

ARCHITECTURAL SUPPORT FOR ENHANCED PERFORMANCE OF OS INTENSIVE APPLICATIONS

PRATHMESH KALLURKAR



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI
JANUARY 2018

©Indian Institute of Technology Delhi - 2018
All rights reserved.

ARCHITECTURAL SUPPORT FOR ENHANCED PERFORMANCE OF OS INTENSIVE APPLICATIONS

by

PRATHMESH KALLURKAR

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of Doctor of Philosophy

to the



Indian Institute of Technology Delhi

JANUARY 2018

Certificate

This is to certify that the thesis titled **ARCHITECTURAL SUPPORT FOR ENHANCED PERFORMANCE OF OS INTENSIVE APPLICATIONS** being submitted by **Mr. PRATHMESH KALLURKAR** for the award of **Doctor of Philosophy** in Computer Science and Engineering is a record of bona fide work carried out by him under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Delhi. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Smruti Ranjan Sarangi

Associate Professor

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

New Delhi- 110016

Acknowledgements

Firstly, I would like to express my sincere gratitude towards my supervisor Dr. Smruti Sarangi for his continuous support and guidance during the entire duration of my PhD. He encouraged me to take a plunge in this research area around seven years back, and I must admit that this was one of the best decision I have made till now. His valuable suggestions helped me improve the quality of research. He also spent a significant amount of time in improving my presentation and writing skills. Thank you Sir.

I thank Dr. Anshul Kumar, Dr. Preeti Ranjan Panda, and Dr. G.S. Visweswaran for providing me valuable feedback and insights on my research work. I would also like to thank Dr. Sorav Bansal for teaching me Operating systems and Virtualization.

I thank all my friends at IIT Delhi. Your constant support and encouragement ensured that I could navigate the highs and lows of life with relative ease. The experiences that we shared were thoroughly enjoyable; I will cherish them forever.

I also thank my parents and my sister for all the constant support and guidance during my PhD. Thank you Aai, Baba, and Prajka.

Prathmesh Kallurkar

Abstract

The OS has a large impact on the performance of important workloads such as web servers, database servers, and file servers. We observe that such workloads spend a considerable portion (20-95%) of their time executing different OS tasks such as interrupt handlers, system call handlers, and the OS scheduler; hence, we call them as OS-intensive workloads. We start by proposing a full-system simulation framework to do an in-depth study of such full-system workloads. Using the developed simulation framework, we show that the instruction footprint of such workloads is typically much larger than the size of instruction caches (16-32 KB). We also bring out interesting insights regarding the execution of such workloads. Based on the detailed insights regarding such workloads, we propose three novel techniques to improve the performance of OS-intensive workloads: (1) an interference-aware instruction cache, (2) an instruction prefetcher that is optimized for OS-intensive applications, and (3) a task scheduler that is optimized for OS-intensive applications.

सार

वेब सर्वर, डेटाबेस सर्वर, और फ़ाइल सर्वर जैसे महत्वपूर्ण वर्कलोड के प्रदर्शन पर ओएस का बड़ा प्रभाव पड़ता है। हम देखते हैं कि ऐसे वर्कलोड्स अलग-अलग ओएस कार्यों जैसे इंटरप्ट हैंडलर, सिस्टम कॉल हैंडलर और ओएस शेड्यूलर को कार्यान्वयन करने में अपने समय का एक बड़ा हिस्सा (बीस से पचानवे प्रतिशत) खर्च करते हैं; इसलिए, हम उन्हें ओएस-गहन वर्कलोड कहते हैं। हम इस तरह के पूर्ण-सिस्टम वर्कलोड के गहन अध्ययन करने के लिए एक पूर्ण-प्रणाली सिमुलेशन फ्रेमवर्क का प्रस्ताव देकर शुरू करते हैं। विकसित सिमुलेशन फ्रेमवर्क का उपयोग करके, हम दिखाते हैं कि ऐसे वर्कलोड का निर्देश पदचिह्न आमतौर पर निर्देश कैश (सोलह से बत्तीस किलो बाइट) के आकार से काफी बड़ा होता है। हम ऐसे वर्कलोड के कार्यान्वयन के संबंध में दिलचस्प अंतर्दृष्टि भी लाते हैं। ऐसे वर्कलोड के बारे में विस्तृत अंतर्दृष्टि पर आधारित, हम ओएस-गहन वर्कलोड के प्रदर्शन को बेहतर बनाने के लिए तीन उपन्यास तकनीकों का प्रस्ताव देते हैं (१) एक हस्तक्षेप-जागरूक निर्देश कैश, (२) एक निर्देश प्रीफेचर जिसे ओएस-गहन अनुप्रयोगों के लिए अनुकूलित किया गया है, और (३) एक कार्य शेड्यूलर जिसे ओएस-गहन अनुप्रयोगों के लिए अनुकूलित किया गया है।

