

**DISTRIBUTED OPTIMISTIC SIMULATION
OF DEVS AND CELL-DEVS MODELS WITH PCD++**

By

Qi Liu, B. Eng.

A thesis submitted to

The Faculty of Graduate Studies and Research

In partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

Canada

© Copyright 2006, Qi Liu

The undersigned hereby recommends to the Faculty of Graduate Studies and Research

Acceptance of the thesis

Distributed Optimistic Simulation of DEVS and Cell-DEVS Models with PCD++

Submitted by Qi Liu

In partial fulfillment of the requirements for the degree of

Master of Applied Science

Thesis Supervisor

Dr. Gabriel A. Wainer

Chair, Department of Systems and Computer Engineering

Dr. Victor C. Aitken

Carleton University

2006

ABSTRACT

DEVS is a sound formal modeling and simulation (M&S) framework based on generic dynamic system concepts. Cell-DEVS is a DEVS-based formalism intended to model complex physical systems as cell spaces. Time Warp is the most well-known optimistic synchronization protocol for parallel and distributed simulations. This work is devoted to developing new techniques for executing DEVS and Cell-DEVS models in parallel and distributed environments based on the WARPED kernel, an implementation of the Time Warp protocol. The resultant optimistic simulator, called as PCD++, is built as a new simulation engine for CD++, an M&S toolkit that implements the DEVS and Cell-DEVS formalisms. Algorithms in CD++ and the WARPED kernel are redesigned to carry out optimistic simulations using a non-hierarchical approach that reduces the communication overhead. The message-passing paradigm is analyzed using a high-level abstraction called *wall clock time slice*. A two-level *user-controlled state-saving* mechanism is proposed to achieve efficient and flexible state saving at runtime. This mechanism is integrated with both the *copy state-saving* and *periodic state-saving* strategies to realize a hybrid technique that gives simulator developers the full power to dynamically choose the best possible combination of state-saving strategies at runtime. An optimization strategy called *one log file per node* is provided to break the bottleneck caused by file I/O operations. The number of file descriptors consumed in the simulation is upper-bounded and the operational overhead is reduced substantially under this strategy. Different Time Warp optimizations are integrated into PCD++, and their effects are analyzed quantitatively. It is shown that PCD++ markedly outperforms other alternatives, and considerable speedups can be achieved in parallel and distributed simulations, indicating that PCD++ is well-suited for simulating large and complex models.

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge the enthusiastic supervision and constant support of Dr. Gabriel Wainer during this work.

I want to thank specially to Narendra Mehta for his assistance with all types of technical problems – at all times.

Finally, I am forever indebted to my parents and Sara for their understanding, endless patience and encouragement throughout the course of my studies.

TABLE OF CONTENTS

ABSTRACT	III
ACKNOWLEDGEMENTS	IV
LIST OF TABLES.....	VIII
LIST OF FIGURES.....	IX
LIST OF ACRONYMS.....	XII
CHAPTER 1 INTRODUCTION.....	1
1.1. Contribution	3
1.2. Thesis Overview	5
CHAPTER 2 REVIEW OF THE STATE OF THE ART	6
2.1. DEVS and Parallel DEVS formalisms	6
2.2. Timed Cell-DEVS and Parallel Cell-DEVS formalisms.....	10
2.3. Parallel and Distributed Simulation.....	13
2.3.1. Conservative parallel discrete event simulation	14
2.3.2. Optimistic parallel discrete event simulation.....	14
2.4. DEVS-based Simulation Toolkits	16
CHAPTER 3 SOFTWARE ARCHITECTURE	19
3.1. Layered Architecture.....	19
3.2. Major Functionalities of the PCD++ and WARPED Layers.....	20
3.2.1. The WARPED layer	21
3.2.2. The PCD++ layer	25
CHAPTER 4 BASIC CONTROL MECHANISMS IN THE WARPED KERNEL. 28	
4.1. Assumptions of the WARPED Kernel.....	28
4.2. Kernel Control Mechanisms	29
4.2.1. Rollback mechanisms and cascaded rollback process	29
4.2.2. GVT calculation and fossil collection.....	33
4.3. Problems and Fixes.....	34
CHAPTER 5 DISTRIBUTED OPTIMISTIC SIMULATION IN PCD++	36
5.1. Flattened Structure for the Simulation Framework.....	36

