TIME-BASED FAIRNESS-AWARE MEMORY SCHEDULING FOR MULTICORE PROCESSORS

By

Amr Saleh AboBakr Khalil Elhelw

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY

GIZA, EGYPT

2015

TIME-BASED FAIRNESS-AWARE MEMORY SCHEDULING FOR MULTICORE PROCESSORS

By

Amr Saleh AboBakr Khalil Elhelw

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Assoc. Prof. Hossam A. H. Fahmy
Electronics and Communications
Faculty of Engineering, Cairo University

Assist. Prof. Ali A. El-Moursy
Computer and Systems
Electronics Research Institute

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2015

TIME-BASED FAIRNESS-AWARE MEMORY SCHEDULING FOR MULTICORE PROCESSORS

By

Amr Saleh AboBakr Khalil Elhelw

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE
in

Electronics and Communications Engineering

Approved by the

Examining Committee

____

Associate Prof. Hossam A. H. Fahmy, Thesis advisor

____

Prof. Amin Mohamed Nassar, Internal member

____

Prof. Elsayed Mostafa Saad, External member

(Professor at Faculty of Engineering, Helwan University)

FACULTY OF ENGINEERING, CAIRO UNIVERSITY

GIZA, EGYPT

2015

Engineer: Amr Saleh Abobakr Khalil Elhelw

Date of Birth: 22/6/1986

Nationality: Egyptian

E-mail: amrkhalil4@hotmail.com

Phone: 01111372223

Address: Mohamed Elsharawy street, behind Mena Palace Hotel, Haram, Giza

Registration Date: 1/10/2010

Awarding:

Degree: Master of Science

Department: Electronics and Communications Engineering

Supervisors:

Associate Prof. Hossam A. H. Fahmy

Dr. Ali A. El-Moursy (Researcher at Electronics Research Institute)

Examiners:

Associate Prof. Hossam A. H. Fahmy (Thesis advisor)

Prof. Amin Mohamed Nassar (Internal examiner)

Prof. Elsayed Mostafa Saad (External examiner)

Title of Thesis:

Time-Based Fairness-Aware Memory Scheduling for Multi-core Processors

Keywords:

Multi-core; Memory Controller; Shared Resources; Memory Interference

Summary:

In the modern chip-multiprocessor system, concurrently executing applications/threads shares common resource such as main memory. Memory scheduling algorithms are developed to resolve memory contention between competing applications/threads so that throughput is high and fairness of the overall multi-core systems is guaranteed. Time-based Least Memory Intensive (TB-LMI) scheduling algorithm is a new memory scheduling algorithm introduced to improve multi-core processor's throughput and fairness.

Acknowledgements

First of all I must thank ALLAH for his great mercy supporting me all the way till the end. If it weren’t for his help, I wouldn’t have reached this point.

I would like to thank my advisors, Dr. Ali El-Moursy, and Prof. Hossam A. H. Fahmy for giving me the opportunity to work in a fruitful research environment and for their continuous guidance and support, as well as for their successful discussions and encouragements.

Most importantly, I would like to thank my parents and my brother for their continuous encouragement, and my colleges in Egyptian Financial Supervisory Authority (EFSA) for their support.

To my parents

Table of Contents

ACKNOWLEDGMENT ii
TABLE OF CONTENTS ii
LIST OF TABLES iv
LIST OF FIGURES v
LIST OF PUBLICATIONS vii
LIST OF ABBREVIATIONS viii
ABSTRACT ix
CHAPTER 1 : INTRODUCTION 1
1.1 SIMULTANEOUS MULTITHREADING 1
1.2 SYMMETRIC MULTIPROCESSING 2
1.3 MULTI-CORE PROCESSORS 2
1.4 SMT/MULTI-CORE THROUGHPUT ENHANCEMENT TECHNIQUES 4
1.5 THESIS CONTRIBUTION 6
1.6 THESIS OUTLINE 7
CHAPTER 2 : MEMORY ACCESS SCHEDULING IN LITERATURE 8
2.1 MAIN MEMORY OPERATION MECHANISM 8
2.2 SCHEMES FOR SINGLE THREADED SINGLE CORE PROCESSORS 10
2.3 SCHEMES FOR SMT AND MULTI-CORE PROCESSORS 11
2.4 SUMMARY 20
CHAPTER 3 : TIME BASED LEAST MEMORY INTENSIVE (TB-LMI) 22
3.1 MOTIVATION 22
3.2 TB-LMI OVERVIEW 23
3.3 TB-LMI SCHEDULING ALGORITHM 23
3.4 IMPLEMENTATION AND HARDWARE COST 28
3.5 TB-LMI OPERATION AND REQUEST SCENARIO 29
CHAPTER 4 : SIMULATOR 34
4.1 OVERVIEW 34
4.2 MULTI2SIM SIMULATOR 35
CHAPTER 5 : EVALUATION METRICS AND WORKLOADS 38
5.1 METRICS 38
5.2 SIMULATION ENVIRONMENT 40
5.3 WORKLOADS 41
CHAPTER 6 : RESULTS 55

