

**Cluster Computing in the Classroom and
Integration with Computing Curricula 2001**

Amy Apon, Jens Mache, Rajkumar Buyya, and Hai Jin

Corresponding author:
Dr. Amy Apon
325 Engineering Hall
University of Arkansas
Fayetteville, AR 72701

aapon@uark.edu
Office: 479-575-6794
Dept.: 479-575-6197
Fax: 479-575-5339

Abstract - With the progress of research on cluster computing, many universities have begun to offer various courses covering cluster computing. A wide variety of content can be taught in these courses. Because of this variation, a difficulty that arises is the selection of appropriate course material. The selection is complicated because some content in cluster computing may also be covered by other courses in the undergraduate curriculum, and the background of students enrolled in cluster computing courses varies. These aspects of cluster computing make the development of good course material difficult. Combining experiences in teaching cluster computing at universities in the USA and Australia, the authors present prospective topics in cluster computing and a wide variety of information sources from which instructors can choose. The course material is described in relation to the knowledge units of the Joint IEEE Computer Society and ACM Computing Curricula 2001 and includes system architecture, parallel programming, algorithms, and applications. Instructors can select units in each of the topical areas and develop their own syllabi to meet course objectives. The authors share their experiences in teaching cluster computing and the topics chosen, depending on course objectives.

Keywords: cluster computing, Computing Curricula 2001, computer science education, parallel programming, system architecture, parallel algorithms.

1. Introduction

Clusters are built using commodity-off-the-shelf (COTS) hardware components and free or commonly used software; they 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. 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 at universities at various levels, from upper-division undergraduate to graduate levels.

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 performance computing.

Because cluster computing is so accessible to many types of colleges and universities, a course in cluster computing can be structured to fit a variety of curriculum

needs. For example, a course in cluster computing can emphasize parallel computing in a curriculum that focuses on theory and algorithm development. Or, the course can emphasize advanced network architecture in a curriculum that emphasizes architecture and computer engineering. Or, a cluster computing course could serve as a capstone project course at a liberal arts undergraduate institution. The goal of a course in cluster computing can vary according to the needs of the curriculum of a particular institution.

Many resources are available for teaching cluster computing. For example, the IEEE Computer Society Task Force on Cluster Computing (TFCC) [1] 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, a good course that covers a reasonable subset of topics of cluster computing is difficult to design. 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 these courses may come at different times 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. For most undergraduate curricula an instructor cannot assume that all the above courses are prerequisites. Otherwise, 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 task with the guidance of a brief brochure. 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. Parallel algorithms need to be developed and implemented. Writing a parallel application that executes correctly and efficiently on a cluster is a difficult task. Students may be unprepared for the challenges involved in these tasks.

To address these difficulties in teaching cluster computing, the authors present a set of sample syllabi of cluster computing for instructors to use. In this paper, the focus is on how to design a good syllabus for a cluster computing course, considering various student backgrounds, and without repeating most of the topics covered by other courses. In Section 2, possible topics of cluster computing are listed and matched with the knowledge units defined in the Joint IEEE Computer Society and ACM Computing Curricula 2001 Computer Science body of knowledge [2]. A set of sample syllabi in Section 3 are provided 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. Finally, in Section 4, the authors discuss their experiences in teaching cluster computing at several universities in the USA and Australia.

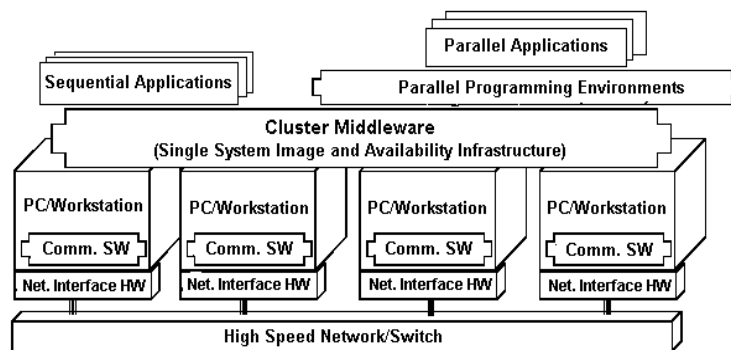


Figure 1: Typical Cluster Architecture. (Source[3])

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 [1]. A generic architecture of a cluster computer is shown in Fig. 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. In addition to describing these general cluster characteristics, topics that may be covered in a course on cluster computing include system architecture, parallel programming, and parallel algorithms and

applications[3][4].

