PARALLEL PROGRAMMING AND OPTIMIZATION WITH INTEL® XEON PHI COPROCESSORS

HANDBOOK ON THE DEVELOPMENT AND OPTIMIZATION OF PARALLEL APPLICATIONS FOR INTEL® XEON® PROCESSORS AND INTEL® XEON PHI® COPROCESSORS

COLFAX INTERNATIONAL ANDREY VLADIMIROV | RYOASAI | VADIM KARPUSENKO

PARALLEL PROGRAMMING AND OPTIMIZATION WITH INTEL[®] XEON PHI[™] COPROCESSORS

HANDBOOK ON THE DEVELOPMENT AND OPTIMIZATION

OF PARALLEL APPLICATIONS FOR INTEL[®] XEON[®] PROCESSORS AND INTEL[®] XEON PHI[™] COPROCESSORS

Second Edition

Andrey Vladimirov, Ryo Asai and Vadim Karpusenko

© Colfax International, 2013–2015

Electronic book built: May 14, 2015 Last revision date: May 13, 2015

Copyrighted Material

Copyright © 2013-2015, Colfax International. All rights reserved.

Cover image Copyright © pio3, 2013. Used under license from Shutterstock.com.

Published by Colfax International, 750 Palomar Ave, Sunnyvale, CA 94085, USA.

All Rights Reserved.

No part of this book (or publication) may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Intel, Xeon and Intel Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

All trademarks and registered trademarks appearing in this publication are the property of their respective owners.

Terms of Use

This book is available in the electronic version and in the printed version. Both versions are accompanied by a set of practical exercises available as an electronic archive. The book and the practical exercises may be used under the following terms:

- 1. You may use book and the code of the practical exercises for your own education.
- 2. If you use this book and/or practical exercises to teach a course,
 - a) every student must purchase their own copy of the book, OR
 - b) you must obtain written authorization from the copyright holder.
- 3. If you wish to use significant portions of the code of the practical exercises for derivative works, you must obtain written authorization from us.
- 4. You MAY NOT distribute the electronic version of the book or the source code of the "labs".
- 5. If you own a printed version of the book, you may lend it to other people, and the borrowers of the book may download the labs as described in Section 6.2 use them under the terms described here. This applies to individual book owners and to libraries (i.e., institutional book owners).

Disclaimer and Legal Notices

While best efforts have been used in preparing this book, the publisher makes no representations or warranties of any kind and assumes no liabilities of any kind with respect to the accuracy or completeness of the contents and specifically disclaims any implied warranties of merchantability or fitness of use for a particular purpose. The publisher shall not be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused, or alleged to have been caused, directly or indirectly, by the information or programs contained herein. No warranty may be created or extended by sales representatives or written sales materials.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Results have been simulated and are provided for informational purposes only. Results were derived using simulations run on an architecture simulator or model. Any difference in system hardware or software design or configuration may affect actual performance.

Because of the evolutionary nature of technology, knowledge and best practices described at the time of this writing, may become outdated or simply inapplicable at a later date. Summaries, strategies, tips and tricks are only recommendations by the publisher, and reading this eBook does not guarantee that one's results will exactly mirror our own results. Every company is different and the advice and strategies contained herein may not be suitable for your situation. References are provided for informational purposes only and do not constitute endorsement of any websites or other sources.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. All products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

ISBN: 978-0-9885234-2-5

About the Authors

Andrey Vladimirov, PhD, is Head of HPC Research at Colfax International. His primary interest is the application of modern computing technologies to computationally demanding scientific problems. Prior to joining Colfax, A. Vladimirov was involved in computational astrophysics research at Stanford University, North Carolina State University, and the Ioffe Institute in Russia, where he studied cosmic rays, collisionless plasmas and the interstellar medium using computer simulations.

Ryo Asai, is a Researcher at Colfax International. He develops optimization methods for scientific applications targeting emerging parallel computing platforms, computing accelerators and interconnect technologies. Ryo holds a B.S. degree in Physics from University of California, Berkeley.

Vadim Karpusenko, PhD, is Principal HPC Research Engineer at Colfax International involved in training and consultancy projects on data mining, software development and statistical analysis of complex systems. His research interests are in the area of physical modeling with HPC clusters, highly parallel architectures, and code optimization. Vadim holds a PhD from North Carolina State University for his research in in the field of computational biophysics on the free energy and stability of helical secondary structures of proteins.

> Additional publications by these authors related to Intel MIC architecture programming may be found at http://research.colfaxinternational.com/









Acknowledgements

Second Edition

We cannot thank enough the people who have contributed their valuable time and expertise to write technical reviews of the 2nd edition of this book. They have provided guidance, fixed misconceptions, future-proofed the messages and caught countless bugs: **Ilya Burylov, Gennady Fedorov, Alexandr Kalinkin, Alexandr Kobotov, Vadim Pirogov** (Intel/MKL), **Joseph Curley** (Intel), **Rob Farber** (TechEnablement.com), **Rakesh Krishnaiyer** (Intel), **Lawrence Meadows** (Intel), **John Pennycook** (Intel), **Troy Porter** (Stanford University), **Frances Roth** (Intel), **Jason Sewall** (Intel), **Gergana Slavova** (Intel). Thank you all very much!

