

5500

681.2:
519.682
W 65A

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

Preface	v
About the Authors	viii
PART I BASIC TECHNIQUES	1
CHAPTER 1 PARALLEL COMPUTERS	3
1.1 The Demand for Computational Speed	3
1.2 Types of Parallel Computers	6
<i>Shared Memory Multiprocessor System,</i>	<i>6</i>
<i>Message-Passing Multicomputer</i>	<i>7</i>
<i>Distributed Shared Memory,</i>	<i>9</i>
<i>MIMD and SIMD Classifications,</i>	<i>10</i>
1.3 Architectural Features of Message-Passing Multicomputers	11
<i>Static Network Message-Passing Multicomputers,</i>	<i>11</i>
<i>Embedding,</i>	<i>16</i>
<i>Communication Methods,</i>	<i>18</i>
Input/Output,	22
1.4 Networked Computers as a Multicomputer Platform	22
1.5 Potential for Increased Computational Speed	26
1.6 Summary	32
Further Reading	32
Bibliography	33
Problems	36

- 2.1 Basics of Message-Passing Programming 38
 - Programming Options, 38*
 - Process Creation, 39*
 - Message-Passing Routines, 41*
- 2.2 Using Workstation Clusters 46
 - Software Tools, 46*
 - PVM, 47*
 - MPI, 51*
 - Pseudocode Constructs, 59*
- 2.3 Evaluating Parallel Programs 61
 - Parallel Execution Time, 61*
 - Time Complexity, 64*
 - Comments on Asymptotic Analysis, 67*
 - Time Complexity of Broadcast/Gather, 68*
- 2.4 Debugging and Evaluating Parallel Programs 71
 - Low-Level Debugging, 71*
 - Visualization Tools, 72*
 - Debugging Strategies, 73*
 - Evaluating Programs Empirically, 74*
 - Comments on Optimizing the Parallel Code, 76*
- 2.5 Summary 77
 - Further Reading 77
 - Bibliography 78
 - Problems 80

- 3.1 Ideal Parallel Computation 82
- 3.2 Embarrassingly Parallel Examples 84
 - Geometrical Transformations of Images, 84*
 - Mandelbrot Set, 89*
 - Monte Carlo Methods, 95*
- 3.3 Summary 100
 - Further Reading 100
 - Bibliography 101
 - Problems 102

CHAPTER 4 PARTITIONING AND DIVIDE-AND-CONQUER STRATEGIES

107

- 4.1 Partitioning 107
 - Partitioning Strategies, 107*
 - Divide and Conquer, 111*
 - M-ary Divide and Conquer, 117*
- 4.2 Divide-and-Conquer Examples 118
 - Sorting Using Bucket Sort, 118*
 - Numerical Integration, 122*
 - N-Body Problem, 126*
- 4.3 Summary 131
 - Further Reading 131
 - Bibliography 132
 - Problems 133

CHAPTER 5 PIPELINED COMPUTATIONS

139

- 5.1 Pipeline Technique 139
- 5.2 Computing Platform for Pipelined Applications 143
- 5.3 Pipeline Program Examples 144
 - Adding Numbers, 144*
 - Sorting Numbers, 147*
 - Prime Number Generation, 151*
 - Solving a System of Linear Equations-Special Case, 153*
- 5.4 Summary 156
 - Further Reading 157
 - Bibliography 157
 - Problems 158

CHAPTER 6 SYNCHRONOUS COMPUTATIONS

162

- 6.1 Synchronization 162
 - Barrier, 162*
 - Counter Implementation, 164*
 - Tree Implementation, 166*
 - Butterfly Barrier, 166*
 - Local Synchronization, 168*
 - Deadlock, 168*

6.2	Synchronized Computations 169
	<i>Data Parallel Computations, 169</i>
	<i>Synchronous Iteration, 172</i>
6.3	Synchronous Iteration Program Examples 173
	<i>Solving a System of Linear Equations by Iteration, 173</i>
	<i>Heat Distribution Problem, 179</i>
	<i>Cellular Automata, 188</i>
6.4	Summary 189
	Further Reading 190
	Bibliography 190
	Problems 191

CHAPTER 7 LOAD BALANCING AND TERMINATION DETECTION

198

7.1	Load Balancing 198
7.2	Dynamic Load Balancing 200
	<i>Centralized Dynamic Load Balancing, 201</i>
	<i>Decentralized Dynamic Load Balancing, 202</i>
	<i>Load Balancing Using a Line Structure, 204</i>
7.3	Distributed Termination Detection Algorithms 207
	<i>Termination Conditions, 207</i>
	<i>Using Acknowledgment Messages, 208</i>
	<i>Ring Termination Algorithms, 209</i>
	<i>Fixed Energy Distributed Termination Algorithm, 211</i>
7.4	Program Example 211
	<i>Shortest Path Problem, 211</i>
	<i>Graph Representation, 212</i>
	<i>Searching a Graph, 214</i>
7.5	Summary 220
	Further Reading 220
	Bibliography 221
	Problems 222