Contents

Certificate	i
Acknowledgements	iii
Abstract	v
1 Introduction	1
2 Infrastructure Development: Full System Simulation Framework	5
2.0.1 Contribution Details	6
2.1 QemuTrace	7
2.1.1 Introduction	7
2.1.2 Information Collected	7
2.1.3 Traces	11
2.2 Tejas: A Java based Versatile Micro-architectural Simulator	17
2.2.1 Emulator Simulator Interface	17
2.2.2 Translation Engine	18
2.2.3 Cache Design	21
2.2.4 Coherence Protocol	22

3	Novel Cache Designs: Advanced OS Cache and Triangle Cache	25
3.1	Introduction	25
3.2	Design of the Advanced OS Cache	27
3.2.1	High Level Design	27
3.3	Design of the Triangle Cache	30
3.4	Benchmark Characterization	32
3.4.1	Instruction Mix	34
3.5	Results	36
3.6	Conclusion	48
4	pTask: A Smart Instruction Prefetching Scheme for OS Intensive Application	49
4.1	Introduction	49
4.2	Related Work	51
4.2.1	Mitigating Application/OS Interference	51
4.2.2	Compiler Optimizations	51
4.2.3	Instruction Prefetching	51
4.3	Definition of a HyperTask	53
4.4	Characterization	56
4.4.1	Instruction Mix in HyperTasks	56
4.4.2	I-cache Hit Rates/Evictions	57
4.4.3	Prefetching HyperTasks	59
4.4.4	Detailed Characterization of HyperTasks	61
4.4.5	Prefetching D-cache Lines	64
4.5	Prefetching HyperTasks	66

4.5.1	Profile Mode	67
4.5.2	Normal Mode	74
4.5.3	Summary	74
4.6	Results	78
4.6.1	Comparison: Performance Improvement	78
4.6.2	Overheads: Profiling and Prefetching	82
4.6.3	Prefetch store	83
4.6.4	pTask: Performance Breakup	85
4.6.5	Granularity: Basic block vs Function	85
4.6.6	HyperTask Size	85
4.6.7	pTask: Using Extra Hardware	88
4.6.8	Multi-programmed Workloads	88
4.7	Conclusion	89
5	SchedTask: A Hardware-Assisted Task Scheduler	91
5.1	Introduction	91
5.2	Related Work	93
5.2.1	Core Specialization	93
5.2.2	Architectural Support	94
5.3	SuperFunction	95
5.3.1	Type of Task	96
5.3.2	Similarity Between Different Types of Tasks	98
5.3.3	Structure Associated with a SuperFunction	99
5.4	Benchmark Characterization	100

5.4.1	Experimental Setup	100
5.4.2	Instruction Breakup	100
5.4.3	Instruction Breakup: Similarity across Epochs	102
5.5	SchedTask	103
5.5.1	The Timeline of a Thread's Execution	103
5.5.2	TAlloc	106
5.5.3	TMigrate	106
5.5.4	Modifications	109
5.6	Results	109
5.6.1	Comparison: Performance Improvement	109
5.6.2	Thread Migrations	120
5.6.3	Impact of the Workload on Performance	121
5.6.4	Impact of Work Stealing	122
5.6.5	Impact of the Page-heatmap Register	123
5.7	Multi-programmed Workloads	124
5.8	Instruction Cache Size	127
5.9	Cache Configuration	129
5.10	Number of Cores	131
5.11	Instruction Prefetcher	131
5.12	Trace Cache	133
5.13	Conclusion	135

CONTENTS

xi

Bibliography

139

List of Publications

145

Biography

147

List of Figures

2.1	Full-system simulation framework	5
2.2	Translation in Tejas	19
2.3	High level working of a cache (simplified)	21
3.1	Design of the Advanced OS Cache	27
3.2	An entry of the set-array in the application sub-cache	31
3.3	Search order of sub-caches	31
3.4	Instruction mix: non-virtualized execution environment	35
3.5	Instruction mix: virtualized execution environment	35
3.6	Impact of cache design on the instruction throughput of OS-intensive workloads in a non-virtualized environment	38
3.7	Impact of cache design on the fraction of inter-code evictions in the instruction cache for OS-intensive workloads in a non-virtualized environment	39
3.8	Impact of cache design on the i-cache hit rate (OS) for OS-intensive workloads in a non-virtualized environment	39
3.9	Impact of cache design on the i-cache hit rate (application) for OS-intensive workloads in a non-virtualized environment	40
3.10	Impact of different <i>Triangle cache</i> configurations on the instruction throughput of OS-intensive workloads in a virtualized environment	41