First Edition

Authors are sincerely grateful to **James Reinders** for supervising and directing the creation of this book, **Albert Lee** for his help with editing and error checking, to specialists at Intel Corporation who contributed their time and shared with the authors their expertise on the MIC architecture programming: **Bob Davies**, **Shannon Cepeda**, **Pradeep Dubey**, **Ronald Green**, **James Jeffers**, **Taylor Kidd**, **Rakesh Krishnaiyer**, **Chris (CJ) Newburn**, **Kevin O'Leary**, **Zhang Zhang**, and to a great number of people, mostly from Colfax International and Intel, who have ensured that gears were turning and bits were churning during the production of the book, including **Rajesh Agny**, **Mani Anandan**, **Joe Curley**, **Roger Herrick**, **Richard Jackson**, **Mike Lafferty**, **Thomas Lee**, **Belinda Liviero**, **Gary Paek**, **Troy Porter**, **Tim Puett**, **John Rinehimer**, **Gautam Shah**, **Manish Shah**, **Bruce Shiu**, **Jimmy Tran**, **Achim Wengeler**, and **Desmond Yuen**.

1	Intr	oduction	1		
	1.1	Intel Xeon Phi Coprocessors	. 2		
	1.2	MIC Architecture: Developer's Perspective	. 13		
	1.3	Applicability of the MIC Architecture	. 30		
	1.4	Preparing for Future Parallel Architectures	. 39		
	1.5	System Administration with Intel Xeon Phi Coprocessors	. 46		
2	Prog	gramming Models	87		
	2.1	Native Applications and MPI	. 88		
	2.2	Explicit Offload Model	. 101		
	2.3	Shared Virtual Memory Model	. 119		
	2.4	Using Multiple Coprocessors	. 132		
	2.5	Official Programming with OpenMP 4.0	. 148		
3	Exp	ressing Parallelism	153		
	3.1	Data Parallelism (Vectorization)	. 154		
	3.2	Task Parallelism in Shared Memory: OpenMP	. 186		
	3.3	Task Parallelism with Intel Cilk Plus	. 212		
	3.4	Process Parallelism in Distributed Memory with MPI	. 229		
4	Optimizing Parallel Applications				
4	Opt	inizing i di dici Applications	201		
1	4.1	Optimization Roadmap for Intel Xeon Phi Coprocessors	. 261		
1	4.1 4.2	Optimization Roadmap for Intel Xeon Phi Coprocessors	. 261 . 261		
•	4.1 4.2 4.3	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization	201 261 267 289		
•	4.1 4.2 4.3 4.4	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading	201 261 267 289 311		
•	4.1 4.2 4.3 4.4 4.5 4.6	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control	201 261 267 289 311 356		
•	4.1 4.2 4.3 4.4 4.5 4.6 4.7	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control Optimization Strategies for MPI Applications	201 261 267 289 311 356 387		
•	4.1 4.2 4.3 4.4 4.5 4.6 4.7	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control Optimization Strategies for MPI Applications	201 261 267 289 311 356 387 396		
5	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control Optimization Strategies for MPI Applications	201 261 267 289 311 356 387 396 427		
5	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft 5.1	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control Optimization Strategies for MPI Applications ware Development Tools Intel Math Kernel Library	201 261 267 289 311 356 387 396 427 427		
5	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft 5.1 5.2	Optimization Roadmap for Intel Xeon Phi Coprocessors	201 261 267 289 311 356 387 396 427 427 444		
5 6	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft 5.1 5.2 Sum	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Optimization Memory Access Optimization Optimization Offload Traffic Control Optimizations Optimization Strategies for MPI Applications Optimizations ware Development Tools Intel Math Kernel Library Intel VTune Amplifier XE Intel Network	201 261 267 289 311 356 387 396 427 427 444 465		
5	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft 5.1 5.2 Sum 6.1	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Optimization of Multi-Threading Memory Access Optimization Optimization Offload Traffic Control Optimizations Optimization Strategies for MPI Applications Optimizations ware Development Tools Intel Math Kernel Library Intel VTune Amplifier XE Intel VTune Amplifier XE Parallel Programming and Intel Xeon Phi Coprocessors Optimizations	201 261 267 289 311 356 387 396 427 427 444 465 465		
5	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft 5.1 5.2 Sum 6.1 6.2	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control Optimization Strategies for MPI Applications Optimization Strategies for MPI Applications ware Development Tools Intel Math Kernel Library Intel VTune Amplifier XE Parallel Programming and Intel Xeon Phi Coprocessors Supplementary Code for Practical Exercises ("Labs")	201 261 267 289 311 356 387 396 427 427 444 465 465 465		
5 6	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft 5.1 5.2 Sum 6.1 6.2 6.3	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control Optimization Strategies for MPI Applications Ware Development Tools Intel Math Kernel Library Intel VTune Amplifier XE Parallel Programming and Intel Xeon Phi Coprocessors Supplementary Code for Practical Exercises ("Labs") Colfax Developer Training	201 261 267 289 311 356 387 396 427 427 444 465 465 465 467 470		
5	4.1 4.2 4.3 4.4 4.5 4.6 4.7 Soft 5.1 5.2 Sum 6.1 6.2 6.3 6.4	Optimization Roadmap for Intel Xeon Phi Coprocessors Scalar and General Optimizations Optimizing Vectorization Optimization of Multi-Threading Memory Access Optimization Offload Traffic Control Optimization Strategies for MPI Applications ware Development Tools Intel Math Kernel Library Intel VTune Amplifier XE mary and Resources Parallel Programming and Intel Xeon Phi Coprocessors Supplementary Code for Practical Exercises ("Labs") Colfax Developer Training Additional Resources	201 261 267 289 311 356 387 396 427 427 444 465 465 465 467 470 471		

