Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Experiences in Data-Parallel Simulation and Analysis of Complex Systems with Irregular Graph Structures

A thesis presented in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy
in
Computer Science

at Massey University, Albany
New Zealand.

Arno Leist

2011

Abstract

The interactions between the components of many natural and artificial systems can be described using a graph. These graphs often have an irregular structure with non-trivial topological features. Complex system behaviour emerges on the macroscopic scale from a large number of relatively simple interactions on the microscopic scale. To better understand the observed behaviour of a complex system, the interactions among its basic elements are commonly described in a computational model. As long as the interactions are defined accurately and the number of elements is large enough for complex patterns to emerge, a simulation based on such a model is expected to produce the same behaviour as the system under investigation.

The difficulty is often to simulate the model on a large enough scale to obtain scientifically meaningful results. Powerful computer systems are required to calculate the effects caused by the interactions of large numbers of elements. Supercomputers that are constructed from hundreds of thousands of processing units can be used to update many components of the system in parallel and thus reduce the overall simulation time, but these systems are expensive to buy and maintain. As the processor architectures used in workstations and regular desktop computers are becoming more powerful, a small cluster constructed from these systems can be a more viable option. In recent years, the highly data parallel architecture of commodity graphics processing units (GPUs) has received a growing amount of attention due to their high peak compute throughput compared to central processing units (CPUs). New software development tools that turn the GPU hardware into a general purpose compute accelerator have become available.

This thesis describes how GPUs can be used to accelerate scientific simulations of complex systems that are based on irregular graph structures. New software development approaches and algorithms are needed to fully utilise the data parallel many-core architecture of today's GPUs. Irregular graph structures are particularly challenging, as the hardware is based on the single instruction, multiple data (SIMD) processor design, where a group of processing elements receives the same instructions. The architecture also imposes strict requirements on memory access patterns, making the optimisation of the memory layout for the irregular data structures and associated information particularly important. Performance suffers dramatically when the algorithm does not comply with these design restrictions.

The author proposes different software design strategies for a number of common graph problems and discusses the advantages and disadvantages of each approach. Two complex system models are used to demonstrate how the GPU can help to accelerate scientific simulations. The first model investigates how the phase transition from ordered to disordered system states in a computational ferromagnet is affected by distortions to the lattice substrate. The second model implements a large scale spiking neural network. The findings show that it is beneficial to utilise the GPU as accelerator to the CPU in almost all scenarios, as long as the project has a long enough run time to justify the more complex software development of the data parallel algorithms. When the model has some regularities in its structure or when some of the design decisions that influence the way memory is accessed can be made with the data parallel architecture in mind, then it is possible to achieve such high performance on the graphics device that it is best to leave the entire computational work to the GPU and use the CPU only to manage the execution of the program.

Acknowledgements

First and foremost I want to thank my parents, Helmut and Gisela, for their continuous support and encouragement. I also want to express my gratitude to my supervisors, Professor Ken Hawick and Associate Professor Chris Scogings, who provided guidance throughout this endeavour. Their experience and expertise have been invaluable to the success of this thesis.

I would like to acknowledge my peers, whose friendship has made the last few years so much more enjoyable and whose knowledge and advice have helped me many times: Dr Daniel Playne, Dr Anton Gerdelan, Dr Andrew Gilman, Dr Guy Kloss, Dr Andre Barczak, Dr Martin Johnson and all the other members of the Complex Systems and Simulations Group at Massey University. I also want to thank everyone who has helped to make me feel welcome and at home in New Zealand.

Finally, I would like to acknowledge the financial support from the Massey University Vice-Chancellor's Doctoral Scholarship and the New Zealand International Doctoral Research Scholarship.

