Performance Tuning of Scientific Applications

Chapman & Hall/CRC
Computational Science Series

SERIES EDITOR

Horst Simon
Deputy Director
Lawrence Berkeley National Laboratory
Berkeley, California, U.S.A.

AIMS AND SCOPE

This series aims to capture new developments and applications in the field of computational science through the publication of a broad range of textbooks, reference works, and handbooks. Books in this series will provide introductory as well as advanced material on mathematical, statistical, and computational methods and techniques, and will present researchers with the latest theories and experimentation. The scope of the series includes, but is not limited to, titles in the areas of scientific computing, parallel and distributed computing, high performance computing, grid computing, cluster computing, heterogeneous computing, quantum computing, and their applications in scientific disciplines such as astrophysics, aeronautics, biology, chemistry, climate modeling, combustion, cosmology, earthquake prediction, imaging, materials, neuroscience, oil exploration, and weather forecasting.

PUBLISHED TITLES

PETASCALE COMPUTING: ALGORITHMS AND APPLICATIONS
Edited by David A. Bader

PROCESS ALGEBRA FOR PARALLEL AND DISTRIBUTED PROCESSING
Edited by Michael Alexander and William Gardner

GRID COMPUTING: TECHNIQUES AND APPLICATIONS
Barry Wilkinson

INTRODUCTION TO CONCURRENCY IN PROGRAMMING LANGUAGES
Matthew J. Sottile, Timothy G. Mattson, and Craig E Rasmussen

INTRODUCTION TO SCHEDULING
Yves Robert and Frédéric Vivien

SCIENTIFIC DATA MANAGEMENT: CHALLENGES, TECHNOLOGY, AND DEPLOYMENT
Edited by Arie Shoshani and Doron Rotem

INTRODUCTION TO THE SIMULATION OF DYNAMICS USING SIMULINK®
Michael A. Gray

INTRODUCTION TO HIGH PERFORMANCE COMPUTING FOR SCIENTISTS AND ENGINEERS, **Georg Hager and Gerhard Wellein**

PERFORMANCE TUNING OF SCIENTIFIC APPLICATIONS, **Edited by David H. Bailey, Robert F. Lucas, and Samuel W. Williams**

Performance Tuning of Scientific Applications

Edited by
David H. Bailey
Robert F. Lucas
Samuel W. Williams

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number: 978-1-4398-1569-4 (Hardback)

Library of Congress Cataloging-in-Publication Data

Performance tuning of scientific applications / David H. Bailey, Robert F. Lucas, Samuel W. Williams, editors.
p. cm. -- (Chapman & Hall/CRC computational science)
Includes bibliographical references and index.
ISBN 978-1-4398-1569-4 (hardback)
1. Science--Data processing--Evaluation. 2. Electronic digital computers--Evaluation. 3. System design--Evaluation. 4. Science--Computer programs. I. Bailey, David H. II. Lucas, Robert F. III. Williams, Samuel Watkins. IV. Title. V. Series.

Q183.9.P47 2011
502.85--dc22 2010042967

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Contributor List

Mark Adams
APAM Department,
Columbia University
New York, New York

David H. Bailey
Lawrence Berkeley National Lab
Berkeley, California

Vincent Beckner
Lawrence Berkeley National Lab
Berkeley, California

John Bell
Lawrence Berkeley National Lab
Berkeley, California

Laura Carrington
San Diego Supercomputer Center
San Diego, California

Jonathan Carter
Lawrence Berkeley National Lab
Berkeley, California

Jacqueline Chame
University of Southern California
Information Sciences Institute
Los Angeles, California

Kaushik Datta
University of California at Berkeley
Berkeley, California

Bronis R. de Supinski
Lawrence Livermore National Lab
Livermore, California

Jack Dongarra
University of Tennessee,
Oak Ridge National Lab
Knoxville, Tennessee

Erik W. Draeger
Lawrence Livermore National Lab
Livermore, California

Stéphane Ethier
Princeton Plasma Physics Lab,
Princeton University
Princeton, New Jersey

