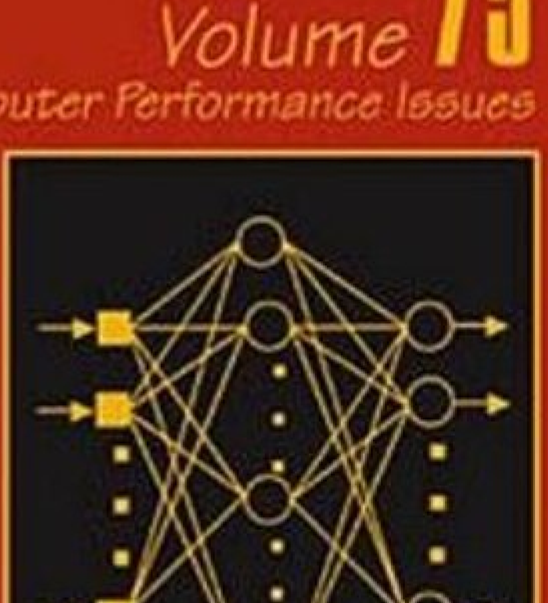
Advances in COMPUTERS

Volume 75

Computer Performance Issues



Edited by

MARVIN V. ZELKOWITZ

AP

Academic Press is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
84 Theobald's Road, London WC1X 8RR, UK
Radarweg 29, PO Box 211, 1000 AE Amsterdam, The Netherlands
525 B Street, Suite 1900, San Diego, CA 92101-4495, USA

First edition 2009

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-0-12-374810-2

ISSN: 0065-2458

Contributors

Dr. Cyntrica Eaton is an Assistant Professor of Computer Science at Norfolk State University (NSU) in Norfolk, Virginia. She received her B.S. in Computer Science from NSU in 2001 and her Ph.D. in Computer Science from the University of Maryland, College Park in 2007. She was awarded fellowships from the National Consortium for Graduate Degrees for Minorities in Engineering and Science, Inc. (GEM) and The David and Lucile Packard Foundation. Her general research interests include software testing and reverse engineering; her current focus is the application of these principles in web-based technology. She can be reached at cneaton@nsu.edu.

Dr. Martyn F. Guest is Director of Advanced Research Computing at Cardiff University. He received his B.Sc. in Chemistry from Sussex University in 1967 and his Ph.D. in Theoretical Chemistry, also from Sussex, in 1971. Following postdoctoral work at the University of Manchester, Dr. Guest joined the Science and Engineering Research Council in 1972, first at the Atlas Computer Laboratory (Chilton, Oxfordshire), and from 1979 at the Daresbury Laboratory near Warrington. He spent 3 years as Senior Chief Scientist and Technical Group Leader of the High Performance Computational Chemistry Group at Pacific Northwest National Laboratory (Richland, Washington). Dr. Guest returned to the UK in 1995 as Associate Director of the Computational Science and Engineering Department at Daresbury before taking up his present position at Cardiff University in March 2007. Prof. Guest's research interests cover a variety of topics in the development and application of computational chemistry methods on high performance computers. He is lead author of the GAMESS-UK electronic structure program and has written or contributed to more than 230 articles in the areas of theoretical and computational chemistry and HPC.

Dr. Christine Kitchen is Manager of the Advanced Research Computing (ARCCA) Division at Cardiff University. She obtained both her B.Sc. in Chemistry (1996) and Ph.D. in Theoretical Chemistry (2001) at Sheffield University. Following the award of her doctorate, Dr. Kitchen spent 3 years in the Molecular Modeling & Drug Design Group at AstraZeneca. Dr. Kitchen joined the Distributed Computing

(DisCo) group at the STFC Daresbury Laboratory in January 2004. Here, she undertook a variety of roles, from system administrator through to interactions with Universities and Industry, providing technical expertise to assist various sites with mid-range cluster computing. Dr. Kitchen started her current position at Cardiff University in September 2007.

Dr. Martin Herbordt received his B.A. in Physics from the University of Pennsylvania and his Ph.D. in Computer Science from the University of Massachusetts. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at Boston University where he directs the Computer Architecture and Automated Design Lab. Dr. Martin's overall research interests are in Computer Architecture and High Performance Computing. Since 2001, he has focused on high-performance computing using configurable logic, particularly in its application to computational problems in biology (Bioinformatics and Computational Biology), and in creating a development environment for those applications.

Dr. Atif M. Memon is an Associate Professor of Computer Science at the University of Maryland, College Park. He received his B.S., M.S., and Ph.D. in Computer Science in 1991, 1995, and 2001, respectively. He was awarded fellowships from the Andrew Mellon Foundation for his Ph.D. research and he received the NSF CAREER award in 2005. His research interests include program testing, software engineering, artificial intelligence, plan generation, reverse engineering, and program structures. He can be reached at atif@cs.umd.edu.

Dr. Justin Moore is a software engineer in the platforms group at Google. His research interests are in automated data center management, improved modeling techniques for power and thermal issues, and the integration of power and thermal considerations in large-scale resource allocation. Dr. Justin received his B.S. degree from the University of Virginia School of Engineering and his Ph.D. from the Duke University Department of Computer Science. He also serves on the Board of Technical Advisers for the Verified Voting Foundation and has advised state legislatures on election reform measures.

Mwaffaq Otoom is a Ph.D. Candidate in the Department of Electrical and Computer Engineering at Virginia Tech (since 2006). He received the M.S. in Computer Systems from Arizona State University in 2006 and the B.S. in Computer Engineering from Yarmouk University (Jordan) in 2003. His thesis topic is in the development of the concept of Workload Specific Processors.

Dr. Joann M. Paul is an Associate Professor in the Department of Electrical and Computer Engineering at Virginia Tech (since 2005). Prior to that, she was Research Faculty in the Department of Electrical and Computer Engineering at Carnegie Mellon University. Her research and teaching interests are in the simulation, evaluation, and architecture of single-chip heterogeneous multiprocessors.

Sean Pieper is a Software Engineer at Viable Inc. (since 2007). He received his M.S. in Electrical and Computer Engineering from Carnegie Mellon in 2004 and was a graduate student at the University of Wisconsin—Madison from 2004 to 2007 where he researched heterogeneous multiprocessors.

Dr. Partha Ranganathan is currently a principal research scientist at Hewlett Packard Labs. His research interests are in low-power system design, system architecture, and performance evaluation. His recent research focuses on designing cost- and energy-efficient systems for future computing environments (from small mobile devices to dense servers in data centers). His past research has focused on application-optimized architecture design (for database, media and communication applications) and performance, programmability, and simulation of parallel systems. He received his B.Tech. degree from the Indian Institute of Technology, Madras and his M.S. and Ph.D. from Rice University, Houston. He was a primary developer of the publicly distributed Rice Simulator for ILP Multiprocessors (RSIM). He was named one of the world's top young innovators by MIT Technology Review, and is a recipient of Rice University's Outstanding Young Engineering Alumni award.

Dr. Suzanne Rivoire is an assistant professor of computer science at Sonoma State University. Her research interests are in computer architecture and energy-efficient computing. Suzanne received her B.S. degree from the University of Texas at Austin and her M.S. and Ph.D. degrees from Stanford University. She serves on the editorial board of IEEE Potentials Magazine, which provides technical and career information to all IEEE student members.

Dr. Michael J. Schulte is an Associate Professor at the University of Wisconsin—Madison (since 2006). He received his B.S. degree in Electrical Engineering from the University of Wisconsin—Madison, and his M.S. and Ph.D. degrees from the University of Texas at Austin. His research and teaching interests include embedded systems, domain-specific processors, computer arithmetic, computer architecture, and digital system design.

Marc Somers is a Patent Examiner with the United States Patent and Trademark Office (since 2007). He received his B.S. in Computer Engineering from Virginia

Tech in 2005 with two minors in Computer Science and Mathematics and received his M.S. in Computer Engineering from Virginia Tech in 2007. He was awarded the Eagle Scout Honor in 2000.

Dr. Tom Vancourt earned his B.S. at the Cornell University and his M.S. and Ph.D. at the Boston University. His industrial experience includes positions at DEC, HP, and a number of smaller firms, and he has taught at the Boston University. He is currently a Senior Member of Technical Staff at Altera Corporation. His research interests include reconfigurable computing for accelerating many kinds of applications and tools to make reconfigurable computing's potential easier to achieve.

Preface

Welcome to volume 75 of the *Advances in Computers*. Every year, since 1960, several volumes are produced which contain chapters by some of the leading experts in the field of computers today. For almost 50 years these volumes have presented chapters describing the current technology and have shown the evolution of the field as it has evolved from the large mainframe, to smaller minicomputers, to the desktop PC to today's evolution of worldwide integrated networks of machines. In this present volume, we present five chapters describing new technology affecting users of such machines.

In this volume, we continue to discuss a theme presented last year in volume 72 – *High Performance Computing*. In volume 72, we described several research projects being conducted in the United States on the development of a new generation of high performance supercomputers. In Chapter 1 of this present volume, we present a similar survey from the perspective of the United Kingdom's efforts in this area. In ''The UK HPC Integration Market: Commodity-based Clusters'' by Christine A. Kitchen and Martyn F. Guest, the authors describe the development of supercomputers as commodity items available to the scientific community in the UK. The chapter is a detailed overview of some 30 different approaches toward building and evaluating such machines.

One of the continuing problems in building high performance computers is that as the speed of the processor increases, the heat generated increases, which requires increasingly larger cooling systems to make them operate. In Chapter 2, Tom Vancourt and Martin C. Herbordt in ''Elements of High Performance Reconfigurable Computing'' describe an alternative approach toward increasing computer capabilities. By using more inexpensive Field Programmable Gate Array (FPGA) processors as part of an inexpensive commodity PC, special purpose high performance machines can be constructed at a lower cost.

As stated above, one of the issues in the evolution of increasingly faster processors is the problem of heat generation and dissipation. In the United States, the power consumed by volume servers doubled between 2000 and 2006, and the Environmental Protection Agency predicts that it will double again between 2006 and 2011. How do we measure the heat generated and how do we measure its impact versus the speed and capabilities of the associated computer? In Chapter 3's ''Models and Metrics for Energy-Efficient Computing'' by Parthasarathy Ranganathan, Suzanne Rivoire, and Justin Moore, the authors discuss current efforts in energy-efficiency metrics and in power and thermal modeling, delving into specific case studies for each. They describe many such metrics and give examples of their use.

In Chapter 4, Joann M. Paul, Mwaffaq Otoom, Marc Somers, Sean Pieper and Michael J. Schulte in ''The Emerging Landscape of Computer Performance Evaluation'' describe models used to evaluate the performance of new technology, such as the machines described in Chapters 1–3 of this volume. Two common measures used are latency, meaning the time to complete a task, and throughput, meaning the time to complete a set of tasks. Decreasing one of these often has the effect of increasing the other. How do we apply these and other measures to the current generation of new technology? The authors develop a new taxonomy representing the user's perspective in viewing performance.

In the final chapter, Cyntrica Eaton and Atif M. Memon in ''Advances in Web Testing'' describe their current interests in testing web applications. With web applications often user input driven, it is often hard to generate effective testing scripts to test such applications. In their chapter they describe a taxonomy of features necessary to test in a web application and then give examples of how to develop such testing plans using various execution models, such as statecharts, object models, object-oriented models, and regular expressions.

I hope that you find these chapters of use to you in your work. If you have any topics you would like to see in these volumes, please let me know. If you would like to write a chapter for a forthcoming volume, also let me know. I can be reached at mvz@cs.umd.edu.

Marvin Zelkowitz
College Park, Maryland

The UK HPC Integration Market: Commodity-Based Clusters

CHRISTINE A. KITCHEN

Advanced Research Computing, Cardiff University, Redwood Building, King Edward VII Avenue, Cardiff CF10 3NB, Wales, United Kingdom

MARTYN F. GUEST

Advanced Research Computing, Cardiff University, Redwood Building, King Edward VII Avenue, Cardiff CF10 3NB, Wales, United Kingdom

Abstract

This chapter considers the ubiquitous position of commodity clusters in providing HPC capabilities to the scientific community, and the many issues faced by organizations when deciding how best to procure, maintain, and maximize the usage of such a resource. With a focus on developments within the UK, we provide an overview of the current high-performance computing (HPC) landscape from both the customer and supplier perspective. Historically HPC provision in the UK has been focused on one or two leading-edge national facilities that have helped the UK to develop and maintain an internationally competitive position in research using HPC. This HPC dynamic has, however, changed dramatically in the period 2005–2008 with the major injection of funding into University HPC sector through the Science Research Investment Fund (SRIF). This sector is now the major provider of HPC resources to the UK research base, with the capability of the sector increased 100-fold.

Our primary focus lies on the role of HPC Integrators in supplying resources into this sector, and the challenges faced by the HPC service providers themselves in sustaining and growing these activities. The host sites through partnership with the selected integrator aim to maximize this entire process, from procurement through system installation and subsequent lifetime of the service. We ask whether those integrators based primarily in the UK have the ability

ISSN: 0065-2458/DOI: 10.1016/S0065-2458(08)00801-2

to provide the necessary level of expertise required in all phases of the process, from procurement to ongoing support of the resource throughout its lifecycle.

We consider how current HPC technology roadmaps might impinge on the role of integrators in responding to the undoubted challenges that lie ahead. Crucial issues when considering potential integrator involvement include both size of the hardware solution, that is, number of cores, number of nodes, and the ongoing robustness of open-source software solutions that might be deployed on these platforms. Informed by developments over the past 24 months associated with the deployment of systems funded under SRIF, we provide an in-depth analysis of the current status and capability of a number of the leading HPC Integrators within the UK. Our primary attention is given to the three major companies who now supply the academic community and hence are well known to us—Streamline Computing, ClusterVision, and OCF. Seven other integrators are also considered, albeit with less rigor. Consideration is also given to several of the Tier-1 suppliers of clusters.

In reviewing the status of commodity-based systems in scientific and technical computing, systems representative of those supported by the integrators, we consider how an organization might best decide on the optimum technology to deploy against its intended workload. We outline our cluster performance evaluation work that uses a variety of synthetic and application-based floating-point metrics to inform this question. Our analysis relies on performance measurements of application independent tests (microbenchmarks) and a suite of scientific applications that are in active use on many large-scale systems. The microbenchmarks we used provide information on the performance characteristics of the hardware, specifically memory bandwidth and latency, and inter-core/interprocessor communication performance. These measurements have been extensively used to provide insight into application performance, with the scientific applications used being taken from existing workloads within the SRIF HPC sector, representing various scientific domains and program structures—molecular dynamics, computational engineering, and materials simulation to name a few.

The chapter concludes with a 10-point summary of important considerations when procuring HPC clusters, particularly those in the mid-to-high-end range.

1. Introduction

Over the past decade there has been a revolution in high-performance computing (HPC) spearheaded by a movement away from using expensive traditional proprietary supercomputers to systems based on relatively inexpensive commodity off-the-shelf (COTS) systems. A direct result of this transition is that the UK academic community has increasingly engaged in research using commodity HPC systems. Major investments into the University HPC sector have resulted in a dramatic increase in the deployment and use of commodity clusters within higher education institutions (HEIs). Today this commodity University HPC sector is by far the dominant provider of HPC resources to academics in the UK.

The chapter is structured as follows. In Sections 2 and 3 we provide a somewhat extended overview of both current and future HPC landscapes. Section 2 should be read in conjunction with the excellent overview of HPC research and development in the USA presented in a previous volume of Advances in Computers [1]. In contrast, however, the present chapter focuses on developments within the UK. Historically the majority of HPC resources were provided by proprietary computer systems via National Supercomputing Centers. These centers are still in operation today but the UK now has a more balanced HPC ecosystem which adds to the resources (both human and capital) available to UK research staff. In considering this landscape, we review briefly developments over the past 12 months associated with the deployment of systems funded under SRIF, the Science Research Investment Fund. We consider the multitude of issues faced by an organization when deciding how best to procure, maintain, and maximize the usage of any associated HPC resource. Our focus lies not only on the role of HPC Integrators in supplying resources into this sector, but also the challenges faced by the HPC service providers themselves in sustaining and growing these activities. Historically many organizations have responded to this challenge by forming partnerships with the associated vendor of the hardware in looking to maximize this entire process. How effective are HPC Integrators in such a partnership, and do such organizations in the UK have the ability to provide the necessary level of expertise required in all phases of the process, from procurement, through installation onto ongoing support of the resource throughout its lifecycle? While these are open questions, there is no doubt that investment in the highly skilled development and support staff fundamental for the efficient management and exploitation of such COTS clusters has not kept pace with the investment in capital equipment.

Sections 4 and 5 provide an in-depth analysis of the status and capability of the suppliers of clusters in the UK, focusing in Section 4 on the leading HPC Integrators and the three major companies who supply the academic community and hence are well known to us—Streamline Computing, ClusterVision, and OCF. To complete this overview, we also consider the smaller, less-known organizations, albeit through information obtained in large part from the cluster integrator's official Web sites. Section 5 considers several of the Tier-1 suppliers of clusters—companies who do not rely on integrators as part of their solution. This analysis has been conducted by a variety of mechanisms that are detailed at the appropriate point. We consider throughout how the associated technology roadmaps might impinge on the role of integrators in responding to the undoubted challenges that lie ahead. We will see that crucial issues involve both size of the hardware solution, that is, number of nodes, and the ongoing robustness of open-source software solutions that might be deployed on these platforms.

In considering the status of commodity-based systems in scientific and technical computing—systems representative of those supported by the integrators—we outline in Section 6 our own cluster performance evaluation work that uses a variety of synthetic and application-based floating-point metrics. Our analysis relies on performance measurements of application independent tests (microbenchmarks) and a suite of scientific applications that are in active use on many large-scale systems. The microbenchmarks we used provide information on the performance characteristics of the hardware, specifically memory bandwidth and latency, and intercore/interprocessor communication performance. These measurements have been extensively used to provide insight into application performance, with the scientific applications used being taken from existing workloads within the SRIF HPC sector, representing various scientific domains and program structures—molecular dynamics, computational engineering, and materials simulation to name a few.

Finally, we provide a 10-point summary of the findings of this review in Section 7.

2. Commodity Clusters and HPC

2.1 The HPC Clusters of Today

HPC using clusters has come a long, long way from its early days. Then, a cluster was a network of disparate workstations harnessed together into a parallel virtual machine (PVM) computation [2]—nascent Beowulf clusters consisted of cheap tower PCs literally stacked up on shelves. The Unix used in those pre-Beowulf days almost certainly belonged to a vendor. Little in the arrangement was free, very little was open source, but it was still the most cost-effective route to compute cycles in large quantities.

Now, of course, using clusters to do HPC is absolutely ubiquitous. Cluster computers have ''always'' been used to perform research in physics, chemistry, mathematics, engineering, etc., but over the last 3 or 4 years, the list of applications has expanded to include biology, economics, sociology, music composition, mainstream gaming, political science, and medical research of all kinds. Even in the world of business, a similar expansion has occurred, with clusters assuming ''traditional'' HPC tasks (largely inherited from their counterparts in academia) in engineering, simulation, and design, and most recently to do advanced modeling, visualization and graphic arts, the making of movies, and far more.

Six or seven years ago, commodity cluster computers—computers built out of COTS parts—would appear, however briefly, in the ''Top 500'' list of supercomputers [3]. Many useful trends in architectural changes over time are available in the list; published twice a year and publicly available at www.Top500.org, the list ranks the most powerful computer systems according to the Linpack benchmark rating system and is an industry respected report which indicates usage trends in computing and interconnect solutions. Now cluster computers of one sort or another are the Top 500 (see Fig. 1)—one can argue whether all of the Top 500 are strictly made with COTS components *per se*, but even the exceptions are usually architected using ''generic'' components that are, at heart, just PCs on a network, regardless of the complexity of the end system. While one can argue over the exact ratio, it is worth

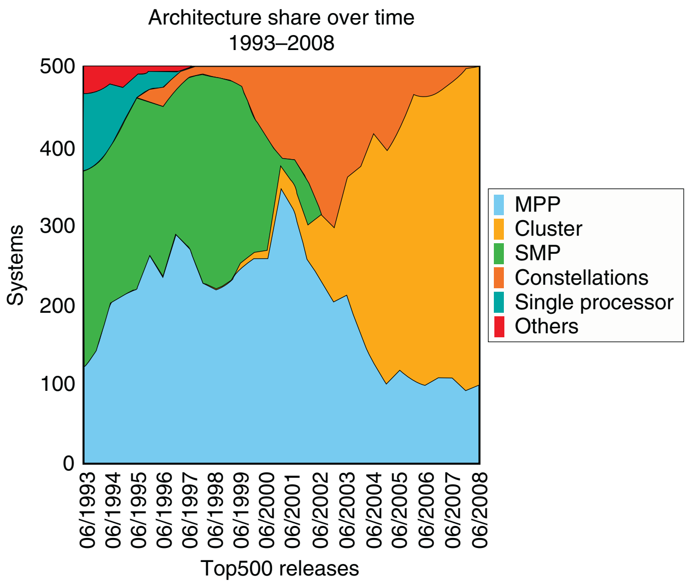


FIG. 1. The evolution of supercomputer architectures in the period 1993–2007.

stressing that commodity clusters based on open-source software (i.e., Beowulf clusters) are far more cost effective than clusters based on proprietary solutions.

An indisputable fact is that the majority of all HPC cluster computers run Linux. Linux has absolutely dominated cluster computing since the mid-1990s, when the Beowulf project in particular demonstrated how easy and cheap it was to purchase COTS computers and turn them into a supercomputer—at a 10th of the cost per floating-point instruction per second (flops) of its ‘‘big iron’’ equivalent from supercomputer vendors and half the cost of network clusters made from proprietary Unix-based systems. Equally telling is the observation that the second-most popular cluster operating system is probably held by FreeBSD, another open-source operating system. ‘‘Open-source operating system’’ is essentially synonymous with cluster computing for good and fairly obvious reasons (see Section 2.2).

Clusters of every size and for nearly any purpose imaginable are clearly here to stay—they are an established part of mainstream computing in both the academic and corporate world. In all cases, cluster computing using COTS components forms an inexpensive, accessible, and scalable route to the kind of computing power required to be competitive in academia and research or world markets. We consider below the future of cluster computing—how will it differ from the way it is today and how it has been in the past?

2.1.1 Hardware and HPC Cluster Performance

The performance of cluster computers on ‘‘most’’ tasks is bounded by one of several relatively simple bottlenecks that appear in the nodes of nearly every cluster design. In order of importance, the bottlenecks [4] are:

1. The central processing unit’s (CPU’s) number of instructions per second, usually floating-point instructions.
2. Memory bandwidth and latency, how efficiently memory accesses can be parallelized with CPU function, and how much memory can be addressed per task thread.
3. Network bandwidth and latency, how efficiently can network utilization be parallelized with CPU and memory function, and how factors such as network topology and details of the network architecture act as discrete rate-limiting parameters.
4. Reading and writing to a ‘‘hard store,’’ factors that include disk bandwidth, latency, and total capacity.
5. Packaging, such as towers, rackmounts, blades, etc.

We consider each of these below; given the breadth of architectural options available, the discussion will necessarily be rather brief.

2.1.1.1 The CPU.

Developments over the past few years have led to processors with the following features:

1. *64-bit*. Very few 32-bit CPUs are now going into new clusters. Until recently the struggle for dominance between Intel and AMD remained fairly evenly balanced. The AMD64 family are cheap, COTS, mainstream 64-bit CPUs. Intel's EM64T was initially more expensive, although competition and time has in part corrected this; today systems featuring EM64T processors dominate the Top 500. Other 64-bit CPUs include the Power PC G5, plus 64-bit variants of non-COTS CPUs (such as SPARC). While 128-bit system buses are available (including Itanium2), these are likely to remain peripheral to HPC cluster computing during the foreseeable future, apart from a few niche areas.
2. *Multicore*. This remains the dominant development over the past couple of years, and is likely to remain so. Following the successful role out of dual-core solutions from AMD and Intel, late 2007 saw the arrival of AMD's Barcelona quad-core processors and the corresponding family of processors from Intel—Clovertown, Harpertown, etc. While multicore developments appeared to have stalled beyond quad-core—with the problems experienced by AMD with its Barcelona processor, and Intel's Roadmap stalling beyond 8-way—the recent announcement by Intel and AMD of the Larrabee and Fusion processors, respectively, has rekindled expectations [5].
3. *Vector support*. This term covers a variety of capabilities—Streaming SIMD Extensions (SSE 1, 2, 3) support, 3DNow! instructions, embedded or attached digital signal processor (DSP) units, and VMX/Altivec.
4. *Multilayered cache*. In particular, the addition of an L3 cache to mainstream processors, on top of the already standard L1 and L2 caches. This process is now commonplace, and is necessary to help narrow the ever widening gap between processor speed and memory speed. Typically, L1 and L2 are on the actual CPU core and L3 sits between processor and memory, but this is very much open to change in future designs.

The advancing level of transistor integration is producing increasingly complex processor solutions ranging from mainstream multicores, heterogeneous many-cores, and also special purpose processors including GPUs. There is no doubt that this advancement will continue into the future until Moore's Law can no longer be satisfied. This increasing integration will require improvements in performance of the memory hierarchy to feed the processors. Innovations such as putting memory on-top of processors, putting processors on-top of memory (PIMS), or a combination of both may be a future way forward. However, the utility of future processor generations will be a result from demonstrable increases in achievable performance from real workloads.

Power consumption, heat production, and CPU clock continue to present serious problems, with CPU and memory design having reached something of a power dissipation, clock speed-scaling crisis as ever more transistors switch ever more times in a limited volume. Features that reduce power during idle periods are of limited use given typical cluster utilization patterns, while features that reduce clock during busy periods act only to seriously impact on performance expectations.

Indeed, there is a growing body of opinion that future processor development is moving away from satisfying the needs of HPC users and that the HPC market needs to look to technical developments in the areas of coprocessors, FPGAs, GPUs, Cell processors, etc. While companies such as Nallatech, ClearSpeed, and nVidia offer significant performance acceleration through such technologies, it remains too early to say whether these developments will become mainstream given the level of effort currently required to deliver a major performance advantage over commodity processors. We refer the reader to the chapter on reconfigurable computing and the use of FPGAs in this volume for an in-depth assessment of the current state of play [6].

2.1.1.2 Memory. The standard DDR memory has largely been superseded by DDR-2 memory that at least helps multicores to function without a significant increase to the already wide gap between CPU clock scaling and memory speed scaling. There are many HPC tasks that are bottlenecked by memory speed for single-CPU systems; others are bottlenecked by memory size available to a single CPU. The disparity will only continue to rise geometrically over the next few years.

CPU memory architecture is also a critical difference in the various multicore architectures. Each has its own solution to the CPU memory bottleneck (and its associated problem, pipeline stalls in ever-longer pipelines). Intel’s long standing solution is hyperthreading, a processor-centric approach with a fairly traditional bridge, which, when combined with multiple cores, permits efficiently parallelized access to heterogeneous resources. Note that hyperthreading is less useful in a homogeneous tasking environment like an HPC cluster, where all the processors (or processor cores) in a system are more likely to be in contention for a single resource more often than trying to access different resources at the same time. AMD’s solution—HyperTransport—can be thought of as placing CPU, memory, and other peripherals on a very small, low-latency, high-bandwidth network, avoiding the bridge concept altogether.

2.1.1.3 System Packaging. There are currently three generic cluster package formats: Tower units, Rackmount units, and Blade clusters. Tower units stacked on ordinary shelving (or the floor) will continue to be popular in small clusters, home clusters, and clusters with very tight budgets. Rackmount units, usually 1U enclosures containing one to four processors, will dominate ‘‘professional’’ grade clusters and larger clusters with prograde infrastructure support in the cluster server rooms. Finally, blade clusters will continue to gain ground in this latter market, but the fact that fully loaded blade enclosures are relatively expensive per CPU, difficult to reliably power and cool (at something like 12 kW/m^2 of occupied floor space), and come with only limited configurability (such as networking) will also continue to limit their growth.

2.1.1.4 Connectivity.
Cluster connectivity has witnessed major changes over the past 2–3 years, with the increasing role of InfiniBand as the communication fabric of choice. While GBit Ethernet (especially its switches) continues to get cheaper and has become the lowest-common-denominator networking standard for vanilla clusters and grids, InfiniBand has increasingly come to dominate the cluster/grid marketplace. These developments have seen some of the traditional suppliers of leading-edge cluster networks rapidly lose both market share, and the ability to fund competitive development. Key technologies that have effectively been displaced include Myricom’s Myrinet [7], Dolphinics’s scalable channel interface (SCI), and Quadrics QsNet [8, 9].

InfiniBand’s position as the fastest growing cluster interconnect—witnessed by the 29th edition of the Top 500 list—plateaued somewhat over the past 12 months. Peaking at 130 InfiniBand-connected supercomputers (26% of the list) in June 2007—260% more than the 36 supercomputers in the June 2006 list-this number of systems remained at 121 in the both the 30th and most recent, the 31st, appearance of the list.