System architecture topics include hardware components, such as the network interface, topology and link characteristics, host node architecture, and system software. Also included in system architecture are the network communication protocols used in the cluster, including TCP/IP programming and low-latency protocols, such as the protocol that executes on the InfiniBand Architecture [5]. Single processor, symmetric multiprocessor (SMP), and Cache-Coherent Non-uniform memory architecture (CC-NUMA) architectures are of interest. 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. Application-level support includes run-time system support, such as software-implemented distributed shared memory (DSM), parallel file systems, and resource management and scheduling systems. Finally, a course covering system architecture may cover case studies, such as Beowulf Clusters, IBM SP2, Digital TruCluster, and Berkeley NOW [6].

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. Parallel programming topics include shared memory programming and tools, such as POSIX and Java threads. Distributed memory programming includes message passing programming tools, such as Message Passing Interface (MPI) and Parallel Virtual Machine (PVM). Middleware programming includes tools, such as Common Object Request Broker Architecture (CORBA), Remote Procedure Call (RPC), Java Remote Method Invocation (RMI), Java Servlets, Java Database Connectivity (JDBC), and Jini.

Parallel algorithms and application topics include high performance algorithms and applications and techniques of algorithm design. In addition, performance evaluation and tuning is of interest, including optimization, visualization, high availability, network security, and benchmark experiments, such as NAS parallel benchmark (NPB) and Linpack benchmark. A comprehensive discussion of cluster-based architectures, programming, and applications can be found in [3][4][7][8][9][10][11][13][14].

Since the areas of interest in cluster computing encompass a wide variety of topics, an instructor may have difficulty selecting the most appropriate topics for inclusion in a course in cluster computing. The Computing Curricula 2001 helps to alleviate this difficulty by defining a set of knowledge units for computer science. The topics in cluster computing come from several knowledge areas. In this section a set of basic, core, and extended units are proposed for a course at the undergraduate level in cluster computing. Basic units for cluster computing include those topics that students should know before proceeding to core cluster computing topics. Core units in cluster computing include those topics that the authors consider to be fundamental to cluster computing. Finally, extended or optional units cover topics such as application areas or advanced areas of cluster computing. By defining basic, core, and extended knowledge units, an instructor is able to design a course that can be lectured at a variety of levels in a variety of institutions, from a liberal arts undergraduate college to a beginning graduate level at a research university.

Eight hours of basic units in cluster computing can be identified from the Computing Curricula 2001 [2]. These units are labeled core topics in the computer science body of knowledge in the Computing Curricula 2001. Since cluster computing is listed as an

advanced course, CS333, in the Computing Curricula 2001, the students coming into the course can be expected to have already completed a majority of the core areas of their curriculum. However, since not all courses are offered every semester at every institution, the possibility exists that students may not have completed certain topics that are basic to cluster computing. An instructor may choose to include or skip these units, depending on the preparation of the students in the course. The following knowledge units can help an instructor identify material that may overlap with other courses in the curriculum, or that should be covered in the cluster computing course if students have not studied this material previously:

- *Programming Fundamentals 2 (PF2), Algorithms and problem-solving*: Divide and conquer strategies, master/slave programming paradigm, and design strategies. *2 basic hours*
- *Programming Fundamentals 5 (PF5), Event-driven programming*: Client/server programming. *1 basic hour*
- *Architecture and Organization 4 (AR4), Memory system organization and architecture*: Cache coherence, memory consistency. *1 basic hour*
- *Architecture and Organization 7 (AR7), Multiprocessing and alternative architectures*: Switch and snoopy bus architectures for symmetric multiprocessors. *1 basic hour*
- *Operating Systems 3 (OS3), Concurrency*: Locking protocols, tools for shared memory programming, solutions to classic synchronization problems. *1 basic hour*

