

Programming Many-Core Chips

András Vajda

Programming Many-Core Chips

With Contributions by Mats Brorsson
and Diarmuid Corcoran

Foreword by Håkan Eriksson

 Springer

András Vajda
Oy L M Ericsson Ab
Hirsalantie 11
02420 Jorvas
Finland
andras.vajda@ericsson.com

ISBN 978-1-4419-9738-8 e-ISBN 978-1-4419-9739-5
DOI 10.1007/978-1-4419-9739-5
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011930407

© Springer Science+Business Media, LLC 2011

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*Dedicated to the loving memory of my
grandmother.
Mama, without your dedication and sacrifices,
I would have never been in a position to
accomplish this.*

*Thank you.
András Vajda*

Foreword

Parallel computing has been confined, for much of its over 40 year history, to highly specialized, technology-wise advanced domains such as scientific computing or telecommunications. There were only a few experts who had the background and experience to write efficient, robust and scalable programs for these parallel machines.

A few years ago all that suddenly changed with the emergence of multi-core and many-core processors, as we reached the end of seemingly endless single-core performance scaling. While Moore's Law still provides chip designers with ever-increasing amount of transistors for building chips, these chips will now all have multiple cores and there's no end in sight for the scalability of computing power through the integration of ever more processor cores on the same piece of silicon. Practically overnight parallel programming skills became mandatory for most experts working in the field of computer programming.

Ericsson has always been at the forefront of parallel programming. We have been delivering best in class massively parallel telecommunication systems for several decades and providing our customers with quality software running on systems with hundreds of processors has always been one of our core assets. Continued leadership in this area is essential for our future success: the amount of traffic in telecom networks and consequently, the amount of computation required to sustain it, is expected to grow orders of magnitude faster than what the chip industry can match, even if it continues to deliver according to Moore's law. I believe we are well positioned to take advantage of emerging chips with hundreds or even thousands of cores in order to sustain our technology leadership in ICT.

This book is the result of the experience we as a company and our experts as individuals have accumulated over the past decades. It is driven by our desire to share with the software community the knowledge we built the hard way: learning from our mistakes and building on our successes. It is our strong belief that co-operation is essential in order to spark and sustain innovation that can generate the cutting

edge technologies from which we all—the computing community and the society in general—can ultimate benefit.

Håkan Eriksson

Senior Vice President, Chief Technology Officer

Head of Group Function Technology & Portfolio Management

Head of Ericsson in Silicon Valley

Contents

1	Introduction	1
1.1	The End of Endless Scalability	1
1.2	The Trouble with Software	3
1.3	The Book	4
1.3.1	Applications	6
1.4	Summary	6
	References	7
2	Multi-core and Many-core Processor Architectures	9
	Mats Brorsson	
2.1	Overview	9
2.2	Architectural Principles	10
2.2.1	Established Concepts	10
2.2.2	Emerging Concepts	21
2.3	Scalability Issues for Many-core Processors	26
2.4	Examples of Multi-core Processors	28
2.4.1	Processors Based on Low Number of Cores	29
2.4.2	Processors Based on Large Number of Cores	35
2.4.3	Heterogeneous Processors	40
2.5	Summary	41
	References	42
3	State of the Art Multi-Core Operating Systems	45
3.1	Definition of an Operating System	45
3.2	Operating System Architecture: Micro-Kernels and Monolithic Kernels	47
3.3	Scheduling	49
3.3.1	Symmetric Multi-Processing	50
3.3.2	Asymmetric Multi-Processing	51
3.3.3	Bound Multi-Processing	52
3.4	Memory Management	52
3.4.1	Virtual Memory and Memory Pages	54

