Second Forum on Parallel Computing Curricula
Held jointly with the *Symposium on Parallel Algorithms and Architectures, 97*
Newport, RI, June 23-25, 1997

Parallel Programming for the Millennium: Integration Throughout the Undergraduate Curriculum

Michael Allen, Barry Wilkinson, and James Alley
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC 28223
cma, abw, jaalley@uncc.edu

Abstract

In 1996, the University of North Carolina at Charlotte was awarded a two-year NSF grant[1] to integrate parallel programming into the Freshman Computer Science curriculum. We are pleased to report on the results of the first full year of this project. In addition, a comprehensive Senior-level course has been developed and delivered to major North Carolina universities through the statewide televideo network (NC-REN network) in a continuing cooperative venture. This cooperation has led to several new aspects of teaching parallel programming because its use of distance learning. Educational materials including web pages for instruction and audio of lectures with "automatically-turning" slides (developed by NC State University). Another new aspect for undergraduate parallel programming education is the use of guest speakers to exposing students to the state of the art. In this paper, we will describe our methods, experiences, and materials in our parallel programming activities. The course materials will form a major textbook which will be published by Prentice Hall in 1997/98, the first undergraduate course textbook concentrating upon the use of workstations in parallel computing.[2]

1 Introduction

The University of North Carolina at Charlotte (UNCC), like many institutions in the country, has offered parallel programming courses at the graduate and senior undergraduate level for several years. Tools that have been used at UNCC for these courses included transputer systems, Lester's parallel programming simulator [5], and more lately networked workstations using pvm/MPI. In 1996, a major effort was started to bring parallel programming truly into the undergraduate curriculum following the award of a grant from the National Science Foundation for this purpose. The catalyst for the development is the widespread availability of high performance networked workstations for student computing, coupled with the emergence of usable and widely available parallel programming tools for these networked workstations. The first really successful software tool was pvm (parallel virtual machine) [1][2], available on a wide range of platforms including version for PCs. More recently, a standard for message-passing has been established, MPI (Message Passing Interface) [3][7][8]. This standard reinforces the acceptance of message-passing in parallel computing.

In this paper, we will describe our two major thrusts; in the Freshman year and in the senior year. In both cases, web-based materials were used for delivery of coursework and program compiling instructions.

2 Computing Platform

We have large numbers of students passing through our various parallel programming courses (several hundred/year), and a suitable computing platform was essential. Our first attempt was to use the small departmental cluster of workstations (SUNs, SGIs, and Pentiums) over which we have total control over. Initially pvm/xpvm was selected. Having a heterogeneous network was particularly attractive as then the effects of different types of computers working collectively on a problem could be illustrated. This was fine for the small pilot courses in Spring and Summer 1996 but it soon became evident that this cluster would be inadequate for large numbers of students. The main problem was that the network is not dedicated to parallel programming and indeed the Pentiums could be switched between Windows NT and UNIX without the knowledge of remote users.

The much larger College of Engineering network consists of 400+ workstations, mainly SUNs and Pentiums. This network does not have remote login/rsh/rexec privileges because of administrative decision. This prevents multiple general-purpose workstations being enrolled in PVM/MPI. To overcome this problem, a dedicated set of SUN boxes has been assembled as a 'PVM/MPI cluster' in a closet. This cluster can only be used by remote access. We have used College of Engineering surplus workstations ini-

1. Research supported in part by a grant from the National Science Foundation, NSF DUE 9554975.
2. *Parallel Programming: Techniques and Applications Using Networked Workstations*, Barry Wilkinson and Michael Allen, Prentice Hall Inc, to be published in December, 1997.

tially (SUNs). We will be adding to this cluster those workstations that become surplus (i.e. last year's models) and so this cluster is expected to grow! Currently the cluster uses a single Ethernet. Research is ongoing in the department to develop a unique interconnection structure for this cluster which takes into account its purpose for parallel programming only.