5.2.	Message Definitions	38
5.3.	Current Status of the CD++ Toolkit	38
5.4.	Structure for Inter-LP Communications.....	40
5.5.	Message-processing Algorithms for PCD++ Processors	41
5.5.1.	Simulator	41
5.5.2.	Flat Coordinator	43
5.5.3.	Node Coordinator	46
5.5.4.	Root Coordinator.....	49
5.6.	A Message-passing Scenario	50
5.7.	Starting and Terminating Simulations	54
5.8.	Saving and Restoring State Variables.....	55
5.9.	Asynchronous State Transitions in Cell-DEVS Models	57
5.9.1.	Cell-DEVS models with transport delay.....	59
5.9.2.	Cell-DEVS models with inertial delay	63
CHAPTER 6 ENHANCEMENTS TO PCD++ AND THE WARPED KERNEL		67
6.1.	An Abstraction for the Simulation Process	67
6.2.	Dormant State of Node Coordinators	70
6.3.	Handling Rollbacks at Time Zero	72
6.4.	User Controlled State Saving Mechanism	75
6.5.	Messaging Anomalies.....	77
6.5.1.	Speculative computation of the Node Coordinator	77
6.5.2.	Two types of messaging anomalies.....	79
6.5.3.	Anomaly with empty NC Message Bag.....	82
6.5.4.	Anomaly with non-empty NC Message Bag.....	83
6.5.5.	Enhanced NC algorithm for done message	91
6.6.	One Log File Per Node Strategy	92
CHAPTER 7 OPTIMIZATION ALGORITHMS IN THE WARPED KERNEL ...		95
7.1.	One Anti-message Per Rollback	95
7.2.	Periodic State Saving.....	97
7.2.1.	Strategy description.....	97
7.2.2.	UCSS mechanism revisited	99

7.2.3.	Integrating PSS strategy in PCD++	100
7.2.4.	Modifications to the fossil collection algorithm	101
7.2.5.	Miscellaneous modifications	103
7.3.	Lazy Cancellation.....	104
CHAPTER 8	EXPERIMENTS AND PERFORMANCE ANALYSIS.....	106
8.1.	Introduction to the Cell-DEVS Models.....	106
8.2.	Performance Metrics	108
8.3.	Effect of One Log File Per Node.....	111
8.4.	Effect of Message Type-based State Saving.....	113
8.5.	Experiments with Standard Time Warp Protocol.....	115
8.6.	Time Warp Optimizations	121
CHAPTER 9	CONCLUSIONS AND FUTURE WORK	126
9.1.	Future Work.....	128
REFERENCES	129

LIST OF TABLES

Table 1. Metrics for performance measurement	109
Table 2. Metrics for system profiling	109
Table 3. Execution results for the 35×35 fire model before and after PSS strategy on 4 nodes	124
Table 4. Execution results for the 35×35 fire model before and after lazy cancellation on 4 nodes	125
Table 5. Execution results for the 35×35 fire model before and after lazy cancellation on 8 nodes	125

LIST OF FIGURES

Figure 1. Layered architecture of the PCD++ optimistic simulator [Gli04].....	19
Figure 2. Major functionalities of the PCD++ and the WARPED layers	20
Figure 3. The clustering levels in the WARPED kernel	21
Figure 4. Runtime representation of a simulation object.....	30
Figure 5. Kernel operations for primary rollback.....	30
Figure 6. Kernel operations for secondary rollback (positive event already processed).....	31
Figure 7. Kernel operations for secondary rollback (positive event not yet processed).....	31
Figure 8. Tree structure of rollback propagation on a processor	32
Figure 9. Status of the queues during fossil collection	33
Figure 10. Model and processor hierarchies in PCD++.....	36
Figure 11. Example model and partition definition	37
Figure 12. Distributed processor structure for the example model.....	37
Figure 13. Simulator algorithm for (I, θ)	41
Figure 14. Simulator algorithm for $(@, t)$	41
Figure 15. Simulator algorithm for $(*, t)$	42
Figure 16. Simulator algorithm for (x, t)	42
Figure 17. FC algorithm for (I, θ)	43
Figure 18. FC algorithm for $(@, t)$	43
Figure 19. FC algorithm for (y, t)	44
Figure 20. FC algorithm for (x, t)	44
Figure 21. FC algorithm for $(*, t)$	45
Figure 22. FC algorithm for (D, t)	45
Figure 23. NC algorithm for (I, θ)	46
Figure 24. Simplified NC algorithm for (x, t)	47
Figure 25. NC algorithm for (y, t)	47
Figure 26. Simplified NC algorithm for (D, t)	48
Figure 27. Root algorithm for (y, t)	49
Figure 28. Example message-passing scenario.....	50

