Preliminaries

ASSUMED KNOWLEDGE

This book assumes a working knowledge of C and/or C++, and many examples are presented in these languages (primarily C++). To get the most out of the book these examples should be studied carefully. No knowledge of assembly language or systems programming is needed. No prior course on algorithms is required. In particular, the basics of asymptotic complexity analysis, needed for parallel performance analysis, are presented in this book in a self-contained way. Prior experience with these concepts would be useful but is not necessary. Detailed knowledge of a specific operating system is not required, although an operating systems course would be useful. We purposefully avoid programming to operating-system-specific threading models and avoid locks in our examples, so prior experience with these concepts is not necessary. Windows, Linux, and Mac OS X are all well supported by the primary programming models used, TBB and Cilk Plus, which allow for a wide selection of operating system choices for practical application of the material in this book. No prior experience with TBB and Cilk Plus is required and we provide enough background in appendices to make the book self-contained. However, for practical application development it is recommended that this text be supplemented with a reading of additional tutorial material and documentation on TBB and Cilk Plus. The section on "Parallel Programming Models" in Appendix A makes specific recommendations for such further reading. The secondary programming models, OpenMP, OpenCL, and ArBB, are not presented in depth; however, these models support many operating systems as well.

FOR INSTRUCTORS

This book supports teaching parallel programming in any programming class using C or C++, or as a focused topic in a semester-long class. If added to an existing course, and only one assignment for parallel programming is to be used, we recommend teaching the map pattern, as that illustrates both parallelism and the concept of patterns quite well. The remaining patterns are generally ordered from simplest to most challenging, so following the chapter order is recommended. We have included a summary chapter for all the patterns but it can be skipped on a first reading if necessary. We have found that a pattern-based approach is an effective way to teach and learn parallel programming in a structured manner.

Teaching material related to the book is available online, and may be used freely for courses taught in conjunction with this book. This material includes slides and example code. This material can be downloaded from

`http://parallelbook.com/download`

An explanation of the available teaching material, as well as additional information on using them in courses, can be found at

`http://parallelbook.com/instructor`

In particular, this material includes suggested reading roadmaps for use of this text in courses of different lengths and at different levels.

We invite you to share your own teaching insights when using our book and the online materials. Suggestions for additional material, such as new patterns or examples, would also be welcome. This book establishes a framework for "structured parallel programming," but it is only the beginning of what can be done in this area. In particular, it does not exhaust the space of examples. There are also some useful patterns that are not included in this book or are mentioned only briefly, so we intend to use online material to expand our discussion of both. Please contact us via our web site or email us at authors@parallelbook.com.

FOR STUDENTS

You are encouraged to download supplemental material from our web site at

http://parallelbook.com/student

This material includes code for the examples used in this book and possibly additional material to be used in conjunction with a course. Patterns are everywhere, but given the limited space in the book we could only print a tiny representative set of examples. We hope to be able to add more examples online over time, perhaps contributed by students like yourself.

We hope this book helps make "Thinking Parallel" an intuitive part of programming for you. Parallelism is now essential for all computing, but due to its complexity it requires a structured, disciplined approach.

We have chosen to organize this book around patterns to provide that structure. Patterns are the best way to share good programming strategies. The patterns we discuss have, in practice, been shown to lead to scalable, high-performance code while being implementable, understandable, and debuggable. These patterns are not a provably perfect set of what you need to know. Rather, patterns represent the best starting points that exist today. Opportunities to refine these patterns, and find more, are certainly there. We're sure that once you understand the concept of patterns, you will begin to see them everywhere and they will become an essential part of your vocabulary as a programmer.

FOR PROFESSIONAL PROGRAMMERS

Regardless of whether you have done some or no parallel programming, this book will help make "Thinking Parallel" an intuitive part of programming for you. We also hope that you will gain an appreciation for patterns as a way to structure your programs. Good patterns will also help you write good programs, since they encapsulate best known methods for achieving scalable and efficient results.