InfiniBand-based clusters continue to take share from clusters using proprietary interconnect solutions as industry-standard InfiniBand is proven to deliver higher scalability, efficiency and performance, especially when connecting multicore computing servers and storage systems. Recent features included the rapid uptake of 20-Gb/s InfiniBand (DDR) usage on the list—this increased significantly to 55 clusters in the 29th edition of the list from only a single cluster 6 months earlier, highlighting the fast adoption of 20-Gb/s technology. Highlights of InfiniBand usage in the June 2008 list include:

– InfiniBand is the most used high-speed, low-latency interconnect and has the greatest increase in the rate of usage, with 24% of the systems in the list.
– The average efficiency of all reported InfiniBand-based supercomputers is significantly higher than that of Gigabit Ethernet-connected clusters—71% compared to 54%, respectively, in the June 2007 list.

– Higher-performing InfiniBand-based supercomputers are replacing systems using proprietary interconnect technologies such as Myrinet (87 systems in June 2006 down to just 12 in June 2008) [8, 9] and lower-performing Gigabit Ethernet.
– The above trend is mirrored by the decline in Quadrics-based systems [8, 9]—from 14 systems in June 2006 to just 5 in June 2008.
– 20-Gb/s InfiniBand is widely used by multicore systems as they impose high demands on the interconnect, and require a scalable and reliable I/O solution.

2.1.1.5 Hard Storage.

For many HPC clusters, hard storage is not a terribly important issue. For others, it is the critical component. For applications in genomics, image rendering for movies, visualization and simulation, and processing accelerator data, the issues focus on moving from today's terabyte stores to tomorrow's petabyte stores in some sort of scalable way. Important technologies for storage are storage area networks (SANs), network-attached storage (NAS), Fiber Channel, and RAID. Cluster Parallel File Systems are achieving greater levels of reliability and robustness, with a number of mature products now in widespread use from, for example, IBM (GPFS), Panasas, Isilon, DataDirect Networks, Hitachi (HSM), Sun (Lustre), etc.

2.1.1.6 Future Developments.

When considering future developments, it is essential to compare the price/performance of the spectrum of hardware alternatives across the various new designs both as they first come to market (when they tend to be relatively costly) and later, as demand and competition stabilize prices and the technology enters the mainstream. Price/performance, in turn, will be dictated by how well each of the competing designs manages access to resources shared by all the processor cores. We would again note the major shift in the semiconductor industry strategy that has become readily apparent during the past 3 years, whereby increased processor throughput is accomplished by providing more computing elements (''cores''), rather than by increasing the operating frequency. Indeed, the 5-year semiconductor industry roadmap only showed a factor of 2 increases in operating frequency from 3.2 GHz (2004) to 6.4 GHz (2009), whereas Intel, AMD, and others are migrating to multiple (2×, 4×, 8×,. . .) cores within a single ''processor.'' Experience has already shown that balance on multicore processors can be a significant issue; this change in processor design has important implications for petascale computing (see Section 2.3).

We would again point to future technical developments in the areas of coprocessors, FPGAs, GPUs, Cell processors, etc. Whether these developments will become mainstream remains in doubt given the level of effort currently required to deliver a major performance advantage over commodity processors. Finally we note that an alternative trend aimed at improving performance/Watt and total cost of operation will be to deploy less powerful processors that use much less power, and constrained memory and I/O. Such a trend will mean that many more processors may be required to reach a desired aggregate performance, further increasing the burden on programmers and users of HPC systems.

This introduction has omitted a key factor in comparing alternative cluster solutions—how a given application performs on the cluster architecture in question. We return to this point in Section 6.

2.2 Supercomputing and Clusters

Today's supercomputing choices are the product of commodity market competition, technology evolution, historical hardware and software legacies, and leadership choices within industry. Computer vendors, driven by developments such as DOE's Advanced Simulation and Computing (ASC) Program (formerly ASCI, Accelerated Strategic Computing Initiative [10]), have aggressively pushed the performance levels of parallel supercomputers higher and higher.

Given the economies of scale, it is clear that the immediate future of supercomputing will be dominated by parallel supercomputers built with commodity compute servers, such as those used as Web servers, tied together by a high-performance communications fabric. Although currently on this plateau in the evolution of parallel supercomputer architectures, this will not last long. New architectures are already on the drawing boards that will be capable of a quadrillion arithmetic operations per second (Pflops). Such computers cannot be built using the same technology in today's Tflops computers—they would require too much space and consume too much power.

2.2.1 Custom and Commodity Clusters

The evolution of cluster technologies has proceeded on two fronts: in the first development, beginning in the late nineties, IBM, Compaq, and SGI—among others—began creating *proprietary clusters* using their shared-memory servers and custom-designed or semicommodity networks. The IBM approach has been to use their own custom multilevel switch fabrics to interconnect shared-memory nodes based on their Power workstation processors. Such nodes have been deployed on the UK's HPCx service—initially 32-way Regatta p690 and p690+ nodes comprising power4 and power4+ processors, and more recently power5 p5-575 nodes comprising 16 processors.

At the same time, true *commodity clusters* were being built and deployed based on uniprocessor or dual-processor nodes utilizing Compaq Alpha or Intel X86 processors. These clusters used mainly semicommodity Myrinet [8, 9] interconnects from Myricom; but smaller examples were sometimes based on gigabit (or slower) Ethernet switch fabrics. In all cases, the system software was built around the Linux open-source operating system.

Moving to the present, more than 80% of the systems in the June 2008 Top 500 list are based on either Intel or AMD processors. The most dominant player is clearly Intel's EM64T—the number of systems featuring this processor has risen from 117 in June 2006 to 356 in June 2008. While this is part reflects a natural replacement for the corresponding IA32 processor, with the number of associated systems declining from 148 to 3 over the same period, it contrasts sharply with the fortunes of AMD who, in part because of the problems with their quad-core Barcelona processor, saw the number of systems housing their processors decline from a maximum of 113 in the November 2006 list to just 55 in June 2008. While Intel dominate the X86 landscape, it is fair to say that their Itanium IA64 architecture has been less successful, with a decline in systems from 31 June 2006 to 16 June 2008.

Also in sharp decline is the number of systems based on proprietary solutions from HP and Sun—with no system in the June 2008 list featuring either PA-RISC or SPARC processors. Finally we would note that while IBM's power processor continues to feature in the top supercomputers, the processor share has decreased from 18% (91 systems) in November 2006 to 14% in June 2008 (68 systems).

Few of these clusters—custom or commodity—have system balance between computation and communications that is competitive with that found historically on true MPPs such as the Cray T3E. Nevertheless, for many important classes of applications, they are capable of achieving high parallel efficiency on a thousand processors or more. They also, in general, lack full-system reliability, availability, and serviceability (RAS) features. For truly large systems, this has caused difficulties in running large jobs with long execution times. In addition, few of the large clusters deployed have truly scalable system software, that is, can provide service to dozens of simultaneous users and have fast, scalable system boot-up, and executable loading capabilities.

Many of the overall trends apparent in the Top 500 are reflected in the leading HPC systems in the UK, with Table I presenting details of the 10 leading UK HPC systems. Five of these systems reside in UK Universities, major beneficiaries of recent investments enabled by the SRIF. The combined computational power of these systems (106 Tflops) is seen to be almost double that of the UK's leading National supercomputer, HECToR (54.6 Tflops). Only three of the Top 10 systems exhibit efficiencies ($R_{\text{max}}/R_{\text{peak}}$) greater than 80%—the two Cray systems—HECToR and the Cray XT3 system at AWE—plus the Merlin InfiniBand cluster

TABLE I
THE 10 LEADING HPC SITES IN THE UK (TOP 500 RATING, JUNE 2008)

Rank (UK)	Rank (Top 500)	System[a]	Site	Processor	R_{max} (Gflops)	R_{peak} (Gflops)	Efficiency (R_{max}/R_{peak}) (%)	Vendor
1	18	Power 575, p6 4.7 GHz, InfiniBand	ECMWF	8320	80,320	156,416	51	IBM
2	29	*HECToR*: Cray XT4, 2.8 GHz DC	HECToR	11,328	54,648	63,437	86	Cray, Inc.
3	55	Cray XT3, 2.6 GHz DC	AWE	7812	33,929	40,622	84	Cray, Inc.
4	66	*BlueCrystal*: x3450 Cluster Xeon 2.8 GHz QC, InfiniBand	Bristol University	3360	28,775	37,632	76	ClusterVision/ IBM
5	85	BladeCenter JS21 Cluster, PPC 970, 2.5 GHz DC, Myrinet	Reading University	2800	20,051	28,000	72	IBM
6	88	*Merlin*: NovaScale R422 Cluster, Xeon E5472, 3.0 GHz QC, InfiniBand	Cardiff University	2048	20,000	24,576	81	Bull
7	99	PowerEdge 1950, Xeon 51xx 3.06 GHz DC, InfiniPath	UCL	1992	18,930	24,382	78	Dell/ ClusterVision
8	108	BladeCenter HS21 Cluster, Xeon DC 3.0 GHz, GigE	Financial DCCoD (B)	3840	18,600	46,080	40	IBM
9	112	*Darwin*: PowerEdge 1950, 3.0 GHz DC, InfiniPath	Cambridge University	2340	18,270	28,080	65	Dell/ ClusterVision
10	120	Cluster Platform 3000 BL460c, Xeon 54xx 3.0 GHz QC, GigE	Food Industry	2704	17,567	32,448	54	HP

[a] QC, quad-core; DC, dual-core.

at Cardiff University. While one would expect higher efficiencies from InfiniBand-based supercomputers than those connected with Gigabit Ethernet (40–54% in Table I), the value for the InfiniPath-based Darwin cluster (65%) appears lower than might be expected.

Open-Source Solutions. Open-source developments have yielded significant enhancements to the state of the art in operating systems (cf., Linux OS) and tools (cf., Apache). Multiple accretions of open software mass have resulted in profitable enterprises that combine these tools into single offerings with support. In addition, there exist many efforts to build clustering tools (cf., LLNL Chaos, LANL Clustermatic, Sandia CPLANT, NPACI Rocks, OSCAR, and others) that extend the desktop environment to medium- and high-performance computing. The potential deployment of open-source solutions in satisfying requirements of high-end, ultrascale technical computing is more an open question. What is clear is that current trends in developing and implementing ultrascale computers fall well below the requirements capable of addressing many scientific challenges. Nevertheless, it appears inevitable that ultrascale platforms will be developed using—among other things—commodity-based hardware components and open-source software. It is clear, however, that a major investment is required in developing these software components, technologies, and enhancements to attain the 100× to 1000× in computational capability and capacity that is vitally needed to address scientific requirements.

In a balanced computational environment, hardware and software improvements have contributed equally to the increase in applications performance. What matters most is the shortest time to solution, with code development efforts definitely on the critical path to success. The delivery of fully integrated heterogeneous, multivendor parallel computing environments pose considerable challenges from system administration to application development to resource management. If these challenges are beyond current Tier-1 vendors, assessing the potential of HPC Integrators to respond to the same challenges raises a number of issues that need to be addressed.

2.3 Operational Challenges with Large Systems

When delivering and supporting the world's largest systems (e.g., those deployed in the American Department of Energy's Advanced Simulation and Computing Program (ASC—formally ASCI) systems—Red, Blue Mountain, Blue-Pacific, White, and Q), the associated sites encountered numerous problems with the software environment. These ''defects'' are often difficult to isolate down to root cause. In addition, many of the vendor partners providing these platforms do not have systems even within 1/10 the size of these platforms for debugging and patch verification. If a Tier-1 vendor does not have these capabilities, there is little or no chance that an integrator could claim to be able address this shortcoming. All these factors contribute to the process of integrating large systems, resulting in inordinately long periods of ''debugging'' time (over a year). Several lessons learned in these efforts suggest that HPC goals might best be served by open-source development efforts where the end sites and their associated communities directly participate in the development and/or testing and/or integration of the technology on platforms at scale.

These conclusions are applicable to the broader high-end technical computing (HEC) community for the following reasons:

- HEC customers require the ability to have operating system and tools source code for root-cause analysis and debugging. In many cases—at least in the USA—the HEC customer sites have the requisite in-house expertise not only to do root-cause fault isolation, but also to formulate and implement bug fixes (that may possibly include rearchitecting portions of the solution). With open-source community development efforts, these fixes can be fed back into the community for the benefit of all.
- Taken as a group, vendors of proprietary solutions indicate that they cannot make a profit providing solutions that span the entire HEC space. Thus HEC sites are left with two options: live with mediocre solutions that fulfill only part of their requirements, or implement major portions of the solution in-house on top of the vendor provided proprietary foundations. Neither of these two options is advantageous for the HEC or vendor community in the long run. With the open-source community development model, HEC sites can contribute their solutions back to the community because it is based on a common software base and can be contributed back without fear that the development would benefit only one particular vendor offering (and thus inhibit competition in the future and lock the HEC site into one proprietary vendor foundation). With this wide HEC scope, the diversity of the debugging environment and developed solutions benefits the entire community.
- Change has been a dramatic feature of the HPC landscape for many years. However, recent introductions of many disruptive technologies (e.g., killer micros, IA-32 commodity hardware, and open-source software) have radically increased the rate of change in the industry. As a result, the time span for vendor support of provided hardware and software platforms has been reduced to shorter than the full lifecycle of implemented solutions at HEC sites. In addition, timescales under which support is withdrawn (from announcement to withdrawal of support) by vendors (measured in quarters) is typically much shorter than the governmental planning period for replacements (measured in years). This mismatch in timescales has led HEC sites to require open-source-based products as a hedge against ''change of support'' status.

Processors Versus Processing Elements. As noted above, a major shift in the semiconductor industry strategy will lead to increased processor throughput being accomplished through more computing elements (''cores''), thereby increasing the burden of parallelization. Future baseline architectures for 100 Tflops and 1 Pflop might thus appear as below:

- 100 Tflops:
  - 5 Gflops each ''processing element''
  - 20,000 processing elements
- 1 Pflop:
  - 10 Gflops each processing element
  - 100,000 processing elements

This increased reliance on parallelization and hence system size will merely act to emphasize the operational challenges associated with large systems, challenges that stretch the resources of proprietary vendors to the limit and are realistically beyond the reach of the HPC Integrators central to this chapter.

2.4 Capacity Through Beowulf Clusters

The success of ASC and international HEC efforts to deliver complex modeling capability to the community has led to an overwhelming demand for ''capacity at the capability level.'' Now that large-scale predictive scientific simulation has been adopted as a critical component of scientific endeavor, there is great demand to run parallel applications for parameter studies, convergence studies and the like at small to medium scale of parallelism. These simulations run at a scale of parallelization that was a ''capability run'' for previous generations of ASC platforms. However, rather than one or even a few capability runs, literally hundreds of delivered Tflops of computation are required for these production calculations.

There is now widespread debate on how best to meet these crushing capacity demands. Although the strategy—and indeed a detailed analysis of the requirements—is still ongoing, one aspect is certain: the solution will be dependent on commodity clusters that are based on open-source software (i.e., Beowulf clusters). Again, this is being driven by the fact that Beowulf clusters are more cost effective than clusters based on proprietary solutions. It would seem clear from the outset that the role of system integrators in providing this scale of resource is far more credible than is the case for their involvement at the very high-end, that is, in the capability regime.

2.5 Power and Space

With both academic and commercial organizations needing more computer systems to run today's HPC applications, high-density racks are growing increasingly popular. However in many cases computer rooms are simply running out of space given the rapid growth in hosting more and more systems over the past decade. Consolidating server farms through the deployment of faster, multicore processors and ultradense board designs into tall racks stacked with dense servers and even denser blades provides an attractive route to maximizing what space is available. The negative side of such deployment is, of course, that heat very quickly becomes a major problem for system administrators [11]. While brand-new data centers with ventilation and cooling systems specially designed to accommodate faster, hotter deployments are designed to deal with such a scenario, the majority of facilities are not new, but are full of legacy systems, each with its own particular thermal management constraints.

Excessive heat can introduce a multitude of problems. Typically IT administrators arrange computer systems by trying to alternate ''hot'' and ''cold'' aisles within the center, that is, they try to distribute exhaust (hot) and intake (cold) aisles throughout the room to avoid hot spots that can push ambient temperatures beyond acceptable limits. Indeed, a growing problem with high, hot racks is the tendency for some systems to emit so much hot air that the computer room's ventilation system simply cannot remove it all. Too often, some of that hot air leaks into the cold aisle, recirculating back into the system and making the rack run hotter than needed. That phenomenon brings with it significant costs:

- *Impact on hardware*. System failures and shortened lifespan of components are an inevitable consequence of repeated exposure to temperatures that exceed prescribed tolerances. Dense 30-kW racks are growing more popular in today's data center—without an effective cooling system, that rack can reach unsafe temperatures in minutes.
- *Impact on energy costs*. As computer rooms grow hotter, they cost more to power and cool. The Green Grid, a nonprofit consortium dedicated to advancing energy efficiency in data centers (http://www.thegreengrid.org/home), has established a metric by which organizations can estimate the total impact of a system's power demands—including the cost of cooling the system. The power usage effectiveness (PUE) metric is essentially a ratio that that provides a helpful multiplier for understanding the ''fully loaded'' energy cost of a system.

Because both system density and processor speeds are increasing—suggesting that heat dissipation will remain a challenge for years to come—both vendors and end-users have pursued a variety of approaches to cooling. Gaining popularity on all fronts is the use of water to carry heat away from systems and the data center at large.

Liquid Cooling is not new. For most modern implementations, water remains the coolant of choice. In a computing system, fans blow heat at a water coil, which takes up the heat and either carries it away or cools it before the air reaches the data center's ambient environment. Some solutions send heated water to ''chillers'' or a central chilling station which safely dissipates the heat and pumps chilled water back to the coils in the rack systems. Plumbing runs beneath the data center floor, sharing the airflow space with cabling. Several types of water cooling implementations are available from the major system vendors, and even from companies who specialize in providing add-on cooling solutions. All of them operate on the same fundamental approach to thermal exchange, even if their particular approach differs.

Today's water-cooled thermal management solutions are designed to solve the thermal challenges posed by systems based on air-cooled, industry-standard components. As it happens, these are the very systems that are driving the majority of the growth in the HPC sector, and their popularity will in turn prompt continued refinements in cooling solutions. Because, as we have established, the future is all about density. Meanwhile, some companies are looking at more specialized approaches to bring liquid cooling further into the rack. Current techniques range dramatically. Some systems spray coolants directly onto heat-generating components such as CPUs. Others use liquid metal to conduct heat away from heat sources. They are all after the same thing as water cooling: to minimize the problems and commensurate costs associated with excess heat.

3. The UK HPC Landscape

Much of the discussion to this point has focused on the technology of cluster computing and associated issues. We now consider the research and scientific drivers that are capitalizing on these advances in technology, with a focus on developments in the UK.

Historically, the UK has performed exciting and innovative scientific research across a vast range of fields, from aerospace to drug design [12]. For these developments to continue, as well as the UK enhancing its position amongst the world leaders in computational science and engineering, there must be significant investment in the development of relevant skills. In addition researchers must be provided with access to a balanced and flexible high-end computing infrastructure that is amongst the best in the world, with regular upgrades to keep pace with the accelerating rate of technological advance. Continuing funding is necessary because simulation is becoming a bigger part of science and engineering. Provision of resources must be made at a variety of levels from the University upward. At the highest end, although many challenges exist, the UK aims to achieve sustained petascale performance as early as possible across a broad field of scientific applications, permitting the UK to remain internationally competitive in an increasingly diverse set of high-end computing grand challenge problems.

With the continuing increase in the power of computers the scope of problems which can be addressed effectively grows relentlessly, so that systems that could not be studied a few years ago have now become amenable to computer simulation. Plainly, there is no limit to the size of problems one may wish to simulate and the wider their scope, the more researchers from different disciplines will turn to simulation in the hope of obtaining solutions. At the pinnacle of computational power today are single multiprocessor supercomputers which are so expensive that they reside in a very limited number of sites (historically no more than 1 or 2) within the UK. As the power of these machines has increased, so too has the technical knowledge to exploit it by optimal algorithm design. As a result, the UK has developed a policy to promote so-called ‘‘capability computing’’ on this class of facility, whereby codes that have a demonstrated ability to run efficiently while consuming a very substantial fraction (up to 100%) of the machine’s resources (processors, memory, and so on) are prioritized by suitable job scheduling. There has been no other way to perform these kinds of computations in the past and the continuation of this policy seems certain to persist in the future. These leading-edge national facilities have helped the UK to develop and maintain an internationally competitive position in research using HPC; continued opportunities for leadership in the field are dependent on sustained access to such resources over the long term, as recognized by the recent International Review of Research Using HPC in the UK [13].

At this highest end, a major aim is to enable UK researchers to reach the petascale level at the earliest moment for a range of grand challenge applications spanning the full gamut of scientific applications—from cosmology to particle physics, embracing computational fluid dynamics, computational chemistry and biology, nanoscience, condensed matter physics, and environmental modeling. An outline of a number of grand challenge projects that may be effectively addressed by the future generation of petascale systems, together with other discussions of science enabled at the petascale may be found in Refs. [14–17].

Because of the immense power of ultra high-end supercomputers, the gulf between them and desktop computers needs to be addressed. This gap has been filled in the UK through University campus-based machines, currently funded in a variety of ways including the SRIF scheme, vendor donations, Research Councils and by the Universities themselves to varying degrees. In the past 5 years, the new grid computing paradigm—distributed computing performed transparently across multiple administrative domains—has begun to take shape and this too can be expected to alter the landscape in the future. The UK now has in place a National Grid Service (NGS) which includes a set of lower-end nodes (similar to those on many campuses and accessible to all) together with the national supercomputing facilities. This new paradigm has the potential to make it possible to undertake new ways of doing computing to solve scientific problems, where international collaborations have already demonstrated the benefits of tying together a multitude of hitherto isolated supercomputers and visualization engines to perform scientific research [18–21]. However, it is fair to say that the potential of this ‘‘new paradigm’’ has not to date delivered real benefit to the vast majority of research scientists in the UK.

As the complexity of the systems studied increases, the necessity of growing a *balanced* computational infrastructure becomes ever more apparent. The quantities of data generated are much greater than before; such data often need to be stored local to compute resources, moved around efficiently between widely disparate geographical locations, and analyzed effectively. Visualization is an adjunct to simulation which is often critical to the interpretation of scientific data; today, however, computational power and/or capability computing policies are leading to situations where it is difficult to visualize the results of large and complex simulations. To do all of these things successfully also requires high quality of service, high-bandwidth production and research networks.

The primary interest of this chapter, however, lies not at the National Services level, but at the so-called midrange facilities within the University community. It is this level that has benefited significantly from recent SRIF expenditure, to an extent that there is now greater peak capacity here than that available at the National Centers. For the sake of completeness, however, we provide a brief overview of the current UK national HPC services, HPCx, and HECToR.

3.1 The UK High-End National Computing Services

There are two current UK national HPC services, HPCx and HECToR.

3.1.1 HPCx

HPCx is provided by a consortium led by the University of Edinburgh, with the Science and Technology Facilities Council and IBM. HPCx offers a ‘‘world-class service for world-class science,’’ by providing access to a 1600-processor power4+ IBM system (see http://www.hpcx.ac.uk).

The Edinburgh Parallel Computing Centre (EPCC), which provides the University of Edinburgh contribution, has a long experience of national HPC services. It provided the Cray T3D/T3E service from 1994 to 2001, and before that, services on a Thinking Machines CM200, and Meiko Computing Surfaces. It continues to be a major center for HPC support and training (see http://www.epcc.ed.ac.uk). EPCC's partner in HPCx, the Computational Science and Engineering Department at STFC Daresbury Laboratory (see http://www.cse.scitech.ac.uk), has for many years worked closely with research groups throughout the UK through many projects, including EPSRC's Collaborative Computational Projects (see http://www.ccp.ac.uk), and has extensive expertise in porting and optimizing large scientific codes.

3.1.2 HECToR

The new UK national HPC Service, HECToR, commenced operation on 16 October 2007. Based at the University of Edinburgh's Advanced Computing Facility, the £113 million service will run for 6 years and is operated by the EPCC; support for applications software is provided by NAG Ltd. The procurement project for HECToR was funded and managed by EPSRC on behalf of the UK Research Councils.

The objectives of the HPC service are:

- To provide a world-class capability-level service for UK-based academic research.
- To support the development of innovative computational technologies.
- To help industry and commerce to make effective use of high-end computing.
- To work with colleagues in Europe and the world.

The HECToR phase 1 hardware configuration is an integrated system known as ''Rainier,'' which includes a scalar Cray MPP XT4 system, a vector system known as ''BlackWidow,'' and storage systems. The XT4 comprises 1416 compute blades, each of which has four dual-core processor sockets. This amounts to a total of 11,328 cores, each of which acts as a single CPU. The processor is an AMD 2.8 GHz Opteron. Each dual-core socket (1) shares 6 GB of memory, giving a total of 33.2 TB in all, and (2) controls a Cray SeaStar2 chip router. This has six links which are used to implement a 3D-torus of processors. The point-to-point bandwidth is 2.17 GB/s, and the minimum bisection bandwidth is 4.1 TB/s. The latency between two nodes is around 6 μs. The theoretical peak performance of the system is 59 Tflops.

There are 24 service blades, each with two dual-core processor sockets. They act as login nodes and controllers for I/O and for the network. In August 2008, a Cray vector system known as ''BlackWidow'' was added to the Rainier system.

It includes 28 vector compute nodes; each node has four Cray vector processors, making 112 processors in all. Each processor is capable of 25.6 Gflops, giving a peak performance of 2.87 Tflops. Each four-processor node shares 32 GB of memory. The BlackWidow interconnection network has a point-to-point bandwidth of 16 GB/s and a bisection bandwidth of 254 GB/s. The average ping-pong MPI latency is approximately 4.6 μs.

The HECToR storage systems are accessible both by the XT4 and the ‘‘BlackWidow’’ systems. Direct-attached storage comprises 576 TB of high-performance RAID disks controlled by three controllers through 12 I/O nodes. The disks are accessible globally from any compute node and use the Lustre distributed parallel file system.

The system runs under UNICOS/lc, which includes SUSE Linux. There is a full range of compilers, tools, and utilities.

3.2 The Science Research Investment Fund

The midrange HEI computing landscape in the UK has changed dramatically over the past 12 months thanks in large part to the major investments coming from SRIF. SRIF is a joint initiative by the Office of Science and Technology (OST) and the Department of Education and Skills (DfES) [22]. Its purpose is to contribute to HEIs’ long-term sustainable research strategies and address past underinvestment in research infrastructure. SRIF also takes into account the need for institutions to make their expertise and facilities more open to access by business and encourage collaboration between HEIs, industry, charitable bodies, government, and other partners. Once grants are assigned to the universities, it is the universities decision on how to allocate resources to the various departments.

3.2.1 SRIF3: 2006–2008

Capital grants made available to institutions from April 2006 had to be spent by March 2008. Funding is distributed by formula as a conditional allocation. £903 million has been allocated for research capital to English HEIs, of which £35+ million has been assigned to computer procurements. Heriot–Watt University led the administrative effort in coordinating these procurements. The idea behind centralizing the procurement process is to deliver maximum return on investment (ROI), while ensuring that the smaller procurements are not overshadowed by some of the more lucrative £1 million+ tenders.

The University HEC sector is now the major provider of HPC resources to the UK research base. As a result of the SRIF3 investment, the capability of this sector increased 100-fold during the 2005–2008 timeframe. Indeed this sector now has the potential to provide four times the capability of the UK National HPC service provider tier (HPCx and HECToR), with a number of 2000+ core systems coming online during 2007–2008, resulting from the three procurement Tranches undertaken during 2006/2007. A summary of 27 of the University installations resulting from SRIF3 funding is given in Table II. The procurement and subsequent installation of major systems at the Universities of Birmingham, Bristol, Cambridge, Cardiff, Edinburgh, Warwick, and UCL has led to major changes within the integrator marketplace, and the evolving relationships between these organizations and their Tier-1 counterparts. This is developed further below.

TABLE II
SRIF3 HPC INSTALLATIONS IN THE UK AND THE ASSOCIATED SUPPLIERS

Institution	Final Supplier
1. Cardiff University	Bull
2. Cranfield University	HP
3. De Montfort University	SGI
4. Keele University	ClusterVision
5. King's College London	Dell with ClusterVision
6. Loughborough University	Apple/Bull
7. Manchester Metropolitan University	NEC
8. Royal Holloway, University of London	ClusterVision
9. UHI Millennium Institute	ClusterVision
10. University College London	Dell with ClusterVision
11. University of Bath	ClusterVision
12. University of Birmingham	ClusterVision
13. University of Bristol	ClusterVision
14. University of Cambridge	Dell with ClusterVision
15. University of Edinburgh	ClusterVision
16. University of Essex	Streamline
17. University of Exeter	SGI
18. University of Greenwich	ClusterVision
19. University of Nottingham	HP
20. University of Oxford	SGI
21. University of Sunderland	Streamline/Dell
22. University of Surrey	OCF and ClusterVision/IBM
23. University of Sussex	Quadrics
24. University of Warwick	IBM with OCF, Qassociates, Apple
25. University of Westminster	IBM with OCF
26. Heriot–Watt University	ClusterVision
27. University of East Anglia	Dell with ClusterVision

3.3 Funding Research in the UK—Full-Economic Costing

The Transparent Approach to Costing (TRAC) methodology has been accepted by the UK Government and major funders of research as a method of costing and much of the funding of research is now based on TRAC costs, known as full-economic costs (fEC). A major feature of the fEC methodology is to ensure that the costing of each project contains an element to sustain the research base in the long term. The background to the implementation of fEC as the means by which HEIs may apply to Research Councils for project funding and a summary of the main features of TRAC are available at Ref. [23].