- *Net-Centric Computing 2 (NC2), Communication and networking:* Simple TCP/IP network architecture, socket programming. *2 basic hours*

Depending on the emphasis of the course, the core units for cluster computing compose from 20 to 29 or more hours from the Computing Curricula 2001. Of these, only six hours are labeled core topics in the Computing Curricula 2001 (only the hours in AL4, AR7, NC2, and SE2). Core units for cluster computing include:

- *Social and Professional Issues 9 (SP9), Economic issues in computing:* Moore's Law, price/performance tradeoffs with clusters. *2 core hours*
- *Architecture and Organization 7 (AR7), Multiprocessing and alternative architectures:* Advanced interconnection architectures. *2 core hours*
- *Architecture and Organization 9 (AR9), Architecture for networks and distributed systems:* The architecture of cluster interconnection networks, such as Myrinet, Scalable Coherent Interface, Gigabit Ethernet, and others. *1 to 4 core hours*
- *Net-Centric Computing 2 (NC2), Communication and networking:* The buffer layer, zero-copy messaging, remote memory operations. *1 core hour*
- *Net-Centric Computing 6 (NC6), Network management:* The setup and administration of IP networks, DNS, subnets, NIS, and NFS. *1 to 2 core hours*
- *Software Engineering 2 (SE2), Using APIs:* Basic MPI or PVM, PVFS. *2 core hours*
- *Computational Science 4 (CN4), High-performance computing:* Advanced message passing programming, parallel file access, optimization. *6 or more core hours*

- *Algorithms and Complexity 4 (AL4), Distributed algorithms:* Ring, tree, and related communication algorithms, broadcast and multicast communication. 1 core hour
- *Algorithms and Complexity 11 (AL11), Parallel algorithms:* Algorithms, such as parallel matrix multiplication, butterfly communication patterns, or others. 3 to 7 core hours
- *Operating Systems 11 (OS11), System performance evaluation:* Amdahl's Law, Gustafson's Law, performance measurement and evaluation. 1 to 2 core hours

To complete the hours for a semester course in cluster computing, any number of optional or extended topics can be included. Suggestions include the following knowledge areas.

- *Software Engineering (SE3), Software tools and environments:* Parallel debugging and environments for parallel application development.
- *Operating Systems (OS8), File systems:* Parallel file systems, advanced techniques for caching, prefetching, process migration, single-system image in clusters.
- *Algorithms (AL11), Parallel Algorithms:* Advanced parallel algorithms.
- *Architecture and Organization (AR9), Architecture for networks and distributed systems:* Interconnection architectures for cluster computing, low-latency protocols.
- *Graphics and Visualization (GV9), Visualization:* Visualization of parallel algorithms or performance of clusters.

- *Intelligent Systems 4 (IS4), Advanced search: Parallel search algorithms.*
- *Information Management (IM8, IM9, IM10, IM11), Distributed databases, physical database design, data mining, and information storage and retrieval on clusters.*
- *Computational Science (CN1, CN3), Numerical analysis, modeling and simulation.*

3. Suggested Course Components

Since the organization of the material into knowledge units according to the Computing Curricula 2001 suggests a minimal core set of about twenty hours, many options exist for filling the remaining hours to form a full-semester course. One option is to construct the course to have a specific emphasis, such as an emphasis in system architecture, or an emphasis in programming environments and languages, or an emphasis in the design of algorithms and applications. This section of the paper suggests courses with these specific emphases. The material in each emphasis area is presented in units, and instructors may select among the units to form a complete course in cluster computing. At the end of each section, the corresponding knowledge units in cluster computing, if applicable, are given in brackets.

