PARALLEL PROGRAMMING

TECHNIQUES AND APPLICATIONS USING NETWORKED WORKSTATIONS AND PARALLEL COMPUTERS

BARRY WILKINSON
University of North Carolina - Charlotte

MICHAEL ALLEN
University of North Carolina - Charlotte

An Alan R. Apt Book

PRENTICE HALL, Upper Saddle River, New Jersey 07458

Preface

The purpose of this text is to introduce parallel programming techniques. Parallel programming uses multiple computers, or computers with multiple internal processors, to solve a problem at a greater computational speed than using a single computer. It also offers the opportunity to tackle larger problems; that is, problems with more computational steps or more memory requirements, the latter because multiple computers and multiprocessor systems often have more total memory than a single computer. In this text, we concentrate upon the use of multiple computers that communicate between themselves by sending messages; hence the term ***message-passing*** parallel programming. The computers we use can be different types (PC, SUN, SGI, etc.) but must be interconnected by a network, and a software environment must be present for intercomputer message passing. Suitable networked computers are very widely available as the basic computing platform for students so that acquisition of specially designed multiprocessor systems can usually be avoided. Several software tools are available for message-passing parallel programming, including PVM and several implementations of MPI, which are all freely available. Such software can also be used on specially designed multiprocessor systems should these systems be available for use. So far as practicable, we discuss techniques and applications in a system-independent fashion.

The text is divided into two parts, Part I and Part II. In Part I, the basic techniques of parallel programming are developed. The chapters of Part I cover all the essential aspects, using simple problems to demonstrate techniques. The techniques themselves, however, can be applied to a wide range of problems. Sample code is given usually first as sequential code and then as realistic parallel ***pseudocode.*** Often, the underlying algorithm is already parallel in nature and the sequential version has "unnaturally" serialized it using loops. Of course, some algorithms have to be reformulated for efficient parallel solution, and this reformulation may not be immediately apparent. One chapter in Part I introduces a type of

parallel programming not centered around message-passing multicomputers, but around specially designed ***shared memory*** multiprocessor systems. This chapter describes the use of Pthreads, an IEEE multiprocessor standard system that is widely available and can be used on a single computer.

The prerequisites for studying Part I are knowledge of sequential programming, such as from using the C language and associated data structures. Part I can be studied immediately after basic sequential programming has been mastered. Many assignments here can be attempted without specialized mathematical knowledge. If MPI or PVM is used for the assignments, programs are written in C with message-passing library calls. The descriptions of the specific library calls needed are given in the appendices.

Many parallel computing problems have specially developed algorithms, and in Part II problem-specific algorithms are studied in both non-numeric and numeric domains. For Part II, some mathematical concepts are needed such as matrices. Topics covered in Part II include sorting, matrix multiplication, linear equations, partial differential equations, image processing, and searching and optimization. Image processing is particularly suitable for parallelization and is included as an interesting application with significant potential for projects. The fast Fourier transform is discussed in the context of image processing. This important transform is also used in many other areas, including signal processing and voice recognition.

A large selection of "real-life" problems drawn from practical situations is presented at the end of each chapter. These problems require no specialized mathematical knowledge and are a unique aspect of this text. They develop skills in using parallel programming techniques rather than simply learning to solve specific problems such as sorting numbers or multiplying matrices.

Topics in Part I are suitable as additions to normal sequential programming classes. At the University of North Carolina at Charlotte (UNCC), we introduce our freshmen students to parallel programming in this way. In that context, the text is a supplement to a sequential programming course text. The sequential programming language is assumed to be C or C++. Part I and Part II together is suitable as a more advanced undergraduate parallel programming/computing course, and at UNCC we use the text in that manner.

Full details of the UNCC environment and site-specific details can be found at http://www.cs.uncc.edu/par_prog. Included at this site are extensive Web pages to help students learn how to compile and run parallel programs. Sample programs are provided. An Instructor's Manual is also available to instructors. Our work on teaching parallel programming is connected to that done by the Regional Training Center for Parallel Processing at North Carolina State University. Additional information about this center can be found at http://renoir.csc.ncsu.edu/RTCPP.

The text is a direct outcome of a National Science Foundation grant awarded to the authors at the University of North Carolina at Charlotte to introduce parallel programming in the freshman year.' It is a great pleasure to acknowledge Dr. M. Mulder, program director at the National Science Foundation, for supporting our project. Without his support, we would not be able to pursue the ideas presented in this text. We also wish to thank the graduate students that worked on this project, J. Alley, M. Antonious, M. Buchanan, and G.

'National Science Foundation grant "Introducing parallel programming techniques into the freshman curricula," ref. no. DUE 9554975.

Robins, and undergraduate students G. Feygin, W. Hasty, C. Beauregard, M. Moore, D. Lowery, K. Patel, Johns Cherian, and especially Uday Kamath. This team helped develop the material and assignments with us. We should like to record our thanks to James Robinson, the departmental system administrator who established our local workstation cluster, without which we would not have been able to conduct the work.

We should also like to thank the many students at UNCC who help us refine the material over the last few years, especially the "teleclasses," in which the materials were classroom tested in a unique setting. These teleclasses are broadcast to several North Carolina universities, including UNC-Asheville, UNC-Greensboro, UNC-Wilmington, and North Carolina State University, in addition to UNCC. We owe a debt of gratitude to many people, among which Professor Wayne Lang at UNC-Asheville and Professor Mladen Vouk of NC State University deserve special mention. Professor Lang truly contributed to the course development in the classroom and Professor Vouk, apart from presenting an expert guest lecture for us, set up an impressive Web page that included "real audio" of our lectures and "automatically turning" slides. (These lectures can be viewed at http:renoir.csc.ncsu.edu/CSC495A.) Professor John Board of Duke University and Professor Jan Prins of UNC Chapel Hill also kindly made expert guest presentations to classes. A parallel programming course based upon the material in this text was also given at the Universidad Nacional de San Luis in Argentina by kind invitation from Professor Raul Gallard — all these activities helped us in developing this text.

We would like to express our appreciation to Alan Apt and Laura Steele of Prentice Hall, who received our proposal for a textbook and supported us throughout its development. Reviewers provided us with very helpful advice.

Finally, may we ask that you please send comments and corrections to us at abw@uncc.edu (Barry Wilkinson) or cma@uncc.edu (Michael Allen).

Barry Wilkinson
Michael Allen
University of North Carolina
Charlotte

Contents