

---

# **ANALYSIS OF CACHE PERFORMANCE FOR OPERATING SYSTEMS AND MULTIPROGRAMMING**

---

**THE KLUWER INTERNATIONAL SERIES  
IN ENGINEERING AND COMPUTER SCIENCE**

**PARALLEL PROCESSING AND  
FIFTH GENERATION COMPUTING**

*Consulting Editor*

Doug DeGroot

**Other books in the series:**

**PARALLEL EXECUTION OF LOGIC PROGRAMS,**

John S. Conery ISBN 0-89838-194-0

**PARALLEL COMPUTATION AND COMPUTERS FOR  
ARTIFICIAL INTELLIGENCE**

Janusz S. Kowalik ISBN 0-89838-227-0

**MEMORY STORAGE PATTERNS IN PARALLEL PROCESSING**

Mary E. Mace ISBN 0-89838-239-4

**SUPERCOMPUTER ARCHITECTURE**

Paul B. Schneck ISBN 0-89838-234-4

**ASSIGNMENT PROBLEMS IN  
PARALLEL AND DISTRIBUTED COMPUTING**

Shahid H. Bokhari ISBN 0-89838-240-8

**MEMORY PERFORMANCE OF PROLOG ARCHITECTURES**

Evan Tick ISBN 0-89838-254-8

**DATABASE MACHINES AND KNOWLEDGE BASE MACHINES**

Masaru Kitsuregawa ISBN 0-89838-257-2

**PARALLEL PROGRAMMING AND COMPILERS**

Constantine D. Polychronopoulos ISBN 0-89838-288-2

**DEPENDENCE ANALYSIS FOR SUPERCOMPUTING**

Utpal Banerjee ISBN 0-89838-289-0

**DATA ORGANIZATION IN PARALLEL COMPUTERS**

H.A.G. Wijshoff ISBN 0-89838-304-8

---

# **ANALYSIS OF CACHE PERFORMANCE FOR OPERATING SYSTEMS AND MULTIPROGRAMMING**

by

**Anant Agarwal**

Massachusetts Institute of Technology

with a foreword by

**John L. Hennessey**



**KLUWER ACADEMIC PUBLISHERS**  
**Boston/Dordrecht/London**

---

**Distributors for North America:**

Kluwer Academic Publishers  
101 Philip Drive  
Assinippi Park  
Norwell, Massachusetts 02061, USA

**Distributors for the UK and Ireland:**

Kluwer Academic Publishers  
Falcon House, Queen Square  
Lancaster LA1 1RN, UNITED KINGDOM

**Distributors for all other countries:**

Kluwer Academic Publishers Group  
Distribution Centre  
Post Office Box 322  
3300 AH Dordrecht, THE NETHERLANDS

---

**Library of Congress Cataloging-in-Publication Data**

Agarwal, Anant.

Analysis of cache performance for operating systems and  
multiprogramming / by Anant Agarwal.

p. cm. — (The Kluwer international series in engineering and  
computer science ; SECS 69)

Bibliography: p.

Includes index.

ISBN-13: 978-1-4612-8897-8

e-ISBN-13: 978-1-4613-1623-7

DOI: 10.1007/978-1-4613-1623-7

1. Cache memory—Evaluation. 2. Operating systems (Computers)  
3. Multiprogramming (Electronic computers) I. Title. II. Series.

TK7895.M4A33 1988

005.4 '3—dc19

88-36500

CIP

---

Copyright © 1989 by Kluwer Academic Publishers

Softcover reprint of the hardcover 1st edition 1989

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061.

# Contents

List of Figures . . . . .	x
List of Tables . . . . .	xv
Foreword . . . . .	xvii
Preface . . . . .	xix
Acknowledgements . . . . .	xxi
<b>1 Introduction</b>	<b>1</b>
1.1 Overview of Cache Design . . . . .	2
1.1.1 Cache Parameters . . . . .	2
1.1.2 Cache Performance Evaluation Methodology . . . . .	4
1.2 Review of Past Work . . . . .	5
1.3 Then, Why This Research? . . . . .	11
1.3.1 Accurately Characterizing Large Cache Performance . . . . .	12
1.3.2 Obtaining Trace Data for Cache Analysis . . . . .	14
1.3.3 Developing Efficient and Accurate Cache Analysis Methods	15
1.4 Contributions . . . . .	16
1.5 Organization . . . . .	17