Mary Hall
University of Utah School of
Computing
Salt Lake City, Utah

Jeffrey K. Hollingsworth
University of Maryland
College Park, Maryland

Khaled Ibrahim
Lawrence Berkeley National Lab
Berkeley, California

Jesus Labarta
Barcelona Supercomputing Center
Barcelona, Spain

Byounghak Lee
Texas State University, San Marcos
San Marcos, Texas

Allen D. Malony
University of Oregon
Eugene, Oregon

John Mellor-Crummey
Rice University
Houston, Texas

Juan Meza
Lawrence Berkeley National Lab
Berkeley, California

Shirley Moore
University of Tennessee
Knoxville, Tennessee

Leonid Oliker
Lawrence Berkeley National Lab
Berkeley, California

Erik Schnetter
CCT, Louisiana State University
Baton Rouge, Louisiana

Martin Schulz
Lawrence Livermore National Lab
Livermore, California

John Shalf
Lawrence Berkeley National Lab
Berkeley, California

Hongzhang Shan
Lawrence Berkeley National Lab
Berkeley, California

Sameer S. Shende
University of Oregon
Eugene, Oregon

Allan Snavely
University of California, San Diego
San Diego, California

Erich Strohmaier
Lawrence Berkeley National Lab
Berkeley, California

Daniel K. Terpstra
University of Tennessee
Knoxville, Tennessee

Ananta Tiwari
University of Maryland
College Park, Maryland

Lin-Wang Wang
Lawrence Berkeley National Lab
Berkeley, California

Harvey Wasserman
Lawrence Berkeley National Lab
Berkeley, California

Vincent M. Weaver
University of Tennessee
Knoxville, Tennessee

Samuel W. Williams
Lawrence Berkeley National Lab
Berkeley, California

Patrick H. Worley
Oak Ridge National Lab
Oak Ridge, Tennessee

Katherine Yelick
Lawrence Berkeley National Lab
Berkeley, California

Zhengji Zhao
Lawrence Berkeley National Lab
Berkeley, California

About the Editors

David H. Bailey is the chief technologist of the Computational Research Department at Lawrence Berkeley National Laboratory (LBNL). Prior to coming to LBNL in 1998, he held a position with the NASA Advanced Supercomputing Center at NASA's Ames Research Center. Bailey's 140 published papers and three books span studies in system performance analysis, highly parallel computing, numerical algorithms, and computational mathematics. He has received the Sidney Fernbach Award from the IEEE Computer Society, the Gordon Bell Prize from the Association for Computing Machinery, and the Chauvenet and Merten Hesse Prizes from the Mathematical Association of America.

Robert F. Lucas is the Computational Sciences Division director of the Information Sciences Institute, and also a research associate professor of computer science in the Viterbi School of Engineering at the University of Southern California. Prior to coming to USC/ISI, he worked at the Lawrence Berkeley National Laboratory and the Defense Advanced Research Project Agency. His published research includes studies in system performance analysis, real-time applications, heterogeneous computing, and system simulation.

Samuel W. Williams is a research scientist in the Future Technologies Group at the Lawrence Berkeley National Laboratory (LBNL). He received his PhD in 2008 from the University of California at Berkeley. Dr. Williams has authored or co-authored 30 technical papers including several award-winning papers. His research includes high-performance computing, automatic performance tuning (auto-tuning), computer architecture, and performance modeling.

Chapter 1

Introduction

David H. Bailey

Lawrence Berkeley National Laboratory

1.1 Background

The field of performance analysis is almost as old as scientific computing itself. This is because large-scale scientific computations typically press the outer envelope of performance. Thus, unless a user's application has been tuned to obtain a high fraction of the theoretical peak performance available on the system, he/she might not be able to run the calculation in reasonable time or within the limits of computer resource allocations available to the user. For these reasons, techniques for analyzing performance and modifying code to achieve higher performance have long been a staple of the high-performance computing field.

What's more, as will become very clear in the chapters of this book, understanding why a calculation runs at a disappointing performance level is seldom straightforward, and it is often even more challenging to devise reasonable changes to the code while at the same doing avoiding doing violence to the structure of the code.

Thus over the past few decades as scientific computing has developed, the field of performance analysis has developed in tandem. Some familiar themes include:

- *Benchmark performance and analysis.* Comparing the performance of a scientific calculation on various available computer systems or studying the performance of a computer system on various scientific applications.
- *Performance monitoring.* Developing hardware/software techniques and

tools that can be used to accurately monitor performance of a scientific application as it is running—floating-point operation counts, integer operations, cache misses, etc.

- *Performance modeling.* Developing techniques and tools to encapsulate the performance behavior of applications and computer system into relatively simple yet accurate models.

- *Performance tuning.* Developing semi-automatic techniques and tools to make necessary code changes to optimize the run-time performance of a scientific application (or class of applications).