Contents

1	Intr	itroduction			
	1.1	Intel 2	Keon Phi Coprocessors	2	
		1.1.1	Technology Overview	2	
		1.1.2	Conventional Programming, Portable Code	4	
		1.1.3	Heterogeneous Computing and Clustering	7	
		1.1.4	Intel Xeon Phi Product Family	8	
		1.1.5	Intel Xeon Processor E3, E5 and E7 Family	11	
	1.2	MIC A	Architecture: Developer's Perspective	13	
		1.2.1	Knights Corner Die Organization	13	
		1.2.2	Core Specifications	15	
		1.2.3	Memory Hierarchy and Cache Properties	17	
		1.2.4	Integration into the Host System through MPSS	20	
		1.2.5	Networking with Coprocessors in Clusters	22	
		1.2.6	File I/O on Coprocessors	24	
		1.2.7	Common Software Development Tools	25	
		1.2.8	Intel Xeon Processors versus Intel Xeon Phi Coprocessors: De-		
			veloper Experience	28	
	1.3	Applie	cability of the MIC Architecture	30	
		1.3.1	Task Parallelism	30	
		1.3.2	Data-Parallel Component	32	
		1.3.3	Memory Access Pattern	34	
		1.3.4	PCIe Bandwidth Considerations	36	
	1.4	Prepar	ring for Future Parallel Architectures	39	
		1.4.1	Exascale Computing for the Rest of Us	39	
		1.4.2	Second Generation MIC Processor, KNL	41	
		1.4.3	Future-Proof Development Options	44	
	1.5	Syster	m Administration with Intel Xeon Phi Coprocessors	46	
		1.5.1	Hardware Compatibility	46	
		1.5.2	Operating Systems	47	
		1.5.3	Installation and Minimal Configuration of MPSS	48	
		1.5.4	Controlling the MPSS service	49	
		1.5.5	Integration of MPSS with InfiniBand: OFED	50	