<b>2</b>	<b>Obtaining Accurate Trace Data</b>	<b>19</b>
2.1	Current Tracing Techniques . . . . .	19
2.2	Tracing Using Microcode . . . . .	22
2.3	An Experimental Implementation . . . . .	24
2.3.1	Storage of Trace Data . . . . .	25
2.3.2	Recording Memory References . . . . .	26
2.3.3	Tracing Control . . . . .	27
2.4	Trace Description . . . . .	27
2.5	Applications in Performance Evaluation . . . . .	29
2.6	Extensions and Summary . . . . .	29
<b>3</b>	<b>Cache Analyses Techniques – An Analytical Cache Model</b>	<b>31</b>
3.1	Motivation and Overview . . . . .	32
3.1.1	The Case for the Analytical Cache Model . . . . .	32
3.1.2	Overview of the Model . . . . .	33
3.2	A Basic Cache Model . . . . .	35
3.2.1	Start-Up Effects . . . . .	37
3.2.2	Non-Stationary Effects . . . . .	37
3.2.3	Intrinsic Interference . . . . .	38
3.3	A Comprehensive Cache Model . . . . .	45
3.3.1	Set Size . . . . .	46
3.3.2	Modeling Spatial Locality and the Effect of Block Size . .	48
3.3.3	Multiprogramming . . . . .	54
3.4	Model Validation and Applications . . . . .	57
3.5	Summary . . . . .	63

<b>4</b>	<b>Transient Cache Analysis – Trace Sampling and Trace Stitching</b>	<b>65</b>
4.1	Introduction . . . . .	65
4.2	Transient Behavior Analysis and Trace Sampling . . . . .	66
4.2.1	Definitions . . . . .	68
4.2.2	Analysis of Start-up Effects in Single Process Traces . . .	70
4.2.3	Start-up Effects in Multiprocess Traces . . . . .	72
4.3	Obtaining Longer Samples Using Trace Stitching . . . . .	74
4.4	Trace Compaction – Cache Filtering with Blocking . . . . .	77
4.4.1	Cache Filter . . . . .	79
4.4.2	Block Filter . . . . .	80
4.4.3	Implementation of the Cache and Block Filters . . . . .	83
4.4.4	Miss Rate Estimation . . . . .	85
4.4.5	Compaction Results . . . . .	86
<b>5</b>	<b>Cache Performance Analysis for System References</b>	<b>89</b>
5.1	Motivation . . . . .	90
5.2	Analysis of the Miss Rate Components due to System References	90
5.3	Analysis of System Miss Rate . . . . .	94
5.4	Associativity . . . . .	96
5.5	Block Size . . . . .	98
5.6	Evaluation of Split Caches . . . . .	101
<b>6</b>	<b>Impact of Multiprogramming on Cache Performance</b>	<b>103</b>
6.1	Relative Performance of Multiprogramming Cache Techniques . .	104
6.2	More on Warm Start versus Cold Start . . . . .	107
6.3	Impact of Shared System Code on Multitasking Cache Performance . . . . .	110

6.4	Process Switch Statistics and Their Effects on Cache Modeling . . . . .	113
6.5	Associativity . . . . .	114
6.6	Block Size . . . . .	117
6.7	Improving the Multiprogramming Performance of Caches . . . . .	120
6.7.1	Hashing . . . . .	121
6.7.2	A Hash-Rehash Cache . . . . .	122
6.7.3	Split Caches . . . . .	124
<b>7</b>	<b>Multiprocessor Cache Analysis</b>	<b>129</b>
7.1	Tracing Multiprocessors . . . . .	130
7.2	Characteristics of Traces . . . . .	131
7.3	Analysis . . . . .	133
7.3.1	General Methodology . . . . .	134
7.3.2	Multiprocess Interference in Large Virtual and Physical Caches . . . . .	135
7.3.3	Analysis of Interference Between Multiple Processors . . . . .	143
7.3.4	Blocks Containing Semaphores . . . . .	146
<b>8</b>	<b>Conclusions and Suggestions for Future Work</b>	<b>149</b>
8.1	Concluding Remarks . . . . .	149
8.2	Suggestions for Future Work . . . . .	152
<b>Appendices</b>		
<b>A</b>	<b>Sensitivity of the Miss Rate on Time Granule <math>\tau</math></b>	<b>155</b>
<b>B</b>	<b>Characterization of the Collision Rate <math>c</math></b>	<b>159</b>
B.1	On the Stability of the Collision Rate . . . . .	159
B.2	Estimating Variations in the Collision Rate . . . . .	162