The purpose of this book is to introduce the reader to research being done in this area as of the current date (summer 2010). The various chapters are written by some of the most notable experts in the field, yet the book is not targeted primarily to experts in the performance field. Instead, the target audience is the much broader community of physicists, chemists, biologists, environmental scientists, mathematicians, computer scientists, and engineers who actually use these systems. In other words, the intent of this book is to introduce real-world users of computer systems to useful research being done in the field of performance analysis.

The hope is that the exposition in this book will assist these scientists to better understand the performance vagaries that may arise in scientific applications, and also to better understand what practical potential there is to actually improve the performance—what changes need to be made; how difficult is it to make these changes; what improvement can be expected if such changes are made; what the are prospects of semi-automatic tools to make these changes, etc.

In general, it should be clear from the scope of research work discussed in this book that the field of performance analysis is a relatively sophisticated field. Indeed, it is now a full-fledged instance of empirical science, involving:

- *Experimentation.* One of the first steps in understanding the performance characteristics of a given application is to perform some carefully conducted measurements, collecting data that can be used to confirm or refute various conjectures, probe anomalies, develop accurate models, and compare results.

- *Theory.* Once a set of validated experiments have been run, performance researchers typically attempt to make sense of this data by developing analytic models that accurately represent this data, and which lead to further insights, both general and specific, that will be of use to other researchers.

- *Engineering.* One important activity in the field of performance analysis is the development of tools and techniques to make more accurate measurements, develop more accurate models, and to assist users in making requisite changes in codes.

Indeed, efforts in this field have led to a significantly increased emphasis on thoroughness and rigor in conducting performance experiments (see next section), and in drawing conclusions from performance data. In this regards, the field of performance analysis is on a trajectory that is typical of almost every other field of science—after some initial work, researchers must develop and adopt significantly more rigorous, reliable methods to further advance the state of the art. We can expect this trend to continue into the future.

1.2 "Twelve Ways to Fool the Masses"

The need for increased sophistication and rigor in the field can be most clearly (and most amusingly) seen in an episode from the early days of parallel computing, which will be briefly recounted here.

Until about 1976, most scientific computing was done on large general-purpose mainframe computers, typically in the central computing centers of universities and research laboratories. Some of the most widely used systems include the IBM 7030, 7040 and 7090 systems, and the CDC 6600 and 7600 systems. Some minicomputers, notably the PDP-11 and DEC-10, were also used for scientific computation.

The Cray-1 made its debut in 1975, designed by legendary computer architect Seymour Cray. This system featured a vector architecture, which proved to be very efficient for many types of scientific applications at the time, because it could sustain large fractions of the system's peak performance while accessing multi-dimensional arrays along any dimension (not just the first), provided that some care was taken in coding to avoid power-of-two memory strides. Many scientists achieved dramatic performance improvements, compared with what they could achieve on other systems available at the time.

As a result of this initial success, systems manufactured by Cray Research dominated the world of scientific computing for at least a decade. The Cray X-MP, introduced in 1982, featured multiple processors. The Cray-2, introduced in 1985, dramatically increased the memory available to scientific programs. A few years later, the Cray Y-MP continued the X-MP architectural line. Several large Japanese firms, including NEC, Fujitsu and Hitachi, also introduced competing systems based on vector designs.

Beginning in 1988, several distributed memory parallel systems became commercially available. These included the Connection Machine, from Thinking Machines, Inc., and systems from Intel and other vendors that were constructed entirely from commodity microprocessors and network components. Some initial work with these systems achieved remarkably high performance rates. These results generated considerable enthusiasm, and soon it became clear that a paradigm shift was underway.

However, some researchers in the field began to grow concerned with what

appeared to be "hype" in the field—in numerous cases, scientists' descriptions of their work with these highly parallel architectures seemed indistinguishable from the sort of salesmanship that one might expect to hear from computer vendor marketing personnel. While some degree of enthusiasm was perhaps to be expected, there was concern that this "enthusiasm" was leading to sloppy science and potentially misleading results. Some researchers noted examples, even in presumably peer-reviewed literature, of questionable research work. The most common example was questionable comparisons between highly parallel systems and Cray vector systems.

In 1991, one of the present editors thought that it was time to publicly express concern with these practices. This was done in the form of a humorous tongue-in-cheek article entitled "Twelve Ways to Fool the Masses When Giving Performance Reports on Parallel Computers" [39], which was published in the now-defunct publication *Supercomputing Review*. This article attracted an astonishing amount of attention at the time, including mention in the *New York Times* [229]. Evidently it struck a responsive chord among many professionals in the field of technical computing who shared the author's concerns.

