

MULTITHREADING FOR OPTIMIZING PDES ON MULTICORE PLATFORMS

BY

DEEPAK A. JAGTAP

BE, University of Pune, India, 2006

THESIS

Submitted in partial fulfillment of the requirements for
the degree of Master of Science in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2012

UMI Number: 1529641

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1529641

Published by ProQuest LLC (2012). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

PREVIEW

© Copyright by Deepak A. Jagtap 2012

All Rights Reserved

Accepted in partial fulfillment of the requirements for
the degree of Master of Science in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2012

March 16, 2012

Dr. Dmitry Ponomarev, Associate Professor
Department of Computer Science, Binghamton University

Dr. Nael Abu-Ghazaleh, Associate Professor
Department of Computer Science, Binghamton University

Dr. Madhusudhan Govindaraju, Associate Professor
Department of Computer Science, Binghamton University

ABSTRACT

Existing parallel discrete event simulation (PDES) kernels use process-based implementations and rely on MPI communication library for message passing. The drawback of these implementations is their reliance on multiple message copying operations, even when the two communicating processes are running on multiple cores on the same chip. As the multicore processors become prevalent and the number of cores per chip rapidly increases, process-based communication model results in a highly suboptimal simulation performance.

We investigate alternative thread-based communication model between PDES processes to take direct advantage of the shared memory hierarchy available on modern multicore chips. The shared memory approach eliminates multiple message copying and significantly minimizes synchronization delays. We implemented multithreaded PDES using ROSS simulator and studied its performance on three hardware platforms: a Intel Core i7, a 48-core AMD Opteron Magny-Cours and 64-core TilePro64 platform. Results shows significant performance improvement on all platforms after carefully addressing some performance bottlenecks.

DEDICATION

I would like to dedicate this work to my family, for there wonderful support throughout these years. I would also like to take this opportunity to convey my gratitude towards my advisors Prof. Dmitry and Prof. Nael for there invaluable guidance, encouragement and support for accomplishing this work.

PREVIEW

ACKNOWLEDGEMENTS

It's been a great pleasure working under the guidance of Dr. Dmitry Ponomarev and Dr. Nael Abu-Ghazaleh. I feel honored and fortunate to work with them, and would like to convey my sincere thanks for all there support for completing this thesis. I would also like to thank Dr. Madhusudhan Govindaraju for evaluating the thesis and providing insightful comments. I am grateful to my colleages Ketan and Jingjing for there valuable discussions and suggestions. It is a pleasure to thank all the Professors at SUNY Binghamton for blessing the knowledge, and enabling me to complete this thesis.

CONTENTS

List of Figures	ix
1 Introduction	1
1.0.1 Sequential Discrete Event Simulation	1
1.0.2 Parallel Discrete Event Simulation	2
1.0.3 PDES on Traditional Multi-core Architectures	5
1.0.4 PDES on Emerging Manycore Architectures	6
2 Related Work	7
2.1 Related Work	7
2.1.1 Optimizing Communication for PDES	8
2.1.2 Shared Memory PDES	8
3 Overview	11
3.1 Overview	11
3.1.1 ROSS	11
3.1.2 Traditional Multi-core Architectures	11
3.1.3 Emerging Many-core Architectures	12
4 Multithreaded ROSS	14
4.1 ROSS: Design Overview	14
4.1.1 Communication in MPI-based ROSS	14
4.1.2 Communication in Multithreaded ROSS	15
4.1.3 ROSS Event scheduler Loop	17
5 Optimizations	18
5.1 Performance Bottlenecks and Optimizations	18
5.1.1 Efficient Barrier Synchronization	19

5.1.2	NUMA-aware free memory management	19
5.1.3	Distributed Locking for the Input Queue	21
6	Evaluation on Traditional Multicores	22
6.1	Performance Evaluation of ROSS-MT on Traditional Multicore	22
6.1.1	Experimental Setup and Benchmark	22
6.1.2	ROSS-MT Performance Analysis	23
7	Tilera: Emerging Manycore Platform	30
7.1	Tilera Tile64Pro Architecture Overview	30
7.1.1	ROSS-MT Performance Optimizations and their Adaptation to Tilera	32
8	Evaluation on Tilera	34
8.1	Performance Evaluation of ROSS-MT and ROSS-MPI on Tilera	34
8.1.1	ROSS-MT Scalability Analysis on Tilera	34
8.1.2	Stress-testing the iMesh	38
9	Conclusion	40
9.1	Concluding Remarks and Future Work	40
	Bibliography	42

LIST OF FIGURES

1.1	Conservative Simulation	3
1.2	Optimistic Simulation	4
3.1	Architecture of the Intel Core-i7	12
3.2	Architecture of the AMD Magny-cours	13
4.1	MPI-based Message Passing Mechanism	14
4.2	Mutlithreaded ROSS Message Passing Mechanism	16
5.1	Performance of Baseline ROSS-MT vs. ROSS using MPI	18
6.1	Optimizing ROSS-MT on Intel core-i7	24
6.2	Execution Time Breakdown – Core i7	25
6.3	Magny-cours performance for different degrees of parallelism	25
6.4	Magny-cours Performance as a function of Remote Events	26
6.5	Performance Improvement Relative to MPI–Magny-cours	27
6.6	Execution time of the optimized ROSS-MT on Intel core-i7	27
6.7	Execution time of the optimized ROSS-MT on AMD Magny-cours	28
6.8	ROSS-MT with increased message size	28
6.9	Speedup for AMD Magny-Cours	29
7.1	Architecture of the Tilera Processor.	31
8.1	Speedup at 20% Remote Communication	34
8.2	Speedup at 40% Remote Communication	35
8.3	Speedup at 100% Remote Communication	35
8.4	Execution Time for ROSS-MT and ROSS-MPI on Tilera	36
8.5	Efficiency at GVT Interval 2048	37
8.6	ROSS-MT Performance with Increasing Event Population	38

8.7 Scalability with Increasing LPs per PE at Different Remote Percentages . .	39
--	----

PREVIEW