**C Inter-Run Intervals and Spatial Locality** 165

**D Summary of Benchmark Characteristics** 169

**E Features of ATUM-2** 175

    E.1 Distributing Trace Control to All Processors . . . . . 175

    E.2 Provision of Atomic Accesses to Trace Memory . . . . . 176

    E.3 Instruction Stream Compaction Using a Cache Simulated in  
        Microcode . . . . . 176

    E.4 Microcode Patch Space Conservation . . . . . 177

**Bibliography** 179

**Index** 187

# List of Figures

1.1	Cache organization. . . . .	3
2.1	Typical VAX computer microcode flows. . . . .	26
3.1	Collision in a direct-mapped cache. Shaded rectangles represent blocks. . . . .	39
3.2	Number of blocks of size one word (4 bytes) that map into a cache set with depth of overlap $d$ . Values on the figure indicate cache size in number of sets. . . . .	40
3.3	Number of collision blocks as a function of the number of sets. Block size is one word (4 bytes). . . . .	41
3.4	Miss rate versus time granule for direct-mapped caches with 1024, 4096, 16384, and 65536 sets. Block size is 4 bytes. Actual values represent simulated miss rates. . . . .	44
3.5	Miss rate versus number of sets $S$ . Caches are direct mapped and block size is 4 bytes. . . . .	45
3.6	Number of colliding blocks vs. $D$ . Cache size is specified in bytes. . . . .	47
3.7	Markov models depicting spatial locality in programs. . . . .	50
3.8	Distribution of run lengths for the benchmark IVEX. $R(l)$ is the probability that a run is of length $l$ . . . . .	51
3.9	Average number of unique blocks, $u(B)$ in a time granule, $\tau$ , vs. the block size $B$ in bytes. . . . .	53

3.10	Miss rate vs. cache size for IVEX. In (a), (b) set size is constant, and in (c), (d) block size is constant. Solid lines are model, dotted lines are simulation with random replacement, and dashed lines are simulation with LRU replacement. All sizes in bytes. . . . .	59
3.11	Miss rate vs. cache size for AL1. In (a), (b) set size is constant, and in (c), (d) block size is constant. Solid lines are model, dotted lines are simulations with for random replacement, and dashed lines are simulations with LRU replacement. . . . .	60
3.12	Miss rate vs. cache size for TMIL1. In (a), (b) set size is constant, and in (c), (d) block size is constant. Solid lines are model, dotted lines are simulations with random replacement, and dashed lines are simulations with LRU replacement. . . . .	61
3.13	Miss rate vs. cache size (K bytes) for multiprogramming. Block size is 16 bytes and cache is direct-mapped. . . . .	63
4.1	Miss rate vs time for benchmark IVEX. Set size is 1; block size is 4. . . . .	67
4.2	Cumulative cold misses for IVEX for a 16K-byte cache (block size is 4 bytes). . . . .	69
4.3	(a) Cumulative cold misses and (b) cumulative misses for direct-mapped caches. . . . .	70
4.4	Cumulative cold misses for a 16K-byte cache for various block and set sizes. . . . .	71
4.5	Comparing cumulative cold misses for uniprogramming and multiprogramming. Cache size is 16K bytes, block size is 16 bytes and set size is 1. . . . .	73
4.6	Cumulative cold misses for various cache sizes, block size 16 bytes, set size 1. . . . .	73
4.7	Cumulative cold misses for three MUL10 traces, with Block size 16 bytes. (a) Individually simulated (b) Concatenated. . . . .	75
4.8	Addressing pattern in a trace segment. . . . .	79
4.9	Addressing pattern of a trace segment after cache filtering. . . .	80
4.10	Addressing pattern of a trace segment after cache filtering and block filtering. . . . .	81