3.4.2	Memory Allocation and Fragmentation	55
3.5	Current Main-Stream Operating Systems	61
3.5.1	Linux	61
3.5.2	Solaris	65
3.5.3	Windows	69
3.6	Summary	74
	References	74
4	The Fundamental Laws of Parallelism	77
4.1	Introduction	77
4.2	Amdahl's Law	77
4.2.1	Amdahl's Law for Many-core Chips	79
4.3	Gustafson's Law	82
4.4	The Unified Amdahl-Gustafson Law	84
4.5	Gunther's Conjecture	86
4.6	The Karp-Flatt Metric	87
4.7	The KILL Rule	87
4.8	Summary	88
	References	88
5	Fundamentals of Parallel Programming	89
5.1	Introduction	89
5.2	Decomposition and Synchronization	89
5.2.1	Functional Decomposition	90
5.2.2	Data-Based Decomposition	91
5.2.3	Other Types of Decomposition	91
5.2.4	Synchronization	92
5.2.5	Summary	93
5.3	Implementation of Decomposition	93
5.3.1	Summary	95
5.4	Implementation of Synchronization	96
5.4.1	Locks	97
5.4.2	Semaphores	99
5.4.3	Condition Variables and Monitors	100
5.4.4	Critical Sections	100
5.4.5	Transactional Memory	101
5.4.6	Shared Memory	102
5.4.7	The Follow the Data Pattern	103
5.4.8	Message Passing Based Communication	106
5.4.9	Partitioned Global Address Space (PGAS)	107
5.4.10	Future Constructs	107
5.4.11	Summary	108
5.5	Patterns of Parallel Programs	108
5.5.1	Structural Patterns	110
5.5.2	Parallel Algorithm Strategy Patterns	111

- 5.5.3 Implementation Strategy Patterns 112
- 5.5.4 Parallel Execution Patterns 114
- 5.6 Summary 115
- References 115

- 6 Debugging and Performance Analysis of Many-core Programs 117**
 - 6.1 Introduction 117
 - 6.2 Debugging 119
 - 6.3 Analysis and Profiling 121
 - 6.4 Performance Tuning 123
 - 6.5 Summary 125
 - References 126

- 7 Many-core Virtualization and Operating Systems 127**
 - 7.1 Introduction 127
 - 7.2 Fundamentals for a New Operating System Concept 128
 - 7.3 Space-shared Scheduling 131
 - 7.3.1 Architecture of a Space-shared Operating System 131
 - 7.3.2 Benefits and Drawbacks of Space-shared Operating Systems 133
 - 7.3.3 Summary 135
 - 7.4 Heterogeneity 135
 - 7.4.1 Managing Core Capabilities in Single-ISA Chips 136
 - 7.4.2 Managing Core Capabilities in Multi-ISA Chips 137
 - 7.4.3 Summary 138
 - 7.5 Power-aware Operating Systems 138
 - 7.6 Virtualization in Many-core Systems 139
 - 7.7 Experimental Many-core Operating Systems 140
 - 7.7.1 Corey 141
 - 7.7.2 fOS 142
 - 7.7.3 Barrelfish 143
 - 7.7.4 Tessellation 145
 - 7.7.5 heliOS 147
 - 7.8 Possible Future Trends: the Return of Speculative Execution 150
 - 7.9 Summary 150
 - References 151

- 8 Introduction to Programming Models 153**
 - 8.1 Introduction 153
 - 8.2 Communicating Sequential Processes (CSP) 154
 - 8.2.1 Practical Implementations of CSP 155
 - 8.3 Communicating Parallel Processes and the Actor Model 157
 - 8.3.1 Definitions and Axioms of the Actor Model 158
 - 8.3.2 Practical Realizations of the Actor Model 160
 - 8.4 Task-Based Programming Models 165

- 8.4.1 Practical Realizations of the Task Based Model 167
- 8.5 Process Versus Task-Based Parallelism and the Usage of Shared Memory 170
- 8.6 Summary 172
- References 172
- 9 Practical Many-Core Programming 175**
 - Diarmuid Corcoran
 - 9.1 Introduction 175
 - 9.2 Task-Based Parallelism 176
 - 9.2.1 Cilk 176
 - 9.2.2 Grand Central Dispatch 180
 - 9.2.3 Intel Thread Building Blocks 185
 - 9.2.4 Microsoft Task Parallel Library 189
 - 9.2.5 OpenMP 3.0 195
 - 9.2.6 Comparison of Task Based Programming Libraries 201
 - 9.2.7 Summary 202
 - 9.3 Data-Parallel Model 203
 - 9.3.1 OpenCL 203
 - 9.4 The Actor Model 207
 - 9.4.1 Erlang’s Actor Model 207
 - 9.5 Summary 210
 - References 211
- 10 Looking Ahead 213**
 - 10.1 What We Learned Until Now 213
 - 10.2 Scalability Bottlenecks 215
 - 10.3 Scaling Hard to Parallelize Applications 216
 - 10.4 Programming at Higher Abstraction Level 218
 - 10.5 Interaction with Related Domains 220
 - 10.5.1 Cloud Computing 220
 - 10.5.2 Exascale Computing 221
 - 10.5.3 Mobile Computing 221
 - 10.6 Summary: the World of Computing in 2020 222
 - References 222
- Index 225**