3.11 Impact of cache design on the instruction throughput of OS-intensive workloads in a virtualized environment	43
3.12 Impact of cache design on the fraction of inter-code evictions in the instruction cache for OS-intensive workloads in a virtualized environment	44
3.13 Impact of cache design on the i-cache hit rate (Hypervisor) for OS-intensive workloads in a virtualized environment	45
3.14 Impact of cache design on the i-cache hit rate (OS) for OS-intensive workloads in a virtualized environment	46
3.15 Impact of cache design on the i-cache hit rate (application) for OS-intensive workloads in a virtualized environment	47
4.1 Impact of task switches on the i-cache hit rate (Apache web server: representative execution)	50
4.2 Decomposition of the <i>Apache</i> benchmark's execution into HyperTasks	55
4.3 Instruction mix in HyperTasks	57
4.4 I-cache hit rate and breakup of evictions	58
4.5 Coverage and Utility	59
4.6 Jaccard distance vs #runs	61
4.7 Efficacy of prefetch list for heap accesses	65
4.8 <i>pTask</i> : Execution overview	67
4.9 Data structures used by <i>pTask</i>	73
4.10 Miss Rate Vector (conceptual view)	73
4.11 Performance impact	79
4.12 Breakup: i-cache accesses	79
4.13 Breakup: prefetch operations	80
4.14 Impact of profiling and prefetching on the hit rate of the d-cache	83

4.15 Breakup: performance improvement	84
4.16 Impact of prefetching first N unique functions of a HyperTask	84
4.17 $pTask$ granularity: basic-block vs function	86
4.18 Performance Impact: h/w support	86
4.19 Performance impact: multi-programmed workloads	87
5.1 An overview of the <i>SchedTask</i> technique	92
5.2 Decomposition of the system execution (Apache)	95
5.3 Quantifying the similarity between two superFuncTypes	99
5.4 Instruction breakup	101
5.5 Timeline of a thread's execution	103
5.6 <i>SchedTask</i> : High level design	105
5.7 Impact of core specialization techniques on application's performance	111
5.8 Impact of core specialization techniques on the instruction throughput	111
5.9 Impact of core specialization techniques on core idleness	112
5.10 Impact of core specialization techniques on i-cache hit rate of the application code	112
5.11 Impact of core specialization techniques on i-cache hit rate of the OS code	113
5.12 Impact of core specialization techniques on d-cache hit rate of the application code	113
5.13 Impact of core specialization techniques on d-cache hit rate of the OS code	114
5.14 Impact of different work stealing strategies on instruction throughput	118
5.15 Impact of different work stealing strategies on core idleness	119
5.16 Impact of different work stealing strategies on the i-cache hit rate	119

5.17	Number of inter-core thread migrations	121
5.18	Impact of the size of the <i>Page-heatmap register</i> on the quality of its ranking . .	123
5.19	Impact of different techniques on the instruction throughput of a system executing multi-programmed workloads	125
5.20	Impact of the instruction prefetcher on the instruction throughput	132
5.21	Impact of the trace cache on the instruction throughput	134

List of Tables

2.1	Illustration: x86 instruction → Packets	14
2.2	Virtual machine configuration	15
2.3	VISA instruction set	18
2.4	Operand types	19
2.5	Example scenarios requiring <i>corrective operations</i>	23
3.1	Advanced OS Cache controller logic	29
3.2	Details of the baseline system	33
3.3	Hardware support: Instruction cache design for non-virtualized environment . .	36
3.4	Details of different <i>Triangle cache</i> configurations	37
3.5	Hardware support: Instruction cache design for virtualized environment	41
4.1	Events related to HyperTasks	53
4.2	HyperTasks: detailed characterization	62
4.3	Hardware support: Prefetching techniques	78
4.4	Details: profiling and prefetching	81
4.5	Size of the prefetch store (KB)	83
4.6	Multi-programmed Workloads	89

5.1	Category and subcategory of <i>SuperFunctions</i>	97
5.2	Configuration of all core specialization techniques	109
5.3	Impact of the workload on the instruction throughput and idle time fractions . .	116
5.4	Constituent benchmarks of multi-programmed workloads	124
5.5	Impact of the size of the instruction cache on the instruction cache hit rate and instruction throughput	126
5.6	Impact of the cache configuration on the instruction throughput	128
5.7	Impact of the number of cores on the instruction throughput	130