CHAPTER 8 PROGRAMMING WITH SHARED MEMORY

227

8.1	Shared Memory Multiprocessors 227
8.2	Constructs for Specifying Parallelism 230
	<i>Creating Concurrent Processes, 230</i>
	<i>Threads, 231</i>

8.3	Sharing Data 236
	<i>Creating Shared Data, 236</i>
	<i>Accessing Shared Data, 236</i>
	<i>Language Constructs for Parallelism, 244</i>
	<i>Dependency Analysis, 245</i>
	<i>Shared Data in Systems with Caches, 248</i>
8.4	Program Examples 250
	<i>UNIX Processes, 251</i>
	<i>Pthreads Example, 254</i>
	<i>Java Example, 256</i>
8.5	Summary 257
	Further Reading 258
	Bibliography 258
	Problems 259

PART II ALGORITHMS AND APPLICATIONS 265

CHAPTER 9 SORTING ALGORITHMS 267

9.1	General 267
	<i>Sorting, 267</i>
	<i>Potential Speedup, 268</i>
	<i>Rank Sort, 268</i>
9.2	Compare-and-Exchange Sorting Algorithms 270
	<i>Compare and Exchange, 270</i>
	<i>Bubble Sort and Odd-Even Transposition Sort, 273</i>
	<i>Two-Dimensional Sorting, 277</i>
	<i>Mergesort, 280</i>
	<i>Quicksort, 282</i>
	<i>Quicksort on a Hypercube, 284</i>
	<i>Odd-Even Mergesort, 290</i>
	<i>Bitonic Mergesort, 291</i>
9.3	Summary 295
	Further Reading 295
	Bibliography 296
	Problems 297

CHAPTER 10 NUMERICAL ALGORITHMS 301

10.1	Matrices — A Review 301
	<i>Matrix Addition, 301</i>

	<i>Matrix Multiplication, 302</i>	
	<i>Matrix-Vector Multiplication, 302</i>	
	<i>Relationship of Matrices to Linear Equations, 303</i>	
10.2	Implementing Matrix Multiplication	303
	<i>Algorithm, 303</i>	
	<i>Direct Implementation, 304</i>	
	<i>Recursive Implementation, 307</i>	
	<i>Mesh Implementation, 309</i>	
	<i>Other Matrix Multiplication Methods, 313</i>	
10.3	Solving a System of Linear Equations	313
	<i>Linear Equations, 313</i>	
	<i>Gaussian Elimination, 314</i>	
	<i>Parallel Implementation, 315</i>	
10.4	Iterative Methods	317
	<i>Jacobi Iteration, 318</i>	
	<i>Faster Convergence Methods, 321</i>	
10.5	Summary	326
	Further Reading	326
	Bibliography	326
	Problems	327

CHAPTER 11 IMAGE PROCESSING

331

11.1	Low-Level Image Processing	331
11.2	Point Processing	333
11.3	Histogram	334
11.4	Smoothing, Sharpening, and Noise Reduction	335
	<i>Mean, 335</i>	
	<i>Median, 336</i>	
	<i>Weighted Masks, 338</i>	
11.5	Edge Detection	340
	<i>Gradient and Magnitude, 340</i>	
	<i>Edge Detection Masks, 341</i>	
11.6	The Hough Transform	344
11.7	Transformation into the Frequency Domain	348
	<i>Fourier Series, 348</i>	
	<i>Fourier Transform, 349</i>	
	<i>Fourier Transforms in Image Processing, 350</i>	
	<i>Parallelizing the Discrete Fourier Transform Algorithm, 352</i>	
	<i>Fast Fourier Transform, 356</i>	

- 11.8 Summary 361
 - Further Reading 362
 - Bibliography 362
 - Problems 364

CHAPTER 12 SEARCHING AND OPTIMIZATION

367

- 12.1 Applications and Techniques 367
- 12.2 Branch-and-Bound Search 368
 - Sequential Branch and Bound, 368*
 - Parallel Branch and Bound, 370*
- 12.3 Genetic Algorithms 372
 - Evolution and Genetic Algorithms, 372*
 - Sequential Genetic Algorithms, 374*
 - Initial Population, 374*
 - Selection Process, 376*
 - Offspring Production, 377*
 - Variations, 379*
 - Termination Conditions, 379*
 - Parallel Genetic Algorithms, 380*
- 12.4 Successive Refinement 384
- 12.5 Hill Climbing 385
 - Banking Application, 386*
 - Hill Climbing in a Banking Application, 388*
 - Parallelization, 389*
- 12.6 Summary 389
 - Further Reading 389
 - Bibliography 390
 - Problems 39 1

APPENDIX A BASIC PVM ROUTINES 398

APPENDIX B BASIC MPI ROUTINES 404

APPENDIX C BASIC PTHREAD ROUTINES 410

APPENDIX D PARALLEL COMPUTATION MODELS 415

INDEX 426