		1.5.6	Restoring MPSS Functionality after Kernel Updates	51
		1.5.7	Installation of Intel Compilers	52
		1.5.8	Installing the OpenCL Runtime and CodeBuilder	54
		1.5.9	Quick Functionality Check	56
		1.5.10	Overview of Intel MPSS Tools	58
		1.5.11	miccheck: Basic Troubleshooting	59
		1.5.12	micctrl: Coprocessor OS Configuration	61
		1.5.13	micflash: Coprocessor Firmware Updates	64
		1.5.14	micinfo: Coprocesssor, Firmware, Driver Info	65
		1.5.15	micrasd: Reliability Monitor, Error Logging	67
		1.5.16	micsmc: Real-Time Monitoring Tool	68
		1.5.17	User Management on Intel Xeon Phi Coprocessors	71
		1.5.18	SSH Client Configuration	76
		1.5.19	NFS Mounting a Host Export	77
		1.5.20	Sharing a Local Disk with VirtIO Block Device	80
		1.5.21	Bridged Networking in Clusters with Coprocessors	82
		1.5.22	Peer to Peer Communication between Coprocessors	84
		1.5.23	Manual Customization of the coprocessor OS	86
2	Prog	grammi	ng Models	87
2	Prog 2.1	grammi Native	ng Models Applications and MPI	87 88
2	Prog 2.1	grammi Native 2.1.1	ng Models Applications and MPI Using Compiler Argument -mmic to Compile Native Applica-	87 88
2	Prog 2.1	grammi Native 2.1.1	ng ModelsApplications and MPIUsing Compiler Argument $-mmic$ to Compile Native Applications for Intel [®] Xeon Phi TM Coprocessors	87 88 88
2	Prog 2.1	grammi Native 2.1.1 2.1.2	ng ModelsApplications and MPIUsing Compiler Argument $-mmic$ to Compile Native Applications for Intel [®] Xeon Phi TM CoprocessorsRunning Native Applications on Using SSH	87 88 88 90
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3	ng Models Applications and MPI	87 88 88 90 91
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4	ng Models Applications and MPI	87 88 88 90 91 93
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5	ng Models Applications and MPI	87 88 88 90 91 93 96
2	Prog 2.1 2.2	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explici	ng Models Applications and MPI Using Compiler Argument -mmic to Compile Native Applica- tions for Intel [®] Xeon Phi [™] Coprocessors Running Native Applications on Using SSH Running Native Applications with micnativeloadex Monitoring the Coprocessor Activity with micsmc MPI Applications on Intel Xeon Phi Coprocessors it Offload Model	87 88 90 91 93 96 101
2	Prog 2.1 2.2	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explice 2.2.1	ng Models Applications and MPI	87 88 90 91 93 96 101
2	Prog 2.1 2.2	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explicit 2.2.1 2.2.2	ng Models Applications and MPI Using Compiler Argument -mmic to Compile Native Applica- tions for Intel [®] Xeon Phi [™] Coprocessors Running Native Applications on Using SSH Running Native Applications with micnativeloadex Monitoring the Coprocessor Activity with micsmc MPI Applications on Intel Xeon Phi Coprocessors it Offload Model "Hello World" Example in the Explicit Offload Model Offloading Functions	87 88 90 91 93 96 101 101
2	Prog 2.1 2.2	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explice 2.2.1 2.2.2 2.2.3	ng Models Applications and MPI Using Compiler Argument -mmic to Compile Native Applica- tions for Intel [®] Xeon Phi [™] Coprocessors Running Native Applications on Using SSH Running Native Applications with micnativeloadex Monitoring the Coprocessor Activity with micsmc MPI Applications on Intel Xeon Phi Coprocessors MPI Applications on Intel Xeon Phi Coprocessors MPI Applications on Intel Xeon Phi Coprocessors Git Offload Model Git Offload	87 88 90 91 93 96 101 101 103 104
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explice 2.2.1 2.2.2 2.2.3 2.2.4	ng Models Applications and MPI	87 88 90 91 93 96 101 101 103 104 106
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explice 2.2.1 2.2.2 2.2.3 2.2.4 2.2.5	ng Models Applications and MPI Using Compiler Argument -mmic to Compile Native Applica- tions for Intel [®] Xeon Phi [™] Coprocessors Running Native Applications on Using SSH Running Native Applications with micnativeloadex Monitoring the Coprocessor Activity with micsmc MPI Applications on Intel Xeon Phi Coprocessors MPI Applications on Intel Xeon Phi Coprocessors MPI Applications on Intel Xeon Phi Coprocessors Git Offload Model Giffload Model Giffload Model Giffloading Functions Giffloading Bitwise-Copyable Data Data and Memory Persistence Between Offloads Asynchronous Offload	87 88 90 91 93 96 101 101 103 104 106
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explice 2.2.1 2.2.2 2.2.3 2.2.4 2.2.5 2.2.6	ng Models Applications and MPI Using Compiler Argument -mmic to Compile Native Applica- tions for Intel [®] Xeon Phi [™] Coprocessors Running Native Applications on Using SSH Running Native Applications with micnativeloadex Monitoring the Coprocessor Activity with micsmc MPI Applications on Intel Xeon Phi Coprocessors MPI Applications MPI Presistence Between Offload Model Asynchronous Offload Asynchronous Off	87 88 90 91 93 96 101 101 103 104 106 108 110
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explice 2.2.1 2.2.2 2.2.3 2.2.4 2.2.5 2.2.6 2.2.7	ng Models Applications and MPI	87 88 90 91 93 96 101 101 103 104 106 108 110
2	Prog 2.1	grammi Native 2.1.1 2.1.2 2.1.3 2.1.4 2.1.5 Explice 2.2.1 2.2.2 2.2.3 2.2.4 2.2.5 2.2.6 2.2.7 2.2.8	ng Models Applications and MPI Using Compiler Argument -mmic to Compile Native Applications for Intel [®] Xeon Phi [™] Coprocessors Running Native Applications on Using SSH Running Native Applications with micnativeloadex Monitoring the Coprocessor Activity with micsmc MPI Applications on Intel Xeon Phi Coprocessors MPI Applications on Intel Xeon Phi Coprocessors "Hello World" Example in the Explicit Offload Model Offloading Functions Offloading Bitwise-Copyable Data Data and Memory Persistence Between Offloads Asynchronous Offload Target-Specific Code Optional and Conditional Offload, Fall-Back to Host	87 88 90 91 93 96 101 101 103 104 106 108 110 111

