Parallel MATLAB

for Multicore and Multinode Computers

SOFTWARE • ENVIRONMENTS • TOOLS

The SIAM series on Software, Environments, and Tools focuses on the practical implementation of computational methods and the high performance aspects of scientific computation by emphasizing in-demand software, computing environments, and tools for computing. Software technology development issues such as current status, applications and algorithms, mathematical software, software tools, languages and compilers, computing environments, and visualization are presented.

Software, Environments, and Tools

Jeremy Kepner, *Parallel MATLAB for Multicore and Multinode Computers*

Michael A. Heroux, Padma Raghavan, and Horst D. Simon, editors, *Parallel Processing for Scientific Computing*

Gérard Meurant, *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*

Bo Einarsson, editor, *Accuracy and Reliability in Scientific Computing*

Michael W. Berry and Murray Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval, Second Edition*

Craig C. Douglas, Gundolf Haase, and Ulrich Langer, *A Tutorial on Elliptic PDE Solvers and Their Parallelization*

Louis Komzsik, *The Lanczos Method: Evolution and Application*

Bard Ermentrout, *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*

V. A. Barker, L. S. Blackford, J. Dongarra, J. Du Croz, S. Hammarling, M. Marinova, J. Waśniewski, and P. Yalamov, *LAPACK95 Users' Guide*

Stefan Goedecker and Adolfy Hoisie, *Performance Optimization of Numerically Intensive Codes*

Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*

Lloyd N. Trefethen, *Spectral Methods in MATLAB*

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide, Third Edition*

Michael W. Berry and Murray Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*

Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst, *Numerical Linear Algebra for High-Performance Computers*

R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*

Randolph E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 8.0*

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*

Greg Astfalk, editor, *Applications on Advanced Architecture Computers*

Roger W. Hockney, *The Science of Computer Benchmarking*

Françoise Chaitin-Chatelin and Valérie Frayssé, *Lectures on Finite Precision Computations*

Parallel MATLAB

for Multicore and Multinode Computers

Jeremy Kepner
Massachusetts Institute of Technology
Lincoln Laboratory
Lexington, Massachusetts

siam®
Society for Industrial and Applied Mathematics
Philadelphia

10 9 8 7 6 5 4 3 2 1

This work is sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Library of Congress Cataloging-in-Publication Data

Kepner, Jeremy.

Parallel MATLAB for multicore and multinode computers / Jeremy Kepner.
p. cm. — (Software, environments, and tools ; 21)
Includes index.
ISBN 978-0-898716-73-3
1. Parallel processing (Electronic computers) 2. MATLAB. 3. Multiprocessors. I. Title.
QA76.58.K46 2009
004'.35–dc22

2009013013

For

Jemma

Alix

Clay

Diane

and

Gordon

Contents

List of Figures

List of Tables

List of Algorithms

Preface

MATLAB® is currently the dominant language of technical computing with approximately one million users worldwide, many of whom can benefit from the increased power offered by widely available multicore processors and multinode computing clusters. MATLAB is also an ideal environment for learning about parallel computing, allowing the user to focus on parallel algorithms instead of the details of the implementation. The succinctness of MATLAB allows many specific examples to be presented to illustrate parallel programming concepts.

The subject of this book is parallel programming in MATLAB and is hopefully the first of many books on this topic, as there are now a wide variety of parallel MATLAB libraries [Choy 2004] for users to choose from. I am fortunate to have been involved in the development of two of these libraries [Kepner 2004, Kepner 2003]. The most widely used of these libraries include pMatlab (developed by MIT Lincoln Laboratory), the Parallel Computing Toolbox (developed by The MathWorks, Inc.), and StarP (developed at MIT, UCSB, and Interactive Supercomputing, Inc.). All of these libraries provide direct support for parallel computing using distributed arrays and other parallel programming models. The specific examples in this book are written using the freely available pMatlab library. pMatlab has the advantage of running either standalone or on top of the aforementioned parallel MATLAB libraries. So all the examples in the book can be run in any of the parallel MATLAB environments. Fortunately, the concepts illustrated in the book are independent of the underlying implementation and valid for any particular parallel programming syntax the user may prefer to use. The software along with installation instructions can be found at the book website:

http://www.siam.org/KepnerBook

The expressive power of MATLAB allows us to concentrate on the techniques for creating parallel programs that run well (as opposed to the syntactic mechanics of writing parallel programs). Our primary focus is on the designing, coding, debugging, and testing techniques required to quickly produce well-performing parallel programs in a matter of hours instead of weeks or months (see Figure 1). These techniques have been developed over the years through thousands of one-on-one interactions with hundreds of parallel MATLAB users.