3 Web-based Materials

Home pages have been created for all the parallel programming courses. For the teleclass especially, it became particularly useful for providing assignments, additional materials and programs that the students could download remotely. However in general providing detailed instructions on how to use the parallel programming tools is very useful. A couple of snapshot of pages from www.cs.uncc.edu/par_prog are shown in Figure 1 and Figure 2. Figure 1 shows the links to the Freshman courses, CSCI 1201 and CSCI 1202 and the teleclass. Figure 2 shows a picture of part of the original departmental network, and the index to the instructions for running pvm programs. A similar page exists for MPI.

4 Freshman Parallel Programming

Our Freshman programming courses are a two-semester sequence consisting of two 2-credit hour lecture courses and two 1-credit hour labs. Currently in the labs, the underlying language in the first semester is C and in the second semester C++. The lecture content is modelled after the CS-1 and CS-2 courses of the ACM. However, we have added some material on parallel programming in what appears to be a unique and innovative way. Our approach relies on our extensive network of Sun workstations and high-end PCs to deliver a distributed workstation parallel computational environment.

4.1 First Semester

Our CS-1 course, (CSCI 1201[lecture] and CSCI 1201L [lab]) is typically taken by our majors in their first semester at UNCC. The lecture portion covers overviews of von Neumann architecture and the compile/link/load/execute process, algorithm development, various searching/sorting algorithms, and basic data structures including arrays, stacks, queues, and linked lists. The lab introduces the basic data types and control structures through programming

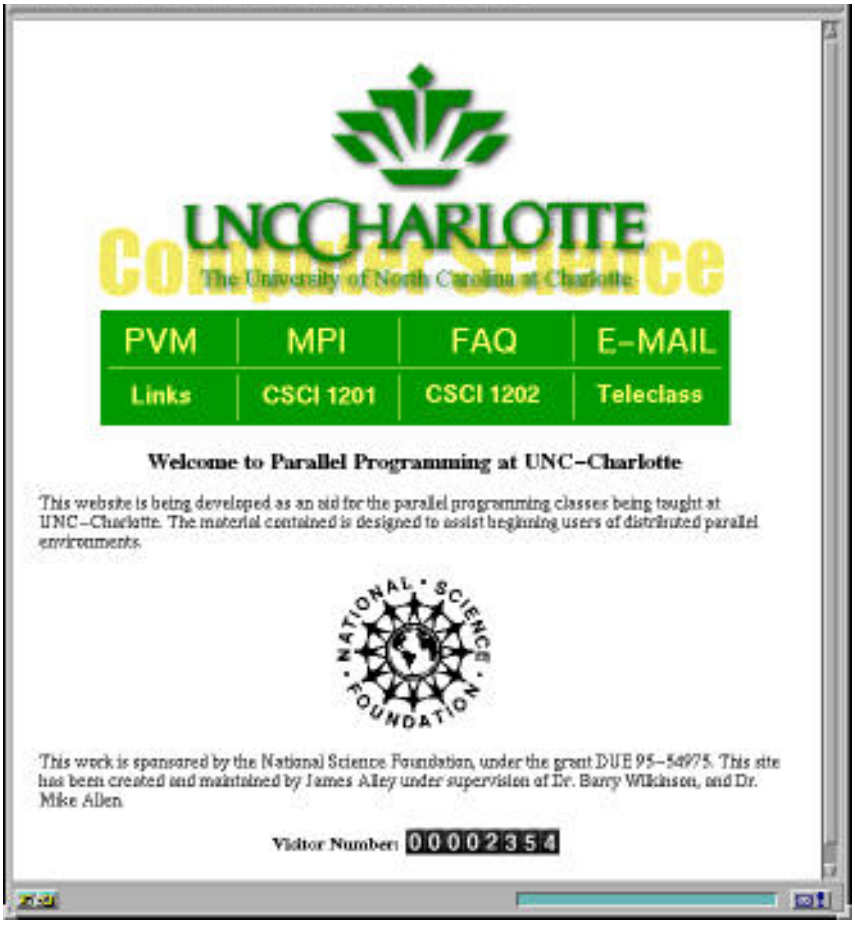

Figure 1 Front page of www.cs.uncc.edu/par_prog

assignments in C during the first part of the semester. Pointers, linked lists, and structures are introduces through the programming assignments in the latter part of the semester. Parallel computing is introduced at roughly the mid-term point through a lab assignment.