3.1 System Architecture

A student having studied computer organization, networking, operating systems, and programming meets the prerequisites for a course on network-based, advanced computer architecture. As cluster-based systems are developed with standard, COTS

hardware and software components, an excellent course exercise is to build one's own cluster-based high-performance and/or high-availability computer system. Such a project can be coupled with the development of software that provides an illusion of a single-system image or on developing scientific and business applications.

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, the largest amount of time should be dedicated to system-level middleware.

Unit 1: Introduction

Many different computer architectures supporting high performance computing have emerged, including 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. Important questions to be addressed are, "What exactly is cluster computing? 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 [8]. [Knowledge units: AR7, AR9, SP9]

Unit 2: Cluster Building Blocks

Clusters are composed of commodity hardware and software components. 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 the InfiniBand communication fabric [5]. 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 low-latency messaging layers, such as VIA. Resources include [3][10][11][13]. [Knowledge units: AR7, AR9, OS3, NC2]

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. 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 [3][9]. A number of operating systems have proposed SSI solutions, including MOSIX [12], Unixware, and Solaris-MC. One or more such systems should be discussed since they help students to understand architecture and implementation issues. [Knowledge units: AR7, AR9, OS8]

Unit 4: Course Projects

Absorbing the entire 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, GLinux, and MOSIX. 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 the following.

- Build a low-cost cluster using PC's, Ethernet, Linux, and MPI.
- Develop tools for system administration, including job submission and management, job scheduling using various scheduling policies, and cluster monitoring.
- Implement a standard user-level communication layer based on VIA.

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 good thesis topics. [Knowledge units: NC6, Optional units]

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 usually 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. Prerequisites for a course in cluster programming include data structures, basic algorithms, computer organization, and

basic operating systems concepts. 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. This unit is appropriate if a prior operating systems course was taught from a primarily theoretical perspective or if the material is used to introduce advanced parallel programming. 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. A prerequisite for Java thread programming is prior experience in Java or another object-oriented programming language.

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 include: processes versus threads, thread libraries, classic concurrent programming problems, parallel algorithms, and applications. Suitable follow-on material for Unit 1 includes advanced parallel algorithms and hardware issues associated with symmetric multiprocessors. A follow-on coverage of symmetric multiprocessors for students with limited hardware

background is [9]. Resources include [14][15][16][17][18]. [Knowledge units: OS3, AL11]

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 systems as a part of coverage of basic units in cluster computing. This unit can include programs using the Sockets interface to Transmission Control Protocol/Internet Protocol (TCP/IP) before introducing more complicated parallel programming with distributed memory programming tools [19]. 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, TCP and UDP Sockets library, client/server programming, stream messaging versus data type messaging, and endian issues.

A project that combines Unit 1 and Unit 2 is programming a multithreaded server application and the corresponding client, using sockets and threads. A good follow-on for Unit 2 is coverage of low latency, message-passing protocols and discussion of the overhead in TCP/IP. [Knowledge units: NC2, PF5, AL4]

Unit 3: Parallel Programming Using MPI

An introduction to distributed memory programming, using a standard tool such as Message Passing Interface (MPI) [20], is crucial to cluster computing. MPI is available

for C, C++, FORTRAN, and Java. 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 after the cluster is set up, it requires little to no maintenance 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
- Algorithm development and advanced 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 [11][14][15][21][22]. [Knowledge units: AL4, AL11, SE2, CN4]

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 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. Advanced middleware products, such

as CORBA, can be taught as an entire course, often forming the basis for topics courses at the advanced undergraduate or beginning graduate level. Resources available to instructors include Jini and Javasoft [23].

3.3 Algorithms and Applications

In a course on cluster computing, students need to study algorithms and applications since high-performance applications are one building block of cluster computing [24]. Moreover, algorithms and applications provide (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 three units: overview of applications, techniques of algorithm design, and evaluation and tuning.

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. Clusters are being increasingly used for high-performance computation, and also 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 the following [24].

- *Internet applications:* Systems like Linux Virtual Server direct clients' network connection requests to multiple servers that share their workload.