The extra funds from the Research Councils for fEC are allocated to ensure that the research infrastructure of universities is *sustainable*. The definition of sustainability includes the requirement to invest in physical, human, and intellectual infrastructure. The definition indicates that the extra fEC funding should be allocated to both Research Schools and Facilities since it covers intellectual and human infrastructure, which is funded by the Schools, as well as estates and IT infrastructure which are typically funded by the center.

Further consideration of the possible mechanisms for fEC reveals two charging extremes, the so-called major research facility (MRF) model and the indirect rate (IR) model. Within the MRF model, all costs are calculated and a directly allocated charge-out rate per ''unit of use'' (e.g., cost per CPU hour) applicable to grants based on estimation of usage per project. In contrast, within the IR model, all costs are calculated, combined with the ''central services'' costs and added to the University's Indirect Rate Charge. For fEC purposes in research grants, the funding bodies, that is, the Research Councils have stated that they will accept either the IR or MRF model.

A variety of component costs typically appear within an MRF model when considering the components of a cluster-based IT service—those included in computing the MRF rate for the Advanced Research Computing Facility at Cardiff University (ARCCA) include (a) depreciated machine replacement cost, (b) other equipment (i.e., noncomputer equipment, such as air conditioning, UPS, etc.), (c) hardware maintenance, (d) software licensing and maintenance, (e) building/floor space charges, (f) computer room maintenance and refurbishment, (g) electricity (including air conditioning/cooling), (h) consumables, (i) support staff (operations, technical, programming, and administrative), (j) training and development, and (k) travel (conferences, workshops, etc.). Note that the dominant contributions typically involve items (a), (g) and (i)-depreciated machine replacement costs, electricity and support staff.

3.4 Challenges Faced by University HPC Facilities

With the increased funding available to local HEI installations through SRIF, many Universities are looking to establish and develop local support services for Advanced Research Computing. The goal for these services is to provide an effective and value for money solution in coordinating, supporting, and developing the associated services for researchers. Typically staffed by a small team, the expectations are that this team will provide effective help and support for research needs.

A number of key challenges face such services. Critical is a consideration of sustainability of both the central equipment and the support staff, based on a funding model that fully captures the impact of full-economic costing (fEC). Assuming a model that is at least in part based on that of an MRF, each service needs to consider all costs and apply a directly allocated charge-out rate per ''unit of use''—for example, cost per processor (CPU) hour—applicable to grants based on estimation of usage per project. Typically these services need also consider a potential funding stream through revenue generation that might be delivered by commercial services.

There are a number of obvious issues facing such services. Clearly the impact of an MRF-approach to fEC in which the proposed cost per CPU hour is perceived by the University's research community in question and/or funding agencies as being ''too high'' and ''too expensive,'' would have a major impact on sustainability. Specifically, (a) funding agencies may refuse resource requests due to apparent high costs in comparison with other HEIs, (b) grants may be funded, but with access to remote systems hosted at other ''competitor'' sites with lower charge rates, and (c) ''peer'' pressure to reduce cost overheads to ensure grants are not rejected due to excessive H/W and associated support costs. Such an MRF rate and the associated impact above can only lead to *reduced funding* into the service. Given these scenarios, there is a strong possibility that departments would revert to procuring their own clusters to ''reduce costs.'' This would be suboptimal—both for the University overall, and for many of the schools/departments concerned. It would lead to sustaining the problems that an institutional approach is designed to solve (additional costs of departments duplicating tasks, risks of depending on a single specialist in each department, etc.).

Using the indirect rate would address the problems of apparently high charges, uncompetitiveness and risks noted above. There is however an immediate concern for using the indirect rate or a combination of indirect and MRF rate. Departments or research projects who do not immediately benefit from HPC will naturally oppose adding an indirect cost to their projects. Addressing this concern relies on the service, through its governance structure, convincing University staff of the benefits to themselves and the University of having a well-funded, sustainable HPC facility, with that facility seen to be a key part of the research infrastructure of the University.

The relative immaturity of Advanced Research Computing at many Universities leads to a number of issues that must be handled in a fair and balanced fashion. Given that utilization rates will take several years to peak, fully costed projects will inevitably ‘‘pay’’ for periods of lower utilization during this period, that is, fully costed projects will have to subsidize ‘‘free’’ projects.

Having quantified the total costs and associated MRF rate, then moving to a sustainable position relies on an accurate and reliable forecast of the total *research income*, and with this an accurate projection of the *number of chargeable projects* and *growth in ARC usage* over time. A variety of schemes might be used to estimate the likely income from grants reliant on research computing. For many institutions, including Cardiff, such an analysis suggests that it could take between 7 and 10 years before near 100% recovery is being achieved, with a major injection of new users needed to approach the income required. Fostering this growth of the User community is a clear goal for ARC services.

Key here is the competitiveness of the computed MRF rate against comparator institutions. Many institutions are in the main using the indirect rate or a combination of indirect and MRF rates, leading to low apparent charges. Comparator rates as low as 3–4p per CPU hour are now in play, with some comparators not charging MRF rates at all. Hence even with more processing power, a given University may well appear to be one of the most expensive of its type in the UK, given an unrealistic charging model. It would seem that the optimum approach is a scheme that is widely favored, namely to use a combination of indirect and MRF rate.

On the question of commercial services, experience suggests this may not be a key contributor to the sustainability of University HPC services in the short term. There may well be the potential of revenue generation through HPC services targeting commercial and other HEI organizations, but the effort in developing and delivering on such services and the impact this would have on the service’s core mission suggest that this is longer term.

Faced with the challenges of sustainability outlined above, we provide below a list of 12 attributes that we feel are needed by local ARC services to successfully address these challenges:

1. The development of a *unified support strategy* and approach that considers the requirements of *all* researchers in *all* disciplines. Central to this strategy should be a philosophy of ‘‘research enablement,’’ with the provision of a responsive support infrastructure that addresses the key requirement of enabling the research of others. Such an approach may be facilitated by a variety of mechanisms:
   - The ARC support team should work in collaboration with University Principal Investigators to ensure that their research agenda and

capabilities are facilitated by IT/Advanced Research Computing, enabling a broader and deeper research portfolio. This will require that the team understand the key research drivers of Investigators within the University and seek their involvement from the outset.

- The provision of a *high-availability* environment that is sympathetic to, supportive of and adds real value to the agendas of individual researchers, that is, *research enablement.*
- An appropriate *governance structure* is crucial; ideally this should be through the University's *Research Committee* or corresponding organization.
- HPC users within most universities are either established practitioners—''power users'' or those with some knowledge of the associated techniques—''casual'' users—who carry out their IT-based research on desktop or laptop processors. The former typically tend to belong to the ''well-established'' disciplines—Engineering, Physics and Astronomy, Chemistry, Earth Sciences, Computer Science, etc.
- The nature of the support for the ''well-established'' disciplines is well understood. Typically this involves assistance and collaboration around the following (1) the development of a new generation of codes, (2) scalable algorithms and load balancing, (3) the management of memory hierarchies, (4) numerically intensive algorithms, (5) optimization of memory access for communications buffers or memory bandwidth, (6) data management and visualization, (7) parallel I/O, database queries, on the fly visualization, (8) performance analysis, and (9) software engineering—assisting developers in adopting modern tools and best practices.
- In contrast, those from the ''emerging'' disciplines—the casual users—are typically unaware of the capabilities of HPC to enhance their research. This group of users is far larger than the expert or power users—users that fall into this category at Cardiff include those from the Schools of Biosciences, Mathematics, Pharmacy, Psychology, Medicine, Dentistry, Architecture, Optometry, History and Archaeology, the Business School, and Social Sciences.
- It is essential for University ARC services to not only address the ''established'' HPC users but also to foster those from the ''emerging'' disciplines, for it is that community who will ultimately provide the breath of user base necessary to sustain local services.

2. In looking to understand the University's research drivers and research portfolio, the ARC service should take responsibility for producing an ARC

roadmap for the University that would comprise both application and technology requirements, being driven both top-down from application needs and bottom-up from technology barriers. Such an ARC roadmap should map to the University's mission needs as well as to ''research'' drivers, and comprise an application and technology map that should address both the ''established'' HPC users and the emerging disciplines. The scientific/applications roadmap should specify the key research needs, goals and requirements, identifying timelines and technology barriers to addressing these goals. The computing technology roadmap would provide a consideration of future trends in hardware, software and networking developments likely to be available to satisfy the research requirements of users.

Such a roadmap should be updated annually and undergo major revision at suitable intervals.

3. A clear requirement in effectively promoting the value of local ARC services within a University is the presence of an Academic Champion for HPC provision and the associated research benefits and deliverables. The multiple tensions for research provision demands that such a voice is heard, and it is unusual for a service to flourish in the absence of such a figure. An Academic Champion, working with the service, can often provide the catalyst to encourage researchers to think more ambitiously—that is, plan to address more complex problems because they were unaware that the infrastructure to do so was available.
4. The committed engagement of researchers demands an environment that will provide continuity of HPC provision and service—realistically within the framework of fEC. Researchers need to be assured that a sustainable approach to both replacement technology and FTE support costs is in place, with an appropriate charging model. They do need that level of commitment from the University toward ongoing provision before committing to the long-term engagement of their research program.
5. Ensuring technology uptake by new users and those from nontraditional areas of HPC demands that the service providers look to simplify use of the technology. There is a clear requirement here to minimize the impact on researchers time and the need for them to directly understand or ''program'' the technology. In the medium term this might effectively be addressed through the provision of portal or portlet access for the nonexpert. Another example would be to provide Windows-based clusters (through Windows Server 2008), a faster route to HPC rather than requiring that PC-based, desktop users ''learn'' Linux.
6. The service should look to provide a range of technology solutions that address the needs of all researchers. There is little chance that a novice user would be in a position to effectively exploit a large commodity cluster, or that

a researcher from the Humanities, with a requirement to interactively data mine through large unstructured data sets, for example, video content, would gain from access to a large cluster that supported only batch scheduling of jobs. This is provided at Cardiff University through a large Condor pool[1] (with some 1600 PCs around the University), a number of small Departmental-Schools-based clusters, plus a high-end cluster comprising 2048 cores for the computationally demanding jobs.

7. Any University service should provide for both ‘‘capability’’ and ‘‘capacity’’ jobs within a high-availability environment. The latter is critical—many national services are criticized for their focus on ensuring a full machine, with the inevitable long wait times for turning jobs around. ‘‘High availability’’ means that the resource must be able to respond to short-term, research critical periods of access that argues against demands on the management of the system to deliver very high levels of utilization. There must also be a seamless path for migration to national services for those application areas that ‘‘outgrow’’ local provision. Regardless of user access patterns and demands, the service must be positioned to provide the appropriate level of support to respond to researchers needs, both those from established and emerging disciplines. It goes without saying that the type of support required, and the amount of that support, is very much a function of the experience and expertise of the user. It is the casual user who typically requires a far greater level of support than his experienced colleague.
8. It can be argued that the most critical requirement of any ARC service is not the support of the technology and systems—that is in many ways taken as read—but is the provision of effective *outreach and appropriate training* of the User community. The latter should be tailored both for experienced and novice users. An effective route to provide and establish a comprehensive training portfolio is to procure that training as part of the technology procurement itself. This lends itself to the approach of ‘‘training the trainers,’’ with the initial running of a course being provided by the technology supplier, and subsequent runs delivered by the ARC support team.
9. Much of the preceding discussion has centered on the role of ARC services in coordinating the research community. A second key role for Institutional ARC services is the provision of effective *coordination* across the IT Support Community itself. Many Universities feature local IT support activities within larger Departments, and it is vital that these local activities do not

[1] Condor, A Workload Management System for High Throughput Computing, ‘‘Introduction to Condor’’, http://www.nanohub.org/resources/1208/.

perceive institutional support as a threat, but rather an ally in providing effective support to the research community at large.

10. It is essential for an ARC service to deliver against an *agreed set of key performance indicators (KPIs)* that fully capture the effectiveness of the facility and its staff. While KPIs that reflect usage of the facility itself—metrics that capture system availability, utilization and research outputs, etc.—are vital, it is also important to develop metrics that capture the effectiveness and development of the support staff against the key attributes of the service. These KPIs should be subject to continual assessment and refinement by the service's governance structure.
11. While the key function of any ARC service is to promote the University's research agenda, most services will naturally wish to develop and grow the support activity to a position of international standing as an organization in its own right. While this may be a long-term goal, it could be promoted through the development of collaborations with other HPC sites to the mutual benefit of the sites involved.
12. Finally, we return to the development of *commercial activities* as a secondary mechanism for contributing to the sustainability demands of an ARC service. While this should not detract from the core mission of such services, the development of added-value packages that mutually benefit both the internal University customers and the external outreach opportunities is clearly a promising avenue, assisting the long-term sustainability of the program.

The preceding discussion has identified 12 service attributes that we suggest are needed by local ARC services to successfully address the challenges of sustainability. We would further recommend that a number of these can be effectively promoted in collaboration with the technology provider as part of the technology procurement, although such an approach does not fit naturally within the standard procurement process, and is not as yet widespread practice within UK Universities. However, at Cardiff University such an approach is embodied within the recently established Cardiff HPC Center of Excellence (CoE), a collaborative undertaking with Bull, the University's SRIF3 technology provider.

The CoE's main objective is to augment skills, expand services, and improve the quality of computer-based research across the whole University, further enhancing its reputation in the UK and beyond. The CoE will champion an outreach agenda for ARCCA, providing a focus for promoting the usage of the facility across all Schools within the University, and will act as the ''commercial'' arm of ARCCA, supporting Cardiff's HPC self-sufficiency by attracting investment, external funding and additional revenue streams. Additionally the center will provide a framework for developing collaborative interactions with other sites in the UK and abroad, through MoUs, etc.

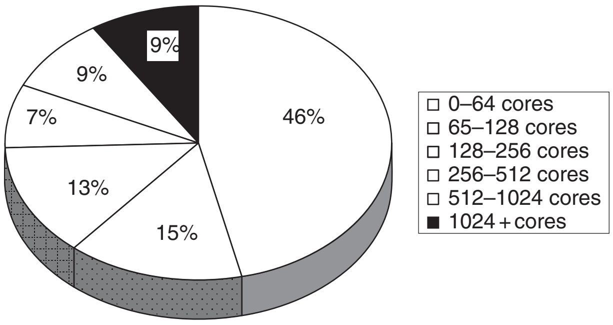
3.5 Brief Summary of the Clusters Deployed in 2007

Returning to the theme of HPC solutions and systems, we would note that commodity-based Linux clusters are the systems of choice for UK academia. To illustrate this, an analysis of the 2007 sales information provided by the principal integrators of Section 4 is presented below. Depicted graphically in Fig. 2, we look to capture the dominant trends across various aspects of the systems procured in this period, summarized under eight headings:

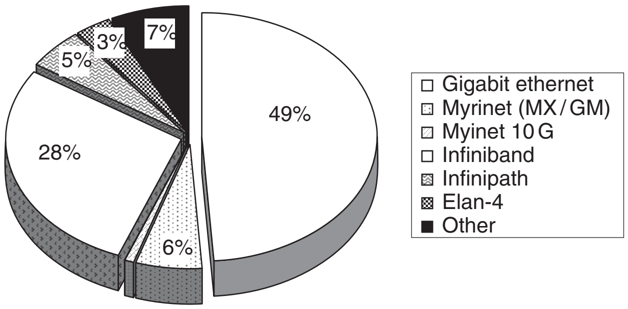
1. *The size of the systems being procured*. While approximately half of the clusters procured featured <64 cores, 2007 witnessed a significant increase in ''large'' systems with 20% of the systems housing 128–512 cores, and 18% involving >512 cores.
2. *Interconnects being deployed*. Half of the cluster systems deployed relied on Gigabit Ethernet as the interconnect, although clusters with high-performance interconnects increased in 2007, with InfiniBand the interconnect of choice—28%—compared to the decrease in Myrinet- and Quadrics-based systems (7% and 3%, respectively).
3. *Processor uptake*. In the first half of 2006, AMD clearly dominated the marketplace, and it was only with the release of the Intel Xeon 5100 series (codenamed ''woodcrest''), in particular the 5150/5160 in Q3, that Intel started to command a major fraction of market share. This trend continued in 2007, with Intel dual- and quad-core solutions commanding 57% of sales compared to AMD's 34%.
4. *Type of systems*—rack-based versus blade centers. Standard rackmount clusters accounted for the majority of the procurements (particularly in the academic sector) in 2006. In 2007 we see that rackmounted systems accounted for 76% of sales compared to 24% of Blade-based systems.
5. *Storage*. The leading storage solution is from Panasas (31%), although there are clearly a wide variety of solutions in play—44% fall into the category of ''others'' while 19% are the default solutions provided by the Tier-1 vendor.
6. *Cluster file systems*. Sixty-three percent of systems rely on NFS as the cluster file system—of the higher performant systems, GPFS and Lustre would appear the leading choices, with 13% and 10%, respectively, of the file systems chosen.
7. *Machine room cooling systems*. Eighty-six percent of the systems are still cooled through standard room-based air conditioning, with 14% using the more efficient water-chilled racks.
8. *Sector sales*. There is effectively an even distribution between academic and commercial installations.

It should be noted that the picture is only a snapshot of the marketplace; these trends can alter rapidly, especially regarding processor uptake, and are extremely dependent

• Size of systems being procured



• Interconnects being deployed;



• Processor uptake

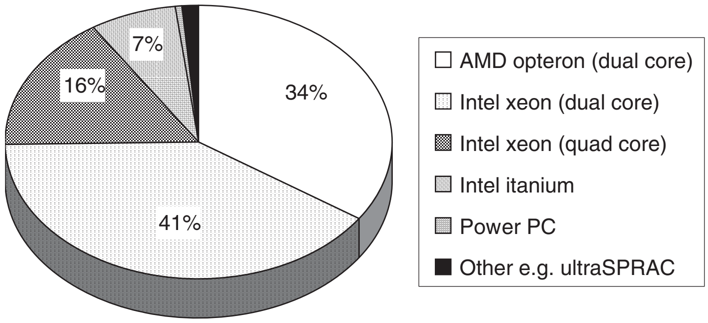
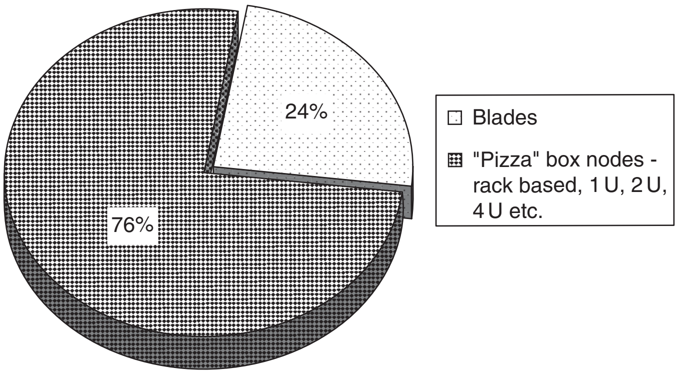
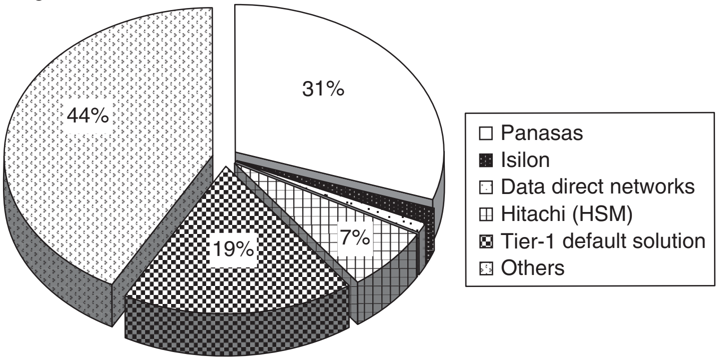


FIG. 2. Continued

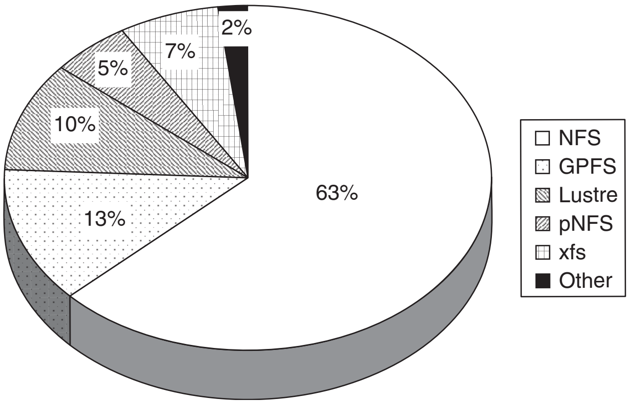
• Types of systems being built (rack-based vs blade centres)



• Storage



• Cluster file system



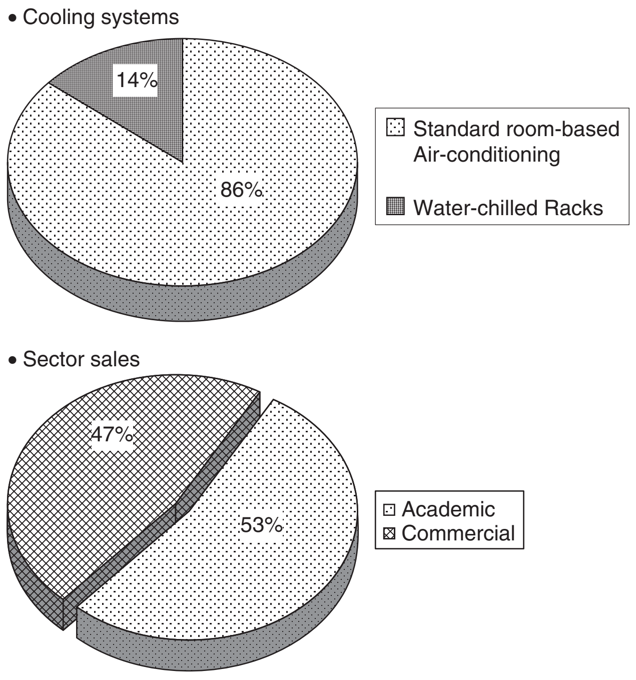


FIG. 2. The dominant technology trends in UK cluster systems (2007).

on developments, forthcoming roadmaps and the timing of chipset releases from both AMD and Intel, particularly in light of the next-generation multicore solutions. Multi-core solutions will continue to make a big impact of the size of the clusters sold, but will also add an additional level of complexity for both the integrator (cluster software scalability) and for the end-user (usability, programmability, etc.).

3.6 Tier-1 and System Integrators

There can be little doubt that the trend toward commodity-based solutions and the resulting cost effectiveness of such solutions has led to a market dynamic that will not support substantial unique developments for ultralarge systems. A guiding principle for many of the proprietary vendors when building HPC systems is to leverage standard technology wherever possible, and to add judiciously custom technology only when this is required. Further, any new technology must ultimately be applicable to mainstream commercial applications and systems for it to be viable from a business perspective.

The vendor is becoming more and more of a system integrator. In this context it should again be stressed that commodity-based Beowulf systems are rapidly becoming the systems of choice, at least within the academic community. This growth is fostered in the UK by the emergence of a number of companies who do provide crucial added-value services around clusters that address existing shortcomings in the standard, open-source-based environments, while keeping costs to a reasonable level (see Section 2.2). Procuring systems from such integrators may be more prudent than relying on the more traditional Tier-1 companies such as IBM or HP given the existing cost differential in the associated products, although caution should be applied and reference sites always sought for feedback regarding the previous installations (preferably of similar sized systems). While some way removed from providing credible high-end alternatives for, for example, HECToR and successor national systems, the emergence of enabling technology infrastructures provided by toolkits such as OSCAR [24] and ROCKS [25] has taken much of the hassle away from supporting such systems, at least up to 128 CPUs. The emergence of essential features such as check point restarting, concurrent file systems, etc., needs to be closely monitored in judging the ''fit-for-purpose'' nature of commodity systems at the high-end.

4. UK Integrators of Commodity Clusters

Consideration is now given to the current status and capability of a number of the leading HPC Integrators within the UK. We provide an in-depth analysis of the three major integrators who supply the academic community and hence are well known to us—ClusterVision, OCF, and Streamline—plus a less rigorous analysis of some of the other players. Our analysis has been conducted by a variety of mechanisms; given that we know each well, we asked for a response to a set of eight points following initial discussions. These points are reproduced in the Appendix. In this section we provide an overview of the response to these points from each of the three major UK HPC Integrators, while Sections 4.5–4.12 summarize data from the smaller, less-known organizations. Note that much of the detailed data provided by each integrator makes up the company overviews below. We should point out these are entirely the companies opinions and do not necessarily reflect the current authors views.

Information on the variety of solutions sold by the main integrators in this document has been presented in Section 3.5.

Prior to our analysis of the integrators, we present in Section 4.1 a summary of the major issues to be considered when deciding whether to procure systems from the integrators central to this review rather than from the traditional Tier-1 companies such as IBM, HP, or Sun, given the undoubted cost differential in the associated products.

Important criteria will be addressed, such as the dividing line between simply rolling out technology infrastructures provided by toolkits such as OSCAR and ROCKS, against providing the level of technical expertise to address the typical RAS requirements associated with high-end solutions, for example, HECToR.

4.1 HPC System Integrator Criteria

We now present an appropriate set of performance criteria that might enable us to differentiate between the various cluster integrators in the market. Based on an objective assessment of their standing against these criteria, it should be possible to identify potential suppliers for high-end solutions—the more performance target compliant a vendor, the more likely they are to provide a viable alternative to the more traditional Tier-1 companies.

We would consider from the outset that the following issues need to be assessed when judging the standing of a given integrator:

1. *In-house technical expertise*. The ‘‘added value’’ that the vendor brings to the procurement through software stacks, finely tuned OS, etc.
   Number of developers—the vendor’s ability to develop, support and maintain the software so as to sustain ‘‘cutting-edge’’ technology and assist with various code porting and optimization tasks.
2. *Size of the company*. Is the company sufficiently staffed to be able to support large-scale compute clusters over multiple installation sites?
   Turnover—is the company a practical long-term prospect with the potential to be ‘‘self-sufficient’’ with respect to large clusters (>1000 processors) within the next few years?
3. *Current install base*. An important factor is the integrator’s current success in the small- to mid range computer cluster market and the actual install base (whether solely in the UK or if they have a presence overseas, particularly in Europe or the USA).
   This again provides information on the potential robustness of an integrator as well as feedback from the community regarding their overall performance.
4. *Support infrastructure*. The number of technical and software engineers in place to support the cluster throughout its lifetime and importantly whether this is in-house or outsourced. If outsourced, whether there are any plans to change this in the near future.

These points formed the basis of discussions with the key cluster integrators at the outset of this analysis exercise—a more detailed account of these points can be found in the Appendix.

In addition to the above, the possibility of an integrator solution rather than one from the more traditional Tier-1 vendor is very much dependent on the nature of the HPC resource(s) under consideration. Some obvious examples would include:

- The *size of the system* in question—is this targeting less than approximately 1000 processing elements (PEs), a domain in which most of the integrators have experience, or does the system in question exceed, say, 15+ Tflop. If the latter, it is worth mentioning that the current national HECToR procurement rejected the use of integrators at a fairly early stage in the proceedings having considered their capabilities through a series of presentations at SC'2003 in Dallas. While a number of US Integrators certainly have experience in the 1000+ processor domain, this was not in general the case for their UK counterparts, at least at the outset of SRIF3. The rollout of several 2000+-core systems is certainly changing this landscape.
- The expected *usage pattern and environment* around the resource—is this being driven by *Capability* or *Capacity* requirements? We would expect, based on some of the considerations above, that integrators would be capable of providing the latter requirement far more effectively than the former.
- The level of *RAS* features expected of the HPC solution. Demanding levels of RAS (say 95+%) around truly large systems are exceptionally difficult to sustain, particularly in a *Capability* regime when running large jobs with long execution times. Few of the large clusters deployed have truly scalable system software, that is, can provide service to dozens of simultaneous users and have fast, scalable system boot-up, and executable loading capabilities. Assuming such features appear in any contract around the services to be provided, it is extremely unlikely that any integrator would be in the position to accept the risk involved in contractually committing to high levels of RAS.

4.2 ClusterVision

ClusterVision have been trading across Europe since 2002 and, as part of their European Expansion, ClusterVision Ltd was introduced as a direct subsidiary responsible for the UK and Ireland HPC market in November 2004. ClusterVision's head office is in Amsterdam with further offices located in Gloucester, Munich, Paris, Geneva, Milano, and Oslo, covering the whole of Europe. Their growth and success since the start of trading has been impressive.