Contents

List of Figures 11

List of Tables 15

I. Introduction 17

1. Overview and Aim of this Thesis 19

1.1. High Performance Computing . . . 19
1.2. The GPU as Compute Accelerator . . . 20
1.3. Thesis Structure . . . 21

2. Graph Structures 23

2.1. A Regular d-Dimensional Lattice . . . 23
2.2. Random Graphs . . . 24
2.3. Graph Complexity . . . 25
2.4. Scale-Free Graphs . . . 27
2.5. The Small-World Phenomenon . . . 28
2.6. Small-World Network Models . . . 30

3. Parallel Processing Architectures 33

3.1. Multi-Core CPU . . . 33
3.2. Cell Broadband Engine . . . 34
3.3. Graphics Processing Units . . . 35
3.3.1. The GPU Architecture . . . 36
3.3.2. The CUDA Programming Model . . . 38
3.4. Test Environment . . . 40

II. Parallel Graph Algorithms 41

4. Parallel Random Number Generation 43

4.1. The Sequential CPU Implementation . . . 44
4.2. The Multi-Threaded CPU Implementation . . . 45
4.3. The CUDA GPU Implementation . . . 46
4.4. Multi-Platform Lagged-Fibonacci Performance Results . . . 48

4.5. GPU Performance Results 50

5. Parallel Graph Generation 53

5.1. Small-World Graphs 54

5.1.1. The Sequential CPU Implementation 54

5.1.2. The Multi-Threaded CPU Implementation 55

5.1.3. The CUDA GPU Implementation 56

5.1.4. Performance Results 59

5.2. Scale-Free Graphs 62

5.2.1. The Sequential CPU Implementation 62

5.2.2. The Multi-Threaded CPU Implementation 63

5.2.3. The CUDA GPU Implementation 64

5.2.4. Performance Results 66

5.3. Memory Allocation Experiment 69

5.4. Discussion 70

6. Parallel Graph Analysis 73

6.1. Component Labelling 73

6.1.1. Data Structures 74

6.1.2. CPU Implementations - Sequential & TBB 74

6.1.3. The CUDA Implementations 76

6.1.4. Performance Results 81

6.2. Clustering Coefficient 86

6.2.1. CPU - Sequential, PThreads & TBB 87

6.2.2. GPU - CUDA 90

6.2.3. Cell Processor - PS3 93

6.2.4. Performance Results 97

6.3. Discussion 99

III. Complex Systems Simulation & Analysis 101

7. The Rewired Ising Model 103

7.1. Metropolis Updates 105

7.1.1. Generating the Small-World Lattice 105

7.1.2. Rewired Irregular Data Structure 106

7.1.3. The CUDA GPU Implementation 107

7.1.4. Sequential & Parallel CPU Implementations 110

7.1.5. Performance Results 111

7.2. Wolff Cluster Updates 113

7.2.1. Rewired Irregular Data Structure 114

7.2.2. The CUDA GPU Implementation 114

7.2.3. Performance Results 118

7.3. Visualisation of the Rewired Ising Model 122

7.3.1. Implementation 123

7.3.2. High Performance Simulation and Visualisation 123

7.4. Spectral Analysis 129

7.5. Ising Model Simulation . . . 132
7.5.1. Equilibration Phase . . . 135
7.5.2. Decorrelated Measurements . . . 135
7.6. Statistics Computation with CUDA . . . 139
7.7. Critical Temperatures . . . 140
7.7.1. Heat Capacity & Magnetic Susceptibility . . . 141
7.7.2. Binder Cumulant Results . . . 142
7.8. Discussion . . . 143

8. A Neural Network Model 147
8.1. The Model . . . 148
8.2. The Network & Data Structure . . . 149
8.3. CUDA Implementation of the Cortical Model . . . 152
8.4. Multi-GPU and Cluster Implementations . . . 160
8.5. Performance Results . . . 162
8.6. Processing the Pseudo-EEG Signal . . . 167
8.6.1. Sampling an Analog EEG Signal . . . 167
8.6.2. Digital Signal Processing . . . 167
8.7. Visual Analysis & Interactive Simulation Control . . . 169
8.8. Discussion . . . 173