4.11	Distribution of the difference between the mean miss rate within a run and the miss rate of a representative reference from that run for PS1. $\sigma_i^2$ is the intra-run variance of the miss rate (percent), $\sigma^2$ is the miss-rate variance in the entire trace, and $s/N$ is the compaction ratio. . . . .	82
4.12	Trace compaction steps. . . . .	84
4.13	Comparison of actual and estimated miss rates for various compactions. The results for benchmarks PS1, AL1 and TMIL1 are averaged. Estimate 1 uses a cache filter $C_f : (256, 1, 1, 1)$ , and the block filter parameters are $w = 128$ and $b = 4$ ; estimate 2 uses $C_f : (1K, 1, 1, 1)$ , $w = 128$ and $b = 16$ . . . . .	86
5.1	Effect of system references on cold-start cache miss rate. Cache block size is 16 bytes. D is set size or degree of associativity. . .	91
5.2	Relative increase in cache miss rate due to system references for VMS. Block size is 16 bytes, and D is set size. . . . .	91
5.3	Components of system and user miss rate. Set size is 1, block size is 16 bytes. . . . .	93
5.4	Components of system and user miss rate as a fraction of the total miss rate. Set size is 1, block size is 16 bytes. . . . .	93
5.5	Comparing inherent system and user miss rate for VMS and Ultrix traces. . . . .	94
5.6	Effect of system references on working-set size. . . . .	95
5.7	The relationship between the fraction of system references and the system miss rate. . . . .	97
5.8	Effect of associativity on system and interference components of the miss rate (a) VMS (b) Ultrix. Block size is 16 bytes. . . . .	98
5.9	Effect of block size on user and system miss rates. Caches are direct-mapped. . . . .	99
5.10	Effect of large block size on system miss rate and user miss rate. . . . .	100
5.11	Split versus unified cache for system and user references. Block size is 16 bytes. . . . .	102
6.1	Multiprogramming cache performance for user workloads. Set size is 1, and block size is 16 bytes. . . . .	105

6.2	Comparison of cold-start and warm-start performance of MUL3 and MUL10. Set size is 1, and block size is 16 bytes. . . . .	108
6.3	On purging and PIDs for user and system workloads. Set size is 1, block size is 16 bytes, and miss rates are warm start. . . . .	111
6.4	Percentage change in warm-start miss rate from uniprogramming to multiprogramming. Set size is 1 and block size is 16 bytes. . .	112
6.5	Distribution of execution time quanta. . . . .	114
6.6	Probability distribution of the LRU stack distance for task switching for MUL3, MUL6, MUL10, and SAVEC. . . . .	115
6.7	On associativity. Block size is 16 bytes and miss rates are warm start. . . . .	116
6.8	Effect of block size on cache miss rate. Miss rates are warm start and caches are direct-mapped. . . . .	118
6.9	Effect of block size on average memory access time. Cache sizes are in bytes. . . . .	119
6.10	Effect of hashing. Trace is MUL3, block size is 16 bytes and D is set size. (a) Hashing (b) Hashing with system not shared (D=1). . . . .	122
6.11	Miss rates of hashing schemes. All statistics assume warm-start. Trace is MUL3, block size is 16 bytes and D is set size. . . . .	124
6.12	Split instruction and data caches for uniprogramming. Set size is one and block size is 16 bytes. . . . .	125
6.13	Split instruction and data caches for multiprogramming. Miss rates are warm start. Set size is one and block size is 16 bytes. . .	126
6.14	Split instruction and data caches for multiprogramming. Miss rates are warm start. Set size is two and block size is 16 bytes. . .	127
7.1	Baseline traffic rates in physical-addressed caches. Block is 16 bytes and caches are direct-mapped. . . . .	136
7.2	Inter-process interference in physical caches. . . . .	137
7.3	Cache occupancy vs. time, no sorting. Process 1 blocks are shown as the hashed region at the bottom, then processes 2, 3, etc., and shared blocks as the cross-hatched section at the top of the graph. Cache size is 256KB. . . . .	138

- 7.4 Cache occupancy vs. time, processes sorted. Cache size is 256KB. 138
- 7.5 Cache occupancy vs. time, processes sorted. Cache size is 16KB. 139
- 7.6 Inter-process interference in virtual caches. . . . . 141
- 7.7 Virtual vs. physical addressing. . . . . 141
- 7.8 Virtual vs. physical addressing in caches for workloads with few context swaps. . . . . 142
- 7.9 Impact of inter-CPU interference in direct-mapped caches. Solid circles are with inter-CPU interference and open circles are without. 144
- 7.10 Impact of process migration. Solid squares are with process migration effects and open squares are without. . . . . 145
  
- A.1 Number of unique references per time granule,  $u(1)$ , versus granule size  $\tau$ . . . . . 157
  
- B.1 Estimating the collision rate  $c$  in a direct-mapped cache. . . . . 162
  
- C.1 The average number of unique blocks in a time granule,  $u(B)$ , versus block size  $B$  in bytes. The estimate using inter-run intervals is more accurate than with just the run-length distribution. . 167