The parallel programming objective of that first semester lab assignment is purely to familiarize the students with the parallel programming environment at UNCC by acquainting them with the process of editing, compiling, and running parallel programs. Midway through the first semester lab they are given a multi-part parallel programming assignment. The first part consists of reviewing web pages of material on PVM and MPI. They are expected to modify their environment (by copying down various configuration files from the web) as part of this review. They are then asked to run a demo program on a multi-workstation platform and explain what it does and why there is a speed-up when multiple processors are introduced into the solution. The objective is to familiarize them with the parallel programming environment available to them and simply orients them to the fact that under certain circumstances it is possible to reduce the computation time by throwing multiple processors at the solution of a problem.

Nothing is covered in the lecture about the theory of parallel computing in that first semester. The lab portion is strictly a “gee whiz, isn’t this a neat way to speed things up” approach. The fact that the first semester lab uses C as the programming language, of course ties in nicely with our PVM or MPI tools. By the midpoint of the first semester lab, the students are fully capable of reading and understanding the demo program with minimal TA support. This parallel computing assignment also gives them an introduction to a topic that will prove invaluable in later courses: makefiles.

4.2 Second Semester

The second semester lecture builds on that first semester lab experience. Topics in the second semester lecture course (CSCI 1202 at UNCC) include more in-depth work with data structures (arrays, stacks, queues, linked lists, trees, graphs) and algorithms incorporating them. For example, we look at time and space complexity of algorithms as well as searching, traversal, and optimization for the first time. The lab portion, (CSCI 1202L at UNCC), introduces them to C++ while reinforcing their first semester C skills.

i Two new innovations have been incorporated, however, in our discussion of algorithms as a result of integrating