A general familiarity with MATLAB is assumed (see [Higham & Higham 2005] and [Moler 2004] for an introduction to MATLAB). The target audience of the book

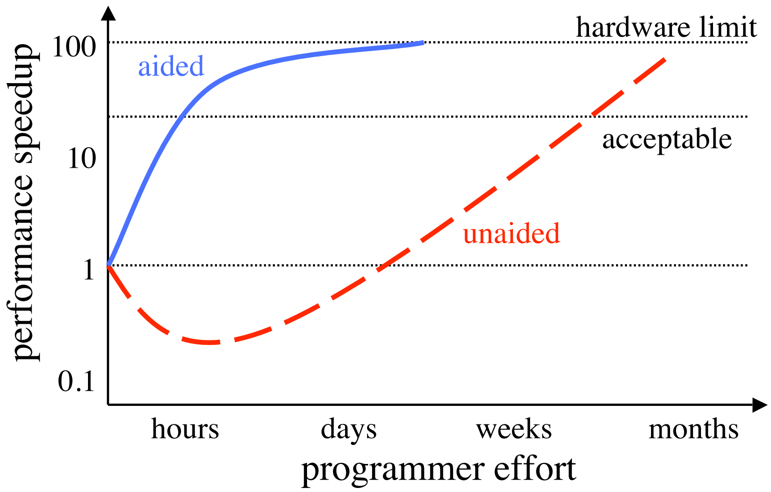

Figure 1. Performance versus effort.
Depiction of the performance achieved versus the user effort invested for a typical parallel MATLAB program. The bottom curve shows the effort required for an unaided user. The top curve shows the effort required for a user aided with expert assistance.

is anyone who needs to adapt their serial MATLAB program to a parallel environment. In addition, this book is suitable as either the primary book in a parallel computing class or as a supplementary text in a numerical computing class or a computer science algorithms class. Ideas are presented using a "hands-on" approach with numerous example programs. Wherever possible, the examples are drawn from widely known and well-documented parallel benchmark codes that have already been identified as representing many applications (although the connection to any particular application may require examining the references). For historical reasons, most of the examples are drawn from the technical computing domain, but an understanding of numerical methods is *not* required in order to use the examples. They are simply convenient and well-documented examples of different types of parallel programming.

The book is organized around two central concepts: the core programming process (i.e., design, code, debug, and test) and the core parallel programming models (i.e., distributed arrays, manager/worker, and message passing) [Lusk 2004]. The distributed array programming model will be the baseline programming model used throughout the book. Distributed arrays are easiest to understand, require the least amount of code to use, and are well-matched to the array nature of MATLAB. Distributed arrays allow very complicated communication patterns to be illustrated far more simply than with a message passing approach and perform well on both multicore and multinode parallel computers. In addition, distributed arrays naturally illustrate the core parallel design concepts of concurrency and locality. Distributed arrays are sufficient for 90% of parallel applications, but there are instances where other programming models are optimal and these are discussed where appropriate. The ultimate goal of this book is teach the reader to "think [distributed] matrices, not messages" [Moler 2005].

Throughout the book the approach is to first present concrete examples and then discuss in detail the more general parallel programming concepts these examples illustrate. The book aims to bring these ideas to bear on the specific challenge of writing parallel programs in MATLAB.

To accommodate the different types of readers, the book is organized into three parts. Part I (Chapters 1–3) provides a general conceptual overview of the key programming concepts illustrated by specific examples. Part II (Chapters 4–6) focuses on the analysis techniques of effective parallel programming. Part III (Chapters 7–11) consists of specific case studies. Parts I, II, and III can be treated independently and may be used as modules in a larger course. Finally, in recognition of the severe time constraints of professional users, each chapter is mostly self-contained and key terms are redefined as needed. Each chapter has a short summary and references within that chapter are listed at the end of the chapter. This arrangement allows the professional user to pick up and use any particular chapter as needed.

Chapter 1 begins by introducing some notation and the basic parallel programming interfaces used throughout the rest of the book. Chapter 2 provides a rapid introduction to creating and running simple parallel programs. Chapter 2 is meant to give readers a taste of the ease of use that parallel MATLAB provides. Chapter 3 focuses on a more complex example and highlights interacting with distributed arrays. Chapter 4 is a broad overview of the field of parallel programming, exposing the key principles that will be covered in more detail in the rest of the book. Chapter 4 uses a more sophisticated example than that used in Chapter 3, exposing the boundaries of the different parallel programming models. In addition, Chapter 4 introduces several key questions that must be dealt with in parallel programming:

Design: when to use parallel processing, concurrency versus locality, how to predict parallel performance, which parallel programming model to use: distributed arrays, client/server, or message passing.