# List of Tables

2.1	Typical trace-technique parameters . . . . .	21
2.2	Distortions introduced by each trace technique (– implies <i>none</i> ). . . . .	22
2.3	Type codes in experimental ATUM output . . . . .	25
3.1	Collision rate for various cache sizes. . . . .	43
3.2	Percentage error in estimated miss rates and hit rates. $\bar{\epsilon}$ and $\hat{\epsilon}$ represent mean and maximum errors respectively. Block sizes are in bytes. . . . .	62
6.1	Fraction of data blocks purged that are dirty as a function of cache size, for the trace MUL3 and a block size of 16 bytes. . . . .	120
6.2	Performance of a direct-mapped hashing and hash-rehash cache for MUL3. Block size is 16 bytes. $m$ denotes cache miss rate, and $T$ denotes average memory access time in cycles. . . . .	124
7.1	Excerpt of raw ATUM-2 trace. . . . .	132
7.2	Cache configuration. . . . .	133
7.3	Distribution of segment lengths. . . . .	146
B.1	Measured variation in the collision rate for IVEX. Coll-blks: number of colliding blks, Ms-lru and Ms-fifo: number of collision induced misses for LRU and random replacement, c-lru and c-fifo: measured values of $c$ for LRU and FIFO replacement. $B$ is in bytes. . . . .	161

- D.1 Summary of benchmark statistics. The cold start (C.S.) and warm start (W.S.) miss rates are for a 64K-byte direct-mapped cache with a block size of 16 bytes using PIDs. The number of references are in thousands. . . . . 170
- D.2 Individual benchmark (VMS) working-set sizes for user references only. T represents the time window. . . . . 171
- D.3 Individual benchmark (VMS) working-set sizes for user plus system references. T represents the time window. . . . . 172
- D.4 Multiprocessor trace characteristics. Instruction counts are in thousands. "Instr" stands for instructions, R for memory references, Proc. for processes, and C.S. for context swap. . . . . 173



## Foreword

As we continue to build faster and faster computers, their performance is becoming increasingly dependent on the memory hierarchy. Both the clock speed of the machine and its throughput per clock depend heavily on the memory hierarchy. The time to complete a cache access is often the factor that determines the cycle time. The effectiveness of the hierarchy in keeping the average cost of a reference down has a major impact on how close the sustained performance is to the peak performance. Small changes in the performance of the memory hierarchy cause large changes in overall system performance. The strong growth of RISC machines, whose performance is more tightly coupled to the memory hierarchy, has created increasing demand for high performance memory systems. This trend is likely to accelerate: the improvements in main memory performance will be small compared to the improvements in processor performance. This difference will lead to an increasing gap between processor cycle time and main memory access time. This gap must be closed by improving the memory hierarchy.

Computer architects have attacked this gap by designing machines with cache sizes an order of magnitude larger than those appearing five years ago. Microprocessor-based RISC systems now have caches that rival the size of those in mainframes and supercomputers. These large cache sizes and the need for more accurate simulation have created the demand for new and more ambitious studies of cache performance. Earlier studies of caches used programs that are much smaller than the programs run today. Those earlier studies cannot adequately predict the performance of very large caches, since the benchmarks essentially fit within the cache.

Since small changes in miss rate cause large changes in system performance, our cache performance prediction must be more accurate than ever before. This requirement for accuracy means that factors that were previously ignored or estimated must be carefully analyzed and measured. A primary example of such a factor is the effect of the operating system's references on cache performance of a user program. The emergence of shared-memory multiprocessors has created another demand for accurate modeling of cache performance. These machines are extremely sensitive to cache misses because such misses cause contention for a shared resource (the bus and memory). In addition, multiprocessors cannot be accurately simulated with simple uniprocessor cache traces, due to the presence of invalidation requests and misses arising from those requests, both of which are absent in a uniprocessor trace.

This work addresses these problems by examining the performance of large

programs running in large caches and measuring an entire range of effects not examined in earlier work. To conduct these experiments the first challenge was to devise a method for collecting traces of large programs, including the operating system and multiprogramming effects that were neglected in earlier studies. A system for collecting such traces was created and is described herein.

Because of the large domain of possible cache designs and the length of the traces needed to simulate them accurately, analytical modeling techniques that can explore a range of cache organizations quickly are important. These techniques allow a designer to concentrate on a small number of alternatives out of the vast range of possible cache designs. This small number of designs can then be fully simulated to obtain accurate performance estimates.