Patterns are effective structures of computation and data access; however, patterns by themselves are insufficient, since they are too abstact. Therefore, we also supply real example code to study. Part I of this book covers the patterns and also has many examples, while Part II has several additional examples.

We do not limit ourselves by using teaching languages in this book. We use proven programming models that are already in serious industrial use by professional programmers around the world. Intel Threading Building Blocks (TBB) and OpenMP are the two most popular models in use today, and are heavily used in this book. Additionally, Intel Cilk Plus and OpenCL have gained sufficient recognition

and usage to be worth exploring as well. You should also look at our web site—we hope to add additional examples there that would not fit in the book, and you can also download the source code for all examples in the book from that site.

A deep knowledge of computer architecture is not needed to understand this book and use the patterns we present. We have purposefully left out any in-depth discussion of parallel computer architecture, except for a short summary of some key points. Instead of teaching computer architecture and then parallel programming, we use patterns to lead to programming styles that map well onto real parallel hardware. Performance matters, so we make sure that our patterns and our discussion of them include the information needed to get excellent results. We do not discourage learning computer architecture, but we feel that it should not be a requirement to learn programming.

We know that vocabulary is important, so we have assembled a lengthy glossary that can be very helpful to review in order to more quickly be able to decipher parallel programming jargon.

USING CODE EXAMPLES

While the book itself is copyrighted, all example programming code found in this book is provided without any restrictions on reuse. You may use this code freely in your own projects, commercial or otherwise. However, we do not provide any promise or warrantee of any kind. The examples can all be downloaded at

```
http://parallelbook.com/download
```

where additional information on our code reuse policy can also be found.

We appreciate, but do not require, attribution. This is true of our teaching materials as well, which are are also available on the same web site. An attribution usually includes the title, author, publisher, and ISBN. For example:

> *Structured Parallel Programming* by Michael McCool, Arch Robison, and James Reinders, copyright 2012, published by Morgan Kaufmann, ISBN 978-0-124-15993-8.

If you have any questions, feel free to contact us at

```
permissions@parallelbook.com
```

HOW TO CONTACT US

We invite you to share your own insights when using our book. We can be reached via our web site or via email at

```
http://parallelbook.com
authors@parallelbook.com
```

This web site provides supplemental material including a list of errata, the ability to download all examples, and additional teaching materials. It is also our intention to distribute additional examples at this site, since limited space in this book did not permit us to include as many examples as we would have liked.

To comment or ask technical questions about this book, send email to:

bookquestions@parallelbook.com

For more information from the publisher Morgan Kaufmann, please visit their web site at

http://mkp.com

ACKNOWLEDGMENTS

This book would not have been possible without the dedicated people at Intel who have tirelessly worked to produce great technology and products. Their passion to help others also helped us produce the best book we could.

Much of the discussion for best practices for OpenMP in the desktop environment was derived from an analysis by Jay Hoeflinger and Brian Bliss of Intel.

The Bzip2 example code is derived from Julian Seward's Bzip2 implementation, available at

http://bzip.org

The TBB code for Bzip2 descends from a port done by Hyojin Sung while she was an intern at Intel. The cache-oblivious stencil code (Section 10.5) was adapted from code by Matteo Frigo and Yuxiong He.

An early draft of this book was reviewed by Jim Cownie, Mark Davis, Michèle Delsol, Stefanus Du Toit, Kathy A. Farrel, Balaji Iyer, Anton Malakhov, Tim Mattson, Priya Natarajan, John Pieper, Krishna Ramkumar, Elizabeth Reinders, Jim Sukha, Peter Tang, Barry Tannenbaum, Michael Voss, Bob Weems, Barry Wilkinson, and Terry Wilmarth. Their feedback was incredibly helpful and vastly improved the book. We would like to especially thank Tim Mattson who provided some additional OpenCL and OpenMP examples.

We all wish to thank our families and friends who put up with the wrinkles, early mornings, and late nights, sometimes simultaneous, that book writing brought into our lives.