

**EMPIRICAL AND STATISTICAL APPLICATION MODELING
USING ON-CHIP PERFORMANCE MONITORS**

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Kirk W. Cameron
B.S., University of Florida, 1994
August 2000

©Copyright 2000

Kirk W. Cameron

All rights reserved

Dedication

This dissertation is dedicated to:

my beautiful and loving wife

Melissa.

You are my compass and my light, without which

this would have never been possible.

And to God for strength, reason, and a wonderful life.

Acknowledgements

In such an enormous, time-consuming endeavor, there are invariably many people to thank for their guidance, support, friendship, and wisdom. Prof. Xian-He Sun, my advisor, mentor and friend, saw ambition and drive in me I didn't know I had. Without his vision, intelligence, and compassion, I'd never have made it this far. While he could not convince me to move to Chicago, he did inspire me to at least attempt the academic life. Most of my academic success is in some way attributable to Xian-He for which I thank him.

We all gain from experience. For that I must thank the LSU faculty, the 82450NX Chipset Validation Team at Intel Corporation, and the PA Team at Los Alamos National Laboratory for invaluable real-world experiences that undoubtedly prepared me for my professional career. I am in particular debt to Yong Luo, my laboratory mentor and friend, for his rebuttals, thoughts, and contributions. I would also like to thank my doctoral committee members for their suggestions and support.

They say a man's stature and character can be measured by observing those he calls friends. If so, I can think of no higher complement than being able to call the following individuals my friends. Dr. Kasidit Chanchio has been there since the beginning, and continues to provide me with support and insight on just about everything. Thanks to Steve O'Neal, without whom I'd have never passed the general exams. Ken Shell and Kimberly Schneider, if there was a beginning to this endeavor, you were there before that even started - I wish you both great success and thank you for always believing in me. Dave and Kara Heckman, I treasure your friendship and hope that one day our kids will play together. To the "Italian mafia", Fabrizio, Mariella,

Allesandro and Federico, thanks for all the advice and especially the tiramisu and accompaniments. To my sister, Tiffany, all my love and one more thing: you're next.

To my family and all my wonderful in-laws, your unending support and encouragement have always inspired me and I am thankful for your thoughts and prayers. And to the unmentioned multitude of others that have helped in many, many ways; thank you.

Table of Contents

Dedication	iii
Acknowledgements	iv
List of Tables	ix
List of Figures.....	xi
Abstract.....	xiii
Chapter 1 Introduction	1
1.1 Performance Analysis	1
1.2 Measuring Performance.....	2
1.3 Our Approach.....	4
1.4 Motivation	5
1.5 Thesis.....	6
1.6 Organization	6
Chapter 2 Literature Review	8
2.1 Superscalar Architectures	8
2.2 Motivation	10
2.3 Our Analytical Approach	11
2.4 Historic Approaches to Modeling	14
2.5 Related Work.....	19
2.6 Summary	27
2.7 References.....	28
Chapter 3 Tools, Testbeds and Workloads	34
3.1 Measurements.....	34
3.2 Hardware Monitors.....	35
3.3 Current Tools.....	37
3.4 Common Problem Set	40
3.5 Testbeds.....	42
3.5.1 MIPS R10000 Machines	42
3.5.2 Intel Machines	45
3.5.3 ASCI Codes	46
3.5.4 Codes for Architectural Evaluation	48
3.6 Workload Characterization.....	49
3.6.1 CPI Characterization of ASCI Codes	49
3.6.2 Steady-State Characterization of ASCI Codes	52
3.6.3 Simulator Characterization of SPEC Codes	53
3.7 Summary	62
3.8 References.....	62