Code: how and where to use parallel code, how to write scalable code (e.g., scalable file I/O), good coding style.

Debug: what the techniques are for going from a serial to a fully parallel execution, what kind of errors to check for at each stage along the way.

Test: how to measure performance achieved and compare with what is predicted.

Chapter 5 introduces the theory, algorithmic notation, and an "under the hood" view of distributed array programming. Chapter 6 discusses metrics for evaluating performance and coding of a parallel program. Chapter 7 is a selected survey of parallel application analysis techniques, with a particular emphasis on how the examples used in the book relate to many wider application domains. Chapters 8–11 use a set of well-studied examples drawn from the HPC Challenge benchmark suite (see http://www.hpcchallenge.org) to show detailed solutions to the challenges raised by parallel design, coding, debugging, and testing.

Throughout this book we draw upon numerous classical parallel programming examples that have been well studied and characterized in the literature. It is my hope that in seeing these examples worked out the reader will come to appreciate the elegance of parallel programming in MATLAB as much as I have.

References

[Choy 2004] Ron Choy, Parallel Matlab survey, 2004, http://supertech.lcs.mit.edu/~cly/survey.html

[Higham & Higham 2005] Desmond J. Higham and Nicholas J. Higham, MATLAB Guide, Second Edition, SIAM, Philadelphia, 2005.

[Kepner 2004] Jeremy Kepner and Stan Ahalt, MatlabMPI, Journal of Parallel and Distributed Computing, Vol. 64, No. 8, pp. 997–1005, 2004.

[Kepner 2003] Jeremy Kepner and Nadya Travinin, Parallel Matlab: The Next Generation, Seventh Annual High Performance Embedded Computing Workshop (HPEC 2003), September 23–25, 2003, MIT Lincoln Laboratory, Lexington, MA, http://www.ll.mit.edu/HPEC/agenda03.htm

[Lusk 2004] Ewing Lusk and Marc Snir, Parallel Programming Models, DARPA HPCS Productivity Team Workshop, January 13–14, 2004, Marina Del Ray, CA.

[Moler 2004] Cleve Moler, Numerical Computing with MATLAB, SIAM, Philadelphia, 2004.

[Moler 2005] Cleve Moler, Householder Meeting on Numerical Linear Algebra, Champion, PA, May 23–27, 2005.

Acknowledgments

There are many individuals to whom I am indebted for making parallel MATLAB and this book a reality. It is not possible to mention them all, and I would like to apologize in advance to those I may not have mentioned here due to accidental oversight on my part.

The development of parallel MATLAB has been a journey that has involved many colleagues who have made important contributions along the way. This book marks an important milestone in that journey: the broad availability and acceptance of parallel MATLAB. At a more technical level, this milestone signifies the broad availability and acceptance of the merger of the distributed array parallel programming model with high level languages such as MATLAB.

My own part in this journey has been aided by numerous individuals along the way who have directly influenced the content of this book. My introduction to numerical computing began at Pomona College in a freshman seminar taught by Prof. Wayne Steinmetz and was further encouraged by my undergraduate advisors Profs. Catalin Mitescu and Robert Chambers.

In the summer of 1990 I worked at the Aerospace Corporation, where I was introduced to high-level array languages by Dr. John Hackwell. This was followed by my first experience with parallel computation in the fall of 1990 provided by Prof. Alexandre Chorin at UC Berkeley [Kepner 1990]. Since that time I have worked to bring these two capabilities together. My first opportunity to do so occurred in 1997 when I worked with Dr. Maya Gokhale and Mr. Ron Minnich [Kepner et al. 1998]. This work was encouraged at Princeton University by my Ph.D. advisor Prof. David Spergel.

In 1998 I joined MIT Lincoln Laboratory, where Mr. Robert Bond was leading a team developing one of the first distributed array libraries in C++ [Rutledge & Kepner 1999]. In the spring of 2001, at the encouragement of Prof. Stan Ahalt, I wrote MatlabMPI in one week [Kepner 2001]. It's first user was Dr. Gil Raz. MatlabMPI, although difficult to use, developed a following at Lincoln and Ohio State (thanks to the efforts of Prof. Ashok Krishnamurthy and Dr. John Nehrbass).