6.1 SENSITIVITY ANALYSIS 55
6.2 MAIN MEMORY EFFECTIVE LATENCY 57
6.3 PERFORMANCE ANALYSIS 60
6.3.1 4-CORE 60
6.3.2 8-CORE 68
CHAPTER 7 : CONCLUSION AND FUTURE WORK 78
7.1 CONCLUSION 78
7.2 FUTURE WORK 78
REFERENCES 80
APPENDIX A : BENCHMARKS 84
A.1 PARSEC 84
A.2 MEDIABENCH 84
A.3 SPEC CPU 2006 84

List of Tables

Table 3.1: Hardware required for each bank memory controller 29
Table 3.2: Hardware required for Meta memory controller 29
Table 5.1: CPU specifications 40
Table 5.2: L1 specification 40
Table 5.3: L2 specification 40
Table 5.4: DRAM chip parameters... 41
Table 5.5: 8-core system workloads 42
Table 5.6: 4-core system workloads 42
Table 5.7: 4-core system workloads memory characteristics........................ 44
Table 5.8: 8-core system workloads memory characteristics........................ 44
Table 5.9: 8mem1 detailed histogram for memory bank 0 \ unlimited memory bank queue 45
Table 5.10: Memory characteristics \ 8 entries bank queue 52
Table 5.11: Memory characteristics \ 16 entries bank queue 52
Table 5.12: Memory characteristics \ 24 entries bank queue 52
Table 5.13: Memory characteristics \ 32 entries bank queue 54
Table 6.1: Main memory effective latency characteristics............................ 59
Table 6.2: Detailed IPC of 4mem2 in case of TB-LMI and FR-LREQ 61
Table A.1: PARSEC Benchmarks 85
Table A.2: Mediabench Benchmarks 85
Table A.3: SPECint CPU2006 Benchmarks.. 86
Table A.4: SPECfp CPU2006 Benchmarks... 87
Table A.5: Single core processor parameters 88
Table A.6: Benchmarks MPKI 89