Chapter 4 Performance Analysis Methods	66
4.1 The General CPI Model	66
4.1.1 Empirical Memory Model.....	68
4.1.2 Validation of the Empirical Memory Model.....	71
4.1.3 Instruction-Level Model.....	74
4.1.4 Instruction-Level Model Validation.....	79
4.1.5 Extended Instruction-Level Model.....	82
4.2 Statistical Analysis Method	85
4.2.1 Background	86
4.2.2 Definitions.....	87
4.2.3 Four-Level Statistical Method for Evaluating Memory Systems.....	88
4.2.3.1 Level One Evaluation: Main Effect	88
4.2.3.2 Level Two Evaluation: Code/Machine Classification	89
4.2.3.3 Level Three Evaluation: Scalability Comparison	90
4.2.3.4 Level Four Evaluation: Memory Hierarchy	91
4.3 The Hybrid Method	91
4.3.1 The Hybrid Approach: Level 1.....	92
4.3.2 The Hybrid Approach: Level 2.....	94
4.4 Summary	95
4.5 References	96
Chapter 5 Application and Experience	98
5.1 SynBAD and PTERA	98
5.1.1 Overview of PTERA	98
5.1.2 The PTERA Prototype	99
5.1.2.1 User Interface Module	101
5.1.2.2 SynBAD and User Application Modules	103
5.1.2.3 Performance Monitor Module.....	103
5.1.2.4 PTERAnalyzer Module.....	104
5.1.3 RISC and CISC Implementation of PTERA.....	104
5.2 Empirical Memory Model	106
5.2.1 Methodology	106
5.2.2 Analysis of Stall Time Due to Memory Accesses	107
5.3 Instruction-Level Model	111
5.3.1 Bottleneck Analysis of MIPS R10000.....	111
5.4 Dependence Analysis of MIPS R10000	115
5.4.1 Methodology	115
5.4.2 Ideal Experiments for Dependence Analysis.....	116
5.5 Architecture Advances via Modeling	123
5.5.1 Mutable Functional Unit (MFU)	123
5.5.2 Alternative Architectures.....	126
5.5.3 Comparisons of the Alternative Architectures	130
5.5.4 Results	131
5.6 Memory Hierarchy Evaluation Using the Statistical Method	136
5.6.1 Main and Interaction Effects.....	137
5.6.2 Code and Machine Classification	140

5.6.3	Scalability Comparison	142
5.6.4	Evaluation of Memory Components	144
5.7	Hybrid Model Analysis	147
5.7.1	Level 1 Results.....	148
5.7.2	Level 2 Results.....	149
5.8	Summary	152
Chapter 6 Conclusions.....		153
6.1	Overall Summary	154
6.2	Scientific Contributions	155
6.3	Future Work	156
Appendix: Performance Monitor Survey		158
Vita		162

List of Tables

Table 3.1 Common problem instruction set	41
Table 4.1 Accuracy of the validation method.....	73
Table 4.2 Validation of t_2 and t_m for the memory model.....	73
Table 4.3 Results for synthetic instruction streams on MIPS R10000.....	80
Table 4.4 Results for ideal synthetic instruction streams on MIPS R10000	81
Table 5.1 Model parameters for PowerChallenge	107
Table 5.2 Memory access times for PC, O2K, and Intel ASCI Red.....	108
Table 5.3 Branch and icache characteristics for measured codes.....	112
Table 5.4 Utilization factors for the measured codes	113
Table 5.5 Ideal cpi_0 calculated using Equation 4.12	114
Table 5.6 MFU mutation penalty	125
Table 5.7 Architecture based on time of analysis and mutation	127
Table 5.8 Mutation frequency.....	134
Table 5.9 Percent of time RS-MFU is full.....	136
Table 5.10 Class level information	137
Table 5.11 Mean effects table.....	138
Table 5.12 Contrast method for pairwise comparison	140
Table 5.13 LSD post hoc comparison for code.....	141
Table 5.14 LSD post hoc comparisons for machines	141
Table 5.15 Scalability comparison of HEAT	142
Table 5.16 Scalability comparison of Sweep	144
Table 5.17 L1 hit ratio comparison for HEAT.....	145