- *Compression*: Systems like Evoke Communications' speech-to-email service use clusters to perform transcoding that meets real-time requirements.
- *Data mining*: Efforts like Terabyte Challenge use 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 use 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.

[Knowledge units: SP9, AL11]

Unit 2: Techniques of Algorithm Design and Specific Algorithms

Most important is showing 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, divide-and-conquer, communication, synchronization, agglomeration, mapping, load balancing, and termination detection. Specific algorithms and applications that may be covered include: sorting, numerical algorithms, image processing, graph algorithms, searching, optimization, genetic algorithms, parallel simulation, molecular modeling, climate ocean modeling, and computational fluid dynamics. Teaching resources include [14][21][25][26][27][28]. [Knowledge units: PF2, AL4, AL11, SE2, CN4, Optional Units]

Unit 3: Evaluation and Tuning

After the fundamental issues of the existence of sufficient parallelism have been addressed, there often are several algorithms or strategies available. Therefore, tradeoffs must be weighed to determine which is most appropriate. How does one choose and develop appropriate algorithms and then evaluate the resulting implementations? How does one 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: modeling, measuring, analysis, visualization, debugging, and optimization. [Knowledge units: OS11, SE3, GV9]

4. Discussions and Experience

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

University of Arkansas

Cluster computing is taught at the advanced undergraduate level with the primary objective of introducing parallel programming and problem solving [29]. Students in the course have had programming, data structures, computer organization, and operating systems.

The first part of the course covers enabling technologies for clusters, including cluster hardware elements, operating systems support for shared memory programming and basic networking concepts. Since many students have not yet studied computer

architecture, some time is spent on hardware issues, such as symmetric multiprocessing. Systems issues, such as network protocol stacks, cost of network communication, interconnection technologies for clusters, Amdahl's Law, and a comparison of clusters to symmetric multiprocessors, are discussed in this portion.

The second and largest part of the course covers distributed-memory programming using MPI. Several programming assignments are given, including at least one basic program such as a matrix/vector multiply program, and several advanced programs. MPI features, such as collective communication, communicators, efficient message passing, and advanced features of MPI-2, are covered. Along with MPI, the course also covers basic design of parallel programs and parallel I/O.

The third part of the course covers miscellaneous topics, as time allows, including high availability, single-system image, tools for cluster setup, administration, and scheduling, and performance testing. A significant portion of the course grade is based on a programming project that each student selects and which is submitted 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

The purpose of the course "CSC433: Parallel Systems" for BSc Honors degree is to build students' knowledge in advanced architecture and parallel programming. About half of the course focuses on (a) parallel systems and machine architectures, and (b) various communication models, and languages for parallel programming [30]. 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. 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.

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 assignment, 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” [31] and “EE657: Parallel Processing” [32]. EE557 focused on system architecture for parallel and distributed systems. They include SMP and CC-NUMA multiprocessors, clusters of servers and PC/workstations, and massive parallel processing (MPP) systems. Another cluster related topic covered is distributed software RAID and parallel I/O. Since 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 a single I/O space in the environment of Linux PC clusters.

Taking EE557 was a prerequisite for the 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. Because this course is research-oriented, only two projects are performed to finish this course. The mid-term project is a research report. Students can select one out of twenty 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 is a semester-long course for undergraduate students. The first part of the course is an introduction to cluster building blocks and MPI programming. Afterwards, most of the time is spent on building the cluster and parallelizing a ray tracer application, including performance measurements and tuning. Experiments include different network technologies (including Gigabit Ethernet), different network topologies, and different file systems (including PVFS). Students seem to particularly enjoy hands-on components like cabling and experimenting with parallel ray-tracing programs. When the cluster is up and several students are ready to run programs, coordination (or a scheduling system) becomes necessary.