		2.2.10	Proxy Console I/O	. 116
		2.2.11	Review: Explicit Offload Model	. 117
	2.3	Shared	Virtual Memory Model	. 119
		2.3.1	Offloading Functions	. 121
		2.3.2	Sharing and Offloading Objects	. 122
		2.3.3	Dynamic Allocation in Shared Virtual Memory	. 123
		2.3.4	Classes in Shared Virtual Memory	. 125
		2.3.5	Placement Operator new for Shared Classes	. 128
		2.3.6	Asynchronous Offload	. 130
		2.3.7	Summary for Shared Virtual Memory Model	. 131
	2.4	Using	Multiple Coprocessors	. 132
		2.4.1	Multiple Coprocessors with Explicit Offload	. 133
		2.4.2	Multiple Coprocessors in the Shared Virtual Memory Model .	. 138
		2.4.3	Multiple Coprocessors with MPI	. 141
	2.5	Offload	d Programming with OpenMP 4.0	. 148
		2.5.1	Offload with Pragma Target	. 149
		2.5.2	Data Persistence with Pragma Target Data	. 150
3	Exp	ressing	Parallelism	153
	3.1	Data P	arallelism (Vectorization)	. 154
		3.1.1	Vector Instructions: Concept and History	. 154
		3.1.2	Intel Architecture Vector Instruction Sets	. 155
		3.1.3	Is Your Code Using Vectorization?	. 156
		3.1.4	Data Alignment	. 157
		3.1.5	Vector Instructions using Inline Assembly, Compiler Intrinsics	
			and Class Libraries	. 163
		3.1.6	Automatic Vectorization of Loops	. 166
		3.1.7	Extensions for Array Notation in Intel Cilk Plus	. 171
		3.1.8	SIMD-Enabled Functions	. 173
		3.1.9	Assumed Vector Dependence	. 175
		3.1.10	Vectorization Pragmas, Keywords and Compiler Arguments	. 178
		3.1.11	Exclusive Features of the IMCI Instruction Set	. 181
	3.2	Task P	arallelism in Shared Memory: OpenMP	. 186
		3.2.1	Multiple Cores and Task Parallelism	. 186
		3.2.2	"Hello World" with OpenMP	. 188
		3.2.3	For-Loops in OpenMP	. 190
		3.2.4	Tasks in OpenMP	. 194

		3.2.5	Shared and Private Variables
		3.2.6	Synchronization: Avoiding Unpredictable Behavior 202
		3.2.7	Reduction: Avoiding Synchronization
	3.3	Task F	Parallelism with Intel Cilk Plus
		3.3.1	"Hello World" in Intel Cilk Plus
		3.3.2	For-Loops in Intel Cilk Plus
		3.3.3	Fork-Join Model and Spawning in Intel Cilk Plus
		3.3.4	Synchronization with Spawned Tasks
		3.3.5	Reduction: Avoiding Synchronization
		3.3.6	OpenMP versus Intel Cilk Plus
		3.3.7	Additional Resources on Shared Memory Parallelism 227
	3.4	Proces	s Parallelism in Distributed Memory with MPI
		3.4.1	Parallel Computing in Clusters with Multi-Core and Many-Core
			Nodes
		3.4.2	Program Structure in MPI
		3.4.3	Point-to-Point Communication
		3.4.4	MPI Communication Modes
		3.4.5	Collective Communication and Reduction
		3.4.6	Further Reading
4	Opti	3.4.6 imizing	Further Reading260Parallel Applications261
4	Opt 4.1	3.4.6 imizing Optim	Further Reading
4	Opt 4.1	3.4.6 imizing Optim 4.1.1	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261
4	Opt 4.1	3.4.6 imizing Optim 4.1.1 4.1.2	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263
4	Opt 4.1	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3	Further Reading
4	Opt 4.1	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266
4	Opt 4.1 4.2	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267
4	Opt 4.1 4.2	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization267
4	Opt i 4.1 4.2	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization269
4	Opt 4.1 4.2	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2 4.2.3	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization269Optimizing Arithmetic Expressions275
4	Opt 4.1 4.2	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2 4.2.3 4.2.3 4.2.4	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization267Compiler Controls for Precision269Optimizing Arithmetic Expressions275Programming Practices for High Performance282
4	Opt 4.1	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2 4.2.3 4.2.4 4.2.5	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization267Compiler Controls for Precision269Optimizing Arithmetic Expressions275Programming Practices for High Performance282Math Kernel Library for Scalar Arithmetic287
4	Opt 4.1 4.2 4.3	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2 4.2.3 4.2.4 4.2.5 Optim	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization267Compiler Controls for Precision269Optimizing Arithmetic Expressions275Programming Practices for High Performance282Math Kernel Library for Scalar Arithmetic289
4	Opt 4.1 4.2 4.3	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2 4.2.3 4.2.4 4.2.5 Optim 4.3.1	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization267Compiler Controls for Precision269Optimizing Arithmetic Expressions275Programming Practices for High Performance282Math Kernel Library for Scalar Arithmetic289Diagnosing the Utilization of Vector Instructions289
4	Opt 4.1 4.2 4.3	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2 4.2.3 4.2.4 4.2.5 Optim 4.3.1 4.3.2	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization267Compiler Controls for Precision269Optimizing Arithmetic Expressions275Programming Practices for High Performance282Math Kernel Library for Scalar Arithmetic287izing Vectorization289Diagnosing the Utilization of Vector Instructions289Unit-Stride Access and Spatial Locality of Reference290
4	Opt 4.1 4.2 4.3	3.4.6 imizing Optim 4.1.1 4.1.2 4.1.3 4.1.4 Scalar 4.2.1 4.2.2 4.2.3 4.2.4 4.2.5 Optim 4.3.1 4.3.2 4.3.3	Further Reading260Parallel Applications261ization Roadmap for Intel Xeon Phi Coprocessors261Optimization Checklist261Expectations263Benchmark Methodology264Benchmark Computing System266and General Optimizations267Compiler Controls for Optimization267Compiler Controls for Precision269Optimizing Arithmetic Expressions275Programming Practices for High Performance282Math Kernel Library for Scalar Arithmetic289Diagnosing the Utilization of Vector Instructions289Unit-Stride Access and Spatial Locality of Reference290Regularizing Vectorization Pattern295