Table 5.18 L2 hit ratio comparison for HEAT.....	145
Table 5.19 L2 hit ratio comparison for HYDRO	146

List of Figures

Figure 3.1 Topology of SGI Origin 2000 at Los Alamos	43
Figure 3.2 Performance of HEAT as a function of linear problem size	51
Figure 3.3 Performance of SWEEP as a function of linear problem size.....	51
Figure 3.4 Performance of HYDRO as a function of linear problem size.....	51
Figure 3.5 Performance of HYDROT as a function of linear problem size	52
Figure 3.6 Performance of NEUT as a function of linear problem size.....	52
Figure 3.7 Instruction distances for Heat on O2K and PC.....	54
Figure 3.8 Instruction distances for Sweep on O2K and PC	54
Figure 3.9 Instruction distances for Dsweep on O2K and PC	54
Figure 3.10 Instruction distances for Hydro on O2K and PC	55
Figure 3.11 Instruction distances for Hydro-t on O2K and PC	55
Figure 3.12 Instruction distance occurrences for Swim.....	58
Figure 3.13 Instruction distance occurrences for Wave5.....	58
Figure 3.14 Instruction distance occurrences for Su2cor	59
Figure 3.15 Instruction distance occurrences for Compress95	59
Figure 3.16 Instruction distance occurrences for ijpeg.....	60
Figure 3.17 Instruction distance occurrences for li	60
Figure 3.18 Instruction distance occurrences for kmeans	61
Figure 4.1 Memory model times.....	69
Figure 4.2 Latency effect on model	69
Figure 5.1 PTERA modules	100
Figure 5.2 MIPS R10000 SynBAD specification and resulting assemble	102

Figure 5.3 Memory stall and overlap parameters (PowerChallenge).....	109
Figure 5.4 Memory stall and overlap parameters (Origin 2000)	109
Figure 5.5 Memory stall and overlap parameters (Intel ASCI Red)	110
Figure 5.6 Performance variation in relation to link-length and link-width	119
Figure 5.7 Performance variation in relation to link-length and link-width	121
Figure 5.8 Performance variation in relation to link-length and link-width	122
Figure 5.9 MIPS R10000 functional unit and reservation station.....	126
Figure 5.10 RS-MFU modification to MIPS R10000.....	129
Figure 5.11 IPC of various architecture schemes	132
Figure 5.12 IPC results for varying number of entries in RS-MFU	135
Figure 5.13 Machine mean distribution.....	139
Figure 5.14 Code mean distribution.....	139
Figure 5.15 m_0 values calculated on the Origin 2000.....	150
Figure 5.16 m_0 values calculated on the PowerChallenge.....	151

Abstract

To analyze the performance of applications and architectures, both programmers and architects desire formal methods to explain anomalous behavior. To this end, we present various methods that utilize non-intrusive, performance-monitoring hardware only recently available on microprocessors to provide further explanations of observed behavior. All the methods attempt to characterize and explain the instruction-level parallelism achieved by codes on different architectures. We also present a prototype tool automating the analysis process to exploit the advantages of the empirical and statistical methods proposed. The empirical, statistical and hybrid methods are discussed and explained with case study results provided. The given methods further the wealth of tools available to programmer's and architects for generally understanding the performance of scientific applications.

Specifically, the models and tools presented provide new methods for evaluating and categorizing application performance. The empirical memory model serves to quantify the hierarchical memory performance of applications by inferring the incurred latencies of codes after the effect of latency hiding techniques are realized. The instruction-level model and its extensions model on-chip performance analytically giving insight into inherent performance bottlenecks in superscalar architectures. The statistical model and its hybrid extension provide other methods of categorizing codes via their statistical variations. The PTERA performance tool automates the use of performance counters for use by these methods across platforms making the modeling process easier still. These unique methods provide alternatives to performance modeling and categorizing not available previously in an attempt to utilize the inherent

modeling capabilities of performance monitors on commodity processors for scientific applications.