Figure 29. Terminating the simulation on a LP	55
Figure 30. Algorithm for function <i>rollbackProcessData</i>	57
Figure 31. Initialization algorithm in Cell-DEVS models with transport delay	59
Figure 32. Algorithm for the λ function in Cell-DEVS models with transport delay	60
Figure 33. Algorithm for the δ_{int} function in Cell-DEVS models with transport delay	60
Figure 34. Algorithm for the δ_{ext} function in Cell-DEVS models with transport delay	61
Figure 35. Initialization algorithm in Cell-DEVS models with inertial delay	63
Figure 36. Algorithm for the λ function in Cell-DEVS models with inertial delay	63
Figure 37. Algorithm for the δ_{int} function in Cell-DEVS models with inertial delay	64
Figure 38. Algorithm for the δ_{ext} function in Cell-DEVS models with inertial delay	64
Figure 39. WCTS representation for the simulation on a LP	68
Figure 40. Typical rollback scenario shown in terms of wall clock time slices	69
Figure 41. Example scenario for state changes of the NC during the simulation	70
Figure 42. Code snippet for entering dormant state in the NC algorithm for (D, t)	71
Figure 43. Enhanced NC algorithm for (x, t)	72
Figure 44. Rollback at virtual time 0	73
Figure 45. Using MPI Barrier to avoid rollbacks at virtual time 0	74
Figure 46. Code snippet for handling rollbacks at time 0 in the NC algorithm for (D, t)	74
Figure 47. UCSS structure with copy state-saving strategy	76
Figure 48. Enhanced kernel algorithm for executing events and saving states (UCSS)	76
Figure 49. Example scenario of messaging anomalies	78
Figure 50. Messaging anomaly with empty NC Message Bag	80
Figure 51. Messaging anomaly with non-empty NC Message Bag	81
Figure 52. NC algorithm for handling anomaly with empty NC Message Bag	82
Figure 53. NC status during anomalies with non-empty NC Message Bag	83
Figure 54. Restoring the event-pointer for undue external events	84
Figure 55. Example scenario for anomalies with non-empty NC Message Bag	86
Figure 56. Enhanced kernel algorithm for state restoration during rollbacks	88
Figure 57. NC algorithm for anomalies with non-empty NC Message Bag	90
Figure 58. Enhanced NC algorithm for (D, t)	92
Figure 59. Periodic state-saving strategy with a static interval of 2	97

Figure 60. Rollbacks with the periodic state-saving strategy	98
Figure 61. UCSS structure for hybrid state-saving strategy	99
Figure 62. Rollbacks under the hybrid state-saving strategy	100
Figure 63. Example scenario for the failure of coasting forward operation	102
Figure 64. Example scenario for fossil collections under the new scheme	102
Figure 65. Definition of the fire propagation model in CD++.....	107
Figure 66. Definition of the watershed model in CD++	108
Figure 67. Execution and bootstrap time before and after one log file per node strategy on 1 and 4 nodes	111
Figure 68. CPU usage before and after one log file per node strategy on 1 node	112
Figure 69. States saved and state-saving time before and after MTSS strategy on 1 and 4 nodes	113
Figure 70. Running and bootstrap time before and after MTSS strategy on 1 and 4 nodes	114
Figure 71. Average and maximum memory consumption before and after MTSS strategy	114
Figure 72. A simple partition strategy for Cell-DEVS models.....	115
Figure 73. Comparison between optimistic and conservative simulators using the fire model .	116
Figure 74. Total execution time for fire model of various sizes on a set of nodes	117
Figure 75. Running time for fire model of various sizes on a set of nodes	118
Figure 76. Overall and algorithm speedups for fire model of various sizes on a set of nodes ...	118
Figure 77. Total execution and running time for the 15×15×2 watershed model.....	119
Figure 78. Overall and algorithm speedups for the 15×15×2 watershed model.....	120
Figure 79. Total execution and running time for the 20×20×2 watershed model.....	120
Figure 80. Overall and algorithm speedups for the 20×20×2 watershed model (false).....	121
Figure 81. Number of rollbacks and anti-messages for the 35×35 fire model	122
Figure 82. Total execution and running time for the 35×35 fire model	122
Figure 83. Execution results for the 35×35 fire model before and after PSS strategy on 1 node	123

LIST OF ACRONYMS

CSS	Copy State Saving
DEVS	Discrete Event System Specification
FC	Flat Coordinator
GVT	Global Virtual Time
LP	Logical Process
LTSF	Least-Time-Stamp-First scheduling
LVT	Local Virtual Time
M&S	Modeling and Simulation
MPI	Message Passing Interface
MTSS	Message Type-based State Saving
NC	Node Coordinator
P-DEVS	Parallel Discrete Event System Specification
PSS	Periodic State Saving
UCSS	User Controlled State Saving
WCTS	Wall Clock Time Slice