ClusterVision specialize in the design, implementation, and support of large-scale compute, storage and database clusters and are the only Euro-wide cluster company to focus solely on cluster technology and development. They are independent in terms of component supply working with both Tier-1 and white box manufacturers and every

ClusterVision cluster is delivered as a fully functional turnkey system with all the hardware and software integrated and configured for immediate deployment.

ClusterVision's sales and technical teams have designed, built, and supported some of the ''largest and complex computational and storage clusters in the UK, Benelux, and Germany.'' Many of the key staff hold Ph.D's in Applied Science disciplines and have years of experience working with both traditional and clustered supercomputers for scientific research. It is this experience in applied scientific research combined with practical experience of a wide range of supercomputing technologies that ''provides insight and understanding of customer's requirements'' and enables ClusterVision to provide tailor-made solutions to meet these requirements.

4.2.1 Install Base

Year 2006 onwards has seen a huge increase in HPC demand and an equivalent increase in sales for ClusterVision as they passed the 3 years trading mark and became firmly established across Europe as a specialist provider of Linux clusters. They have continued their position at the forefront of new clustering technologies and responded to the increasing demand for high-speed interconnects with new variants such as Myrinet-10G and InfiniPath; accelerator board technology and new operating systems including the Microsoft Windows cluster software.

Reference sites include the University of Surrey, one of the first European sites installed with Myrinet-10G alongside the DAS3 grid in the Netherlands; billed as the ''fastest grid in the world''; utilizing Myricom's Myrinet-10G technology. The five Linux supercomputer clusters will have an aggregate theoretical peak performance of more than 3.8 Tflops. ClusterVision have also installed the ScotGrid node at the University of Glasgow (140 nodes + storage) and were awarded the contract for the NGS 2 (over 550 dual-core processors in dual- and quad-configuration). The service will provide HPC access to the academic community and includes in its configuration four ClearSpeed technology accelerator cards, capable of 50 Tflops performance with only 25 W power. This procurement coordinated through CCLRC was one of several successes for ClusterVision at that Laboratory including a number of Compute and Storage nodes to increase capacity for the Particle Physicists as the UK LCG Tier-1 site (over 80 nodes supplied).

Working with the British Antarctic Survey engaged ClusterVision in the delivery not only of a turnkey cluster solution but also in the power supply and air conditioning, using their own in-house expertise, and working with subcontractors to deliver the compute resource and the environment in which it will be housed (41 nodes). At the University of Cambridge, ClusterVision worked with partner Dell Corp. to install, integrate and provide support on 600 dual-processor Xeon Woodcrest nodes; with GBitE and the then new InfiniPath interconnect throughout. The Cluster will run the ClusterVisionOS software. A second delivery to the Department of Chemistry at the University of Cambridge, with DDR InfiniBand across 47 nodes, coincided with the installation of the HPC facility.

The installation of the HPC Facility at Cambridge and the installation at Surrey were awarded under the SRIF [22] Collaborative procurement coordinated by Heriot–Watt University. ClusterVision have also been awarded contracts from The Scottish Association for Marine Science, Keele University and the largest award currently direct to ClusterVision, at the University of Bristol. This large cluster—with two associated clusters—was the first project where ClusterVision partnered with IBM (it will total over 630 compute nodes) and a fore runner to more success for ClusterVision/IBM in more recent rounds of the SRIF procurement at the University of Edinburgh (128 nodes) and the University of Birmingham (256 nodes). Outside of SRIF but maintaining aspects of hardware and software collaboration ClusterVision has partnered with IBM to install the NEMO cluster at the Proudman Oceanographic Laboratory (90 nodes) and are working with Dell to deliver and support both 120 nodes at King's College London plus a far larger cluster at University College, London.

The current division between Academia and industry installations for ClusterVision is approximately 70–30%. While there has been a greater degree of interest from the private sector in the last 12 months this has been matched and surpassed by large projects across the public sector such as the current SRIF collective procurements in which ClusterVision has been especially successful (see Table II). Inside and outside of the current Heriot–Watt collective procurement there has been a shift toward larger procurements in general either across departments or the entire campus to provide resources for all schools, departments, and associated Research Bodies. This pooling of resources, encouraged by fEC models has meant that the average cluster size has increased to greater than 35 nodes with the largest individual cluster to date being 630 nodes for the University of Bristol.

4.2.2 Company Details and Size

To deliver these systems ClusterVision have also undergone a rapid growth in key staff areas to allow for their continued concentration on core areas of development such as the ClusterVisionOS^{TM} and to achieve their Customers schedules. As of mid-2007, the company totaled 28 permanent staff across four offices. The Head Office is in Amsterdam, with offices in Germany, France, and the UK. Their market stretches across Northern Europe and expansion includes new markets in Ireland, at the University of Ireland: Galway and Africa where success has already been achieved, for example, with the University of Limpopo. Any of the projects referred to and many others could be counted as a reference and a contact can be made available on approach.

The headcount of permanent staff as of mid-2007 is broken down as:

Function	Number of FTEs
Management	2
Sales	4
Logistics/projects	3
Cluster engineers	7
Software developers	4
Hardware/productions engineers	5
Office administration	3
Total	28

There is also a number of staff available under part-time contract.

4.2.3 International Presence

ClusterVision's own expansion in the Netherlands, Germany, France, Switzerland, and the UK mirrors the growth in HPC within these and the surrounding countries in Northern Europe and also the collaboration between them in developing Grid initiatives, even a single European Grid or Resource. Success for ClusterVision has been achieved already in what is one of the largest worldwide Grid Projects to date; The Large Hadron Collider Computing Grid at CERN, Switzerland. The company supplied over 425 dual-processor nodes to the world's largest Particle Physics Laboratory (currently the only Tier-2 vendor to do so from the UK).

4.2.4 Company Expertise

ClusterVision's profile within Europe has risen strongly in the last 12 months and has drawn significant interest; the company has been approached by, engaged with and been successful with several of the Tier-1 suppliers. This has been due to the company's expertise in their chosen market of cluster computing, an understanding of the components to identify the most resilient on the market, plus the development of their own cluster software stack, the ClusterVisionOS^{TM}.

The ClusterVisionOS is a Linux-based operating system and software environment specifically developed to ease the administration and use of any Linux cluster. It includes all the software required for the effective utilization of cluster computers. The cluster software stack has been a key component of recent awards, for example, that at the University of Cambridge HPC Facility with partner Dell that heralded an expansion of the role of ClusterVision as a system integrator working with Tier-1 partners.

4.2.5 Marketplace

The cluster market has developed around commodity-based components which, coupled with open-source software technology like ClusterVisionOS, can match the performance and stability of traditional supercomputers for a fraction of the cost. However fEC looks not only at the initial financial outlay but also at the surrounding costs, the power requirements and cooling and this has had a large impact on the market from Manufacturers to Users. Every quotation produced by ClusterVision includes information on the power, heat output, and weight of the equipment to allow decisions to be made within the fEC criteria.

ClusterVision have maintained their position at the forefront of Cluster technologies, developing working and collaborative relationships with manufacturers and users to provide access to information. To support technical changes ClusterVision continually run their own and customer benchmarks to identify the optimum configuration, either for a specific application or a range of crosscampus projects. They also provide remote access to customers to trial the ClusterVisionOS and to compare configurations. In focusing on designing the best available resource for their customers, the company has produced a series of ''firsts'' both in the UK and in Europe. These include the first Opteron cluster in the UK at the University of Manchester, the first production ready InfiniBand cluster in Europe at the University of Utrecht, the first Opteron dual-core cluster in the UK at CCLRC RAL, the first with a cluster incorporating the ClearSpeed card in the UK, and the first to deliver IBM x3455 servers incorporating AMD's socket-F technology at the Universities of Surrey and Bristol.

The flexibility of this early adoption pattern allows ClusterVision to maintain a technical competitive edge which in turn can fund future technology growth and customer support.

4.2.6 Relationship to Tier-1 Organizations

With the growth in the HPC Market—both in terms of user numbers and cluster size—there has been recognition of the role of a dedicated ''system integrator'' by the Tier-1 suppliers, whereby an opportunity is coordinated and sold as a turnkey solution with ongoing collaboration and contact. As this is an area in which ClusterVision excel, the relationship with each Tier-1 has also progressed and brought success. In 2006 and 2007 ClusterVision have worked successfully as both a prime and a subcontractor, whilst maintaining their own independence in the marketplace.

4.2.7 Relationship to Other System Integrators

ClusterVision maintains links with certain established system integrators with particular software developments and collaborations in mind to support the technology roadmap, for example.

4.2.8 Additional Information

The ClusterVisionOS has become a prime tool for collaboration to develop key features for large-scale clusters, such as software-driven power management. There are a number of projects underway which will see ClusterVision answer the long-term needs for customers in both academic and industrial markets. ClusterVision's added value remains (1) in the expert supply of tailor-made turnkey solutions that can be deployed upon delivery to benefit the science of the user, (2) that they are a single point of contact for hardware and software, and (3) that ClusterVisionOS is widely acknowledged as the most useful means of managing a cluster.

4.3 OCF plc

OCF designs, integrates, and supports innovative high-performance technical compute (HPTC) solutions, high-performance visualization (HPV) solutions, and enterprise computing infrastructure for storage and server technologies. The company ''continues to evolve its skills in accordance with technological advances and breakthroughs to remain at the cutting edge of HPTC and HPV developments and infrastructure provision.'' OCF has forged a strong relationship with IBM and the majority of their solutions are now based upon IBM hardware. These solutions range from individual workstations and servers to large bespoke enterprise systems. OCF works closely with a number of ''technology partners'' in the following areas:

- Development and deployment of server infrastructure.
- Complete data management solutions providing automated ILM (information lifecycle management).
- Manufacturers of HPV workstations.
- Manufacturers of high-performance immersive group visualization environments.
- Manufacturers of the whole range of compute servers, from individual Linux servers, to clustered Linux-based solutions, through to high-end SMP servers.
- Manufacturers of computer interconnect technology.
- Independent software vendors.

4.3.1 Install Base

The division between public/private installations has historically been 70/30%. This ratio has recently been moving toward a 30:70 split due to two factors (1) the successful deployment of some large private sector contracts and (2) delays in the implementation of the public sector SRIF program.

OCF does keep some private installations confidential for commercial reasons. A number of customer sites would ‘‘be happy to act as a reference’’ for the UK and the associated market developments—please contact OCF for named contacts. A number of its customers are named on its Web site—see http://www.ocf.co.uk.

Perhaps of most current relevance would be the work done by OCF at Southampton University with the IRIDIS cluster. From the initial installation of 330 Opteron cores linked by a Gigabit Ethernet network almost 3 years ago, the system has been increased to over 1000 Opteron cores (with a section connected by a Myrinet high-performance interconnect), over 700 GB of RAM, 25 TB of storage, and a full remote management solution. OCF’s largest recent installation in terms of the TOP 500 is a 3000-core BladeSystem utilizing power processors which entered the June 2007 list at number 36.

As far as high-performance storage is concerned, a recent GPFS installation by OCF has yielded aggregate performance figures of over 1.5 GB/s with InfiniBand, writing 16 GB files with 1 MB block sizes, a 33× speedup over the previous ‘‘traditional cluster storage’’ architecture. A recent win by OCF will provide over 4 GB/s of throughput to over 100 TB of storage. In addition single-client throughput is now reaching 1.2 GB/s using GPFS with DDR InfiniBand.

4.3.2 Company Details and Size

The overall size of the company is shown in the table below:

Function	No. of FTEs
Managerial/supervisory	1
Sales	7
Service	2
Operational/administrative	4
Design experts	6
Technical (implementation, rollout, support, and maintain)	5
Others	–
Total	25

OCF’s technical team comprises a team of eleven. This team has ‘‘expertise to configure, supply, and install large server and storage solution infrastructure,’’ and includes three software engineers. Further investments have been made in accelerator technology and the company is active in porting code for a number of customers.

In-house warehousing, and engineering workshops at OCF's Sheffield Headquarters enable OCF's technicians to be able to test all equipment, evaluate and benchmark new technologies as they are released. The company also provides a customer helpdesk and support facility. OCF build and support procedures are governed as part of their ISO9000 accreditation and IBM reseller partnership agreement. Strict internal procedures are in place to ensure that all IT equipment delivered to OCF plc is fully tested prior to shipment to their customers.

OCF provides maintenance for IBM equipment through IBM Global Services, with OCF acting as the first point of contact for complete projects. Maintenance options include service upgrades for in-warranty machines; extended maintenance for postwarranty machines; experienced technicians; and extensive parts distribution. OCF has a dedicated Support Hotline to which all calls (technical queries, delivery information, returns, repairs, etc.) are logged through the Hotline onto an electronic Call Management system. This in turn is set to escalate any calls that exceed the agreed time span. OCF can also run reports for specific problems, all customers with specific machines that have been notified with a problem, etc.

OCF has been specializing in HPC and HPV perhaps longer than any other UK integrator. This has enabled it to develop an unrivalled ecosystem of HPC and HPV partners, ensuring that technical challenges can be referred to technical specialists outside its own employee base. IBM alone has a team of over 50 people in the UK dedicated to HPC and HPV; by far the greatest commitment to the sector of any vendor. As a Premier Partner, OCF enjoys unrivalled access to this team and further through to the worldwide resources contained within IBM. As far as it is aware, OCF has the largest UK-based HPC team of any integrator.

4.3.3 Geographical Outreach

OCF's primary market area is the UK and Southern Ireland. In the past OCF has performed contracts in Scandinavia and Holland and has acted as a European business partner for a Life Sciences ISV with business currently quoted in Italy, Switzerland, and France. OCF has no overseas locations and all contracts are serviced with staff based in their Sheffield office.

4.3.4 Company Expertise

OCF has supplied computational clusters to a broad range of academic and corporate clients, and as a result have implemented a wide range of CPU architectures utilizing nodes from a variety of Tier-1 vendors and white box manufacturers.

OCF has also integrated clusters across a wide variety of interconnects whilst implementing management tools such as CSM, IBM Director, MOAB, GridEngine, Tivoli Storage Manager, etc. OCF's engineers are certified to an expert level in both Linux and AIX and have been early partners and implementers of Microsoft for Windows CCS. Many of OCF's solutions are delivered on IBM hardware and software environments—indeed OCF has implemented more IBM-based technical computing and cluster solutions than any other company (other than IBM itself) in Europe. However, OCF is not simply a ''cluster company''—it specializes in the whole spectrum of HPC and HPV and includes large SMP solutions and other traditional alternatives to clustered solutions in its portfolio.

The company has a strong dedicated visualization team and has done extensive work with IBM on DeepView to provide immersive multidimensional environments and deliver high-performance collaboration solutions to remote users. OCF is the only company to partner with IBM to demonstrate DeepView and has been involved with this technology since before its release to the market.

OCF has developed advanced capabilities to enable the delivery of high-performance storage solutions, with a focus on IBM's GPFS parallel file system. OCF specializes in integrating these storage environments into the customer's existing infrastructure, extending what has traditionally been considered a ''cluster-only'' technology across the organization it serves. These storage technologies have been successfully implemented in a number of commercial and academic customers.

The company is fully accredited to the ISO9000 standard and has a clearly defined methodology for the design and delivery of all of its HPC and HPV solutions. Wherever possible this includes the staging of the complete solutions at its workshops in Sheffield, with benchmarks and acceptance tests completed to identify any problems prior to installation at the customers premises.

OCF believes that the first step to getting the optimum performance from any compute resource is to optimize the design of the hardware and operating environments, concentrating its expertise on the infrastructure and operating system framework. For code-level performance tuning, the company looks to its extensive partner ecosystem with commercial and academic organizations to provide services that complement the core abilities of its engineers.

4.3.5 Marketplace

There is a view that the impact of public sector organizations having to base their procurement decisions upon the full-economic cost will be considerable. OCF is not convinced that the guidance issued to such organizations is sufficiently proscribed for them to accurately assess the impact. Before considering the impact of fEC it is necessary to take into account the full cost of ownership of a facility including all of the following factors:

- Purchase cost
- Installation
- Configuration
- Power
- Cooling
- Management personnel
- Maintenance
- Upgrade
- Disposal
- Real estate cost

These make up different proportions of the total and have varying degrees of importance.

It is difficult to justify the use of valuable inner city real estate as a viable location. It is cost advantageous to position systems in lowest cost facilities taking advantage of regional development sites and even offshore hosting and management capability.

The location of systems where the cost of power is minimized is a sensible model. This viewpoint becomes increasingly interesting if we take into account the development of GRID—e-science initiatives—where a distributed facility could be constructed taking advantage of the regional development locations (objective 1 regions) and then linked as a grid to fulfill users requirements. However, OCF has not yet seen any significant moves in this direction. Whilst public sector customers are increasingly identifying running costs as a factor to be taken into account in their decision making, it has not yet become apparent to OCF that such considerations override the traditional drive for the maximum compute power for the minimum cost. This clearly has to change if the current relentless increases in power costs continue. OCF believes that private sector customers tend to be more sophisticated in their capital budgeting processes and take into account fEC as a matter of process.

As regards OCF's own internal investments, it is continuing to increase the people's skills in its core competency areas, that is, infrastructure and operating systems. Taking the solution up the food chain from there requires the creation of strong mutually beneficial relationships—hence its continued focus upon a partner-based approach and evolving relationships with such specialist organizations as Pathscale, Mellanox, Voltaire, IBM, Allinea, Arup, Fluent, Force 10, Level 5, Cluster Resources, etc.

There is a strong body of opinion that future processor development is moving away from satisfying the needs of HPC users and that the HPC market needs to look to technical developments in the areas of coprocessors, FPGAs, Cell processors, etc. OCF has already supplied Cell processor-based systems and has a working relationship with Nallatech to incorporate its FPGA technology into its HPC solution portfolio. It has also developed relationships with ClearSpeed and nVidia to allow its customers to take advantage of the performance acceleration provided by these technologies.

The number of HPC users has grown exponentially over the last 10 years and it has moved outside the realm of traditional ''expert users.'' This has led to a requirement for systems to be far more user friendly and it is OCF's view that supported toolsets become evermore important, preferably supported by organizations with the credibility and critical mass to provide support and development over the long term. The recent launch by Microsoft of its Compute Cluster software will help at the lower end of the demand scale, whilst such developments as portals for access to large centrally managed systems may well assist the user with higher demands.

Furthermore it is evident that as the need for petascale computing capability expands, the scale of such systems will likely best be met by the implementation of robust and flexible grid infrastructures. OCF continues to work with its HPC partners to develop solutions to the many problems still facing such complex systems.

4.3.6 Relationship to Tier-1 Organizations

In responding to the question—''Does your company have the strength and depth to build high-performance, scalable systems which can support tera- and petascale solutions in the not-too-distant future?—Are you able to provide this independently?''—OCF believes that the only honest answer is ''yes we can build them but not independently.'' They do not feel able to address the relative abilities of UK and US Integrators.

OCF's view is that ''the only sensible way forward for commodity-based tera- and petascale computing is to leverage the financial strength of a Tier-1 vendor.'' OCF has backed that view by developing a very strong relationship with IBM. Risk, reliability, and long-term support are vitally important in most areas of business, and they are critical in large-scale computing environments. From an integration perspective, the major problem of fault identification and subsequent resolution means that preconfigured supported solutions are very much in vogue with the Tier-1's, with IBM's 1350 and 1600 clusters and HP's XC. The downside of such solutions is that they are significantly more expensive and customers face difficult decisions, especially those funded by the public purse.

In the medium-term OCF is not convinced that today's delivery models will be appropriate. How users obtain their compute cycles may well change radically if there are advances in middleware, bandwidth, etc. There has to come a time when vendors question whether delivering kit is an economic model, or whether the delivery of cycles is more appropriate. IBM currently operate a very crude version of this (on-demand computing) and the vision of them simply taking servers off the production line and into huge server farms serviced on demand may not be fanciful.

4.3.7 Relationship to Other System Integrators

OCF believes that the role of these organizations depends to a large degree on the business model being adopted by the provider of the compute cycles. Such organizations are good at billing and ongoing facilities management. Hence CSC's tie up with SGI at CESA. They tend to be exceptionally expensive having had a reasonably easy life looking after corporate networks and charging absolute fortunes.

4.3.8 Added Value

OCF has a long history of proposing innovative and flexible collaboration schemes for public sector procurements. In particular for the current SRIF procurements a detailed package of measures has been designed, in conjunction with its technology partners, to further evidence OCF's commitment to and support of UK Academic Research. These include access to new and exciting technologies on a no cost basis, funded placements of engineering resource to assist in the management of large systems, access to the research departments of our technology partners (particularly IBM), free quarterly health check visits to ensure continuing efficiency and a whole host of other initiatives.

Whilst OCF believes that the collaborative opportunities that it offers clearly differentiate it from its integrator peers, it is not yet clear whether such initiatives have a great deal of influence upon procurement decisions when, as with the fEC cost model discussed above, the emphasis continues to be the most kit for the least initial cost.

4.4 Streamline Computing

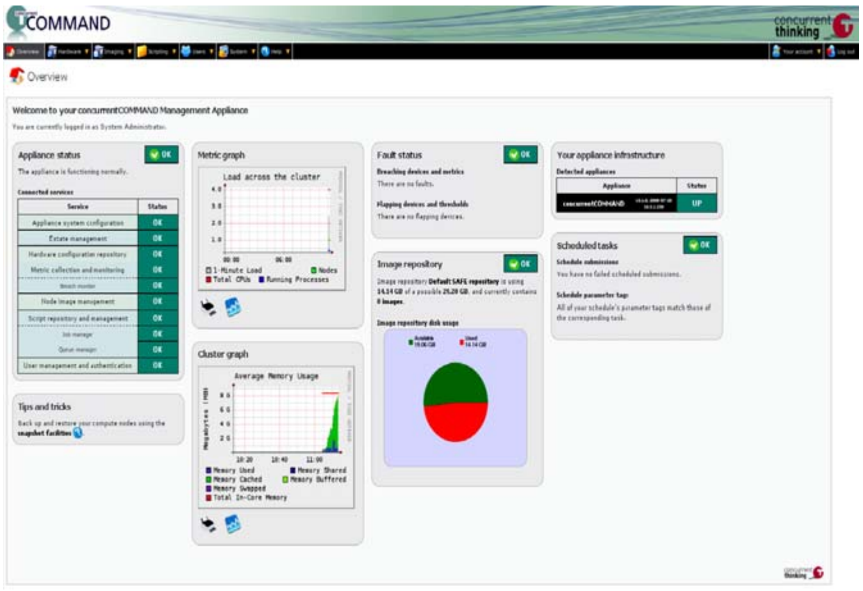
Streamline Computing is a trading division of Concurrent Thinking Ltd alongside its sister division Allinea Software, an HPC tools provider. Streamline Computing endeavors to provide state-of-the-art, commodity-based HPC systems to solve its clients' technical and scientific computing problems with the best performance and quality possible. It also provides consultancy and services to customers wishing to optimize their current HPC environments, whether on Streamline delivered or third-party systems.

Streamline Computing engineers HPC solutions from commodity components whether sourced internally through Streamline's own supply chain or in partnership with Tier-1 and Tier-2 vendors (through its OEM channel), both delivering and supporting these solutions. It has provided, since its formation in December 2000, many hundreds of Linux clusters to blue chip companies both nationally and internationally, as well as to nearly all the top-ranking research-led academic institutions in the UK.

Streamline delivers the solution through the preconfiguration of front-end node(s) with an HPC-optimized software stack and a customized (and fully customizable) cluster management appliance. During 2007 a major release of the appliance extended the control, management, and (dynamic) repurposing of Linux clusters. In the future this solution will be sourced under OEM from Concurrent Thinking Ltd as the appliances are made generally available to third parties. As a discrete cluster appliance, concurrent command is available on a dedicated hardware platform installed into the cluster as part of the cluster-build process. Streamline Computing engineers use the Streamline Advanced Front-End (SAFE) configuration tools included with concurrent command to build and configure new clusters, providing a reliable and consistent software configuration. In so doing the appliances alleviate much of the burden of administration and management—a significant contributor to the overall cost of commodity clusters.

Streamline Partners. Streamline Computing partners, not only with Tier-1 vendors, but with a number of technology organizations in the HPC sector to deliver solutions. These range across the complete stack from microprocessor through networking, parallel file systems, to application development and integration; the company also works closely with AMD and Intel and their respective platform providers to best deliver these ‘‘core’’ technologies to market. The company is an authorized reseller for most of the software companies providing compiler technology and tools to the HPC market, and if purchased as a component within a Streamline solution, these software products come preinstalled and ready for use and integrated with the Allinea tools for debugging and optimization.

The majority of clusters delivered by Streamline in this period of 2007 (some 60%) offer Gigabit Ethernet as the networking solution (70% of the clusters delivered are <64 nodes) which meets the demands of the majority of commercial ISV codes as well as meeting the primary requirements of academia in respect of maximizing CPU count. In this respect it has worked closely with Nortel in providing multiple stacked configurations for large cluster solutions as well as providing 10 Gb Ethernet core capabilities for I/O. In terms of high-performance low-latency networking Streamline continues to see increasing demand for InfiniBand solutions for both I/O and message passing which we expect to increase as well as other interconnects too, due to larger core count SMP nodes in the future.



Overview

Welcome to your concurrentCOMMAND Management Appliance

You are currently logged in as System Administrator.

Appliance status ✓ OK

The appliance is functioning normally.

Connected services

Service	Status
Appliance system configuration	OK
Estate management	OK
Hardware configuration repository	OK
Metric collection and monitoring	OK
[illegible]	OK
Node image management	OK
Script repository and management	OK
Job manager	OK
Queue manager	OK
User management and authentication	OK

Metric graph

Cluster graph

Tips and tricks

Back up and restore your compute nodes using the **snapshot facilities**.

Fault status ✓ OK

Breaching devices and metrics

There are no faults.

Flapping devices and thresholds

There are no flapping devices.

Image repository ✓ OK

Image repository **Default SAFE repository** is using **14.14 GB** of a possible **29.20 GB**, and currently contains **8 images**.

Image repository disk usage

Your appliance infrastructure

Detected appliances

Appliance	Status
concurrentCOMMAND [illegible]	UP

Scheduled tasks ✓ OK

Schedule submissions

You have no failed scheduled submissions.

Schedule parameter tags

All of your schedule's parameter tags match those of the corresponding task.

Use of Cluster Appliances. Streamline clusters are shipped with the Streamline Cluster Management Appliance where IPMI or proprietary lights-out service and management interfaces are available. It is delivered as an appliance based on a 1U form factor commodity server, and ostensibly provides a Web services interface (or CLI) rationalizing much of the complexity of Linux clusters to an intuitive set of tools based around management, monitoring and imaging—alleviating much of the administrative burden inherent in Linux. clusters. In particular the appliance features comprise:

- Script and task management:
  - Managed repository of preloaded and user-defined scripts
  - Scheduled and manual execution of tasks across a cluster
- Metric collection and display:
  - Multiple data collection methods, including ganglia, SNMP, etc.
  - New grid/cluster/rack view to highlight problems at a glance
  - Highly configurable actions on metric threshold breaches

- Hardware configuration and control:
  - Database of hardware, configuration, and historical data
  - Support for blade servers and virtual machines
  - Out-of-band management features for compatible nodes
- Image management:
  - Image collection and deployment
  - Provides full audit trail of deployed images for client nodes
  - Support for RAMdisk and diagnostic boot mode

Streamline's Software Stack. Streamline Computing installs and configures a custom combination of open-source and licensed software on all systems as part of its package. Most systems ship with SuSE Professional or Enterprise Linux or RedHat Enterprise Server Linux (or derivatives thereof) where, for example, particular support of an ISV package is required. This stack is provisioned by the Management and Control appliances discussed above.

For parallel applications using MPI, Streamline has installed most of the numerous choices available, and provides these within a consistent ''modules'' environment allowing users to switch easily between application dependencies. Streamline also has unique access to an open-source version of MPI called SCore, developed and widely used in Japan. The SCore parallel computing environment is installed as standard on systems shipped by Streamline based on the prevailing professional version of Linux. When installed and configured, SCore provides a parallel computing environment, including a custom MPI layer, that can provide significant performance increases for parallel applications compiled to use SCore's drivers and subsystems. SCore also provides deadlock detection, fault tolerance with pre-emptive check-pointing, parallel process migration and flexible job distribution including gang and batch scheduling. In addition Score provides a general parallel shell which offers a powerful system administration tool for cluster-wide operations. Further development of Score is in process in collaboration with the Score consortium (see http://www.pccluster.org/).

Streamline also has a wide range of expertise in distributed resource management (DRM) and scheduling software. Most Streamline clusters ship with either SGE with ISV application-specific scripts if required. Support is also available for Torque and LSF where appropriate.

4.4.1 Install Base (During 2007)

During 2007 Streamline continued its success in the penetration of the Industrial HPC market with many solutions entering industry for first time cluster users. Typically these solutions are for less than 64 nodes (256 processors reflecting ISV support) but notably it has provided 300- and 800-core solutions in the support of a large multiuser engineering simulation production system combining multi-Tbyte storage.

4.4.2 European Presence

Streamline have installed clusters in France, Germany (as well as the Middle East, USA, and Canada), although its primary focus has been within the UK. The approach in other geographies has been to provide software and support skills to local cluster builders (and is extending this with appliances).