5. Conclusions

The variety of references cited illustrates that cluster computing ties together systems, communications, architecture, programming, applications, and algorithms. While this variety 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. The authors 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] IEEE Task Force on Cluster Computing (TFCC), <http://www.ieeetfcc.org/>. Last accessed December, 2002.
- [2] Computing Curricula 2001, The Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery, December 15, 2001, <http://www.computer.org/education/cc2001/>.
- [3] R. Buyya (ed.), *High Performance Cluster Computing: Systems and Architectures*, Prentice Hall, 1999.
- [4] R. Buyya (ed.), *High Performance Cluster Computing: Programming and Applications*, Prentice Hall, 1999.
- [5] G. Pfister, "An Introduction to the InfiniBand Architecture", *High Performance Mass Storage and Parallel I/O*, IEEE Press, 2001. <http://www.infinibandta.org/>.
- [6] UC Berkeley NOW project, <http://now.cs.berkeley.edu/>. Last accessed December, 2002.
- [7] Cluster Info Centre: <http://www.buyya.com/cluster>. Last accessed December, 2002.
- [8] K. Hwang and Z. Xu, *Scalable Parallel Computing*, McGraw-Hill, 1998.
- [9] G. Pfister, *In Search of Clusters*, Prentice Hall, 1998.
- [10] D. Spector, *Building Linux Clusters*, O'Reilly, 2000.
- [11] T. Sterling, J. Salmon, D. Becker, and D. Savarese, *How to Build a Beowulf*, MIT Press, 1999.
- [12] MOSIX – <http://www.mosix.cs.huji.ac.il/>. Last accessed December, 2002.
- [13] T. Sterling (ed.), *Beowulf Cluster Computing with Linux*, MIT Press, 2002.
- [14] B. Wilkinson and M. Allen, *Parallel Programming*, Prentice Hall, 1999.

- [15] G. Andrews, *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison-Wesley, 1999.
- [16] D. Butenhof, *Programming with POSIX Threads*, Addison-Wesley, 1997.
- [17] S. Hartley, *Concurrent Programming, the Java Programming Language*, Oxford Press, 1998.
- [18] D. Lea, *Concurrent Programming in Java: Design Principles and Patterns*, Addison-Wesley, 2000.
- [19] B. Hall, *Beej's Guide to Network Programming Using Internet Sockets*, <http://www.ecst.csuchico.edu/~beej/guide/net/>, Last accessed December, 2002.
- [20] MPI Software - <http://www-unix.mcs.anl.gov/mpi/mpich/>. Last accessed December, 2002.
- [21] P. Pacheco, *Parallel Programming With MPI*. Morgan Kaufmann Publishers, 1996.
- [22] M. Snir, S. Otto, S. Lederman, D. Walker, J. Dongarra, *MPI: The Complete Reference*, MIT Press, 1996.
- [23] W.K. Edwards, Core Jini, *The Sun Microsystems Press Java Series*, Prentice Hall, 1999.
- [24] M. Baker, A. Apon, R. Buyya, H. Jin, "Cluster Computing and Applications", *Encyclopedia of Computer Science and Technology*, Vol.45, Marcel Dekker, Aug. 2001.
- [25] I. Foster, *Designing and Building Parallel Programs*, Addison-Wesley, 1995
- [26] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures*, Morgan-

Kaufman Press, 1992.

- [27] S. Roosta, *Parallel Processing and Parallel Algorithms: Theory and Computation*, Springer Verlag, 2000.
- [28] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin/Cummings, 1994.
- [29] A. Apon, CSCE 4253: Cluster Computing, <http://csce.uark.edu/~aapon/courses/cluster/>. Last accessed December, 2002.
- [30] T. Dix and R. Buyya, CSC433: Parallel Systems, <http://www.buyya.com/csc433/>, Monash University. Last accessed December, 2002.
- [31] K. Hwang, EE557: Computer Systems Architecture, <http://www-classes.usc.edu/engr/ee-s/557h/>. Last accessed December, 2002.
- [32] K. Hwang, EE657: Parallel Processing, <http://www-classes.usc.edu/engr/ee-s/657h/>. Last accessed December, 2002.