		4.3.5	Compiler Hints: Pointer Disambiguation
		4.3.6	Strip-Mining for Vectorization
		4.3.7	Additional "Tuning Knobs" for Vectorization
	4.4	Optim	ization of Multi-Threading
		4.4.1	Avoiding Synchronization through Parallel Reduction 311
		4.4.2	Elimination of False Sharing with Padding
		4.4.3	Resolving Load Imbalance with Scheduling Control
		4.4.4	Dealing with Insufficient Parallelism
		4.4.5	Thread Affinity Optimization
		4.4.6	Diagnosing Parallel Efficiency, Scalability Tests
	4.5	Memo	ry Access Optimization
		4.5.1	General Considerations
		4.5.2	Loop Tiling
		4.5.3	Cache-Oblivious Recursive Methods
		4.5.4	First Touch Allocation and NUMA Policy
		4.5.5	Cross-Procedural Loop Fusion
		4.5.6	Advanced Topic: Prefetching
	4.6	Offloa	d Traffic Control
		4.6.1	Bandwidth Optimization with Persistent Buffers
		4.6.2	Masking Offload Latency with Double Buffering
	4.7	Optim	ization Strategies for MPI Applications
		4.7.1	Static Load Balancing
		4.7.2	Dynamic Work Scheduling
		4.7.3	Multi-threading within MPI Processes
		4.7.4	Fabric Control 420
5	Soft	ware D	evelopment Tools 427
	5.1	Intel M	Aath Kernel Library 427
		5.1.1	Functions Offered by MKL
		5.1.2	Linking Applications with MKL. Link Line Advisor 430
		5.1.3	MKL on Intel Xeon Phi Coprocessors
		5.1.4	Automatic offload
		5.1.5	Compiler-Assisted Offload
		5.1.6	Native Execution
		5.1.7	Benchmarks of Select MKL Functions
	5.2	Intel V	Tune Amplifier XE 444
		5.2.1	System Administration

		5.2.2	Running VTune	446
		5.2.3	Project Management	447
		5.2.4	Analysis on the Host CPU	448
		5.2.5	Analysis on an Intel Xeon Phi Coprocessor	459
6	Sum	mary a	nd Resources	465
	6.1	Paralle	l Programming and Intel Xeon Phi Coprocessors	465
	6.2	Supple	mentary Code for Practical Exercises ("Labs")	467
	6.3	Colfax	Developer Training	470
	6.4	Additio	onal Resources	471
Bil	Bibliography		475	

Foreword to the First Edition

We live in exciting times; the amount of computing power available for sciences and engineering is reaching enormous heights through parallel computing. Parallel computing is driving discovery in many endeavors, but remains a relatively new area of computing. As such, software developers are part of an industry that is still growing and evolving as parallel computing becomes more commonplace.

The added challenges involved in parallel programming are being eased by four key trends in the industry: emergence of better tools, wide-spread usage of better programming models, availability of significantly more hardware parallelism, and more teaching material promising to yield better-educated programmers. We have seen recent innovations in tools and programming models including OpenMP and Intel Threading Building Blocks. Now, the Intel[®] Xeon PhiTM coprocessor certainly provides a huge leap in hardware parallelism with its general purpose hardware thread counts being as high as 244 (up to 61 cores, 4 threads each).

This leaves the challenge of creating better-educated programmers. This handbook from Colfax, with a subtitle of "Handbook on the Development and Optimization of Parallel Applications for Intel Xeon Processors and Intel Xeon Phi Coprocessors" is an example-based course for the optimization of parallel applications for platforms with Intel Xeon processors and Intel Xeon Phi coprocessors.

This handbook serves as practical training covering understandable computing problems for C and C++ programmers. The authors at Colfax have developed sample problems to illustrate key challenges and offer their own guidelines to assist in optimization work. They provide easy to follow instructions that allow the reader to understand solutions to the problems posed as well as inviting the reader to experiment further. Colfax's examples and guidelines complement those found in our recent book on programming the Intel Xeon Phi Coprocessor by Jim Jeffers and myself by adding another perspective to the teaching materials available from which to learn.