4.4.3 Company Expertise

Streamline has a proven track record in tuning operating systems and specialized MPI and scheduling software layers, and has developed expertise in specific applications in the areas of engineering simulation. Streamline Computing has considerable experience in networking solutions (commodity Ethernet or low-latency alternative), storage and file systems, visualization and integration of these within the operational environment and policies of the end-user and is an active contributor to open-source projects.

Streamline maintains a thriving research and development group which actively pursues new technologies and market opportunities. It executes this R&D with internal investment (as witnessed through the development of the Allinea trading division), close involvement with the Score consortium in Japan, and through external funding via the DTI (at a National or Regional level). Of particular note, during 2007, is a grant for Research and Development which has been investigating Fault-Tolerant Parallel Computing solutions, and the BROADEN Inter-Enterprise Computing project (Business Resource Optimization for Aftermarket and Design on Engineering Networks) that aims to build an internal GRID at Rolls Royce plc as a proving ground for utilizing Web/grid service technology to fully exploit available IT resources.

4.4.4 Marketplace

Streamline/Concurrent Thinking Strategy. As previously mentioned, cluster management will become an increasingly barrier to the wider adoption of the cluster architecture. As cluster sizes grow and commodity hardware becomes cheaper, the ‘‘value’’ in the market will be the ability to make sure that clusters work efficiently ‘‘hitting the ground running’’ and providing the means to support through the lifecycle. In many circumstances the cluster needs to support a broad range of applications (with a range of run-time issues) and function in a more scalable manner operating seamlessly across subsystems including file systems and visualization.

To increase the accessibility to HPC, system management and monitoring capabilities will therefore become more important and proffer a more service-based approach to HPC. The ability to provide a high level of support, not only on a system level, but also on an application level, will help differentiate the ''box-shifters'' from the serious HPC-oriented companies, ideally separating out the commodity aspects of the system to a value integration.

Previous investment has been made in building this knowledge base within the company around this strategic view. In particular it is of worth noting:

- Collaboration with the Score consortium in Japan and the provision of the MPI Score environment widely used in the UK; this technology offers the best latency/bandwidth profile for commodity Ethernet.
- The spinout of DDT and OPT within Allinea Software now providing software tools critical to parallel and cluster computing in general Linux cluster solutions; this technology is becoming more increasingly important as the amount of parallelism increased within multicore and embedded technologies.
- Continued research initiatives supported by the DTI for high-availability computing, GRIDs and visualization involving major industrial customers.
- The development of cluster appliance products toward the general provisioning of optimized parallel software environments for cluster computing as well as the management and control of these systems.

Streamline continues to invest in such capabilities and is keen to work with partners and customers to ensure our cluster technologies maintain Streamline's position as the leading UK integrator in this field. We also value such collaboration as a means to not only develop the right products but also note that having partnerships can ensure that Streamline can deliver these products in a timely manner to the market—in a sense getting the product right. Primary R&D areas for Streamline concentrate on:

- Distributed and parallel file systems
- Optimized MPI environments
- Cluster installation management and provisioning
- Parallel job control and GRID environments
- Software tools and code optimization
- Distributed, remote visualization of very large computational models

Following the recent investment round, Concurrent Thinking has focused on bringing these technologies to marketplace within its own cluster architectures, as well as those of its partners.

Trends in the Marketplace. A significant proportion of cluster sales have been based, and continue to be based, on Gigabit technology, or a mix of Gigabit and high-speed interconnect reflecting the state of parallel applications for cluster computing. For this reason, Streamline has invested significantly in R&D relating to Score providing maximum efficiency on Gigabit, although continued work will be required to support larger SMP nodes in the future. Streamline observe very few customers who run a large proportion of capability jobs on their clusters; the majority run mildly parallel applications on up to 32 nodes (and an expectation that with increasing core count per node will achieve the same scalability). Benchmarks have shown that Score can outperform MPICH and LAM by over 20% on standard test cases using Computational Chemistry and Computational Fluid Dynamics.

Notwithstanding this, with the new wave of multicore solutions, with four- and eight-socket servers, Streamline is witnessing an increased requirement for clusters of modest SMPs connected by Myrinet-10G, InfiniBand/InfiniPath, or Quadrics Interconnects. With the advent of low-latency/high-bandwidth networking for MPI, increasing use of these is being made in the support of storage too. Challenges remain in the multiprotocols required to share networking for both MPI and I/O.

Challenges. It should be noted that Streamline has had exposure to, and has overcome, some very interesting technical and support challenges relating to large cluster systems (buffer overflows on switches and NICs; processor timing problems; interaction between job schedulers and Linux modules, etc.) and it is this level of systematic support that is critical to many operations.

A potential major challenge to systems integrators is whether all public procurement decisions relating to clusters become a question of price—if so, then companies like Streamline will be unable to sustain the skill-base needed to resolve such complex problems. Similarly, records demonstrate that Streamline currently provide a significant level of customer support (all of their support queries are logged in a company database) yet that the delivery of this level of service is clearly not profitable for academic customers on a tight budget. As a growing company, Streamline see this as an investment for the future. However if University procurement procedures do not provide the mechanisms for companies like Streamline to demonstrate their value, then the provision of this level of service will become unviable.

4.4.5 Relationship to Tier-1 Organizations

Very large systems require the financial strength of a Tier-1 vendor to execute larger contracts but need the specialist skills of companies like Streamline to provide the knowledge and skills to build and support the systems. Relationships at all levels with Tier-1 vendors are thus critical as a success factor for future growth—Streamline have built systems based on the major Tier-1 hardware providers. Streamline have also provided support for a number of other vendors, both in UK (for systems shipped from the USA) and in Europe, the Middle East, and USA, where Streamline partners build and Streamline support. Experience so far demonstrates that Streamline's technical staff has more than equal skills to any other integrator globally, and can compete effectively if these skills can be channeled profitably.

4.4.6 Relationship to Other System Integrators

Streamline has developed relationships with large-scale systems integrators where commercial and industrial end-users have outsourced supply to these organizations. This arrangement meets the market demand for delivery of the commodity servers in the most efficient manner but enables Streamline to transfer its value in providing the solution. Streamline is continuing to develop further relationships in this area as part of a definitive strategy aimed at primarily the industrial and commercial HPTC sectors and where appropriate, in the academic sector too.

4.4.7 Additional Information

''With a strong UK-based team with deep technical skills'' Streamline has much to offer UK and European customers and as a company ''can keep a UK technology flag flying in a global market.'' Whilst establishing their own Tier-1 relationships, Streamline is grateful for the support from organizations such as CCLRC and other UK government organizations in allowing Streamline to demonstrate its value and the ability to succeed in this market and ultimately benefit UK plc with exports and technology from a strong UK base. In particular, Streamline wishes to take the opportunity to thank the DTI and its project partners for their support in the execution of R&D projects and looks forward to continued involvement in similar regional, national and international initiatives.

4.5 Other Cluster Suppliers in the UK

Having considered the current status and capability of the three leading UK HPC Integrators—Streamline Computing, ClusterVision, and OCF—we now provide a far briefer overview of some of the other, less recognized players. Note that much of the information here has been obtained from the organizations Web pages and does not map naturally onto the discussion points raised with each of the integrators of Section 4.1. Companies considered below include Compusys (originally classed as a leading integrator, but recently absent from any of the major procurement exercises), Cambridge Online Systems, Western Scientific, SCC, Silicon Mechanics (USA-based, but branching out into European solutions), Workstations UK, and as a cautionary tale, the now defunct Linux Networx (USA-based, but with a footprint in the UK). In only three cases, Cambridge Online Systems, Compusys and Silicon Mechanics, did we have the opportunity to raise and discuss the points noted in Section 4.1.

4.6 Compusys

Compusys is a privately owned, UK-based Computer Solutions Provider and Systems Integrator, and was formed in 1987. Originally an Olivetti PC Systems reseller, Compusys soon began building and assembling its own brand of PC systems, and by 1989, this had overtaken branded systems sales.

In 1997, Compusys formed its Networking Division, to provide turnkey networking and integration projects for new and existing customers. Within 2 years, this Division enjoyed significant success with many turnkey campus-wide LAN and WAN deployments, for further and higher education establishments, the Police, and commercial customers. Compusys formed its HPC division in 1999; the company also has an e-Business Consulting Division, providing Web, content management and e-business solutions to local government, emergency services and academic institutions.

Install Base. Compusys HPC have been building, supplying, installing and supporting HPC Linux Clusters since 1999. In their first 2 years of HPC, Compusys deployed Alpha Clusters to a significant number of Academic Sites. With the improvements in Intel Architecture performance in 2000–2001, the company enjoyed significant deployments at a number of leading research Universities in the UK, continuing to supply cluster solutions to UK, European and Intercontinental customers between 2002 and 2006. During this time the company's average run rate for the delivery and deployment of systems had been around 50 per annum. The majority of these systems had been below 64 nodes, with around 15–20% of these being above 128 nodes. These systems had been supplied largely to academic institutions, as Compusys found the market for HPC to be far more mature in the academic than the commercial sector. However, the company continued to expand into commercial markets, with successes in the Automotive, Manufacturing, Bioinformatics, and Financial sectors. In 2006 the commercial markets accounted for around 15% of Compusys HPC.

While Compusys HPC had a proven track record of over 250 successful cluster deployments to this point, it is unclear, however, whether the company has enjoyed any significant deployment of HPC systems beyond 2006.

Company Details and Size. Compusys provided its own service, support, and maintenance for all systems sold by the company. This included desktops, servers, laptops, and HPC cluster solutions. Furthermore, the company employed its own field engineering, support, helpdesk, and administration staff, with no reliance on any third party in the provision of any service delivery. Providing ''a broad range of solutions, to an even broader range of customers,'' Compusys employed over 120 staff in 2006.

International Presence. Between 2001 and 2005, Compusys HPC provided and supported cluster solutions outside of the UK. Their first significant international installation was a 1000 CPU Alpha Cluster for the Moscow Academy of Science, which was installed in 2001. Since then, Compusys have continued to provide systems and solutions to clients across mainland Europe, with installations in Germany and Austria, plus a number of HPC clusters to the United States.

Company Expertise. Compusys solutions are integrated using 100% in-house resources. No third parties are used in any part of the solution; these are built on their own systems hardware, manufactured in their own assembly facilities, adjacently located to their HPC Labs. All of their HPC clusters are fully built, configured, tested, and signed off in the HPC Labs prior to shipment to site. This ensures the systems are fully operational on day 1.

Compusys have direct relationships with all of the leading vendors of HPC cluster products; both hardware and software. For example, they ''collaborate with motherboard and systems hardware manufacturers to help them to design suitable HPC platforms. This collaboration is reciprocated when it comes to support, allowing Compusys to efficiently resolve any operational issues that may occur through direct contact with the designers and engineers of the products sold. The direct relationships also support a preferential pricing and commercial support model—essential when fighting against stiff competition from Tier-1 vendors.''

Marketplace. Compusys see the HPC market continuing to grow, as the technology continues to mature, and new innovations drive performance ever higher. However, the issue of fEC is now playing an increasing part in decision-making process, as the factors that affect fEC are becoming more visible, and sites are now reaching their capacity to provide the space, mains power, and air conditioning required to run a supercomputer cluster.

Compusys is part of a group turning over in excess of $100 Million a year, and as such are able to finance contracts of well in excess of £2 million and have done so several times in the past. Tier-1 support is only an issue, if a particular Tier-1 vendor has decided to offer products at prices below cost. The company believes that most of the time, Tier-1 vendors bring a low cost price for the computer hardware, and the comfort factor of a known name.

4.7 Cambridge Online Systems Ltd

Response from Cambridge Online Systems Ltd (http://www.cambridgeonline.net).

Install Base. The customer base is within the UK and made up of Research (40%) and Higher Education establishments (30%) and commercial organizations (30%). The main architectures installed are HP Proliant and Integrity (Itanium2) with Linux, and legacy HP (Digital) Alpha with Tru64 Unix. The majority of systems are either clusters or compute farms with blade servers being the most popular form factor. The split between ''small'' (32–64) systems and larger (128) systems is roughly 75:25.

For integer heavy computations, HP Servers with dual-core AMD Opteron CPUs have proven popular, having high performance for a low price. With the introduction of the new HP c-Class BladeSystems with improved Linux management software, ''the stage is set for users who previously had to share HPC resource, to afford dedicated systems. For users with advanced visualization and floating-point calculation requirements requiring full 64-bit computing, Itanium2 came of age with the release of Intel Montecito-based HP Integrity systems in September 2006.''

Over the past 4 years, noticeable trends have been (1) the shift from proprietary Unix to Linux, (2) the increasing use of industry-standard, ''commodity'' computing systems, (3) heterogeneous, mixed-vendor Linux environments, and (4) the requirement for performant file system, such as Lustre/HP scalable file share.

Company Details and Size. Cambridge Online was established in 1978. Their Applied Technology Group specializes in the provision, development and support of IT infrastructure, covering computer systems, storage, networking, and telecommunications. The company partners with industry leading technology vendors and hold ISO9001:2000 quality accreditation. With a particular focus upon HPC systems, the company delivers ''a highly technical and consultative approach to meeting solution requirements.''

Total staffing of 60 employees is split by sales (5), technical support/engineers (15), software development (30), and management and administration (10). More than 250 customers are served. Relevant customer names include the Wellcome Trust Sanger Institute, European Bioinformatics Institute, Universities of Cambridge and East Anglia, Cranfield University, and the Medical Research Council.

Company Outreach, Presence, and Expertise. Cambridge Online's business is largely UK-based with only a small, overseas presence (<10 customers). They have a number of Biotech, Life Sciences and Research customers based in the Midlands/South East of England. Their portfolio of HPC products from industry leading vendors is complemented by value-added services which include technical consulting, system and network design, system build and configuration, systems integration, network infrastructure design and installation, technical support, and system maintenance.

Linux, clustering, and open-source software are key drivers for the company to provide customers with solutions to their computations needs. Increasingly the company is consulted on storage and file systems. Cambridge Online is one of the UK experts in the HP productization of Lustre, SFS (scalable file share). The company's HPC lab provides facilities for proof-of-concept demonstrations, benchmarking, training, and support.

Relationship to Tier-1 Organization. Cambridge Online state they have the ''strength and depth'' to provide HPC, scalable systems supporting tera/petascale solutions and this experience can be demonstrated. The company works in close partnership with leading vendors for full support and has accredited relationships with HP, Intel, Platform Computing, and RedHat amongst others. They do feel that UK integrator partners are competitive with those in the USA, provided always that strong Tier-1 relationships exist.

4.8 Silicon Mechanics

Response from Silicon Mechanics (http://www.siliconmechanics.com).

Silicon Mechanics are different from other companies in the cluster marketplace; they are simply focused on building very reliable ''industry-standard'' (or ''white-box'') servers and storage systems. Silicon Mechanics is not a company engaged with the system integration of the cluster layer or other applications, but rather they work with the customer to complete the physical framework: systems, racks, power, cooling, integration into racks, site deployment, etc.

A major aim of the company is to make the whole process a simple one for the customer. Silicon Mechanics are very flexible. Most of their customers are highly technical IT staff who prefer a fairly direct, simple experience. This strategy of keeping the interaction simple starts with their Web site (http://www.siliconmechanics.com) and continues through email or verbal communications with the sales and support team.

Install Base. In the 6 years since the founding of Silicon Mechanics, their customer base has been represented by both the industrial (commercial) and HEI (public sector, higher education, government research) sectors. The focus is on providing high-quality rackmount server and storage systems (along with the rack, power distribution, network switching, and hardware-only infrastructure), and not on providing the software infrastructure for building out a cluster. To date, Silicon Mechanics have been very successful in doing this, primarily in the USA. Part of the process is to ensure that their products will work with over a dozen open-source operating systems, plus Windows (though the vast majority of their systems are used with some flavor of Linux).

Silicon Mechanic's primary customer is someone who knows how to build the cluster infrastructure, or who has consulting expertise available to do so. Silicon Mechanics will work closely with their customers to ensure they are planning to use the best hardware platforms for their applications.

Company Details and Size. The company's average annual growth rate since 2001 is 118%. The current employee count as of 2006/2007 is 42 employees and continues to grow.

The areas most important to server acquisition and support are staffed as follows:

Function	Number of FTEs
Sales	7
Product engineering	6
Postsales support	5
Production personnel	12
Total	30

In the period 2005–2007 Silicon Mechanics executed server system sales with 600+ customers. The customer base is quite varied, with the following industries the most notable:

- College and Universities (for both academic and research use)
- Federal Government Research Laboratories
- Web Retailers
- Online Web Hosting
- Game Developers
- Data Storage and Archiving

Below is a list of some of Silicon Mechanics public sector installations (not listed in any particular order):

- Massachusetts Institute of Technology
- Fred Hutchinson Cancer Research Center
- Harvard University
- University of California, Berkeley
- Los Alamos National Lab
- Lawrence Livermore National Lab
- Stanford University
- University of San Diego, Scripps Oceanographic Institute
- University of Washington

- Johns Hopkins University and Medical Institute
- Carnegie Mellon University

Company Outreach and Presence. Whilst Silicon Mechanics are located in the northwest United States near Seattle, Washington, most of their customers are outside the ''local area.'' Many are outside North America, with customers on all the continents. Silicon Mechanics have considerable experience in working with and supporting distant customers and systems—the distribution includes Washington state (30%), USA (other states) (65%), and International (5%).

Company Expertise. Silicon Mechanics' engineering and development staff design, develop, validate, and document one of the most comprehensive rack-optimized product offerings in the industry. Their product development efforts focus on the needs of customers deploying large, rack-optimized server installations and take into consideration density, power, thermal efficiency, performance, and reliability. They have strong, cooperative relationships with the most advanced technology companies in the industry, such as Intel Corporation, AMD, and Supermicro, and leverage their design efforts and research capabilities in their development process. Silicon Mechanics are able to cost effectively and efficiently develop products specifically for each customer.

The focus is on developing solutions that make it simple and efficient for a customer to acquire, manage, and service a product through its entire lifecycle. This holistic approach sets Silicon Mechanics apart and ensures that they will remain competitive in environments where IT staff are continually asked to do more with limited resources.

Their products are built to order, and the operations and manufacturing are key components in maintaining a high level of quality and customer satisfaction.

Recently, Silicon Mechanics purchased 19,000+ square feet of state-of-the-art manufacturing space, with an additional 23,000+ square feet of contiguous space available when needed. They expect this facility to accommodate system unit shipment growth of over 600% over the current unit production volume. This is believed to be central to producing integrated commodity server and cluster configurations for their customers.

Overview of the Marketplace Perspective. Silicon Mechanics' believe cluster computing hardware has been the beneficiary of a host of commodity technologies, such as the rack-optimized server, general-purpose microprocessor, Linux, networking, and others. To date, success has been dependent on the integration of these technologies with minimal modification or value-adds. While the forces shaping these technologies are much larger than the influence of the commodity cluster market alone, recognition of clustering is steadily increasing among the companies responsible for these technologies. Silicon Mechanics see the further commoditization of clustering features as they apply to these technologies, making it easier for users to cost-effectively source the hardware framework for powerful clustering configurations.

Relationship to Tier-1 Organizations. Currently, Silicon Mechanics has several large-scale server installations at customer sites, with the largest being greater than 1000 nodes. They believe that their infrastructure is capable of handling tera- and petascale solutions. Silicon Mechanics work concurrently with Intel Corporation, as a Premier Provider, and Advanced Microdevices (AMD), as a Platinum Provider, as well as additional manufacturers like Supermicro Computer, Inc., in the proposal process for large-scale installations. They use these resources readily in an effort to increase the efficiency of acquisition and deployment of these installations.

Many of their customers have made a business decision to move away from Tier-1 hardware platforms, realizing that they can obtain more economical and better-customized solutions by working with Silicon Mechanics as the integrator, and that they themselves have the ability to implement a cluster environment. However, on occasions there are a small number of opportunities that carry a greater risk and liability profile than our infrastructure will support, and for those Silicon Mechanics will seek to utilize Tier-1 relationships. Silicon Mechanics maintain a relationship with Hewlett-Packard (HP) as a Certified Higher Education Partner, and can work with any customer to determine the benefits of using Silicon Mechanics versus a Tier-1, multinational corporation.

4.9 Linux Networx

Linux Networx Inx (LNXI) enjoyed a period of substantial growth as an HPC cluster vendor between 2000 and 2006. However it had become clear [26] that the company stagnated after a series of big procurement wins in 2005 and 2006. When LNXI did not attend the Supercomputer Conference (SC07) in Reno, it was clear that the company was in trouble, a position advertised by their becoming increasingly sensitive about how the company was portrayed in the media. So the demise of LNXI in late 2007 was not entirely unforeseen, the end coming with the sale of its remaining assets to SGI in February 2008.

The decline and fall of Linux Networx, Inc. (LNXI) may serve as a cautionary story to other struggling HPC vendors. The increased competition from larger, more established companies was certainly a factor in its demise. As Tier-1 vendors like HP and IBM moved in with their own cluster computing portfolios, they were able to wield a lot of brand power against their smaller rivals. According to IDC, IBM and HP alone captured two thirds of the $11.6 billion HPC server market. Of course, that still leaves a large section of the market for other vendors.

But competition for that section is fierce. In October 2007, when Appro was awarded the $26.1 million contract to deliver capacity computing to the three NNSA

ASC labs (Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratories), the deal effectively closed the door on all other cluster vendors at those Laboratories for the next 2 years. Linux Networx was almost certainly a bidder for that contract. If large deals like this become the norm, it will cause problems for those smaller vendors who are not successful. While the NNSA had good reasons to make a major one-off investment in their computing capacity, over the long term this strategy may, limit their choice of vendors.

But this alone would probably not have saved LNXI. The resurrection of SGI may have been the final straw, for when SGI emerged from bankruptcy in 2006, LNXI was faced with a resurgent competitor with fresh funding, an established HPC brand, and a new product portfolio aimed squarely at Linux Networx customers. The fact the SGI ultimately acquired the company's assets is bitter irony, especially considering that SGI's current CEO, Bo Ewald, was at the helm of Linux Networx up until April 2007.

SGI and Linux Networx's major investors have exchanged specific company assets—system management software products, intellectual property, and intellectual property rights—for SGI stock, with a number of senior LNXI people joining their former competitor. However, it is likely that the majority—probably around 60–70%—of the 140 or so LNXI employees faced redeployment.

As far as the LNXI products themselves, SGI does not intend to offer Linux Networx cluster systems, presumably since the overlap with SGI's portfolio would be confusing to customers and expensive to support. SGI will however attempt to leverage some of the intellectual property from the defunct company. Linux Networx was working on its next-generation cluster management software, which, according to Ewald, is a great fit for SGI's recently announced Industrial Strength Linux Environment (ISLE), so that work will be preserved. In addition, SGI intends to support LNXI's system management tool, Clusterworx, and is considering folding it into its system management offerings. LNXI also had patents related to cluster system packaging that will now be added to SGI's own patent portfolio.

The problem that faced Linux Networx is that most of its products were not differentiated enough to offer special value to either its customers or its competitors. If there is general agreement that one vendor has ''systems that are every bit as good'' as another, then it is only a matter of time before competition eliminates the weaker company.

Install Base. Linux Networx had been responsible for building some of the most powerful supercomputers in the world, and as such enjoyed an install base of large systems far greater than any of their UK counterparts. Linux Networx client base covered a range of sectors including Manufacturing, Life sciences, Government and Research, Entertainment, and Oil and Gas. From the Web site it was not possible to provide a breakdown of academic to commercial sales ratios.

Company Details and Size. Linux Networx was based in the USA but had offices/subsidiaries worldwide. The main objective of the company was to help customers improve product development and scientific research by delivering high productivity computing systems. Linux Networx's workforce consisted of hardware and software engineers, installation and integration, staff and sales representatives, although the distribution of employees in these fields was not provided.

Company Outreach and Presence. Unlike many of the other integrators considered in this section, Linux Networx could point to a successful UK installation, having installed an Evolocity II Linux cluster computing system at the European Centre for Medium-Range Weather Forecasts (ECMWF). The system was to be used to evaluate the suitability of cluster technology for broader deployment within ECMWF's high-performance production environment, primarily as a test bed for various aspects of ECMWF's operational workload. Linux Networx successfully met ECMWF's requirements for acceptance testing on June 18, 2005. The cluster was fairly modest in size, consisting of 64 AMD Opteron 2.2 GHz processors, 128 GB of memory and used InfiniBand high-speed interconnects from Mellanox. It is now clear that Linux Networx failed to truly capitalize on this installation, and with no support staff in the UK, failed to win any significant subsequent UK business.

Company Expertise. Linux Networx clearly had a proven track record in the HPC market with a range of systems in the Top 100. The company had incorporated an Active cooling technology to the cluster designs, developed their own management software together with storage to provide a unified HPC solution.

4.10 Western Scientific

Unfortunately little in the way of information on client base/technical assistance is provided at the company's Web site (http://www.wsm.com). Founded in 1978, the company is a global provider of HPC and storage solutions—they supply an extensive line of computing solutions including the latest Beowulf/HPC clusters, RAID and tape storage, high-performance workstations and servers, and networking solutions for the multiuse Linux, Unix, and Windows marketplace.

Western Scientific's main Headquarters are located in the United States. The company also targets the European audience and has an office located in the United Kingdom. They did put in an appearance at the 2004 Daresbury Machine Evaluation Workshop, and while promising much, have been effectively invisible since that event.

The company has partnerships with key Tier-1 organizations including AMD, IBM, and Intel. It also has key collaborations with major high-performance interconnect providers (Mellanox and Cyclades) and mainstream Linux OS (RedHat and SuSE).

4.11 SCC

SSC has a 30-year history of successful growth, with a business that has developed from an initial investment in the UK of 3000 into a 3 billion turnover business with leading positions in seven key European markets and business partners in over 65 countries (see http://www.scc.com). SCC is a strong company within the European marketplace—no information is provided for activities outside the European region.

SCC's international client base spans both public and private sectors with specific expertise in Banking, Financial and Professional Services, Manufacturing, Pharmaceuticals, Retail, Leisure, Telecommunications, Transport and Utilities, together with Defense and Intelligence, Education, Health, and Local and Civil Government.

The company has offices in the following European countries:

- United Kingdom
- Belgium
- France
- Germany
- Italy
- Netherlands
- Spain

To develop and deliver core solution sets and leverage, SCC has developed relationships with key technology vendors. The ''Enterprise Solutions'' section of the company is formally organized into six key technology pillars.

Each pillar operates under its own management with specialist sales, consultancy, and vendor relationship management resources. Each management team ''expends considerable time and resource with vendors and service providers from both a leading and emerging marketplace position. This ensures a continual flow of new ideas and technology components for solutions architects to design, test and deliver best of breed solutions with rapid time to market. Distilling their experience with bringing solutions composed of leading technology components to a wide variety of customers allows SCC to develop thought leadership and trusted advisor status.''

Enterprise Solutions does not operate independently of the traditional customer account manager or sales contact. Customer engagement is on a planned, integrated basis based on business needs which have been intelligently identified. Enterprise Solutions is a rich source of solutions expertise to be introduced and managed by SCC account management teams assigned to customers.

At all stages of engagement SCC Enterprise Solutions works to a process that ensures that value goals are clearly defined at the outset, audited during the project lifecycle, and measured at its conclusion:

- *Enterprise Computing*—process and transact—Key vendor relationships are HP, IBM, Sun, and SGI.
- *Enterprise Storage*—store and retrieve—Key vendor relationships are HP, IBM, Sun, Veritas, Network Appliance, and EMC.
- *Enterprise Communications*—join and protect—Key vendor relationships are Cisco, Nortel, CheckPoint, Nokia, APC, and QinetiQ.
- *Enterprise Software*—collaborate and analyze—Key vendor relationships are IBM, Oracle, Microsoft, and Citrix.
- *Enterprise Print Solutions*—copy and archive—Key vendor relationships are HP and Xerox.
- *Enterprise Solutions Architects*—design and transform—Key vendor relationships are all the above, ISVs and emerging technology partners (e.g., VMWare).

As outlined above, SCC covers a variety of commercial clients as well as academic institutions. No data is provided on the Web site regarding the proportion of commercial to academic install bases The company has a number of key partnerships with Tier-1 organizations including IBM, HP, Sun, and SGI. See above for further details.

4.12 Workstations UK

Historically Workstations UK Ltd were one of the leading cluster companies in the UK, but have largely withdrawn from this marketplace over the past 3 years, and now specialize in consultancy. The company provides consultancy services to eXludus (http://www.eXludus.com) and ScaleMP, and is currently the European agent for Terrascale Technologies (see below).

Workstations UK is based in Amersham, Buckinghamshire in the United Kingdom. Inventors of the blade server, Workstations UK has experience of MPI, PVM, SCI, InfiniBand interconnects, and NAS and SAN storage. The company operates within EMEA space, with a customer base that has included:

- Conoco/Phillips
- Sandia National Laboratory
- EBI
- Raytheon

- NNSA
- Defense Intelligence Agency

The company has been focused on the TerraGrid parallel storage platform from TerraScale technologies. TerraGrid is used in Geophysical processing, Biotechnology, Digital media, Mechanical and Electrical Engineers, and HPC, where TerraGrid ''makes a Linux cluster behave like an SMP.''