IV. Discussion & Conclusions 175

9. Discussion 177
9.1. Parallel Processing Architectures . . . 177
9.2. Data Parallel Programming Models . . . 178
9.2.1. The Compute Unified Device Architecture . . . 178
9.2.2. The Open Computing Language . . . 180
9.2.3. Other Programming Models . . . 180

10. Conclusions & Future Work 183
10.1. The GPU as Compute Accelerator . . . 183
10.2. Irregular Graph Structures on a Data Parallel Architecture . . . 185
10.3. Opportunities for Future Work . . . 187

Appendices 191

A. Milgram's Small-World Experiment 191

B. The Small-World α-Model 193

C. Binder Cumulant Results: Continued 195

D. Different Types of Spiking Neurons 199

E. List of Publications 201

Bibliography 203

List of Figures

2.1. The simple cubic crystal structure. 24
2.2. A regular 2-dimensional lattice with periodic boundaries. 24
2.3. The characteristic path length $L(p)$ and clustering coefficient $C(p)$ for Watts' β-model. . . 30
2.4. Watts' small-world α-model. 31

3.1. A high-level view of a multi-core CPU architecture. 33
3.2. A high-level view of the architecture of the Cell Broadband Engine. 35
3.3. A high-level view of the "Fermi" GPU architecture. 36
3.4. A detailed view of a "Fermi" Streaming Multiprocessor. 37

4.1. Random number generator: Scaling the number of TBB tasks. 46

5.1. A "caveman" network generated with Watts' α-model. 54
5.2. Graph generation: Performance results for the α-model when scaling parameter α. 60
5.3. Graph generation: Performance results for the α-model when scaling the allocation size. I. 60
5.4. Graph generation: Performance results for the α-model when scaling the allocation size. II. 60
5.5. Graph generation: Performance results for the α-model when scaling the network size. . . 61
5.6. Graph generation: Performance results for the α-model when scaling the mean degree. . . 61
5.7. Graph generation: Performance results for the scale-free network model when scaling the allocation size. 67
5.8. Graph generation: Performance results for the scale-free network model when scaling the network size. 68
5.9. Graph generation: Performance results for the scale-free network model when scaling the mean degree. 68
5.10. Allocation size performance test. 70

6.1. Component labelling: The graph data structures for the CPU and CUDA algorithms. . . . 75
6.2. Component labelling: Performance when processing graphs of type Disjoint 1. 83
6.3. Component labelling: Performance when processing graphs of type Disjoint 2. 83
6.4. Component labelling: Performance when processing graphs of type Disjoint 3. 84
6.5. Component labelling: Performance when processing line graphs. 84
6.6. Component labelling: Performance when processing scale-free graphs. 84
6.7. Component labelling: Performance when processing complex graphs 1. 84
6.8. Component labelling: Performance when processing complex graphs 2. 85
6.9. Component labelling: Performance when processing random graphs. 85

6.10. Clustering coefficient: The data structure for the Cell BE implementation. 93
6.11. Clustering coefficient: The phases of the Cell BE implementation. 96
6.12. Clustering coefficient: Performance results for small-world graphs. 97
6.13. Clustering coefficient: Performance results for scale-free graphs. 97