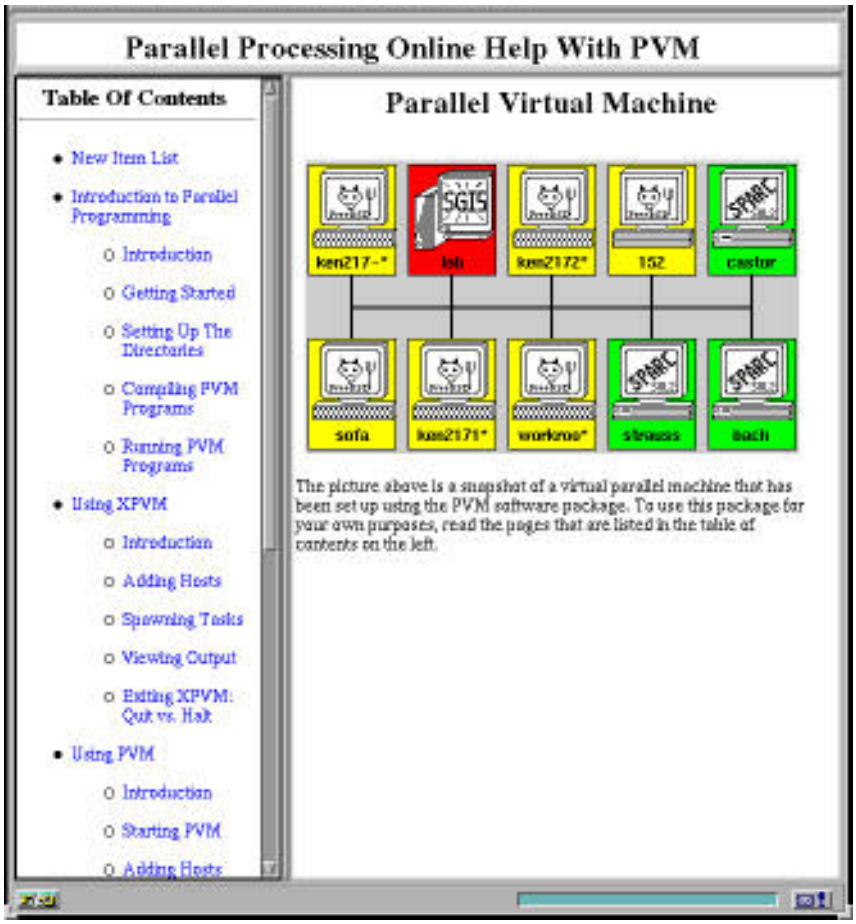

Figure 2 Front page of www.cs.uncc.edu/par_prog/pvm_index.html

parallel computing theory into the second semester lecture. First, when we discuss algorithms we do it from two perspectives ... "How would you approach this problem using sequential computation?", and "How would you approach this problem if you had a large number of workstations you could devote to assisting you?". Second, we ask them to develop, code, and run a parallel solution to a simple "divide and conquer" type problem. At this point they are fairly comfortable with C (they have a semester of it under their belts) and have at least a superficial exposure (from the first semester lab) to PVM and MPI. As a result, most students are able to complete the assignment without excessive difficulty.

In the second semester of the lecture course, an effort is made to bring out the factors limiting performance improvement when multiple workstations are thrown at a problem, However, the in-depth course devoted strictly to parallel computation is a Junior/Senior level elective. Between the Freshman year when they are introduced to the concepts of parallel computation and that Junior/Senior level elective we are in the process of working additional theory of parallel programming into our Analysis of Algorithms course as well. When this integration of material is complete, all of our majors will have had a basic introduction to parallel algorithms and programming spread, in pieces, over three semesters while those taking the Junior/Senior elective also will have extensive depth.

5 Senior Parallel Programming Teleclass Course

Concurrently with the Freshman development, a comprehensive parallel programming course has been developed for senior undergraduates and delivered to major North Carolina universities through the statewide televideo network (NC-REN network) in a cooperative venture. Institutions involved in the 1996 offering were NC State University, UNC-Asheville and UNC-Greensboro, in addition to UNCC. The next offering of this course in Fall 1997 will include Duke University. Guest speakers are also involved from Duke University and NC State University. About 37 seniors enrolled in this first offering with UNC-Asheville closing their section at 12 students.

Because this is a "teleclass" broadcast to other universities, faculty were actively involved at each site. They could answer questions and provide other materials specific to their site. Finally all lectures were recorded so that students could review any lecture, or catch up with anything they missed. In a related activity at NC State University, the sound of complete lectures, with "automatically-turning" slides, have been integrated on a home page. This technology may be the future of distance learning at home.

Students do their assignments either on their local computer network, or by remotely accessing UNCC's cluster (or NC State's ATM cluster). Fortunately each site had, or could very quickly establish a common environment so that remote access proved unnecessary. Instructional materials were given to each student in the form of 185+ pages of typed lecture notes, so that students did not need to take notes in class and could concentrate upon the lectures. The notes themselves were developed in preparation for publication as a textbook by Prentice Hall in December 1997. (The final book will be about 450 pages.) The organization of the material is purposely divided into two parts, one part describing the basic techniques of parallel programming, and one part describe applications and specialized algorithms. The first part requires no specialized mathematical knowledge and could be used at lower levels in the curriculum, specifically as a supplement to CS-1, CS-2 and sophomore algorithms courses. The second part does have some mathematical prerequisites (linear equations, partial different equations, matrices, etc.,) but nothing beyond that expected of seniors.