In the summer of 2001 I was first exposed to the StarP work of Profs. Alan Edelman and John Gilbert and their students Dr. Perry Husbands, Mr. Ron Choy, and Dr. Viral Shah [Husbands 1999]. At this point I realized the distributed array concepts developed in C++ libraries could be brought to MATLAB with MatlabMPI underneath. From this concept Ms. Nadya Bliss created the pMatlab

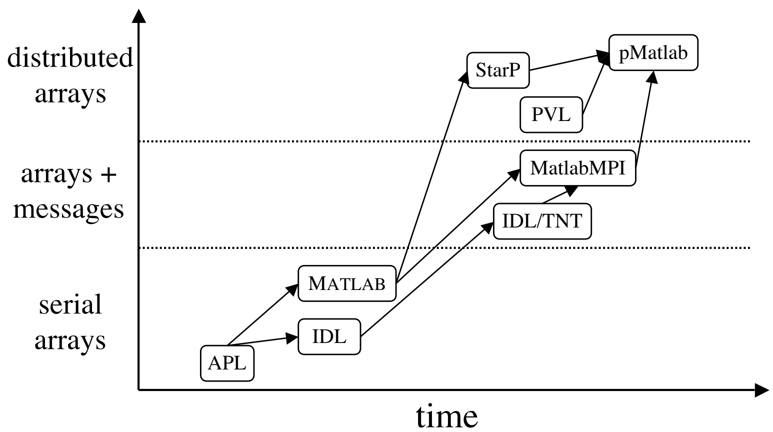

Figure 2. Development of parallel MATLAB.
A selected view of the evolution of some of the technologies that led to the development of pMatlab beginning with the first interactive array based language APL and ending with the current pMatlab library.

library [Kepner & Bliss 2003], which when coupled with the interactive LLGrid concept developed with Dr. Albert Reuther and Mr. Andrew McCabe became very popular at Lincoln [Reuther et al. 2004]. This early parallel MATLAB work was supported internally by Mr. David Martinez and externally by Mr. John Grosh.

Over the next few years I had the opportunity to interact closely with Dr. Cleve Moler and many others at The MathWorks as they brought parallel MATLAB out of the laboratory and made it into a product.

At the same time, I was also involved in the development of a suite of new parallel benchmarks designed to test new parallel architectures and parallel languages. The HPC Challenge benchmark suite was developed with Prof. Jack Dongarra, Dr. Piotr Luszczek, and Dr. Bob Lucas [Luszczek et al. 2006]. Mr. Andrew Funk, Mr. Ryan Haney, Mr. Hahn Kim, Dr. Julia Mullin, Mr. Charlie Rader, as well as many of the aforementioned individuals contributed to the parallel MATLAB implementations of these benchmarks. This later parallel MATLAB work was supported internally by Dr. Ken Senne and externally by Mr. Robert Graybil.

A visual depiction of this particular technology thread leading to pMatlab is shown in Figure 2.

In addition to those folks who have helped with the development of the technology, many additional folks have helped with the development of this book. Among these are my editor at SIAM, Ms. Elizabeth Greenspan, my copy editor at Lincoln, Ms. Dorothy Ryan, and the students in the spring 2008 MIT Applied Parallel Computing course. Finally, I would like to thank several anonymous reviewers whose comments enhanced this book.

References

[Husbands 1999] Parry Husbands, Interactive Supercomputing, 1999, Ph.D. Thesis, MIT, Cambridge, MA.

[Kepner 1990] Jeremy Kepner, Comparison of Canonical and Micro-Canonical Sampling Methods in a 2D Ising Model, 1990, Lawrence Berkeley Lab report LBL-30056.

[Kepner et al. 1998] J. Kepner, M. Gokhale, R. Minnich, A. Marks & J. DeGood, Interfacing Interpreted and Compiled Languages for Computing on a Massively Parallel Network of Workstations (MP-NOW), HPEC98, September 23–24, 1998, MIT Lincoln Laboratory, Lexington, MA.

[Kepner 2001] J. Kepner, Parallel Programing with MatlabMPI, HPEC 2001, September 25–27, 2001, MIT Lincoln Laboratory, Lexington, MA.

[Kepner & Bliss 2003] Jeremy Kepner & Nadya Travinin Bliss, Parallel Matlab: The Next Generation, 7th High Performance Embedded Computing Workshop (HPEC 2003), September 23–25, 2003, MIT Lincoln Laboratory, Lexington, MA.

[Luszczek et al. 2006] P. Luszczek, J. Dongarra & J. Kepner, Design and Implementation of the HPC Challenge Benchmark Suite, CT Watch, Vol. 2, Number 4A, November 2006.

[Reuther et al. 2004] A. Reuther, T. Currie, J. Kepner, H. Kim, A. McCabe, M. Moore & N. Travinin Bliss, On Demand Grid Computing with Grid Matlab and pMatlab, DOD HPCMP User's Group Conference 2004, June 8, Williamsburg, VA.

[Rutledge & Kepner 1999] Eddie Rutledge & Jeremy Kepner, PVL: An Object Oriented Software Library for Parallel Signal Processing, Cluster 2001, October 9, 2001, Newport Beach, CA.