems (including PVFS).

Students seemed to particularly enjoy hands-on components like cabling and experimenting with parallel ray-tracing programs. When the cluster was up and several students were ready to run programs, coordination (or a scheduling system) became necessary.

5. Conclusions

The variety of references cited illustrates that cluster computing ties together systems, communications, architecture, programming, applications, and algorithms. While this can make the selection of course topics difficult, the sample courses described in this paper can help instructors to design a course in cluster computing at their own institutions. Our experience with teaching cluster computing has been very favorable. The nature of cluster computing allows students to tie together material from a number of different courses in their curriculum to provide a sort of "capstone" experience in an undergraduate education, or to provide a source of thesis topics at the graduate level.

References

[1] G. Andrews, *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison-Wesley, 1999.

[2] A. Apon, CSCE 4253: Concurrent Computing, http://csce.uark.edu/~aapon/courses/concurrent/

[3] A. Apon, CSCE 4253: Concurrent Computing, http://csce.uark.edu/~aapon/courses/cluster/

[4] M. Baker, A. Apon, R. Buyya, H. Jin, "Cluster Computing and Applications", *Encyclopedia of Computer Science and Technology, Vol.45*, Marcel Dekker, Aug. 2001.

[5] D. Butenhof, *Programming with POSIX Threads*, Addison-Wesley, 1997.

[6] R. Buyya (ed.), *High Performance Cluster Computing: Systems and Architectures*, Prentice Hall, 1999.

[7] R. Buyya (ed.), *High Performance Cluster Computing: Programming and Applications*, Prentice Hall, 1999.

[8] Cluster Info Centre: http://www.buyya.com/cluster

[9] T. Dix and R. Buyya, CSC433: Parallel Systems, http://www.buyya.com/csc433/, Monash Uni., 2000.

[10] W.K. Edwards, Core Jini, *The Sun Microsystems Press Java Series*, Prentice Hall, 1999.

[11] I. Foster, *Designing and Building Parallel Programs*, Addison-Wesley, 1995

[12] S. Hartley, *Concurrent Programming, the Java Programming Language*, Oxford Press, 1998.

[13] K. Hwang and Z. Xu, *Scalable Parallel Computing*, McGraw-Hill, 1998.

[14] K. Hwang, EE557: Computr Systems Architecture, http://www-classes.usc.edu/engr/ee-s/557h/

[15] K. Hwang, EE657: Parallel Processing, http://www-classes.usc.edu/engr/ee-s/657h/

[16] IEEE Task Force on Cluster Computing (TFCC), http://www.ieeetfcc.org/.

[17] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin/Cummings, 1994.

[18] D. Lea, *Concurrent Programming in Java: Design Principles and Patterns*, Addison-Wesley, 2000.

[19] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures*, Morgan-Kaufman Press, 1992.

[20] MHPCC, *SP Parallel Programming Workshop – MPI Tutorial*, http://www.mhpcc.edu/doc/mpi/mpi.html, 2000.

[21] Monash Parallel Parametric Modeling Engine (PPME), http://hathor.csse.monash.edu.au/

[22] MOSIX – http://www.mosix.cs.huji.ac.il/.

[23] MPI Software - http://www-unix.mcs.anl.gov/mpi/mpich/

[24] P. Pacheco, *Parallel Programming With MPI*. Morgan Kaufmann Publishers, 1996.

[25] G. Pfister, *In Search of Clusters*, Prentice Hall, 1998.

[26] G. Pfister, "An Introduction to the InfiniBand Arechitecture", *High Performance Mass Storage and Parallel I/O*, IEEE Press, 2001. www.infinibandta.org

[27] S. Roosta, *Parallel Processing and Parallel Algorithms: Theory and Computation*, Springer Verlag, 2000.

[28] M. Snir, S. Otto, S. Lederman, D. Walker, J. Dongarra, *MPI: The Complete Reference*, MIT Press, 1996.

[29] D. Spector, *Building Linux Clusters*, O'Reilly, 2000.

[30] Thomas Sterling, J. Salmon, D. Becker, and D. Savarese, *How to Build a Beowulf*, MIT Press, 1999.

[31] UC Berkeley NOW project, http://now.cs.berkeley.edu/

[32] Virtual Interface Architecture, http://www.viarch.org/.

[33] B. Wilkinson and M. Allen, *Parallel Programming*, Prentice Hall, 1999.