The topics of these notes is given below:

PART I Basic Techniques

PART II Algorithms and Applications

An important aspect of this material is that it is language and system-independent yet targeted towards message-passing on networked workstations. All example code is in a pseudo-code such that students can very easily adapt it to MPI or PVM, the two systems currently used at UNCC. Both these tool, including their differences, are described quite early in the course, but it is left to students to learn the details of the library calls through writing programs. (All our parallel programming courses are hands-on "programming" courses; students write programs.)

The first assignment is in fact the same as used for the Freshman; a familiarization assignment in which a working program is provided. Students set up their system environment, compile the program and obtain results after reading the instructions on the home page. The program simply adds numbers together in parallel. At this stage only one workstation is used, but multiple workstations are used in

subsequent assignments. The students also have to modify the program to find the maximum value. This assignment only requires a few hours to do. The concepts of creating makefiles and other compiling matters are of course very familiar to most senior students (which is not the case with the Freshman).

The second assignment, which is still quite simple, illustrates an embarrassingly parallel program, the Mandelbrot computation. For this assignment, X-window code for generating graphical output is provided for downloading – the course does not assume any prior knowledge of graphics. The graphics code is in fact useful for the subsequent assignments as well.

The next assignments are increasingly difficult. The third assignment involves solving Laplace's equation to obtain the heat distribution in a room which has a fireplace. Graphical output is required in the form of temperature contours. Synchronous iteration is used. Another assignment involves solving the same problem by direct means (Gaussian elimination). It is useful to show different methods to solve the same problem and the speed implications of the different methods. Load balancing is a key aspect of all the assignments, and timing information must be provided, usually by instrumenting the code with the time() system call. All assignments have "open-endedness" in that extra credit can be obtained by additional work.

6 Expert Guest Speakers

Part of the NSF project is to introduce expert guest speakers who would talk about some practical aspect of parallel programming. In Fall 1996, we were fortunate to have two such presentations. (Only one in each semester was originally planned.) Professor John Board of Duke University gave a presentation entitled "Networks of Workstations: The Plodding Workhorses of Parallel Computing" and also outlined modeling DNA. Professor Mladen Vouk of NC State University gave a presentation on the current state of supercomputers and supercomputing conferences. In both cases, the teleclass NC-REN facility was used. The response to these presentations was very positive and gave the material in the courses a greater significance. We intend to continue using expert guest speakers. We also arranged for two graduate students to make presentations of their parallel programming project dealing with parallel genetic algorithms.

7 Conclusions

In this paper we have described our work in bringing parallel programming into the Freshman year and a Senior parallel programming class to various NC universities. Freshman in particular have responded very enthusiastically when it was explained that multiple-processor parallel computation is the future of their field. They realize that they are getting a "leg up" on their competition through this early introduction to the area.

The Senior class incorporated several new aspects partly because of its use of a televideo network. First, it was necessary for remote sites to have an adequate parallel computing platform. Networked workstations are idea since nearly every university has them. Second, very significant preparatory materials were necessary in the form of notes and web materials. Finally the use of a televideo facility allowed experts from different sites to participate in the course and give presentations to undergraduates. We will be offering the teleclass on a continuing basis with faculty from different universities cooperating on lecture materials.

Acknowledgments

It is a great pleasure to acknowledge Dr. M Mulder, program director at the National Science Foundation for supporting our project. We should like to thank the many students at UNCC who help us refine the material over the last few years, especially the talented "teleclass" of Fall 1996 when the material was finally classroom-tested. We owe a debt of gratitude to many people. Professor Lang of UNC-Asheville truly contributed to the course development in the classroom and Professor Vouk of NC State University, apart from presenting a expert guest lecture for us, set up an impressive web page which included "real audio" of the lectures and "automatically-turning" slides. Professor John Board of Duke University also kindly made an expert guest presentation to the class – all these activities helped us in developing our materials.

References

[1] Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam (1994), *PVM3 User's Guide and Reference Manual*, Oak Ridge National Laboratory: Tennessee.