List of Figures

Figure 1.1: SMT architecture 2
Figure 1.2: SMP architecture 3
Figure 1.3: Multi-core processors 4
Figure 1.4: Multiple cache copies 5
Figure 1.5: MSHR entry 6
Figure 2.1: Memory bank 9
Figure 2.2: DRAM FSM 9
Figure 2.3: FR-LREQ 14
Figure 2.4: FLRMR 15
Figure 2.5: Modified_ROB 16
Figure 2.6: FIQMR 17
Figure 3.1: TB-LMI in case warm-up cycles 25
Figure 3.2: TB-LMI in case SQ cycles 26
Figure 3.3: Threads priorities calculation in Meta memory controller 27
Figure 3.4: End of warm-up cycle 31
Figure 3.5: Bank memory controllers during SQ cycles 32
Figure 3.6: Bank memory controllers at the end of SQ cycles 33
Figure 5.1: 4-core workloads histogram \ unlimited memory queue 43
Figure 5.2: 8-core workloads histogram \ unlimited memory queue 43
Figure 5.3: 8mem1 histogram \ 8 entries bank queue 46
Figure 5.4: 8mix1 histogram \ 8 entries bank queue 47
Figure 5.5: 8mix2 histogram \ 8 entries bank queue 47
Figure 5.6: 8mem1 histogram \ 16 entries bank queue 48
Figure 5.7: 8mix1 histogram \ 16 entries bank queue 49
Figure 5.8: 8mix2 histogram \ 16 entries bank queue 49
Figure 5.9: 8mem1 histogram \ 24 entries bank queue 50
Figure 5.10: 8mix1 histogram \ 24 entries bank queue 51
Figure 5.11: 8mix2 histogram \ 24 entries bank queue 51
Figure 5.12: 8mem1 histogram \ 32 entries bank queue 52
Figure 5.13: 8mix1 histogram \ 32 entries bank queue 53
Figure 5.14: 8mix2 histogram \ 32 entries bank queue 53
Figure 6.1: Weighted speedup versus Maximum slowdown 8-core \ 8 entries memory bank queue (SQ sensitivity analysis) 55
Figure 6.2: Weighted speedup versus Maximum slowdown 8-core \ 8 entries memory bank queue (FR sensitivity analysis) 56
Figure 6.3: Weighted speedup versus Maximum slowdown 8-core \ 32 entries memory bank queue (FR sensitivity analysis) 57
Figure 6.4: Main Memory Effective Latency Histogram for 8mem1 58
Figure 6.5: Main Memory Effective Latency Histogram for 8mix1 58
Figure 6.6: Main Memory Effective Latency Histogram for 8mix2 59

Figure 6.7: Weighted speedup 4-core \ 8 entries memory bank queue 61
Figure 6.8: ANTT 4-core \ 8 entries memory bank queue 62
Figure 6.9: Maximum Slowdown 4-core \ 8 entries memory bank queue 62
Figure 6.10: Weighted speedup versus Maximum slowdown for 4-core \ 8 entries memory bank queue.. 63
Figure 6.11: Weighted speedup 4-core \ 16 entries memory bank queue .... .64
Figure 6.12: ANTT 4-core \ 16 entries memory bank queue 64
Figure 6.13: Maximum Slowdown 4-core \ 16 entries memory bank queue 65
Figure 6.14: Weighted speedup 4-core \ 24 entries memory bank queue .... .65
Figure 6.15: ANTT 4-core \ 24 entries memory bank queue 66
Figure 6.16: Maximum slowdown 4-core \ 24 entries memory bank queue...... 66
Figure 6.17: Weighted speedup 4-core \ 32 entries memory bank queue .... .67
Figure 6.18: ANTT 4-core \ 32 entries memory bank queue 67
Figure 6.19: Maximum slowdown 4-core \ 32 entries memory bank queue...... 68
Figure 6.20: Weighted speedup 8-core \ 8 entries memory bank queue 69
Figure 6.21: ANTT 8-core \ 8 entries memory bank queue 69
Figure 6.22: Maximum slowdown 8-core \ 8 entries memory bank queue........ 70
Figure 6.23: Weighted speedup 8-core \ 16 entries memory bank queue 71
Figure 6.24: ANTT 8-core \ 16 entries memory bank queue 71
Figure 6.25: Maximum slowdown 8-core \ 16 entries memory bank queue...... 72
Figure 6.26: Weighted speedup 8-core \ 24 entries memory bank queue 72
Figure 6.27: ANTT 8-core \ 24 entries memory bank queue 73
Figure 6.28: Maximum slowdown 8-core \ 24 entries memory bank queue...... 73
Figure 6.29: Weighted speedup 8-core \ 32 entries memory bank queue 74
Figure 6.30: ANTT 8-core \ 32 entries memory bank queue 74
Figure 6.31: Maximum slowdown 8-core \ 32 entries memory bank queue...... 75
Figure 6.32: Average Weighted speedup\different memory bank queue sizes...76
Figure 6.33: Average ANTT \ different memory bank queue sizes.. 77
Figure 6.34: Average Maximum slowdown \ different memory bank queue sizes. 77

List of Publications

Amr Elhelw, Ali A. El-Moursy, and Hossam A. H. Fahmy. Time-based least memory intensive scheduling. *In The 8th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-14), Aizu-Wakamatsu, Japan,* September 2014.