5. Tier-1 Suppliers of Commodity Clusters in the UK

Having provided an in-depth analysis of the status and capability of the leading HPC Integrators in the UK, and the three major companies who supply the academic community—Streamline Computing, ClusterVision, and OCF—we now consider two of the Tier-1 suppliers of clusters—Bull and SGI—who do not rely in the UK on integrators as part of their solution.

5.1 Bull

Bull is an information technology company, dedicated to helping corporations and public sector bodies develop open and sure information systems to sustain their business strategies. The premier European-based global IT supplier, Bull has a presence in more that 60 countries, and is particularly active in the defense, finance, healthcare, manufacturing, public and telecommunication sectors.

Bull brings its customers the expertise and know-how to help them transform their information systems. With open technologies and solutions that fully support business strategies, reduce operating costs and risks, Bull helps private and public sector organizations develop their activities, build trust, and adapt to changing market demand as it happens.

Bull offers an impressive vision and expertise in today's market:

- A global commitment to open, standards-based information technologies for total flexibility
- An integrated expertise along the whole IT value chain to build end-to-end solutions
- A comprehensive understanding of secure solutions for mission-critical environments

5.1.1 Breakthrough Technologies

Open Servers and Storage for New-Generation Data Centers. With its NovaScale server range, Bull delivers breakthrough open and scalable platforms and mainframes for data centers and HPC. For example, Bull has delivered to the CEA (the French Atomic Energy Authority) the most powerful supercomputer ever designed in Europe and one of the world's most powerful supercomputers. With its Escala and StoreWay product ranges, Bull also delivers open Unix servers and comprehensive storage solutions.

Service-Oriented Architecture (SOA)-Based Middleware for Open Computing. A key player in middleware, Bull offers open infrastructure software, ranging from workflow and service bus tools to process orchestration and application platform suites. Bull was a founder member of ObjectWeb, which is now the premier consortium worldwide for open-source middleware. Based on open-source components, Bull's open middleware solutions combine all the freedom of open source with comprehensive manufacturer support.

Security Solutions for a Safer World. A European leader in IT security, Bull delivers advanced security solutions for an open world, including identity and access management, cryptography, e-signature, payment security, and Web services security. Many of the most sensitive high-tech, financial and defense organizations in Europe rely on Bull solutions for securing identities, exchanges and access to their information systems.

5.1.2 Comprehensive Services and Solutions

IT Consulting. From IT architecture and ''urbanization'' to project management support, Bull helps organizations design or re-engineer their information systems to support their business strategies. Seven of the 10 new countries that recently joined the European Union have chosen Bull to help them upgrade their taxation and customs systems.

IT Integration. With an innovative development factory, inshore and offshore services, Bull enables businesses to build SOA-based applications and ''industrialize'' their information systems, for reliability and flexibility. Half the top 10 European telecom players have relied on Bull services and solutions to build secure, new generation mobile applications.

IT Operations. With 24/7 support, insourcing and outsourcing services, Bull helps customers hand over responsibility for all or part of their IT operations, so they can concentrate on their core business priorities. Many sensitive public sector organizations in Europe outsource their recovery centers to Bull.

5.1.3 Research and Development

Bull's R&D labs are working hard today on the technologies for building the open, flexible and secure IT infrastructures of tomorrow. Bull has a long history of R&D success, from the premier mainframe computers in the 1960s and the invention of smart cards in the 1980s, to the next-generation data center computing, SOA-based middleware for open computing and advanced security technologies for a trusted world today.

Bull has also developed closed R&D partnerships with lead edge partners such as Intel, IBM, Microsoft, Novell-Suse, RedHat, SAP, and other corporations, as well as open-source communities (OSDL, ObjectWeb, etc.) and European R&D programs (ITEA, etc.). Thanks to its advanced research on new technologies, as well as its collaborative programs with a large network of partners and communities worldwide, Bull customers benefit from advanced IT solutions that help them grow ahead of the competition and deliver exceptional performance.

5.1.4 High-Performance Computing

Although virtually unknown in the world of HPC as recently as 2005, Bull has now won worldwide recognition thanks to Tera-10, the first large-scale supercomputer designed and developed by Bull for the CEA, ranked in June 2006 as number one in Europe and number five in the world. Since then, Bull has enjoyed significant growth with HPC customers in 15 countries across three continents, including Cardiff University who purchased a 25 Tflop cluster in February 2008. In industry, prestigious customers include Alcan, Pininfarina, Dassault-Aviation, and Alenia. To further develop and grow its business in the HPC marketplace, Bull has recruited more than 100 additional engineers with expertise in HPC technology in the past 2 years, to boost the existing teams and specialists. This recruitment drive has covered every area, from HPC development and benchmarking, to marketing, sales, maintenance, and management. These technical teams now make up Europe's largest group of experts in this sector amongst any of the companies involved in HPC.

5.1.5 Install Base

Major HPC installations in the UK are shown below:

2004 National Oceanographic Centre (NOC)—2 $\times$ 32 Itanium core SMP + 8 $\times$ 4 Itanium core servers/Quadrics Interconnect
2005 Irish Centre for HE Computing (ICHEC)—32 Itanium core SMP system
2006 Manchester University—26 $\times$ 8-core Itanium cluster with Quadrics Interconnect and Lustre File System
2006 Loughborough University—Department of Mathematical Sciences; 22 $\times$ 8-core Itanium cluster with Quadrics Interconnect and NSF

2008 Plymouth Marine Laboratory—320-core Intel Xeon Harpertown cluster/ InfiniBand Connect-X Interconnect
2008 Cardiff University—2048-core Intel Xeon Harpertown + 64-core Intel Tigerton cluster with InfiniBand Connect-X Interconnect and Lustre File System

5.1.6 Company Details and Size

Employee numbers	UK	Group
Administration	18	674
Development		777
Distribution		316
Management	19	74
Marketing	4	111
Procurement/purchasing		302
Sales	15	477
Technical support	220	2133
Project management	9	500
Others [R&D, functional consultants (business intelligence, ERP, architects), system engineers, integration/infrastructure]	4	2200
Total	289	7564

5.1.7 International Presence

Bull is Europe’s leading computer manufacturer, with Headquarters in Paris and R&D facilities in Paris, Grenoble, and Phoenix. Bull is a publicly owned company quoted on the French stock market. Present in over 100 countries, Bull is particularly strong in the government, banking, finance, telecommunication, and industry sectors and has a portfolio which includes server, storage, and security products.

In recent years, building on the strength of Bull NovaScale server technology, Bull has established itself as a leading European supplier of HPC solutions with a turnover for HPC systems growing from € 1.4 M in 2003 to an estimated € 70 M this year. In 2007 Bull installed 50,000 cores for HPC applications and grew its HPC user base to over 100 customer sites in 13 countries.

Overall revenues last year were in excess of €1 billion.

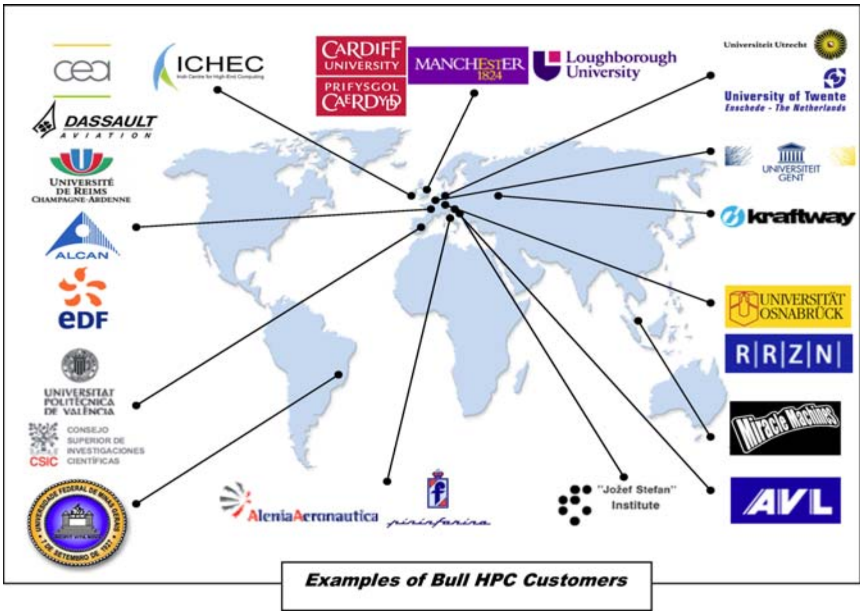
5.1.8 Company Expertise

HPC is a key area of investment and growth by Bull. This is demonstrated by creating the Bull’s HPC Competence Center, which is now established as the leading HPC expertise center in Europe. It provides Bull customers with:

- Expertise in choosing the most appropriate HPC architectures to match their needs

- Expertise in optimizing user applications to fully exploit the available power
- High-level skills relating to Linux and its support
- In-depth knowledge of the open-source software that forms an integral part of Bull's HPC offering
- Close proximity to Bull experts—in a European time zone

The mission of the Bull HPC Competence Center is to develop the clustering and software solutions for the HPC market with Bull NovaScale computers and to be a competence center for Bull HPC customers. The center participates in workshops and different HPC seminars; it provides expert to expert support to Bull customers and partners.


cea
ICHEC
Cardiff University Prifysgol Caerdydd
Manchester 1824
Loughborough University
Universiteit Utrecht
University of Twente Enschede - The Netherlands
Dassault Aviation
Université de Reims Champagne-Ardenne
Universiteit Gent
kraftway
Alcan
EDF
Universität Osnabrück
R|R|Z|N|
Universitat Politècnica de València
Consejo Superior de Investigaciones Científicas CSIC
Miracle Machines
Alenia Aeronautica
pininfarina
"Jožef Stefan" Institute
AVL
Examples of Bull HPC Customers

Located in Grenoble, France, the center is close to scientific universities and research laboratories and takes advantage from this environment in various areas, such as Ph.D. students, collaborative experiments, and projects. The center is also involved in different European projects under organizations like Information Technology for European Advancement (ITEA). ITEA is an 8-year strategic pan-European program for advanced precompetitive research and development in embedded and distributed software. These projects are supported financially by all 33 countries in the EUREKA framework. Through these projects Bull has direct links with Universities specialized in HPC research and development like Stuttgart, Dresden, or Versailles and also with industrials like Intel Gmbh, CEA, Dolphin, MandrakeSoft, etc.

The teams in the Competence Center include around 70 dedicated people, which cover all the different aspects of HPC components, for example:

Linux Kernel Group. This group's focus is the optimization of the scheduler for HPC applications, and the placement for the threads, processes and data for the NovaScale architecture. The group has contributed to Atlas (consortium for the support of Itanium®2 and Linux high-end features) in the domain of recovery. It contributes to the kernel development in the domains of debuggers (kdb project), tests (Linux Test Project), and high-performance features (Linux Scalability Effort) with the development of CPUSET.

Special attention is also paid to the optimization of the I/O for huge data throughput. Bull validates and supports a Linux distribution well adapted to clusters of NovaScale machines.

MPI and Global File System Group. This group develops an MPI library especially optimized for the NovaScale architecture.

Cluster Management Group. This group develops and integrates all the components that are used to monitor, control, install, and debug the hardware and software components. Different open-source components from the scientific community are used and integrated, in addition to specific components that are developed for a large number of nodes. The group includes specialists in human interfaces, data bases and system monitoring for the developments of monitoring solutions, and engineers trained in the use of clusters for the development and integration of resource and batch managers.

Benchmark and Scientific Application Group. This group provides technical support to customer requests and calls for tender. It includes engineers and scientific researchers well trained in multithreaded applications (OpenMP), Fortran programming, parallelization using MPI, and all the profiling tools. It also performs the porting of scientific applications as well as the technical relationship with ISVs in the area of HPC. Certified measurements are performed for a group of well-known scientific applications.

This group interfaces with Intel compiler specialists and provides expertise to customers in the domain of programming environments. Access to machine time is provided to partners on a special case basis through secured links.

Performance Group. This group develops and supports the measurement tools that are used for the system tuning and benchmark analysis. It covers both hardware and software aspects, to gain insight into benchmark behavior and allows the modeling of systems under development. It includes senior specialists in performance analysis and Ph.D. students. The manager of this group is the Bull representative at SPEC, for both

OSG and HPC committees. This group interfaces with the University of Versailles, in the domain of performance tuning, especially code optimization.

Storage Group. This group supports the storage that is attached to the I/O nodes of the NovaScale cluster: interfaces with the storage vendors, optimization of parameters of the drivers and connections, techniques for high-availability solutions of the I/O nodes within the global file system.

5.1.9 Relationship to Tier-1 Organizations

Bull works closely with major industrial partners and software vendors to ensure that their product portfolios are available on the NovaScale® server ranges. In collaboration with its ISV partners, Bull can offer various middleware software to augment cluster productivity, as well as optimized vertical offerings for the following sectors:

- Industrial production (automotive, aeronautics, etc.)
- Life sciences (medical research, pharmaceutical industry, chemistry, biotechnologies, etc.)
- Oil and gas industry (reservoir exploration and production)

Software partners
Allinea
Altair
Ansys
AVL
DC-Adapco
ESI Group
Fluent
LSTC
Platform
Simula
TotalView Technologies

Hardware partners
Cisco
DataDirect Networks
EMC
Foundry Networks
Intel
Mellanox Technologies
NEC

nVIDIA
Quadrics
Supermicro
Voltaire

Bull is already engaged in the design of petascale solutions. The Military Applications Department (DAM) at the French Atomic Energy Authority (CEA) and Bull have signed a collaboration contract to design and build Tera-100, the future supercomputer to support the French nuclear weapons Simulation Program. This long-term project will consist of two phases:

1. An initial R&D stage will enable the technologies needed by the new computer to be validated; these new technologies will have many repercussions in the future, both for industry and society.
2. A second phase will enable the CEA to acquire and implement Tera-100, the first petaflops-scale system to be designed in Europe. To meet the needs of the CEA's Simulation Program, the supercomputer will stand out by its capacity to run a wide spectrum of applications, by the fine balance between its processing power and data throughput, as well as by its fault-tolerant capability. As a real multipurpose system with extremely high levels of productivity, Tera-100 will also be developed on the basis of open-source software and X86 architecture processors.

Developing the new Tera-100 supercomputer will involve significant upfront research and development work. Bull and the CEA will combine their respective skills and expertise in this area: in particular, Bull will provide its know-how in the design and operation of high-performance servers, along with the software development expertise needed to operate large-scale systems; for its part, the CEA will contribute its expertise in specifications, computer architecture and application development, as well as its in-depth understanding of infrastructures for very large data centers. As a result, there will be several hundred highly qualified engineers and researchers working on this project.

5.1.10 Relationship to Other System Integrators

None in the UK. Corporately Bull owns Serviware, an HPC Integrator operating principally in the French market.

See http://www.wcm.bull.com/internet/pr/rend.jsp?DocId=301835&lang=en and http://www.serviware.fr/.

5.2 SGI

SGI is a company of 26 years standing graduating from the design of graphic workstations to large shared-memory systems, visualization and storage products to their entry into the commodity cluster market in 2006. Since that date SGI has shipped in excess of 150,000 cores of Intel Itanium and Xeon dual- and quad-core processors.

5.2.1 Install Base

The median cluster size in the January–December 2007 time period was 128–256 cores but SGI has been awarded contracts both in the small or departmental category of 0–64 cores (previously white-box territory based on budget) as well as maintaining their traditional customer market of large and complex research systems of 512 cores and above. The immediate success enjoyed by SGI on entering the cluster market was driven by existing customer expectations of the technical ability and services support demonstrated by SGI in MPP computing. Also by the ISV's focusing on commodity X86 products as they were already working with SGI's Application Engineering and the new novel architecture designed and offered by SGI.

The cluster product portfolio includes the commodity Altix XE 1U server chassis incorporating the Atoka motherboard (dual-motherboard technology), designed jointly by SGI and Intel and manufactured by Supermicro; and the innovative bladed Altix ICE (Integrated Computing Environment) with integrated switch technology.

The ICE cluster is SGI's response to the rapid growth in commodity clusters' size and complexity to deal with larger data sets creating in turn issues for system administration, manageability, space, and power. ICE also features SGI's ''Power Up and Go'' technology, factory integrated and delivered to site fully racked. In 2007 New Mexico became the host of the 14,336-core SGI® Altix® ICE system; the supercomputer is noteworthy for more than its sheer power: the new system was up and running only 48 h after it arrived; a significant accomplishment because many top-ranked supercomputers can take weeks or even months to deploy. Configured with Intel® Xeon® processors, 28 TB of memory, and a 172 TB SGI® InfiniteStorage 4500 solution, the state of New Mexico acquired the system as part of an economic and educational development effort driven by Governor Richardson. Through the New Mexico Computing Applications Center (NMCAC), the state plans to leverage the supercomputer and storage resource to partner with private businesses and New Mexico universities on research and development projects, attract top academic researchers, and help communities solve complex problems.

Reference sites in the UK for ICE include the University of Exeter and the National Oceanographic Laboratory, Southampton, Honda, and McLaren Formula 1.

In June 2007 Exeter University purchased an SGI Altix Ice 8200 compute cluster with 128 quad-core Intel Xeon 5300 processor cores and 11 TB of useable disk space. The system gave the group several times the computational capacity of the UKAFF facility and enabled them to undertake similar sorts of calculations in greater numbers. Then in December the University took delivery of an 1152-core SGI Altix Ice 8200 based on quad-core Intel Xeon 5400 series processors. The new solution is an order of magnitude more powerful than anything the Astrophysics Group previously had access to.

SGI was selected because it offered a complete solution incorporating superior application expertise, best of breed support and strong collaboration ties enabling SGI's specialist knowledge to be embedded within the group's core engineering teams and so maximize their scientific output. Because the Altix ICE features optional water-cooled doors capable of dissipating 95% rack heat, it also had almost no effect on their data centers ambient temperature. This had played a key role in the contract award at Honda.

NOC, part of the University of Southampton, has installed an SGI Altix ICE bladed cluster, a total of six racks with water cooling and 768 Intel processor cores based on Intel Harpertown quad-core technology. As with all medium to large installations The NOC was assigned a single point of contact for the management of this project and provided with a detailed statement of Work both as part of the Tender Submission and ahead of delivery to ensure all aspects of delivery and installation were catered for. The main or core code for this system, NEMO, was heavily benchmarked by SGI to demonstrate the suitability of the innovative ICE architecture and the dual-InfiniBand network within the individual rack units. SGI provided all of this information to NOC so that the system could begin to run code at the earliest opportunity.

SGI market sectors are classic HPC—for Government, Research, and Industry—and Enterprise. Worldwide this splits out as government 57%, Research 16%, Industry 23%, and Enterprise 4%. The current division between HEI's and industry installations for SGI Ltd is approximately 73% Academic and 27% commercial but this is project and funding dependant year on year.

5.2.2 Company Details and Size

SGI Ltd is headquartered in Thames Valley Park, Reading with an outreach office at the Daresbury Innovation Park, Warrington. Currently there are 134 full-time employees in the UK and plans to expand across Sales and Technical services. The global Headcount is 1600+ employees working with 4000+ customers.

All hardware R&D is done in the USA offices but core application specialists work out of each country office. SGI is a technically focused company using the expertise of its 660 technical staff to drive their market (20 of these are UK based). The global turnover is $100 million.

Recently, SGI relaunched themselves into the visualization market with a new range of products designed to further enhance the heterogeneous cluster environment. VirtuVN and VUE are the products that provide users of the ICE and Altix environment with the ability to visualize locally and remotely. Returning to their traditional marketplace, VirtuVN and VUE are based upon commodity infrastructure and open source with exciting roadmaps into the future.

SGI still remains at the forefront of MPP and shared-memory technology, with the Altix 4700 and the Altix 450. Combining both SMP and MPP creates a hybrid cluster allowing cluster administrators to leverage MPP and MPI, visualization and novel architectures in a heterogeneous environment. This delivers a wide range of capability that a generic compute resource often fails to deliver.

SGI recognizes then the increasing complexity in the *ad hoc* evolution of cluster technologies and in the growth of HPC generally and makes the transition from hardware sale by a Tier-1 manufacturer to solution sale with the addition of their Professional Services Group and Applications Engineering Team. Recently referring to themselves as a systems company, SGI places value on ''fit-for-purpose'' solutions for their customers, working with a number of third-party products, for example, in providing sales and support of file systems such as GPFS, Lustre, and Panasas, and archiving and backup products such as Backbone and Riverbed.

Professional Services within SGI are the division that has additional expertise in non-SGI products and supplies and supports all third-party partner offerings. Support is provided in the UK. SGI remains the support contact and can offer all levels of support directly via our Award winning ''follow the sun'' Technical Support and Customer Services teams. The Professional Services consulting team has been rated No. 1 in the industry by SatMetrix.

SGI is also building relationships with Channel partners, currently working with EDS, Accenture, and CSC, alongside the technical and development relationships, for example, with ISVs to build on access to current and expanding markets for HPC.

5.2.3 International Presence

SGI has demonstrated solid success in deploying clusters of varying size and complexity. See HLRN example where ICE is being deployed over multiple sites and managed with MOAB from Cluster Resources and supported by SGI (http://www.sgi.com/company_info/newsroom/press_releases/2007/december/hlrn.html).

A small snapshot of worldwide reference sites is below:

Organization	System size (CPU cores)
NASA (Ames), US	40,000
HLRN (German Supercomputing Centre), Europe	25,000
NMSC (New Mexico Supercomputing Center), US	14,336
CINES (Centre Infor. Nat. d'Enseignement Superieur), Europe	12,288
TOTAL, Europe	10,240
NASA (Ames), US	4096
Idaho National Labs, US	2816
NASA (Langley), NRL	2048
McLaren F1 racing, Europe	Classified
Honda F1 racing, Europe	Classified
Europe: University of Exeter, Roshydromet (RHM)	1000+
NEW: Mazda, APAC	
IFREMER, Europe	768
Europe: National Oceanographic Centre Southampton (NOCS), MIMOS, TU Dresden Data Center 1, TU Dresden Data Center 2	512+
USA: UFSC, Sandia National Lab	

5.2.4 Company Expertise

Every company in the world, ultimately, is a reflection of its customers. SGI as a Tier-1 vendor has delivered high-performance and computing solutions to the scientific, government and engineering communities for more than 26 years. As a result, SGI is very familiar with the demanding requirements of technical computing customers, and SGI systems are built to satisfy these requirements, not only in terms of delivered hardware performance, but also in broad software functionality and robustness.

SGI continues to build strong relationships between our customers and encourage partnerships to help drive research productivity, knowledge transfer, and institutional awareness. Examples include relationships with the following world-established institutes:

Organization	Principalities	Codes
TU Dresden/Wellcome Trust Sanger Institute, Europe (New)	Bioinformatics	BLASTn/p

(*continued*)

(*continued*)

Organization	Principalities	Codes
Leibniz-RechenZentrum Muenchen (LRZ), Europe (New)	Materials Science	VASP
NASA Goddard/Ames/Langley, US	Weather, Climate, CFD	EC002 MITgcm
University of Exeter, Europe (New)	Theoretical Physics	in-house
University of Cambridge (Stephen Hawkings), Europe	Applied Mathematics	in-house
National Oceanographic Centre Southampton, Europe (New)	Ocean/Marine	NEMO
Mclaren F1 racing/Honda F1 racing, Europe (New)	Aero/computational fluid dynamics	Fluent, Star-CD
Unilever R&D Port Sunlight, Europe (New)	Pharmacology, safety, and health quality	Gromacs, DLPOLY

Computer architects and system designers have made tremendous advances in the performance of computer systems over the past several decades. However, there is clear evidence that with each new generation of computing systems, regardless of architectural approach, the gap between actual applications performance and theoretical peak performance is growing. One of the reasons is that applications performance is increasingly dependent on a range of systems resources, in extremely complex and nonlinear ways.

At SGI, we have a goal to base our future systems architectural design, on these dependencies of a broad spectrum of compute intensive applications used in scientific and engineering communities:

- Computational Structural Mechanics (CSM)
- Computational Fluid Dynamics (CFD)
- Computational Electromagnetics (CEM)
- Computational Chemistry and Materials Science (CCM)
- Computational Biology (BIO)
- Seismic Interpretation (SEI)
- Reservoir Simulations (RES)
- Climate/Weather/Ocean Modeling and Simulation (CWO)
- Data Analytics (DA)

We therefore believe that comprehensive profiling and characterization of the above applications set is the key to a successful next-generation systems architectural design. Our application engineers started developing a set of profiling tools that facilitate better understanding of the needed compute resources and an accurate performance estimation of future architectures. Understanding the computational profile and communication pattern of an MPI application is essential to achieve scalability on thousands of cores with sustained performance in the Tflop range. SGI recently optimized HIRLAM for a partner customer, for example.

5.2.5 Marketplace

Moving forward SGI reacts and responds well to their technical users' growing requirements, as well as to those user markets joining HPC. With the development of ProPack software tools and the strategy for ISLE, SGI wants to reduce the complexity of HPC management, avoiding proprietary OS platforms, and attract more of these new markets. Large HPC systems are difficult to build, use, and manage where users do not necessarily have an easy and uniform way to manage the large collection of management software—BIOS, OS, system tools, libraries, file system, profilers, debuggers, compilers, job scheduler, and storage management—which are necessary. ISLE is the SGI solution; a robust and integrated software environment built on open standard Linux with a comprehensive set of products and services to support developers, users, and administrators. It incorporates a SOA that provides a framework and interfaces to combine service components (services) into management applications. Effectively SGI's ISLE will unify the various management applications (SGI and third party) into a cooperative, cohesive product set for administrators and users, thus simplifying development and deployment by facilitating interactions among services and applications and enabling control and optimization across an entire computing environment. It is worth noting here that SGI is also working closely with Microsoft Corp. with its own HPC Roadmap.

There are a number of software developers alongside the application specialists within SGI who continue to optimize for our hardware platforms, interconnects, and infrastructure but also support the multiprocessor multicore technologies, working with key partner and processor manufacturer Intel Corporation.

In the June 2008 Top 500 compute efficiency between the various systems could be highlighted by the core count difference but performance efficiency is demonstrated clearly below:

Position 319 SGI ICE	1152 cores	Achieved: 11.21 Tflops	Theoretical: 13.04 Tflops	86% efficient
Position 320: HP cluster	2560 cores	Achieved: 11.20 Tflops	Theoretical: 20.48 Tflops	55% efficient

The system at Position 319 is the University of Exeter, part of the SGI Early Access program to ensure delivery of a working, reliable system and a collaboration partnering in practice.

Maintaining hardware and software development teams has allowed SGI to continue both at the forefront of their technical development and early adoption of novel architectures. For example, SGI's own accelerator technology focused on the Blast application and the sale and support of GP GPU, FPGA, and accelerator cards from other suppliers (Xtremedata, Inc., ClearSpeed and CellBE).

5.2.6 Relationship to Other Organizations

As we move further into the realms of multicore processing and such environments become accessible and affordable, then their management and reliability requires a paradigm shift in thinking. The attainment of truly petascale computing brings with it issues of operational management in addition to resolving the issues of multicore scaling. There is a high probability that the first true petaflop environments will be pure computational science research projects. Next will follow top tier academic and government funded laboratories using the environment as a capacity compute resource whilst as we enter the world of multicore proper, petaflop compute environments will be affordable by all central academic institutions. This is to happen within the next 3–5 years.

Some of the top technology topics at the current time are multicore and hybrid processors, contextual computing, augmented reality, semantics on the Web, social networks, and software and cloud computing; SGI Ltd is currently working in partnership with Constellation Technologies for this. As the requirement to run larger more complex applications continues to grow other considerations are the power, cooling and footprint of new system architectures and the environmental aspect or impact of these systems.

SGI is a supporting member of the Green Agenda and a member of The Green Grid—a global consortium dedicated to advancing energy efficiency in data centers and business computing ecosystems. The Green Grid is focused on promoting the adoption of energy-efficient standards, processes, measurements, and technologies. SGI's technology R&D reflects this commitment in the highly efficient power supplies, power-efficient DIMMS and data center efficiencies, and savings via their water-cooled doors.

SGI also has a remanufacturing Division (RPG) where working systems are effectively recycled and returned to the market.

A large part of the HPC market history from development to today is tied into SGI as a company and the longevity of the business—continuing to deliver new concept but production ready systems has always been the key to the Success of SGI. As the

HPC market matures, there is a recognition amongst customers that this expertise and knowledge of the total cluster solution is indeed to be valued—where ‘‘time to science’’ is factored in SGI can deliver a cost-effective HPC solution.

6. Evaluating the Performance of Commodity Clusters

Up to this point we have not considered how an organization might best decide on the optimum cluster technology to deploy against its intended workload. While price and the supplier credentials are clearly critical factors, what about the technology itself? Just what node configuration and processor will prove optimal? What level of interconnect capability is required—both latency and bandwidth? What are the likely demands on the associated storage configuration—both in terms of overall capacity and performance? In looking to answer these questions, a clear understanding of the likely performance of the future workload and the applications that will comprise that workload is clearly vital in deciding the optimum cluster configuration to deploy.