In the quest to learn, it takes multiple teaching methods to reach everyone. I applaud these authors in their efforts to bring forth more examples to enable either self-directed or classroom oriented hands-on learning of the joys of parallel programming.

James R. Reinders

Co-author of "Intel[®] Xeon Phi[™] Coprocessor High Performance Programming" © 2013, Morgan Kaufmann Publishers Intel Corporation March 2013

Preface to the Second Edition

A lot has happened in Intel's "parallel universe" since the publication of the first edition of this book in March 2013. The family of Intel Xeon Phi coprocessors has grown to three series: 3100, 5100 and 7100, offering a range of performance tiers and prices. Active-cooling Intel Xeon Phi coprocessors were introduced, allowing workstation users to take advantage of the Intel Many Integrated Core (MIC) architecture. Plans were released for future Intel MIC architecture products, based on the Knights Landing chip, and capable of acting as a stand-alone CPU. In the CPU domain, Intel Xeon processors based on the Haswell architecture were released, supporting a new instruction set AVX2 and new functionality.

On the software tools side, the Intel Parallel Studio XE 2015 suite was improved to accommodate the new parallel framework standards: OpenMP 4.0 and MPI 3.0. The evolution of Intel VTune Amplifier XE has added many useful functions for automated diagnostics of performance issues. Intel compilers produce more user-friendly optimization reports than before, and have become even smarter about automatic vectorization and other optimizations.

The work in the users' domain did not stand still, either. With a large number of case studies and research articles on applications for the Intel MIC architecture, it is accurate to say that the developer ecosystem has been established. We are proud to say that Colfax has made a considerable contribution to this progress with the first edition of "Parallel Programmin and Optimization with Intel Xeon Phi Coprocessors". In the years 2013 and 2014, over 1000 science and industry experts at tens of locations across North America have been students of the Colfax Developer Training based on this book. Their experience and feedback, along with the innovations in the Intel tools, have built a solid case for the publication of the second edition of "Parallel Programming and Optimization with Intel Xeon Phi Coprocessors".

Among the numerous new features of the second edition, the ones that stand out are:

- 1. The details unveiled by Intel of the present and future MIC processors, including Knights Landing;
- 2. Discussion of configuration and system administration of clusters with Intel Xeon Phi coprocessors, including InfiniBand support, bridged network configuration and storage setup;

- 3. Additional applications based on case studies of our research in 2013–2014 included in the text as references, as well as practical exercises;
- 4. Console listings, example codes and hyperlinks to online manuals accurate as of Intel Parallel Studio XE 2015, Intel MPSS 3.4.1 and CentOS 7.0 Linux;
- 5. New programming models made available in OpenMP 4.0;
- 6. Deeper review of the Intel Math Kernel Library support for the MIC architecture;
- 7. More convenient page format and font size for on-screen reading, and
- 8. Numerous updates to the text improving the clarity and depth of the discussion.

We hope that you find this book to be a valuable resource on "all things Xeon Phi", and, as always, we value your feedback. The HPC research department of Colfax International can be reached by email at phi@colfax-intl.com, and the latest updates on our work can be found at research.colfaxinternational.com.

Preface to the First Edition

Welcome to the Colfax Developer Training! You are holding in your hands or browsing on your computer screen a comprehensive set of training materials for this training program. This document will guide you to the mastery of parallel programming with Intel[®] Xeon[®] family products: Intel[®] Xeon[®] processors and Intel[®] Xeon PhiTM coprocessors. The curriculum includes a detailed presentation of the programming paradigm for Intel Xeon product family, optimization guidelines, and hands-on exercises on systems equipped with Intel Xeon Phi coprocessors, as well as instructions on using Intel[®] software development tools and libraries included in Intel[®] Parallel Studio XE.

These training materials are targeted toward developers familiar with C/C++ programming in Linux. Developers with little parallel programming experience will be able to grasp the core concepts of this subject from the detailed commentary in Chapter 3. For advanced developers familiar with multi-core and/or GPU programming, the training offers materials specific to the Intel compilers and Intel Xeon family products, as well as optimization advice pertinent to the Many Integrated Core (MIC) architecture.

We have written these materials relying on key elements for efficient learning: practice and repetition. As a consequence, the reader will find a large number of code listings in the main section of these materials. In the extended Appendix, we provided numerous hands-on exercises that one can complete either under an instructor's supervision, or autonomously in a self-study training.

This document is different from a typical book on computer science, because we intended it to be used as a lecture plan in an intensive learning course. Speaking in programming terms, a typical book traverses material with a "depth-first algorithm", describing every detail of each method or concept before moving on to the next method. In contrast, this document traverses the scope of material with a "breadth-first" algorithm. First, we give an overview of multiple methods to address a certain issue. In the subsequent chapter, we re-visit these methods, this time in greater detail. We may go into even more depth down the line. In this way, we expect that students will have enough time to absorb and comprehend the variety of programming and optimization methods presented here. The course road map is outlined in the following list.