List of Abbreviations

ATLAS	Adaptive per-Thread Least-Attained-Service
BW	Bandwidth
CMP	Chip-level Multiprocessing
CPU	Central Processing Unit
DRAM	Dymanic Random Access Memory
FCFS	First Come First Serve
FIQMR	Fair Issue Queue Most Related
FLRMR	Fair Least Request Most Related
FR	First Ready
FR-FCFS	First-Ready First Come First Serve
FR-LREQ	First-Ready Least REQuest
FSM	Finite State Machine
Id	Identification number
ILP	Instruction Level Parallelism
IPC	Instruction Per Cycle
IQ-based	Issue Queue based
LAS	Least-Attained-Service
LREQ	Least Request
ME	Memory Efficiency
ME-LREQ	Memory Efficiency with Least REQuest
Modified-ROB_PF	Modified Reorder buffer Prioritization Factor
MPKI	Misses Per Kilo Instructions
MSHR	Miss Status Holding Register
MSU	Memory Scheduling Unit
OS	Operating System
PAR-BS	Parallelism Aware Batch Scheduling
RAM	Random Access Memory
ROB-based	Reorder Buffer based
RR	Round-Robin
SCHED	Stall Time Fair Memory
SMP	Symmetric Multiprocessing
SMT	Simultaneous Multithreading
SQ	Schedule Quantum
TB-LMI	Time-Based Least Memory Intensive
TCM	Thread Cluster Memory
TLP	Thread Level Parallelism
TMA	Total Memory Access
TMAPB	Thread Memory Access Per Bank
TPSR	Thread Priority Storage Register

Abstract

In the modern chip-multiprocessor system, concurrently executing applications/threads shares common resource such as main memory. Memory scheduling algorithms are developed to resolve memory contention between competing applications so that throughput is high and fairness of the overall multi-core systems is guaranteed. This emphasizes the importance of the memory access scheduling to efficiently utilize memory bandwidth. Although memory access scheduling techniques have been recently proposed for performance improvement, most of them have overlooked the fairness among the running applications.

In this thesis, we present Time-based Least Memory Intensive (TB-LMI) scheduling that address both fairness and system performance. The main idea of TB-LMI is to prioritize threads according to their memory contentions every pre-defined period of cycles to improve system throughput and to guarantee fairness. We evaluate TB-LMI on a variety of multi-programmed workloads with different queue sizes of memory controllers and compare its performance to six previously proposed scheduling algorithms. TB-LMI achieves the best system throughput and fairness. Previously proposed algorithms were First-Ready First Come First Serve (FR-FCFS) scheduling, First-Ready Fair Least-Request Most Related (FR-FLRMR) scheduling, First-Ready Fair Issue-Queue based Most Related (FR-FIQMR) scheduling, First-Ready Modified Reorder Buffer based (FR-Modified_ROB-based) scheduling, First-Ready Least REQuest (FR-LREQ) scheduling and Thread Cluster Memory (TCM) scheduling. TCM, FR-LREQ, and FR-FLRMR showed competitive results against the new scheduling TB-LMI. On 8-core system, TB-LMI improves system throughput and fairness on average by 4.22% and 11.7% respectively compared to TCM which was the previous work that provides the best system throughput and fairness.

Chapter 1. Introduction

A Central Processing Unit (CPU) is typically referred to as a processor. A processor contains memory caches, decoders, and execution units. Memory caches may be separated to a cache for instructions (Instruction cache) and another one for data (data cache) or unified caches where one cache for both instruction and data. Execution units such as Arithmetic Logic Unit (ALU) are used in performing arithmetic or logical operations. In order to increase processor's throughput, recent processors are tending now days towards parallel architectures. In early days, Operating Systems (OS) were developed to support multiprogramming. Multiprogramming is a kind of parallel processing in which several applications can run at the same time. In case of single CPU, OS executes part of one program, then part of another. All programs are appeared to be executed at the same time. Recent processors contain more than one CPU (core), allowing different applications to execute in parallel such as Simultaneous Multi Processing (SMP), Chip-Level Multi Processing (CMP), and Simultaneous Multi Threading (SMT) (described later). In order to get the highest benefit from recent processors, running applications must have a lot of routines that can run simultaneously. As an example a user may use the desktop to surf the web, watch a video and play a flash game at the same time. In general, hardware these days is trending toward highly parallel architectures.