[2] Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam (1994), *PVM: Parallel Virtual Machine*, The MIT Press: Cambridge, Massachusetts.

[3] Gropp, W., E. Lusk, and A. Skjellum (1994), *Using MPI Portable Parallel Programming with the Message-Passing Interface*, The MIT Press: Cambridge, Massachusetts.

[4] Kumar, V., A. Grama, A. Gupta, and G. Karypis (1994), *Introduction to Parallel Computing*, Benjamin/Cummings Publishing Company Inc.: Redwood City California.

[5] Lester, B. (1993), *The Art of Parallel Programming*, Prentice Hall: Englewood Cliffs, New Jersey.

[6] Nevison, C. H. (1995), "Parallel computing in the

undergraduate curriculum", *IEEE Computer*, **28**, no. 12, 51–6.

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

[8] Snir, M., S. W. Otto, S. Huss-Lederman, D. W. Walker and J. Dongarra (1996), *MPI The Complete Reference*, The MIT Press: Cambridge Massachusetts.

Parallel programming Python Performance optimization GPU computing PDC Undergraduate education. This is a preview of subscription content, log in to check access. Notes. Acknowledgements. The authors would like to thank Professor Nicolae TÄƒpuÅŸ, Alexandru HeriÅŸanu, RÄƒzvan Dobre, Vlad SpoialÄƒ, Dan Dragomir, Alexandru Olteanu, and VoichiÅ£a Iancu for their valuable contributions to the CSA and PPA curriculum.Â Cite this paper as: CarabaÅŸ M., DrÄƒghici A., Lupescu G., SamoilÄƒ CG., SluÅŸanschi EI. (2019) Integrating Parallel Computing in the Curriculum of the University Politehnica of Bucharest. In: Mencagli G. et al. (eds) Euro-Par 2018: Parallel Processing Workshops. Euro-Par 2018. An integrated curriculum allows children to pursue learning in a holistic way, without the restrictions often imposed by subject boundaries. In early childhood programs it focuses upon the inter-relatedness of all curricular areas in helping children acquire basic learning tools. It recognizes that the curriculum for the primary grades includes reading, writing, listening, speaking, literature, drama, social studies, math, science, health, physical education, music, and visual arts. The curriculum also incorporates investigative processes and technology. It emphasizes the importance of maintai M. Allen, B. Wilkinson, and J. Alley, "Parallel Programming for the Millennium: Integration Throughout the Undergraduate Curriculum," 2nd Forum on Parallel Computing Curricula, June 22nd, 1997. Paper2. B. Wilkinson and M. Allen, "A State-Wide Senior Parallel Programming Course," IEEE Transactions on Education, Vol. 42, no. 3 (August), 1999, pp. 167-173. : Instructor's Manual ISBN 0-13-085041-1Now available from publishers or Authors (abw@uncc.edu) - only to Instructors. Curriculum Guidelines for Undergraduate Programs in Data Science. Park City Math Institute (PCMI) Undergraduate Faculty Group. â‡¤.Â The Park City Math Institute (PCMI) 2016 Summer Undergraduate Faculty Program met for the purpose of composing guidelines for un-dergraduate programs in Data Science. The group consisted of 25 un-dergraduate faculty from a variety of institutions in the U.S., primarily from the disciplines of mathematics, statistics and computer science.Â problem-solving skills recur throughout the workow of the data scientist. As Jean-nette Wing put it, â€œThinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstractionâ€ (Wing). Only by integrating project-based learning experiences throughout the undergraduate curriculum will we give students the opportunity to develop a mastery of the fundamentals of science, engineering and mathematics along with providing them with the contextual environment for developing the skills necessary to practice engineering such as project management, teamwork and effective communication.Â A project plan is usually developed to guide students through the process, support teamwork, focus communication and evaluate if the economic objectives of the project are being achieved. Throughout this process the students are challenged to learn how to work in teams and to practice systems level thinking when integrating technologies.