Preface

The idea to write this book arose from a lecture that Michael was giving at the Technical University Munich. You know, just for the fun of it. The lecture was called “Parallel Programming Systems” and taught students the basics of how to bring a parallel programming model to a parallel machine. Exactly the topic of this book. What a coincidence!

The topic of parallel runtime systems is a really interesting one. Our day jobs are to work on all sorts of performance questions and low-level machine details on various processors. However, thinking about performance in an application code is quite different from thinking about performance at the lower levels of a software stack and inside a runtime library that supports a parallel programming model. Many more machine details are exposed at this low level and must be taken into account when aiming for the highest possible performance.

After the first couple of series of said lecture, it seemed that the students were really interested in the low-level stuff that Michael was teaching them, so he felt it would be a good idea to move to the next level and write a book! With Jim joining the team of authors, we were ready to write the book. In addition to his many years of experience, Jim brought his extensive knowledge of parallelism and his interest in low-level machine details to the table.

We hope that you’ll like the topic as much as we do. It has certainly been fun writing this fine piece of text (and the accompanying code), and we hope that you will enjoy reading it. We use many code examples and figures to illustrate the concepts and to help you to understand the details of how to write critical parts of the runtime system. Most of these code samples will be in C and C++, with some assembly language where we need it to demonstrate what modern compilers do to your source code.

Fortran programmers, or adepts of other languages: don’t feel left out; there is still a lot that you can take away from these discussions, and you will certainly be able to translate the key concepts into your language of choice. Our choice of C/C++ is not because we consider other languages uninteresting or irrelevant, but rather because we are considering implementation details that are very close to the machine, where C and C++ are still the dominant languages no matter which language the higher-level user code is written in. So please keep an open mind, even if the trouble with having an open mind is that people will insist on coming along and trying to put things in it.

We would like to thank our publisher, De Gruyter, for accepting the book into their portfolio and their patience while we were busy working on the book. We would like to thank Ute Skambraks at De Gruyter who worked with us patiently to prepare the manuscript. We would also like to express our gratitude to the reviewers who read the early versions of the book and did not run away screaming. They are Mark Bull, Barbara Chapman, Florina M. Ciorba, Chris Dahnken, Alex Duran, Wooyoung Kim, Will Lovett, Larry Meadows, Jennifer Pittman, Carsten Trinitis, and

https://doi.org/10.1515/9783110632729-202

Terry Wilmarth. Others who deserve thanks are our proofreader, Matthew Robertson (https://checkmatteditorial.com), and Randall Munroe, creator of the wonderful https://xkcd.com, for giving us permission to use one of his creations. Of course, all errors that still exist in the book are ours. Finally, this work used the Isambard UK National Tier-2 HPC Service (http://gw4.ac.uk/isambard/) operated by GW4 and the UK Met Office, and funded by EPSRC (EP/P020224/1), and some machines at the University of Bristol, who all deserve special thanks.

Now, without any further ado, please enjoy the book!

Michael & Jim