The following is a brief summary of the "Twelve Ways:"

1. Quote only 32-bit performance results, not 64-bit results, and compare your 32-bit results with others' 64-bit results.

2. Present inner kernel performance figures as the performance of the entire application.

3. Quietly employ assembly code and other low-level language constructs, and compare your assembly-coded results with others' Fortran or C implementations.

4. Scale up the problem size with the number of processors, but don't clearly disclose this fact.

5. Quote performance results linearly projected to a full system.

6. Compare your results against scalar, unoptimized, single processor code on Crays [prominent vector computer systems at the time].

7. Compare with an old code on an obsolete system.

8. Base megaflops operation counts on the parallel implementation instead of on the best sequential implementation.

9. Quote performance in terms of processor utilization, parallel speedups or megaflops per dollar (peak megaflops, not sustained).

10. Mutilate the algorithm used in the parallel implementation to match the architecture. In other words, employ algorithms that are numerically inefficient in order to exhibit artificially high megaflops rates.

11. Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.

12. If all else fails, show pretty pictures and animated videos, and don't talk about performance.

Subsequently, the present editor previously wrote a more detailed paper, which quoted actual passages and graphs taken from peer-reviewed work [40]. To avoid undue public embarrassment of the scientists involved, which was not the purpose of the article, the source articles were not explicitly given in that paper. This material was also presented in a session at the 1992 Supercomputing conference. These actions caused considerable consternation in some quarters at the time (and the present editor does not recommend repeating this experiment!), but many in the field appreciated these much-needed reminders of the pressing need for greater professional discipline in the field.

1.3 Examples from Other Scientific Fields

We do not need to look very far to see examples in other fields where well-meaning but sloppy practices in reporting and analyzing experimental results have led to distortion. One amusing example is the history of measurements of the speed of light [105]. Accurate measurements performed in the late 1930s and 1940s seemed to be converging on a value of roughly 299,760 km/s. After World War II, some researchers made more careful measurements, and thereafter the best value converged to its present-day value of 299,792.458 km/s (which is now taken as a standard, indirectly defining the meter and the second). No researcher seriously believes that the speed of light changed between 1930 and 1950. Thus one is left with the sobering possibility that sloppy experimental practices and "group-think" led earlier researchers to converge on an incorrect value.

Difficulties with experimental methods and dispassionate assessment of results have also been seen in the social sciences. In his book *The Blank Slate: The Modern Denial of Human Nature* [277], Harvard psychologist Steven Pinker chronicles the fall of the "blank slate" paradigm of the social sciences, namely the belief that heredity and biology play no significant role in human psychology—all personality and behavioral traits are socially constructed. This paradigm prevailed in the field until approximately 1980, when a wave of indisputable empirical evidence forced a change of paradigm. The current consensus, based on latest research, is that humans at birth universally possess sophisticated facilities for language acquisition, visual pattern recognition, numerical reasoning and social life, and furthermore that heredity, evolution and biology are major factors in human personality—some personality traits are as much as 70% heritable.

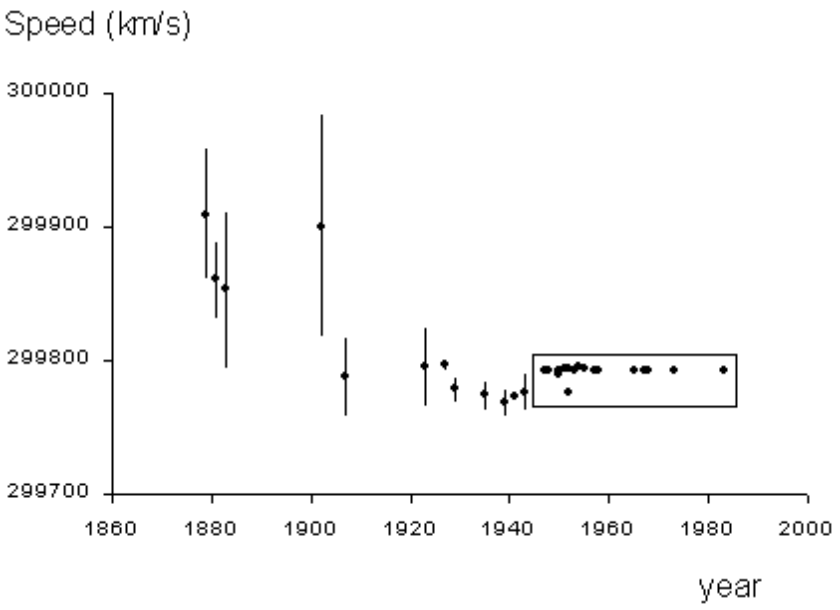

FIGURE 1.1: History of measurements of the speed of light.