Finally, exploring the effect of multiprocessing required obtaining traces from a multiprocessor and then constructing accurate simulation models. Since the effects of multiprocessing can have a significant impact on cache performance and an enormous impact on system performance, it is vital to obtain these traces and model these effects accurately.

The research results described here are an important resource for all computer architects. The increasing importance of the memory hierarchy in determining the performance of high speed machines and multiprocessors makes this type of exploration and analysis invaluable if our computer systems are to continue to grow in performance.

*John L. Hennessy*  
*Stanford, California*

# Preface

Advances in high-performance processors continue to create an increased need for memory bandwidth. Migration to multiprocessor architectures exacerbates this need causing the well known Von Neumann bottleneck to main memory. Caches can provide this bandwidth cost-effectively. However, minimizing the performance loss due to caching requires that our analysis and prediction of cache performance become more exact. Although previous studies have shown that operating system and multiprogramming activities affect the cache performance, those studies did not deal with these issues in detail, nor did they address multiprocessor cache performance, largely because of the unavailability of efficient analysis techniques and the difficulty in collecting data for these analyses. To obtain the higher hit rates needed to sustain the effectiveness of caches, we must address these issues completely.

This book investigates the performance of large caches for realistic operating system and multiprogramming workloads. Cache analysis of bus-based multiprocessors is also presented. A suite of efficient and accurate cache evaluation techniques is developed. These include: a mathematical cache model, a trace sampling and a trace stitching procedure, and a trace compaction method. The analyses use a data collection technique called ATUM to obtain realistic system traces of multitasking workloads with little distortion. An extension of ATUM that traces multiprocessors is also described.

Accurately characterizing cache behavior using ATUM traces shows that both operating system and multiprogramming activities significantly degrade cache performance, with an even greater proportional impact on large caches. Multiprocessor interference further reduces cache performance. From a careful analysis of the causes of this degradation, we explore various techniques to reduce this loss. While seemingly little can be done to mitigate the effect of system references, multitasking cache misses can be reduced with little effort. The impact of process migration, virtual versus physical addressing, and the effect of synchronization on large write-back caches in multiprocessors is investigated. We also demonstrate how analytical cache modeling, and trace sampling – with a new approach to cold-start and warm-start analysis – can be used to make large cache studies insightful and efficient.

This book is largely based on my Ph.D. thesis submitted in May 1987 at Stanford University. Chapter 7 contains new material on multiprocessor caches extending the earlier work on uniprocessor cache behavior and represents recent work done at Stanford and also at M.I.T. in collaboration with Dick Sites at Digital Equipment Corporation, Hudson, Massachusetts.

## Acknowledgments

I wish to thank Professor John Hennessy for his support, guidance and encouragement, and most of all, for his enthusiasm, throughout the course of my research. Even in his terribly busy moments, he would be happily willing to discuss problems. I am indebted to Professor Mark Horowitz, who was a source of innumerable suggestions and ideas, and to Dick Sites, who was instrumental in getting me going in my research. I am also indebted to Professors Thomas Kailath, Robert Matthews and John Newkirk, who offered me the opportunity for graduate studies at Stanford.

My gratitude to the MIPS-X crew, including Arturo Salz, Paul Chow, Malcolm Wing, Scott McFarling, C. Y. Chu, Steve Richardson, Don Stark, Rich Simoni, John Acken, Steve Przybylski, Peter Steenkiste, Steve Tjiang, and Glenn Gulak, for their suggestions and criticisms, for asking all those questions that motivated my research, and for simply making my experience at Stanford a very enjoyable one.

I must not fail to acknowledge my Indian Institute of Technology clan in the valley, had it not been for whom, I would have completed my thesis far sooner, but with far more of my sanity lost. I am also grateful to Madan Valluri, Sailesh Rao, Barbara and Steve Mckee, and Yvonne and Jim Weiberg for their friendship and affection when I first came to this country.

My research was supported in part by Defense Advanced Research Projects Agency under contract # MDA903-83-C-0335 at Stanford, and under contract # N00014-87-K-0825 at MIT during the preparation of this book. I am grateful to Digital Equipment Corporation, Hudson, for making trace data available for this research.

My wife Anu was a constant source of moral support and encouragement throughout my graduate study, going through some stressful times very patiently and happily.

Finally, I dedicate this book to my parents, for their love, care and sacrifice.

*Anant Agarwal*  
*Cambridge, Massachusetts*

---

# **ANALYSIS OF CACHE PERFORMANCE FOR OPERATING SYSTEMS AND MULTIPROGRAMMING**