• Chapter 1 presents the Intel Xeon Phi architecture overview and the environment provided by the MIC Platform Software Stack (MPSS) and Intel Parallel Studio XE on Many Integrated Core architecture (MIC). The purpose of Chapter 1 is

to outline what users may expect from Intel Xeon Phi coprocessors (technical specifications, software stack, application domain).

- Chapter 2 allows the reader to experience the simplicity of Intel Xeon Phi usage early on in the program. It describes the operating system running on the coprocessor, with the compilation of native applications, and with the language extensions and CPU-centric codes that utilize Intel Xeon Phi coprocessors: offload and virtual-shared memory programming models. In a nutshell, Chapter 2 demonstrates how to write serial code that executes on Intel Xeon Phi coprocessors.
- Chapter 3 introduces Single Instruction Multiple Data (SIMD) parallelism and automatic vectorization, thread parallelism with OpenMP and Intel Cilk Plus, and distributed-memory parallelization with MPI. In brief, Chapter 3 shows how to write parallel code (vectorization, OpenMP, Intel Cilk Plus, MPI).
- Chapter 4 re-iterates the material of Chapter 3, this time delving deeper into the topics of parallel programming and providing example-based optimization advice, including the usage of the Intel Math Kernel Library. This chapter is the core of the training. The topics discussed in this Chapter 4 include:
 - i) scalar optimizations;
 - ii) improving data structures for streaming, unit-stride, local memory access;
 - iii) guiding automatic vectorization with language constructs and compiler hints;
 - iv) reducing synchronization in task-parallel algorithms by the use of reduction;
 - v) avoiding false sharing;
 - vi) increasing arithmetic intensity and reducing cache misses by loop blocking and recursion;
 - vii) exposing the full scope of available parallelism;
 - viii) controlling process and thread affinity in OpenMP and MPI;
 - ix) reducing communication through data persistence on coprocessor;
 - x) scheduling practices for load balancing across cores and MPI processes;
 - xi) optimized Intel Math Kernel Library function usage, and other.

If Chapter 3 demonstrated how to write parallel code for Intel Xeon Phi coprocessors, then Chapter 4 shows how to make this parallel code run fast.

• Chapter 6 summarizes the course and provides pointers to additional resources.

Throughout the training, we emphasize the concept of portable parallel code. Portable parallelism can be achieved by designing codes in a way that exposes the data and task

parallelism of the underlying algorithm, and by using language extensions such as OpenMP pragmas and Intel Cilk Plus. The resulting code can be run on processors as well as on coprocessors, and can be ported with only recompilation to future generations of multi- and many-core processors with SIMD capabilities. Even though the Colfax Developer Training program touches on low-level programming using intrinsic functions, it focuses on achieving high performance by writing highly parallel code and utilizing the Intel compiler's automatic vectorization functionality and parallel frameworks.

The handbook of the Colfax Developer Training is an essential component of a comprehensive, hands-on course. While the handbook has value outside a training environment as a reference guide, the full utility of the training is greatly enhanced by students' access to individual computing systems equipped with Intel Xeon processors, Intel Xeon Phi coprocessors and Intel software development tools. Please check the Web page of the Colfax Developer training for additional information: http://www.colfax-intl.com/xeonphi/

Welcome to the exciting world of parallel programming!

THIS IS A PREVIEW

COMPLETE BOOK Is available at <u>Xeonphi.com/book</u>

508 PAGES Electronic or print edition

IT IS ALL ABOUT OPTIMIZING PARALLELISM

Parallelism has long been a nonnegotiable requirement of all high performance computing applications in supercomputer sites. These days, parallel computing is also becoming commonplace in smaller computing environments: private clusters, workstations and portable computers. Computer architectures grow in size (more compute nodes in a cluster, more cores on a chip) and evolve in depth (wider SIMD registers, deeper pipelines). Harnessing this rocketing growth of hardware capabilities to tackle new fascinating computing problems requires that software developers continually learn to optimize their applications to utilize all available levels and scope of hardware parallelism.

In Parallel Computing and Optimization with Intel Xeon Phi Coprocessors, Colfax International presents to high performance application developers the state-of-the-art programming paradigms and best optimization practices for modern computing platforms based on the Intel multi-core and Many Integrated Core (MIC) architectures.

In this example-based intensive guide to programming Intel Xeon Phi coprocessors, you will find:

- An overview of Intel MIC processors of the first generation (Knights Corner) and second generation (Knights Landing);

- Introduction to task- and data-parallel programming with MPI, OpenMP, Intel Cilk Plus, and automatic vectorization with Intel C and C++ compiler;

- Extensive discussion of high performance application optimization on the Intel Xeon and Intel Xeon Phi platforms, including scalar optimizations, improvement of SIMD operations, multithreading, efficient cache utilization, communication control, and scaling across heterogeneous distributed-memory computing systems;

- A discussion of system administration tasks for workstations and clusters with Intel Xeon Phi coprocessors;

- Supplementary code for practical exercises (self-study "labs") comprising 30 guided exercises with solutions, also used in the Colfax Developer Training program.