Along this line, anthropologists, beginning with Margaret Mead in the 1930s, painted an idyllic picture of primitive societies, notably South Sea Islanders, claiming that they had little of the violence, jealousy, warfare, and social hangups that afflict the major Western societies. But beginning in the 1980s, a new generation of anthropologists began to re-examine these societies and question the earlier findings. They found, contrary to earlier results, that these societies typically had murder rates several times higher than large U.S. cities, and death rates from inter-tribe warfare exceeding those of warfare among Western nations by factors of 10 to 100. What's more, in many of these societies, complex, jealousy-charged taboos surrounded courtship and marriage. For example, some of these societies condoned violent reprisals against a bride's family if she was found not to be a virgin on her wedding night. These findings represented a complete reversal of earlier claims of carefree adolescence, indiscriminate lovemaking, and peaceful coexistence with other tribes.

How did these social scientists get it so wrong? Pinker and others have concluded that the principal culprit was sloppy experimental methodology and wishful thinking [277].

1.4 Guidelines for Reporting High Performance

When the pressures of intense marketplace competition in the computer world are added to the natural human tendency of scientists and engineers to be exuberant about their own work, it should come as little surprise that sloppy and potentially misleading performance work has surfaced in scientific papers and conference presentations. And since the reviewers of these papers are themselves in many cases caught up in the excitement of this new technology, it should not be surprising that they have, in some cases, been relatively permissive with questionable aspects of these papers.

Clearly the field of technical computing does not do itself a favor by condoning sloppy science, whatever are the motives of those involved. In addition to fundamental issues of ethics and scientific accuracy, the best way to ensure that computer systems are effective for scientific or engineering applications is to provide early feedback to manufacturers regarding their weaknesses. Once the reasons for less-than-expected performance rates on certain problems are identified, improvements can be made in the next generation.

Virtually every field of science has found it necessary at some point to establish rigorous standards for the reporting of experimental results, and the field of high-performance computing is no exception. Thus the following guidelines have been suggested to govern reporting and analysis of performance [40]:

1. If results are presented for a well-known benchmark, comparative figures should be truly comparable, and the rules for the particular benchmark should be followed.

2. Only actual performance rates should be presented, not projections or extrapolations. For example, performance rates should not be extrapolated to a full system from a scaled-down system. Comparing extrapolated figures with actual performance figures, such as by including both in the same table or graph, is particularly inappropriate.

3. Comparative performance figures should be based on comparable levels of tuning.

4. Direct comparisons of run times are preferred to comparisons of Mflop/s or Gflop/s rates. Whenever possible, timings should be true elapsed time-of-day measurements. This might be difficult in some environments, but, at the least, whatever timing scheme is used should be clearly disclosed.

5. Performance rates should be computed from consistent floating-point or integer operation counts, preferably operation counts based on efficient implementations of the best practical serial algorithms. One intent here is to discourage the usage of numerically inefficient algorithms, which may exhibit artificially high performance rates on a particular system.

6. If speedup figures are presented, the single processor rate should be based on a reasonably well tuned program without multiprocessing constructs. If the problem size is scaled up with the number of processors, then the results should be clearly cited as "scaled speedup" figures, and details should be given explaining how the problem was scaled up in size.

7. Any ancillary information that would significantly affect the interpretation of the performance results should be fully disclosed. For example, if performance results were for 32-bit rather than for 64-bit data, or if assembly-level coding was employed, or if only a subset of the system was used for the test, all these facts should be clearly and openly stated.

8. Due to the natural prominence of abstracts, figures and tables, special care should be taken to ensure that these items are not misleading, even if presented alone. For example, if significant performance claims are made in the abstract of the paper, any important qualifying information should also be included in the abstract.

9. Whenever possible, the following should be included in the text of the paper: the hardware, software and system environment; the language, algorithms, the datatypes and programming techniques employed; the nature and extent of tuning performed; and the basis for timings, flop counts and speedup figures. The goal here is to enable other scientists and engineers to accurately reproduce the performance results presented in the paper.