This move resulted in increasing the number of threads that execute in parallel. All these threads are competing for shared resources. One of these most important resources is system main memory. While execution, each thread sends requests to the main memory to serve the cache misses. This introduces the need of memory access schedulers to decide which requests should be served first to either improve throughput or fairness or both.

1.1 Simultaneous Multithreading

In processor design, there are two ways to increase the on-chip parallelism: the first is superscalar technique which tries to make full use of Instruction Level Parallelism (ILP); the second is Thread Level Parallelism (TLP). Superscalar means that a processor tries to execute multiple instructions at the same time within a single processor core. TLP means that a processor tries to execute instructions from multiple applications/threads at the same time.

Some core components are duplicated in SMT. For example; an SMT core might have duplicate resources of thread scheduling, so that the core looks like two separate processors although it has a single execution unit. One of the SMT processors implementations is Hyper-threading processors which were introduced by Intel [1]. A processor with Hyper-Threading Technology consists of two logical processors per core. Each logical processor has its process state (logical registers and program counter). Each logical processor acts as a single processor where it can be individually interrupted, stalled, or directed to execute a specific application independently from the other logical processor. As shown in Figure 1 Hyper-threading processors, logical processors share the instruction cache, fetch queue, decoder, and L2 cache but they have different execution unit.

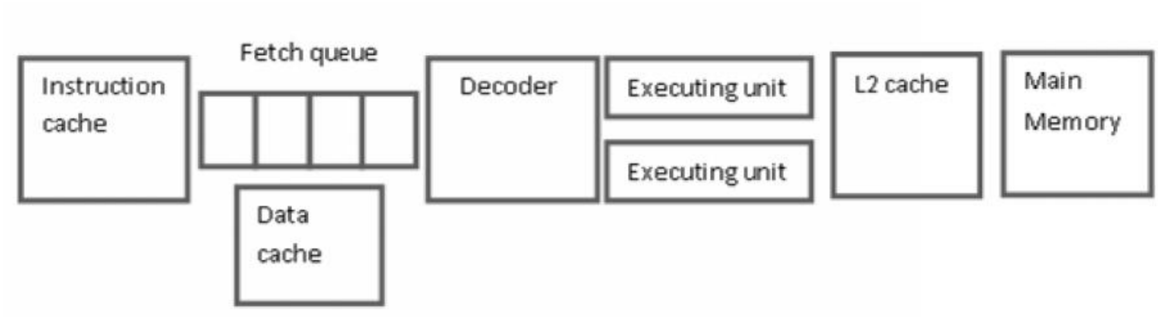

Figure 1.1: SMT architecture

1.2 Symmetric Multiprocessing

SMP stands for a symmetric multiprocessor system in hardware and software architecture. SMP consists of two or more identical processors that share common resources such as main memory interrupt system, and I/O devices. Each processor has its own Instruction cache, data cache, fetch unit, decoder, execution unit, and L2 cache. These identical processors are implemented on different chip. Each processor has its own chip and they share the common resources through a bus or a crossbar. Figure 1.2 shows SMP architecture. One of the first SMP processors is that what was introduced by IBM s/360 series in 1960s [2]. One of the main advantages of SMP processors are that if one processor fails the other can handle system requests. Also, if one application is multithreaded it can use more than one processor. Multithreaded applications may arise data inconsistency problem where data required may be obsolete (wrong). It is possible to have multiple copies of any instruction from a running application. One copy is in the main memory and a copy in each cache. Cache coherence guarantee that the changes in any shared data is updated to all caches and main memory. SMP disadvantages are its waste in power, energy, and area.

1.3 Multi-core processors

Multi-core processors are kind of processor that contains more than one core in one chip. These cores have their own instruction cache, Fetch stage, Decode stage, data cache, and execute stage but they share the same main memory and L2 cache. Figure 1.3 shows the architecture of multi-core processors. Multi-core processors are classified into homogenous multi-core which includes identical cores and heterogeneous multi-core that have non-identical cores [3].