7.1. Ising model: An example of a 2-dimensional Ising system. 104
7.2. Ising model: Rewiring procedure for the irregular lattice Ising model. 106
7.3. Ising model: The mapping of threads to cells used for the Metropolis algorithm. 106
7.4. Ising model: The vertex arrays used by the implementations of the Metropolis algorithm. . 107
7.5. Ising model: The arc arrays used by the implementations of the Metropolis algorithm. . . . 107
7.6. Ising model: Execution times for Metropolis updates with increasing rewiring probability. 111
7.7. Ising model: Execution times for Metropolis updates with increasing system size. 112
7.8. Ising model: Execution times for the Wolff algorithm with local updates in shared memory. 120
7.9. Ising model: Execution times for Wolff updates when approaching the equilibrium state. . 120
7.10. Ising model: Execution times for Wolff updates with increasing rewiring probability. . . . 121
7.11. Ising model: Execution times for Wolff updates with increasing system size. 122
7.12. Ising model visualisation: 2D regular and rewired lattice. 125
7.13. Ising model visualisation: Inside a 3D model. 126
7.14. Ising model visualisation: A rewired 3D model and interactive parameter changes. 126
7.15. Ising model visualisation: Wolff cluster updates. 127
7.16. Ising model analysis: Spherical mean of the 3D FFT. 130
7.17. Ising model analysis: Cluster size distribution of the 3D Ising system. 131
7.18. Ising model analysis: The FFT of a 2D system visualised. 131
7.19. Ising model analysis: Hyper-surfaces of the 3D FFT for a regular lattice. 133
7.20. Ising model analysis: Hyper-surfaces of the 3D FFT for a lattice rewired in all dimensions. 133
7.21. Ising model analysis: Hyper-surfaces of the 3D FFT for a lattice rewired in one dimension. 134
7.22. Ising model analysis: Hyper-surfaces of the 3D FFT for a lattice rewired in two dimensions. 134
7.23. Ising model simulation: The equilibration phase. 136
7.24. Ising model simulation: Histogram of the energy distribution. 137
7.25. Ising model simulation: Decorrelated measurements. 138
7.26. Ising model analysis: Heat capacity and magnetic susceptibility. 141
7.27. Ising model analysis: Binder cumulant results for systems with $p = 10^{-4}$. 142
7.28. Ising model: The shift in the critical temperature with respect to the rewiring probability p. 144

8.1. Cortical model: The mapping of threads to neurons. 151
8.2. Cortical model: The neighbour distribution. 151
8.3. Cortical model: The neighbours structure. 151
8.4. Cortical model: Action potentials from neighbouring neurons. 158
8.5. Cortical model: The distributed system model for multi-GPU processing. 160
8.6. Cortical model: A pseudo-EEG. 164
8.7. Cortical model: Execution times when scaling the system size. 164
8.8. Cortical model: Execution times when scaling the average degree. 164
8.9. Cortical model: Execution times when scaling the maximum neighbour distance. 166
8.10. Windowing functions for signal processing. 168
8.11. An interactive visual analysis tool for the cortical model. 170
8.12. A visualisation of the pseudo-EEG and individual neural firing rates. 172

C.1. Ising model analysis: Binder cumulant results for systems with $p = 10^{-7}$. 195

C.2. Ising model analysis: Binder cumulant results for systems with $p = 10^{-6}$. 196
C.3. Ising model analysis: Binder cumulant results for systems with $p = 10^{-5}$. 196
C.4. Ising model analysis: Binder cumulant results for systems with $p = 10^{-3}$. 197
C.5. Ising model analysis: Binder cumulant results for systems with $p = 10^{-2}$. 197

D.1. Cortical model: The different types of spiking neurons. 199

List of Tables

4.1. Random number generator: Performance comparison on different architectures. 50
4.2. Random number generator: Performance of different algorithms on the GPU. 51

5.1. Graph generation: Performance summary. 71

6.1. Component labelling: Summary of the component labelling algorithms. 81
6.2. Component labelling: The graph types used for the performance measurements. 82
6.3. Component labelling: The slopes of the least square linear fits to the data sets. 86
6.4. Component labelling: Performance summary. 86
6.5. Clustering coefficient: Performance summary. 98

7.1. Ising model: Performance summary for the Metropolis algorithm. 113
7.2. Ising model: Wolff and Metropolis update performance compared. 121
7.3. Ising model visualisation: Performance results for single and dual GPU simulations. 128
7.4. Ising model simulation: Overview of the system configurations. 140
7.5. Ising model analysis: The critical temperatures for $p = 10^{-2}$ to $p = 10^{-7}$. 143

8.1. Cortical model: The device memory requirements per neuron. 159
8.2. Cortical model: A list of the test systems used for the performance measurements. 165
8.3. Cortical model: The slopes of the least square linear fits to the data sets. 166
8.4. Cortical model: The range of frequency bands in the periodogram. 171