As part of our support for the Distributed Computing community in the UK, we have carried out a variety of performance evaluation exercises that look to assess current and emerging commodity systems in scientific and technical computing through a variety of synthetic and application-based floating-point metrics. The primary goals of these evaluations are to (a) determine the most effective approaches for using each system, (b) evaluate benchmark and application performance, both in absolute terms and in comparison with other systems, and (c) predict scalability, both in terms of problem size and in number of processors. Our analysis relies on performance measurements of application independent tests (microbenchmarks) and a suite of scientific applications that are in active use on many large-scale systems. The microbenchmarks we used provide information on the performance characteristics of the hardware, specifically memory bandwidth and latency, and intercore/interprocessor communication performance. The goal is to use these measurements to provide insight into application performance. The scientific applications we use are taken from existing workloads within the SRIF3 University community, with a focus on Cardiff University itself, and represent various scientific domains and program structures—molecular dynamics, computational engineering, and materials simulation to name a few.

6.1 Node Performance

As noted at the outset, the dominant trend in the semiconductor industry strategy during the past 2–3 years has been to increase processor throughput by providing more PEs or ‘‘cores,’’ rather than by increasing the operating frequency. Intel, AMD, and others are migrating to multiple ($2\times$, $4\times$, $8\times$,...) cores within a single ''processor.'' Thus the next few years of server design will involve more cores and higher integration. While dual-core processors dominated server solutions from all leading suppliers during 2007, quad-core processors are set to dominate the server landscape during 2008. Indeed, the Cardiff SRIF3 procurement was faced with this choice of either staying with the proven, tested dual-core technology, or grasping the opportunity represented by quad-core, and with it the potential benefits of price–performance, packaging and lower power consumption.

The choice of optimal node and associated processors (performance vs cost) has been informed by an ongoing analysis of the serial performance of a wide variety of processors across a number of floating-point intensive benchmarks. Originally conducted as part of the Distributed Computing support program at STFC Daresbury Laboratory, we have been involved over the past two decades in an ongoing comparison of a variety of different computer systems across a variety of application areas.

One of the most useful indicators of CPU performance is provided by the SPEC (''Standard Performance Evaluation Corporation'') benchmarks [27]. This benchmark suite contains nontuned application-based code to measure processor speed for both integer (SPECint) and floating-point (SPECfp) arithmetic. SPECfp95 and SPECint95, their successors, SPECfp2000 and SPECint2000, and more recently SPECfp2006 and SPECint2006 [28] have become industry standards in measuring primarily the performance of a system's processor, memory architecture, operating system, and compiler. CFP2006 is derived from the results of 17 floating-point benchmarks compiled with aggressive optimization, and is the geometric mean of 17 normalized ratios (one for each benchmark). CINT2006 is derived from the results of 12 integer benchmarks compiled with aggressive optimization, and represents the geometric mean of 12 normalized ratios (one for each benchmark). Note that the level of optimization is not mandated. While highly aggressive optimization is permitted, results derived from benchmarks compiled with conservative optimization (SPECfp_base2006) can be submitted.

Having considered CPU performance based on the general SPEC benchmarks, we mention a performance benchmark in the area of computational chemistry which, developed by the authors at the Daresbury Laboratory, has been used to compare the performance of *ca.* 120 computers, ranging from supercomputers to scientific workstations and Pentium, Athlon and Itanium-based PCs [29, 30]. This multicomponent Computational Chemistry Benchmark Suite had been developed to incorporate those tasks typically undertaken by the computational chemist:

1. Matrix Operations, including both matrix multiplication and matrix diagonalization. Given the memory intensive nature of many matrix operations, the suite also incorporated the STREAM (memory bandwidth) benchmark [29, 30].

2. Computational Chemistry Kernels—four typical application kernels (direct-SCF, MD, QMC, and Jacobi eigensolver).
3. End-user application packages featuring both electronic structure and molecular dynamics calculations, using the GAMESS-UK [31] and DL_POLY [32] packages, respectively. The quantum chemistry benchmark included 12 typical applications, including SCF, direct-SCF, CASSCF, MCSCF, direct-CI, MRD-CI and MP2 second derivatives. The molecular dynamics benchmarks included six representative molecular simulations.

It is perhaps worthwhile at the outset to consider the potential shortcomings of using single-processor benchmarks, such as SPECfp2006 or SPECfp2006_base, on the current generation of multicore clusters. The nature of the associated cache hierarchy and memory architecture need to be considered, both arguing for a consideration of not only the SPECfp values, but also the related RATE benchmarks (SPECfp_rate). The complex cache hierarchy of systems such as the quad-core AMD and Intel processors means in practice that with $(n - 1)$ of the n-cores unused, a given single-processor benchmark is actually running in an environment comprising the total L2/L3 cache associated with the entire processor. The recorded level of performance may well bear little resemblance to what might be seen if, for example, all n-cores were each running the same job. A second effect is that of the memory architecture of the processor in question; in many instances the effective memory bandwidth available to a given job will decline significantly as more of the cores are involved in processing. The impact of this effect will be critically dependent on the nature and bandwidth demands exhibited by the application. The impact of cache utilization and memory architecture may be partially quantified by ensuring that all cores are populated in associated ''rate'' or ''throughout'' benchmarks, as captured in the associated SPECfp2006_rate figures for such systems.

In line with the realization that rate or throughput benchmarks provide a far more realistic measure of processor performance than single-core benchmarks, the chemistry serial benchmarks described above have been recast into a rate format. This multicomponent rate benchmark has been developed to incorporate both matrix operations (matrix multiplication and diagonalization) and chemistry applications [both Quantum Chemistry (GAMESS-UK) and Molecular Dynamics (DL_POLY)]. The rate procedure may be described as follow, where on a system with n-processing elements (NPEs):

- For each benchmark (i) run NPEs instances at once and take the elapsed time (last to finish—first to start).
- The rate for this benchmark is given by

$$R_i = \text{NPEs} \times T_{\text{ref}}/T_i,$$

where T_{ref} is the elapsed time on a reference system. In the following examples the reference system is taken to be an AMD Opteron 852 2.6 GHz processor with code compiled with the Pathscale pathf90 compiler (V2.2) scaled to a single processor (NPEs = 1) elapsed time of 100 units.

- Take ''the geometric mean'' of all the benchmarks (with the same NPEs).

The detailed components of this rate benchmark are as follows:

1. **Matrix Operations**—with three distinct classes of operation considered:
   - *Sparse matrix multiply operations (MMOs).* A series of MMOs ($R = A \times B$) involving matrices of increasing order, 100, 200,. . .,1200 (where B is sparse). Three matrix multiplication algorithms—MMO, MXMB0, and DGEMM—are used:
     - *MMO*—A vectorized FORTRAN code from the Cray vector era (outer product), where the treatment of sparsity is included
     - *MXMB0*—An optimized FORTRAN code for scalar processor in which the inner loop is unrolled by 8 to optimize for 64-byte cache lines
     - *DGEMM*—Simply using the Level 3 BLAS DGEMM routine from the associated Maths library, omitting any consideration of sparsity
   - *Diagonalization benchmark.* The performance of seven routines from Maths libraries and quantum chemistry codes are measured, each diagonalizing a number of matrices of increasing size (100, 200,. . .,600).
   - A similarity transformation ($Q^{\dagger}HQ$) widely used in Quantum Chemistry codes. This involves use of library routines, for example, BLAS and is carried out using both a scalar and vector implementation. The QHQ-S benchmark minimizes the number of floating-point operations using a scalar algorithm (based on the dot product, DDOT), while QHQ-V, a vector algorithm, is based on two successive matrix multiplications (using DGEMM), one involving a matrix transpose. Each transformation is carried out on a number of matrices of increasing size (100, 200,. . .,1000).
2. **Molecular Simulation**—five molecular dynamics simulations.
3. **Electronic Structure Calculations**—eight calculations using a variety of electronic structure methods (direct-SCF, DFT B3LYP, MCSCF, direct-CI, MP2-geometry, SCF second derivatives, MP2 second derivatives, and direct-MP2).

We consider here the current status of a number of quad-core solutions from AMD and Intel—AMD's Opteron ''Barcelona'' and Intel's Clovertown/Harpertown processors. Specifically, we present a performance evaluation of three such processors: the AMD Opteron 2350 (Barcelona) and Intel's Xeon X5365 (''Clovertown'') and X54xx (''Harpertown'') processors. We note here that the Barcelona

TABLE III
NODE CONFIGURATIONS UNDER INVESTIGATION

Compute Node/PEs	Processor Attributes
Dell PE 1955, Intel Xeon 5160/3.0 GHz DC	Dual-processor, dual-core
HP DL585, AMD Opteron 885/2.6 GHz DC (PGI6.2)	Quad-processor, dual-core
AMD Barcelona Opteron 2352 2.1 GHz QC (PGI)	Dual-processor, quad-core
Intel Harpertown E5472 2.8 GHz QC	Dual-processor, quad-core
Intel Clovertown 2.66 GHz QC 4MBL2	Dual-processor, quad-core
SGI Ice X3465 Intel Clovertown 3.0 GHz QC 4MBL2	Dual-processor, quad-core

processor is fabricated as a single die whereas Harpertown incorporates two dual-core dies into a single package. We examine the suitability of these processors in dual-socket compute nodes as building blocks for large-scale scientific computing clusters, comparing the performance of several 8-core nodes, one with two Barcelona processors and the other with either two Clovertown or Harpertown processors. These nodes have quite different performance characteristics in terms of both peak and sustained. As will be seen, the best observed performance is not necessarily when all processing cores within a socket, or all cores within a node, are in use. This is heavily dependent on the application characteristics. We would note that while the ultimate goal is to understand performance on large clusters, this performance results from both the performance of the computational nodes as well as their integration into the system as whole. The single node performance is vitally important in large systems for capability computing.

To illustrate the relative performance of the variety of dual- and quad-core systems of Table III, we first show in Figs. 3 and 4 the results of two of the components of the ChemRate benchmark that exhibit quite different performance attributes. The DLPOLY rate benchmark of Fig. 3 shows a linear increase in performance with increased core count for each of the processors under consideration, suggesting that DLPOLY is not subject to memory bandwidth constraints. This is further emphasized by the quad-core Intel Clovertown and Harpertown processors showing similar levels of performance, similar to that found on the dual-core Woodcrest processor. The leading processor is seen to be the Intel quad-core Harpertown—with all Intel processors significantly faster than both dual- and quad-core AMD processors. Considering the latter, we find that processor clock speed drives performance, with the dual-core 2.6 GHz Opteron processor outperforming the quad-core 2.1 GHz Barcelona processor.

In contrast, the Similarity Rate Benchmark (QHQ-S) exhibits quite different characteristics. Memory bandwidth is now a dominant constraint on performance, with Fig. 4 showing that all Intel processors suffer a significant decrease in

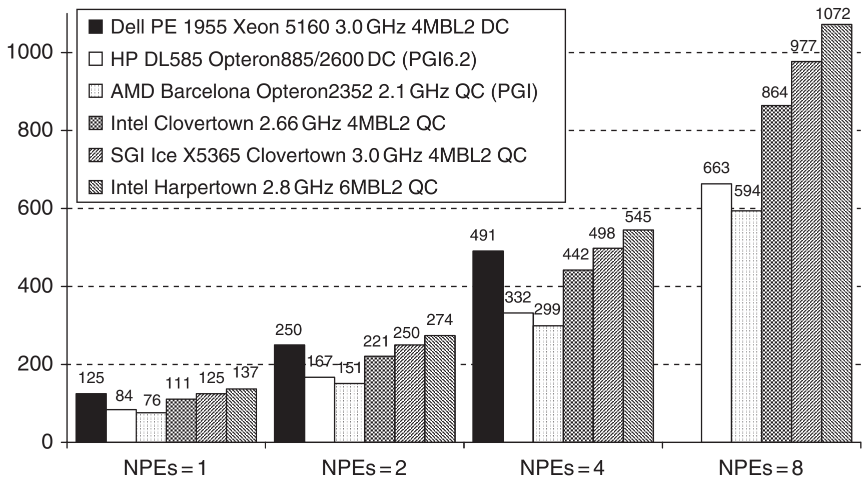


FIG. 3. The DLPOLY rate benchmark. Performance on the node configurations of Table III as a function of the number of processing elements (NPEs). The ratings are computed relative to a single AMD dual-core Opteron 852/2.6 GHz processor.

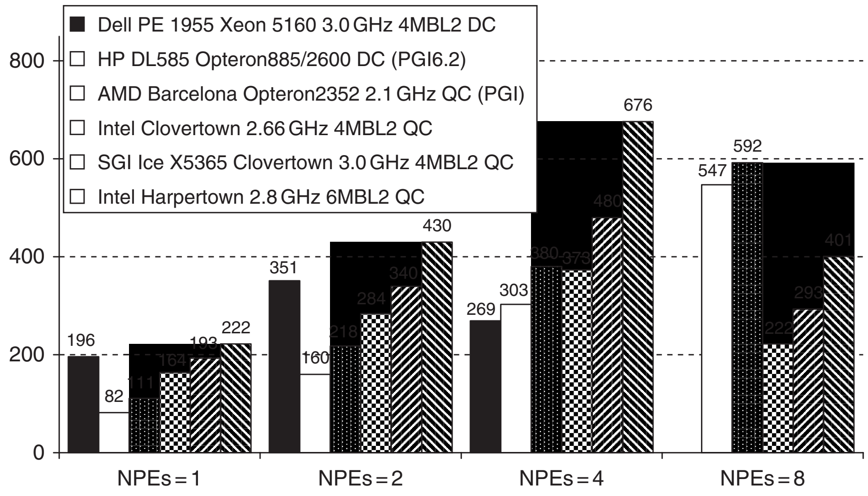


FIG. 4. The QHQ-S rate benchmark. Performance on the node configurations of Table III as a function of the number of processing elements (NPEs). The ratings are computed relative to a single AMD Opteron 852/2.6 GHz processor.

performance when all cores of the processor are running the application. In contrast, both dual- and quad-core AMD systems continue to show improved performance with increasing occupancy of the processor, such that at NPEs = 8 the Barcelona system is some 50% faster than the Intel Harpertown node. We also find that the improved Barcelona architecture results in superior performance compared to the dual-core socket-F Opteron processor.

Having derived the individual rates for each component (i) of the rate benchmark—$R_i = \text{NPEs} \times T_{\text{ref}}/T_i$ —(where again T_{ref} is the elapsed time on an AMD Opteron 852 2.6 GHz processor scaled to a single-CPU time of 100 units), we now need to consider which of the components should be included in an overall performance indicator. This is derived from a consideration of the component tasks that would typically be undertaken by the computational chemist on the system in question. Clearly this should be driven by the applications themselves—in this case DLPOLY and GAMESS-UK, but might also include key kernels likely to be deployed by other computational chemistry codes. This leads us to the following definition for the final Rate, R-Chem,

$$\begin{aligned}\mathrm{R-Chem} = {} & 0.1 \times R_{\text{DGEMM}} + 0.1 \times R_{\text{Diagonalization}} \\ & + 0.4 \times R_{\text{DLPOLY}} + 0.4 \times R_{\text{GAMESS-UK}},\end{aligned}$$

in which a weight of 40% is given to each of the applications, plus a weight of 10% for both matrix multiplication (using the most efficient DGEMM kernel) and matrix diagonalization.

The Final ChemRate benchmark of Fig. 5 shows a number of interesting features. First, we find an excellent increase in performance with increased core count for each of the processors under consideration; in some ways this is not surprising, for none of the four component codes is subject to demanding memory bandwidth constraints. This is further emphasized by the quad-core Intel Clovertown (3.0 GHz) and Harpertown processors showing similar levels of performance to that found on the dual-core Woodcrest processor. The leading processor is seen to be the Intel quad-core Harpertown—with all Intel processors faster than both dual- and quad-core AMD processors, although this performance advantage is less than that found in DLPOLY alone (see Fig. 3). Considering the AMD processors, we find that the quad-core 2.1 GHz Barcelona processor outperforms the dual-core 2.6 GHz Opteron processor at all core counts.

We would conclude from this evaluation that quad-core-based nodes have a compelling performance advantage over their older dual-core counterparts. However, we would stress that this conclusion is highly dependent on application code and discipline—it is almost certain that the recognized memory bandwidth constraints exhibited by the current generation of Intel quad-core processors would be far more apparent in other application areas.

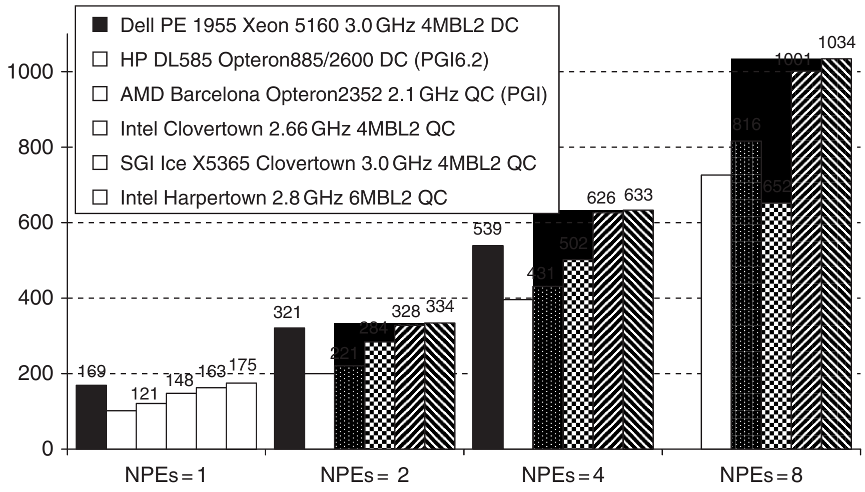


FIG. 5. The ChemRate benchmark. Performance on the node configurations of Table III as a function of the number of processing elements (NPEs). The ratings are computed relative to a single AMD Opteron 852/2.6 GHz processor.

6.2 Evaluation Systems

Having considered the performance attributes of a variety of cluster nodes, let us now turn to a consideration of the performance of the cluster system itself. Our ongoing assessment of a variety of commodity-based systems (CS) has produced a wealth of data generated through access to some 50 systems since 2002 [33]. Nine such systems have been used in the present study (see Table IV), with a particular focus on the emerging quad-core systems from AMD and Intel. Three of these systems feature dual-core processors, CS36 (the ''Darwin'' system at Cambridge University) with Intel Xeon 5160 ''Woodcrest'' 3.0 GHz dual-processor nodes and InfiniPath interconnect, CS44, the cluster at King's College with Intel Xeon 5150 ''Woodcrest'' 2.66 GHz dual-processor nodes and Mellanox InfiniBand interconnect and CS42, an InfiniBand-connected cluster from IBM/ClusterVision with AMD 2.6 GHz Opteron 2218-F dual-processor nodes.

The remaining six systems are based on quad-core processors with a variety of high-performance interconnects. One features AMD ''Barcelona'' Opteron 2350 processors (CS49) with clock speed of 2.0 GHz and Connect-X Interconnect. The remaining systems feature Intel quad-core offerings—two from SGI with the initial Clovertown-based processors (CS50) and the more recent E5440 Harpertown-based

8200EX system (CS54), plus systems from ClusterVision (CS52), Bull (CS51), and Intel (CS47), with InfiniPath, Connect-X, and InfiniBand interconnects, respectively.

6.3 Interconnect Performance

While we have measured point-to-point bandwidth and latency on a wide variety of interconnects, experience suggests that these are of limited value and provide no more than an indication of network performance likely to be encountered in parallel applications. To provide a more quantitative assessment, we have adopted a number of benchmarks designed to provide a systematic evaluation of both point-to-point and collective operations.

We have continued to use the IMB Parallel Communications benchmarks from Intel (the former Pallas (PMB—Pallas MPI Benchmarks [34]) across a variety of parallel hardware. IMB considers a number of point-to-point communications (e.g., Ping-Pong, Sendrecv, and Exchange) plus a selection of MPI Collective Operations (Allreduce, Reduce, Reduce_scatter, Allgather, Allgatherv, Alltoall, Alltoallv, Bcast, and Barrier). Results for the former are reported in Mbytes/s, results for the latter as time (μs) to complete. Each operation is run for a variety of message lengths (0–4,194,304 bytes), with the collective operations performed for various combinations of the number of CPUs available. Of the nine commodity clusters considered here (see Table IV), two feature the low-latency InfiniPath HTX interconnect and seven InfiniBand networks. To provide an example of the output, we show in Fig. 6 the performance of MPI_Alltoall on 64 processing elements—''cores''—of the evaluation systems.

While the performance at large message sizes is largely as expected, the irregular shape of the curves for a number of systems—notably the SGI Ice 8200 cluster and the AMD quad-core Opteron cluster—clearly point to performance issues when dealing with messages of size 512–4096 bytes. Interestingly this behavior on the Ice 8200 is a function of MPI library—using SGI's MPT library [35] removes the performance bottleneck in contrast to MPAVICH. Similar problems are evident at very short messages, when the SGI Ice system (with both MPAVICH and MPT) looks to be significantly slower than the other systems. The two most efficient systems would appear to be the dual-core CS36 Xeon and the quad-core CS51 Bull cluster, the former with InfiniPath HTX fabric, the latter with InfiniBand Connect-X.

It is worth stressing that the IMB benchmarks have proven an invaluable tool in the early identification of interconnect issues on a variety of clusters, issues that became major bottlenecks to performance in the rollout of any subsequent communication-intensive application.

TABLE IV
COMMODITY-BASED SYSTEMS (CSX)

System	Compute Nodes/PEs	Number of PEs	Interconnect	Location
CS36	Intel Xeon 5160 3.0 GHz DC	256	InfiniPath	Cambridge
CS42	IBM x3455 Opteron 2218-F, 2.6 GHz DC	1024	InfiniBand	Birmingham
CS44	Intel Xeon 5150 2.66 GHz DC	480	InfiniBand	King's College, London
CS47	Intel Harpertown E5472 3.0 GHz QC	128	InfiniBand	Intel
CS50	SGI Ice—Clovertown X5365 3.0 GHz QC	1024	InfiniBand	Chippewa Fall
CS49	AMD Barcelona 2350 QC (Scali + PSC)	128	Connect-X	AMD Developer Center
CS51	Bull R422 Xeon E5472 3.0 GHz QC	2048	Connect-X	Cardiff University
CS52	Supermicro Xeon E5430 2.66 GHz QC	480	InfiniPath	East Anglia University
CS54	SGI Altix Ice 8200—Intel Xeon E5440 2.83 GHz QC	768	InfiniBand	NOC, Southampton

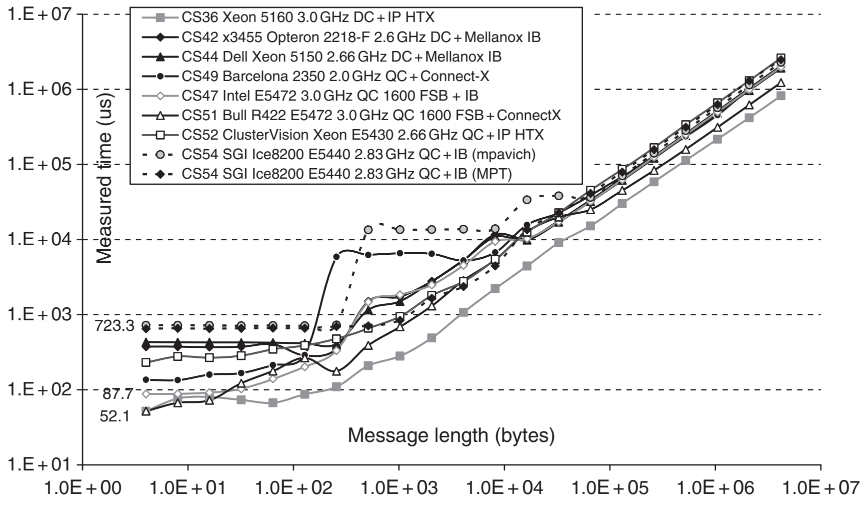


FIG. 6. Performance of the MPI_Alltoallv collective on 64 processing elements of the evaluation systems of Table III.

6.4 Application Performance

The present evaluation concentrates on four application codes—DL_POLY [32], CPMD [36], PDNS3D [37], and ANGUS [38]. These have been chosen given their performance dependency on differing aspects of cluster architecture. DL_POLY is the parallel molecular dynamics simulation package, with both replicated data (DL_POLY2) and distributed data (DL_POLY3) versions. The three benchmark simulations for the replicated data version of the code feature (1) 27,000 ions of NaCl, (2) 8640 ions of NaK disilicate glass with three-body forces, and (3) a 12,390 atom macromolecular system, including 4012 TIP3P water molecules solvating the Gramicidin-A protein. Three simulations have been used for DL_POLY3—the distributed data version of the code—a NaCl simulation with 216,000 ions plus a simulation of a macromolecular system involving a number of Gramicidin-A molecules in water, yielding a total 792,960 atoms. The third benchmark features a system with short-range potentials that should exhibit linear scaling across all processor counts. In common with many simulation codes, DL_POLY typically has little dependency on memory bandwidth and interconnect performance, at least for modest processor counts. More details on these benchmark cases are provided below.

The second application, CPMD, is the *ab initio* Car–Parrinello plane-wave pseudopotential code. In contrast to DL_POLY, the performance of CPMD is critically dependent on a low-latency interconnect, given the central role of the MPI collective MPI_Alltoall. Three test cases have been considered involving the C_{120} and Si_{512} species. In the former we consider both closed-shell singlet (S) and open-shell triplet (T) single-point density optimizations using the BLYP functional. Singlet calculations on Si_{512} used the LDA functional.

Finally, two engineering direct numerical simulation (DNS) codes, ANGUS and PDNS3D, have been included given their known dependency on memory bandwidth, the Achilles Heel of multicore processors. PDNS3D is a simple turbulent channel flow example using the shock/boundary-layer interaction approach; the two test cases (T1 and T2) have a cubic grid size of (120^3) and (240^3), respectively. Test cases for the ANGUS combustion code benchmark include two cubic grid sizes—T1 (144^3) and the larger T2 (288^3).

6.5 Molecular Simulation—DL_POLY

DL_POLY [32] is a general-purpose molecular dynamics simulation package designed to cater for a wide range of possible scientific applications and computer platforms, especially parallel hardware. Application areas include [39] ionic solids, solutions, metals, zeolites, surfaces and interfaces, complex systems (e.g., liquid crystals), minerals, biosystems, and those in spectroscopy. Comprehensive benchmarking of the replicated data (RD) version (Version 2.11) of DL_POLY [40] revealed the limitations inherent in the RD strategy, with restrictions in the size of system amenable to study, and limited scalability on current high-end platforms. These limitations apply not only to systems possessing complex molecular topologies and constraint bonds, but also to systems requiring simple atomic descriptions, systems that historically exhibited excellent scaling on systems with ''slow'' processors and ''fast'' proprietary interconnects, for example, the Cray T3E/1200E. The release of the distributed data (or domain decomposition) version (DL_POLY3) [41] provided significant enhancements to the code's capabilities.

Evaluation of the Coulomb potential and forces in DL_POLY is performed using the smooth particle mesh Ewald (SPME) algorithm [42]. As in all Ewald [43] methods, this splits the calculation into two parts, one performed in real space and one in Fourier space. The former only requires evaluation of short ranged functions, which fits in well with the domain decomposition used by DL_POLY3, and so scales well with increasing processor count. However the Fourier component requires three-dimensional fast Fourier transforms (FFTs) to be performed. These are global operations and so a different strategy is required if good scaling is to be achieved.

The original implementation involved replicating the whole FFT grid on all processors and performing the FFTs in serial after which each processor could evaluate the appropriate terms for the atoms that it held. This method clearly has a number of well-known drawbacks. While both open-source 3D parallel FFTs (such as FFTW [44]) and proprietary routines (such as Cray's PCCFFT) are available, neither adequately address all the issues. The problem is that they impose a data distribution, typically planes of points, that is, incompatible with DL_POLY's spatial domain decomposition, so while a complete replication of the data is not required, it is still necessary to perform extensive data redistribution which will limit the scaling of the method.

To address these limitations, a parallel 3D FFT has been written [41] which maps directly onto DL_POLY's data distribution; this involved parallelizing the individual 1D FFTs in an efficient manner. While the method will be slower than the proprietary routines for small processor counts, at large numbers it is attractive, since (a) while moving more data in total, the method requires much fewer messages, so that in the latency dominated regime it should perform better, and (b) global operations, such as the *all-to-all* operations used in both FFTW and PCCFFT, are totally avoided. More generally the method is extremely flexible, allowing a much more general data distribution than those of other FFTs, and as such should be useful in other codes which do not map directly onto a ''by planes'' distribution.