1.5 Modern Performance Science

In the chapters that follow, we will present a sampling of research in the field of modern performance science and engineering. Each chapter is written by a recognized expert (or experts) in the particular topic in question. The material is written at a level and with a style that is targeted to be readable by a wide range of application scientists, computer scientists and mathematicians. The material is organized into these main sections:

1. *Architectural overview* (Chapter 2). This chapter presents a brief overview of modern computer architecture, defining numerous terms, and explaining how each subsystem and design choice of a modern system potentially affects performance.

2. *Performance monitoring* (Chapters 3–5). These chapters describe the state-of-the-art in hardware and software tools that are in common use

for monitoring and measuring performance, and in managing the mountains of data that often result in such measurements.

3. *Performance analysis* (Chapter 6). This chapter describes modern approaches to computer performance benchmarking (i.e., comparative studies of performance), and gives results exemplifying what insights can be learned by such studies.

4. *Performance modeling* (Chapters 7–9). These chapters describe how researchers deduce accurate performance models from raw performance data or from other high-level characteristics of a scientific computation.

5. *Automatic performance tuning* (Chapters 10–13). These chapters describe ongoing research into automatic and semi-automatic techniques to modify computer programs to achieve optimal performance on a given computer platform.

6. *Application tuning* (Chapters 14–16). These chapters give some examples where appropriate analysis of performance and some deft changes have resulted in extremely high performance, in some cases permitting scientific applications to be scaled to much higher levels of parallelism, with high efficiency.

1.6 Acknowledgments

Writing, compiling and reviewing this book has involved the efforts of many talented people. In addition to the authors of the respective chapters, the editors wish to thank officials at Taylor and Francis for their encouragement and support. Also, this work was supported by numerous research grants, mostly from the U.S. Department of Energy, but also from other U.S. and Spanish research agencies. The particular agencies and grants are summarized in each chapter.

David Bailey is supported in part by the Performance Engineering Research Institute, which is supported by the SciDAC Program of the U.S. Department of Energy, and by the Director, Office of Computational and Technology Research, Division of Mathematical, Information and Computational Sciences, U.S. Department of Energy, under contract number DE-AC02-05CH11231.

Chapter 2

Parallel Computer Architecture

Samuel W. Williams

Lawrence Berkeley National Laboratory

David H. Bailey

Lawrence Berkeley National Laboratory

In this chapter we provide a brief introduction to parallel computer architecture. The intent of this material is to provide a survey of the principal features of computer architectures that are most relevant to performance, and to briefly define concepts and terminology that will be used in the remainder of this book. Obviously there is substantial literature on the topic of computer architectures in general, and even on the subtopic of computer architectures within the realm of high-performance computing. Thus, we can only provide a sampling here. For additional details, please see the excellent references [104, 164, 268].

We commence this chapter with a discussion of the fundamental forces enabling advances in architecture, proceed with a high-level overview of the basic classes of parallel architectures, and then provide a retrospective on

the challenges that have driven architecture to date. Along the way, we note particular challenges that face present-day designers and users of computer systems, such as the recent re-emergence of heterogeneous architectures.

2.1 Introduction

2.1.1 Moore's Law and Little's Law

In a seminal 1965 article, little noticed at the time, Intel founder Gordon Moore wrote

> The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. ... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. [247]

With these sentences, Moore stated what is now known as Moore's Law, namely that advances in design and fabrication technology enable a doubling in semiconductor density every 12 months (a figure later modified to roughly 18 months). Astonishingly, Moore's Law has now continued unabated for 45 years, defying several confident predictions that it would soon come to a halt. This sustained and continuing exponential rate of progress is without peer in the history of technology. In 2003, Moore predicted that Moore's Law would continue at least for another ten years [18]. As of the writing of this volume (summer 2010), it appears that at least an additional ten years are assured.

It should be emphasized that not all parameters of computer processor performance have improved. In particular, processor clock rates have stalled in recent years, as chip designers have struggled to control energy costs and heat dissipation. But their response has been straightforward—simply increase the number of processor "cores" on a single chip, together with associated cache memory, so that aggregate performance continues to track or exceed Moore's Law projections. And the capacity of leading-edge DRAM main memory chips continues to advance apace with Moore's Law.

Beyond the level of a single computer processor, computer architects capitalize on advances in semiconductor technology to maximize sustained performance, productivity, and efficiency for a given application (or set of applications), within constraints of overall system cost. Indeed one over-arching design objective of computer architecture is to ensure that progress in the aggregate computational power and memory storage of computer systems ad-