The 216,000 ion Coulombic-based NaCl simulation of Table V involves use of the particle mesh Ewald scheme, with the associated FFT treated by the algorithm outlined above [41] in which the traditional all-to-all communications are replaced by the scheme that relies on column-wise communications only. The reported

Table V
Time in Wall Clock Seconds for Three DL_POLY3 Benchmark Calculations on a Variety of Dual- and Quad-Core Clusters (see Table IV)

No. of Processing Elements (Cores)	CS36 Intel Xeon 5160 3.0 GHz DC (InfiniPath)	CS42 IBM x3455 Opteron 2218-F, 2.6 GHz DC (InfiniBand)	CS49 AMD Barcelona 2350 QC (Connect-X)	CS51 Bull R422 Xeon E5472 3.0 GHz QC (Connect-X)	CS54 SGI Altix Ice 8200, Xeon E5440 2.83 GHz QC (InfiniBand) MPT	CS54 SGI Altix Ice 8200, Xeon E5440 2.83 GHz QC (InfiniBand) Mpavich
NaCl; 216,000 ions, 200 time steps						
16	151	199	267	139	154	160
32	75	97	136	71	83	79
64	40	49	71	38	44	43
128	23	28		22	28	28
256	18			15	20	24
Gramicidin-A; 792,960 atoms, 50 time steps						
16	261	293	371	273	350	332
32	145	172	215	138	192	174
64	80	96	117	69	99	107
128	52	66		45	68	81
256	39			35	55	75
Argon LJ potential; 4,000,000 atoms, 100 time steps						
16	242	307	424	245	276	294
32	121	154	214	115	130	130
64	62	79	109	58	66	65
128	31	42		29	35	32
256	16	23		14	16	15

timings are for 200 time steps. The second benchmark of Table V is a macromolecular simulation based on a system of eight Gramicidin-A species (792,960 atoms), with the timings reported for just 50 time steps. The third benchmark—a system of 4,000,000 argon atoms—is a straightforward Lennard–Jones short-range potential that should exhibit linear scaling across all processor counts considered, regardless of interconnect.

The results of Table V show a marked improvement in performance compared to the replicated data version of the code [40]. Considering the NaCl simulation, we find speedups of 134 and 148, respectively, on 256 cores of the CS36 dual-core Woodcrest and CS51 quad-core Harpertown, respectively, with similar times to solution. The reduced scalability on the CS54 SGI Ice system under mpavich—

a speedup of 107 on 256 cores—points to the inferior performance noted in Section 6.3; this figure increases to 123 under MPT. Considering the performance of quad-core versus dual-core, the little demand on memory bandwidth made by DLPOLY means that quad-core technology is performing extremely well on this code. We note that the total times to solution on the AMD-based systems point to the socket-F dual-core processor outperforming the quad-core Barcelona processor, in line with the relative clock speeds. In both cases the Intel-based systems significantly outperform their AMD counterparts.

A more compelling improvement with system size compared to the replicated data version of the code is found in the macromolecular Gramicidin-A simulation. Again the SPME algorithm is used for evaluation of the Coulomb field, but there is now the extra complication of constraints on the atoms' motions, which reflects chemical bonds in the system. The shake algorithm is used to evaluate the constraints, and this is again potentially a global operation and so, as for the FFT, good scaling is difficult to achieve. In the distributed data implementation, both SHAKE and short-range forces require only nearest neighbor communications, suggesting that communications should scale well with the number of nodes, in marked contrast to the replicated data implementation. This is borne out in practice—we find speedups of 107 and 125 on 256 cores of the CS36 dual-core Woodcrest and CS51 quad-core Harpertown, respectively. This level of scalability represents a significant advance over that exhibited by both DL_POLY2 and CHARMM [40]. The timings of Table V again point to reduced scalability on the CS54 SGI Ice system when running mpavich—a speedup of 71 on 256 cores. An increased figure of 102, comparable to that seen on the CS36 dual-core Woodcrest, is found under SGI's MPT library. Considering the performance of quad-core versus dual-core, we again note that the total times to solution on the AMD-based systems point to the socket-F dual-core processor outperforming the quad-core Barcelona processor. In both cases the Intel-based systems outperform their AMD counterparts. As expected, the Argon simulation is seen to exhibit close to linear scaling at all processor counts, with the times to solution on the Intel dual- and quad-core solutions effectively identical at higher processor counts.

6.6 Materials Simulation—CPMD

The CPMD code is based on the original code by Car and Parrinello. It is a production code with many unique features and currently has about 200,000 lines of code, written in FORTRAN 77 with the MPI communications library. Besides the standard Car–Parrinello method, the code is also capable of computing many different types of properties, including the inclusion of quantum effects on nuclei with the path-integral method and interfaces for QM/MM calculations. Since January 2002 the source code has been freely available for noncommercial use.

More than 6000 registered users from more than 50 countries have compiled and run the code on platforms ranging from notebooks to some of the largest high-performance parallel computers.

FFTs are an essential part of all plane-wave calculations. In fact, it is the near-linear scaling of FFTs that make large-scale DFT calculations with plane-wave basis sets possible. FFTs are used to transform the charge density and local potential between real space and reciprocal space. The number of these transforms is fixed and does not depend on the system size. On the other hand the transforms of wave functions from reciprocal space to real space and back (needed in the force calculation) has to be done for each state and dominates execution time for small and medium sized systems. Only for large systems (number of atoms larger than 1000) do the cubic scaling inner products and orbital rotations become dominant.

Different strategies are followed in parallel implementations of plane-wave/pseudopotential codes. Parallelization of the CPMD code was done on different levels. The central parallelization is based on a distributed-memory coarse-grain algorithm that is a compromise between load balancing, memory distribution, and parallel efficiency. This scheme achieves good performance on computers with up to about 200 CPUs, depending on system size and communication speed. In addition to the basic scheme, a fine-grain shared-memory parallelization was implemented. The two parallelization methods are independent and can be mixed. This allows us to achieve good performance on distributed computers with shared-memory nodes and several thousands of CPUs, and also to extend the size of the systems that can be studied completely *ab initio*, to several thousand atoms.

Some methods implemented in CPMD allow a further level of parallelization. These methods, such as path-integral molecular dynamics or linear response theory, are embarrassingly parallel on the level of the energy calculation. Typically 2–16 copies of the energy and force calculation can be run in parallel. For these methods, an efficient use of computers with tens of thousands of CPUs can be envisaged. However, in this work only the main Car–Parrinello molecular dynamics part of the code has been used.

The coarse-grain distributed-memory parallelization is driven by the distribution of wave-function coefficients for all states to all CPUs. Real-space grids are also distributed, whereas all matrices that do not include a plane-wave index are replicated (especially overlap matrices). All other arrays are only distributed if this does not cause additional communications. For a general data distribution in both spaces, each transform making up the 3D FFT would include communication between all processors. The data distribution in CPMD tries to minimize the number of communication steps while still having optimum load balancing in both spaces. This scheme requires only a single data communication step after the first transform.

In addition, we can make use of the sparsity of the wave-function representation still present after the first transform and only communicate nonzero elements.

The various load-balancing requirements are interrelated, with a heuristic algorithm used to achieve near-optimum results. Experience suggests that for all cases good load balancing is achieved for the reciprocal space. The restriction to full-plane distributions in real space, however, introduces severe problems in the case of a large number of processors. The number of planes available is typically about 50 for small systems and 200–300 for large systems. This restricts the maximum number of processors that can be used efficiently. The coarse granularity of this approach is also responsible for the appearance of magic numbers of processors where especially good performance can be achieved. This is no major problem because the appearance of these numbers is fully transparent. The efficiency of the scheme described above has a number of limitations which are discussed elsewhere [36].

Shared-memory parallelization on the loop level is achieved by using OpenMP compiler directives and multithreaded libraries (BLAS and FFT) if available. Compiler directives have been used to ensure parallelization of all longer loops (those that depend on the number of plane waves or the number of grid points in real space), and to avoid parallelization of the shorter ones. This type of parallelization is independent of the MPI parallelization and can be used alone or in combination with the distributed-memory approach. Tests on various shared-memory computers have shown that an efficient parallelization up to 16 processors can be achieved.

The performance of CPMD on a variety of dual- and quad-core clusters using up to 256 processors is given in Table VI. The timings suggest a marked dependence of performance on both the respective cluster interconnect and the MPI library in use. Consider the C_{120} simulations involving both closed and open-shell density calculations. For those clusters with recognized low-latency interconnects—*ca.* 1.5 μs MPI latency on InfiniPath HTX and Connect-X—we find speedups in the closed-shell calculations of 103 and 118, respectively, on 128 cores of the CS36 dual-core Woodcrest and CS51 quad-core Harpertown, respectively, with similar times to solution. A reduced speedup of 63 is found on the CS42 Opteron cluster that features standard InfiniBand (*ca.* 5 μs MPI latency). In marked contrast there is clearly no performance scaling on the CS54 SGI Ice system under mpavich beyond 32 cores—hardly surprising in light of the MPI_Alltoall performance noted in Section 6.3. This is to be compared with the performance when running SGI's MPT library. Now we find a speedup of 102 in the closed-shell calculation, with a factor of 14 improvement in time to solution compared to that found with the mpavich library. Considering the performance of quad-core versus dual-core, it is gratifying that the dual-core CS36 and quad-core CS51 clusters exhibit almost identical times to solution in this and the other benchmarks of Table VI. Clearly the demands on memory bandwidth made by CPMD, while somewhat greater than that found with DLPOLY, do not

TABLE VI
TIME IN WALL CLOCK SECONDS FOR FOUR CPMD BENCHMARK CALCULATIONS ON A VARIETY OF DUAL- AND QUAD-CORE CLUSTERS (SEE TABLE IV)

					CS54	
					SGI Altix Ice 8200, Xeon E5440 2.83 GHz QC (InfiniBand)	
No. of Processing Elements (Cores)	CS36 Intel Xeon 5160 3.0 GHz DC (InfiniPath)	CS42 IBM x3455 Opteron 2218-F, 2.6 GHz DC (InfiniBand)	CS49 AMD Barcelona 2350 QC (Connect-X)	CS51 Bull R422 Xeon E5472 3.0 GHz QC (Connect-X)	MPT	Mpavich
			C_{120}; total density calculation of the singlet state			
16	334	368	448	353	458	485
32	115	180	210	122	137	216
64	67	111	124	69	75	375
128	52	94		48	72	979
			C_{120}; total density calculation of the triplet state			
16	682	741	904	726	884	992
32	236	364	426	246	262	441
64	138	223	249	137	150	772
128	109	188		102	142	1991
			Si_{512} total density calculation (20 Rydberg)			
16	542	669	738	550	752	747
32	311	436	448	309	393	423
64	182	258	270	181	226	491
128	118	148		116	178	1042
			Si_{512} total density calculation (60 Rydberg)			
32	1636	1801		1908		
64	789	1040		826		1185
128	587	733		555	709	1224
256	308	432		280	404	2429

inhibit quad-core technology performing extremely well on this code. We note that the total times to solution on the AMD-based systems point to the socket-F dual-core processor outperforming the quad-core Barcelona processor, in line with the relative clock speeds, although the difference is smaller than that found with DLPOLY. In both cases the Intel-based systems again significantly outperform their AMD counterparts.

Two series of single-point density optimizations are reported for the Si_{512} species. The first uses a wave-function cutoff of 20 Rydberg, resulting in approximately 320-K plane waves and a real-space mesh of $108 \times 108 \times 108$ when using the LDA functional and Kleinman pseudopotentials. The second more demanding calculation increases the wave-function cutoff to 60 Rydberg, resulting in approximately 1680-K plane waves, and a real-space mesh of $180 \times 180 \times 180$. Considerably improved scaling is found in this second optimization—increasing, for example, from 73 and 96 on 128 cores of the CS36 dual-core Woodcrest and CS51 quad-core Harpertown clusters in the 20 Rydberg case, to 170 and 218 on 256 cores of CS36 and CS51, respectively. Again the overall times to solution on dual- and quad-core Xeon clusters are similar in both optimizations. We again note the extremely poor performance of the SGI Ice system at higher processor counts when using the mpavich library. The total times to solution on the AMD-based systems point to the quad-core Barcelona and socket-F dual-core clusters showing similar performance. We again see that the performance differential between the Intel-based systems and their AMD counterparts is significantly less compared to that found with DL_POLY.

6.7 Computational Engineering—PDNS3D

Fluid flows encountered in real applications are invariably turbulent. There is, therefore, an ever-increasing need to understand turbulence and, more importantly, to be able to model turbulent flows with improved predictive capabilities. As computing technology continues to improve, it is becoming more feasible to solve the governing equations of motion—the Navier–Stokes equations—from first principles. The direct solution of the equations of motion for a fluid, however, remains a formidable task and simulations are only possible for flows with small to modest Reynolds numbers. Within the UK, the Turbulence Consortium (UKTC) has been at the forefront of simulating turbulent flows by DNS. UKTC has developed a parallel version of a code to solve problems associated with shock/boundary-layer interaction. The code (PDNS3D) was originally developed for the Cray T3E and is a sophisticated DNS code that incorporates a number of advanced features: namely high-order central differencing; a shock-preserving advection scheme from the total variation diminishing (TVD) family; entropy splitting of the Euler terms and the stable boundary scheme [37]. The code has been written using standard FORTRAN 90 code together with MPI in order to be efficient, scalable and portable across a wide range of high-performance platforms.

The PCHAN benchmark is a simple turbulent channel flow benchmark using the PDNS3D code. Performance with both the T1 ($120 \times 120 \times 120$) and T2 ($240 \times 240 \times 240$) Grid benchmark data cases shows close to ideal scaling on all the present cluster systems (see Table VII). Note that the timings reported under the tag ‘‘fully populated nodes’’ refer to CPU configurations in which all cores on a given node are involved in the computation. These timings show several distinct features. All machines, with the exception of the CS42, the dual-core Opteron 2218-F cluster,

TABLE VII

TIME IN WALL CLOCK SECONDS FOR FOUR PCHAN BENCHMARK CALCULATIONS ON A VARIETY OF DUAL- AND QUAD-CORE CLUSTERS (SEE TABLE IV)

	CS36	CS42	CS49	CS51	CS54	
					SGI Altix Ice 8200, Xeon E5440 2.83 GHz QC (InfiniBand)	
No. of processing elements (cores)	Intel Xeon 5160 3.0 GHz DC (InfiniPath)	IBM x3455 Opteron 2218-F, 2.6 GHz DC (InfiniBand)	AMD Barcelona 2350 QC (Connect-X)	Bull R422 Xeon E5472 3.0 GHz QC (Connect-X)	MPT	Mpavich
*T1 GRID (120**3) fully populated nodes*						
16	297	249	301	343	456	457
32	139	128	149	171	216	216
64	60	62	68	69	88	89
128	29	34		25	32	37
*T1 GRID (120**3) half-populated nodes*						
16	146			168	213	238
32	65			75	89	115
64	27			26	29	49
128	17			10	11	20
*T2 GRID (240**3) fully populated nodes*						
16	2388	1980		2756	3673	3673
32	1202	942		1413	1897	1893
64	584	476		709	926	925
128	298	258		356	459	463
256	139	134		167	217	224
*T2 GRID (240**3) half-populated nodes*						
16	1825	1463		1430	1853	
32	662	694		758	937	1004
64	298	350		333	428	482
128	145	195		168	213	241
256					89	119

appear to exhibit superlinear speedups. Thus for the T1 benchmark, effective speedups of 10.2, 13.7, and 12.3 are found for the CS36, CS51, and CS54 Xeon clusters, respectively, on moving from 16 to 128 cores.

While this behavior is at first sight confusing, it may be rationalized from a consideration of the driving force behind this benchmark, namely memory bandwidth. First, we note that additional insight can be gained by varying the distribution of processors over the available nodes, initially by only populating half the cores on each node. As can be seen from Table VII, we find that the performance of each system is effectively doubled by depopulating the nodes.

Thus, for example, the time to solution for the T2 Grid on 128 cores of the CS51 Bull Xeon cluster is reduced from 356 to 168 s when using half-populated nodes. We would also note that in the smaller T1 benchmark, the optimum time to solution on 128 cores is given by the quad-core CS51 system, while on the larger T2 benchmark the dual-core CS36 Xeon 5160 cluster outperforms the quad-core cluster. This may be attributed to the impact of the increased L2 cache on the E5472 processor relative to that on the Xeon 5160 when running the smaller case at a sufficiently high processor count.

The most important communications structure within PCHAN is a halo-exchange between adjacent computational subdomains. Providing the problem size is large enough to give a small surface area to volume ratio for each subdomain, the communications costs are small relative to computation and do not constitute a bottleneck.

6.8 Application Performance

The performance and scaling behavior of each application on the quad-core commodity-based systems is now compared with results from the dual-core ''Darwin'' cluster that features Intel Xeon 3.0 GHz Woodcrest nodes. We limit the discussion here to a consideration of the percentage of a 32-core partition of the Woodcrest cluster delivered by the quad-core commodity-based systems (i.e., $T_{32\text{-core}}$ dual-core Xeon 5160 cluster/$T_{32\text{-core}}$ quad-core cluster, CSx). Figure 7 shows the percentage of a 32-core partition of the dual-core cluster delivered by 32 cores of both the InfiniBand-connected CS51 Bull R422 E5472/3.0 GHz and the Connect-X CS49 AMD Opteron 2350 ''Barcelona'' 2.0 GHz quad-core systems.

At the simplest level, assuming that:

1. The cost per socket in dual- and quad-core systems remains constant.
2. Our end system comprises the same number of sockets.

Then given twice the number of cores in our quad-core system, this system will be more cost performant than the corresponding dual-core system if the individual performance ratios of Fig. 7 exceed 50%. As we can see from both Table VIII and Fig. 7, these ratios are far higher than this figure—averaging across all the applications suggests that 32 quad-cores of the Bull Harpertown cluster delivers 97% of the corresponding performance shown by the dual-core Woodcrest cluster, while 32 cores of the AMD quad-core cluster achieves some 68% of the dual-core system. These figures clearly show that (1) the overall computational capacity delivered by both quad-core systems is far higher than that delivered by the dual-core system and (2) Intel's 3.0 GHz Harpertown processor has a significant performance advantage

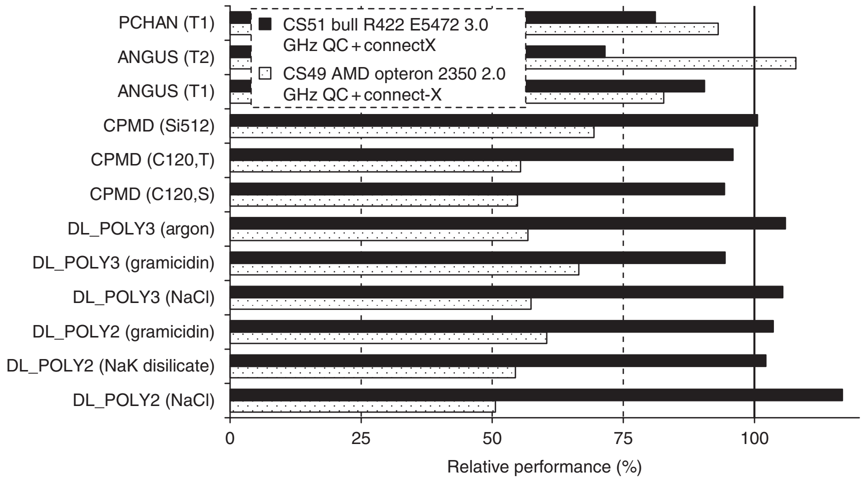


FIG. 7. Application performance: percentage of 32 cores of the dual-processor Darwin Xeon 5160 dual-core CS36 cluster achieved by 32 cores of the CS51 Bull R422 E5472/3.0 GHz (solid) and AMD Opteron 2350 Barcelona (dotted) quad-core systems.

over the Opteron 2350/2.0 GHz processor. Thus the CS51 Bull Harpertown cluster outperforms the CS49 Barcelona cluster in 10 out of the 12 tests—the only exception is the larger of the two ANGUS benchmarks and the PDNS3D benchmark. Considering each code, we find the largest performance differential between Harpertown and Barcelona is with the DLPOLY simulation code—on average 194% and 166% for DL_POLY2 and DL_POLY3, respectively. The advantage is less marked in CPMD (127%) and ANGUS (113%). Only in the memory bandwidth-intensive PDNS3D code (88%) does the Barcelona system outperform the Xeon cluster.[2]

Our analysis suggests that quad-core processors can deliver an improvement in performance of up to 4× per processor but is heavily dependent on the workload being processed. While the Intel processors have a higher clock rate and peak performance, the AMD processors have higher memory bandwidth and intranode scalability. The scientific applications we analyzed exhibit a range of performance improvements from only 3× up to the full 8× speedup over a single core. Also, we note that the maximum node performance is not necessarily achieved by using all 8 cores.

[2] For each code, we have averaged over the individual performance ratios for each of the associated data sets.

TABLE VIII
APPLICATION PERFORMANCE: PERCENTAGE OF 32 CORES OF THE DUAL-PROCESSOR DARWIN XEON 5160 DUAL-CORE CLUSTER ACHIEVED BY 32 CORES OF THE BULL R422 E5472/3.0 GHZ AND AMD OPTERON 2350 BARCELONA QUAD-CORE CLUSTERS

		$T_{32\text{-core}}$ DC Cluster/$T_{32\text{-core}}$ CSx QC Cluster	
Application Code	Data Set	CS51 Bull R422 Xeon E5472 3.0 GHz Quad-core + Connect-X Cluster (%)	CS49 AMD Opteron 2350 2.0 GHz Quad-core + Connect-X Cluster (%)
CPMD	C_{120} (S)	94	55
	C_{120} (T)	96	55
	Si_{512}	101	69
DL_POLY2	NaCl	117	51
	NaK disilicate	102	54
	Gramicidin	105	60
DL_POLY3	NaCl	105	57
	Gramicidin	94	67
	Argon LJ	106	57
ANGUS	T1 (144**3)	91	83
	T2 (288**3)	72	108
PDNS3D	T1 (120**3)	81	93
	Average performance ratio	97	68

The present results suggest that the balance in the battle for processor supremacy between AMD and Intel lies in favor of Intel at this point, although we would suggest that this battle is far from over. The availability of Barcelona processors with clock speeds higher than 2.0 GHz—that on the system tested—would markedly reduce the performance advantage of the Harpertown processor shown in the figure.

7. Summary and Conclusions

In overviewing the current HPC landscape, this chapter has considered the multitude of issues faced by an organization when deciding how best to procure, maintain, and maximize the usage of any associated HPC resource. We have concentrated on the potential role of HPC Integrators in any partnership that looks to maximize this entire process, and whether such organizations in the UK have the ability to provide the necessary level of expertise required in all phases of the process, from procurement, through installation onto ongoing support of the resource throughout its lifecycle. Our primary conclusions are as follows:

1. Crucial issues when considering potential integrator involvement include both size of the proposed hardware solution, that is, number of nodes, and the ongoing robustness of open-source software solutions that might be deployed on these platforms. Specifically:
    a. The *size of the system* in question—is this targeting less than 1000 processing elements or cores, a domain in which most of the integrators have experience, or does the system in question exceed, say, 15 Tflop. If the latter, it is worth mentioning that the current national HECToR procurement *rejected* the use of integrators at an early stage having considered their capabilities through a series of presentations at SC'2003 in Dallas. While US Integrators certainly have extensive experience in the 1000+ CPU domain, this is not in general the case for their UK counterparts, although the current procurements associated with SRIF3 are rapidly changing this landscape. Tenders of 1000+ CPU systems are increasing, which has repercussions on companies experience in this marketplace. Thus while the UK's national facility. HECToR has 11,328 cores, 5 of the 10 Leading HPC sites in the UK house cluster systems with more than 2000 cores.
    b. The increased reliance on parallelization and hence system size to accomplish the highest levels of performance will merely act to emphasize the operational challenges associated with extremely large systems, challenges that stretch the resources of proprietary vendors to the limit and are realistically beyond the reach of many of the HPC Integrators central to this chapter.
    c. The expected *usage pattern and environment* around the resource—is this being driven by *Capability* or *Capacity* requirements? We would again suggest that integrators remain capable of providing the latter requirement far more effectively than the former.
    d. The level of *RAS features* expected of the HPC solution. Demanding levels of RAS (say 95+%) around truly large systems are exceptionally difficult to sustain, particularly in a *Capability* regime when running large jobs with long execution times. Assuming such features appear in any contract around the services to be provided, it is

extremely unlikely that any integrator would be in the position to accept the risk involved in committing to high levels, and would need the partnership framework in partnership with an experienced Tier-1 organization.

2. Our considered view is that while existing UK HPC Integrators certainly do have a valuable role to play in the ongoing provision of capacity-based resources, that is, less than 1000 processing elements, the majority *are less able to provide added value to high-end capability machines* where the focus lies on stringent RAS requirements. A notable exception based on the past 12 months is ClusterVision who with IBM and Dell supplied three of the aforementioned UK cluster systems.
3. The HPC integration marketplace is growing rapidly. The install base of commodity-based clusters is accelerating at pace in the UK, funded through initiatives such as SRIF, and a large portion of that business is going to the integrators identified in this chapter, and not to Tier-1 vendors such as IBM and HP bidding in their own right. The reasons for this are easy to understand:
   a. The focus of Tier-1 activity remains on the larger, proprietary-based machines where the margins are greatest. Companies such as IBM remain primarily focused on their proprietary CPU offerings—for example, the power series—with much of their pre- and postsales support targeting such solutions. Interestingly, however, IBM has become far more engaged in the SRIF arena over the past 12 months, typically in partnership with either OCF or ClusterVision.
   b. The margins remain less attractive for Tier-1 organizations when dealing with commodity solutions. We have certainly witnessed Tier-1 vendors discouraging commodity solutions in favor of their own proprietary-based solutions.
4. Most integrators see the HPC market continuing to grow, as the technology continues to mature, and new innovations drive performance ever higher. One possible caveat here however is the issue of full-economic costing. This is now playing an increasing part in the decision-making process as the factors that affect fEC are becoming more visible, and sites are now reaching their capacity to provide the space, mains power and air conditioning required to run a supercomputer cluster. The days of major injections of capital funding though Universities and SRIF may be drawing to a close, and with it a much needed funding stream for many of the UK integrators.
5. There is some confusion over the role that Tier-1 vendors actually play in the UK market, a point made by some of the integrators. Eighty to ninety percent

of all HPC cluster developments in the UK have been carried out by systems integrators, even those ‘‘sold’’ by a Tier-1 vendor. For example, IBM and HP cluster solutions have been integrated and installed by OCF for a number of years. Dell worked historically with Scali to position their PC offering into a ‘‘Dell Cluster,’’ and more recently with ClusterVision. Streamline recently built most of the large HPC clusters sold in the UK by SUN, while Compusys historically were the integration and support specialists for the Cray XD1 supercomputer.

6. This issue of technical competence is seen by all the integrators as the key differentiator, and key to the future of their organizations. As cluster sizes grow but commodity hardware becomes cheaper and cheaper, the ‘‘value’’ in the market will be the ability to make sure that clusters work efficiently for a broad range of applications and function in a more scalable manner and operate seamlessly across subsystems. System management and monitoring capabilities will become more important, and will help the ability of the integrator and the end-user to support such a system through its lifetime. The ability to provide a high level of support, not only on a system level, but also on an application level, will help differentiate the ‘‘box-shifters’’ from the serious HPC-oriented companies. Notable developments over the past 2 years have been the increasing maturity and features associated with ClusterVisionOS and the Streamline Cluster Management Appliance.
7. In the commodity-based solutions market, integrators provide cost-effective, technology compelling solutions rivaling and often exceeding those of alternative Tier-1 organizations.
8. One potential engagement model would be to form an *Integrator Technology Partnership* with those integrators who are deemed appropriate to the task in hand. These organizations typically do not have legacy turf wars that have made previous attempts to structure multi-Tier-1 vendor consortiums around high-end HPC solutions extremely difficult (e.g., in the UK’s national HPC procurements—HPC’97 and HPCx), and are far more able to accept such a solution. This would have the obvious advantage of pooling highly competent, but thinly spread, technical expertise.
9. All integrators have existing relationships with Tier-1 vendors, although the nature of these interactions varies considerably. It is clear that deployment of very large systems requires the financial strength of a Tier-1 vendor to execute larger contracts and arguably needs the specialist expertise of key integrators to provide the knowledge and skills to build and support the systems. Productive relationships with Tier-1 vendors are seen as critical success factors for future growth of many of the integrators, and

does provide an *engagement model for other organizations*, assuming that the Tier-1 vendor of choice does not inflict an ineffective integrator (or vice versa).

10. We do not feel that traditional SI houses have a role to play in this arena—they are invisible within the academic space, and would realistically have a steep learning curve to climb to be in a position to deal with the technology issues central to HPC provision. We would suggest that their value is clearly ‘‘only perceived at a corporate rather than operational level.’’

Appendix: Integrator Questionnaire

Initial Integrator Discussion Points Around HPC Provision

To best inform the data gathering exercise associated with this chapter, a set of preliminary questions were devised and discussed with each of the integrators. These questions are sketched out below, with the responses of Section 4 driven off the following eight points:

1. Understanding the current install base (both in the UK and abroad) and trends in the cluster marketplace. Information on current install base (ideally over the last 4–5 years, providing a picture of changing trends), including where possible the site, architecture, size, procurement date, etc. (naturally no financial details are expected). An idea of the relative number of ‘‘small’’ (32–64) systems compared to larger (128+) machines. What is the approximate split between HEI’s and industrial installations?
2. Details and company size/status, etc.: Company overview—background, status, size, etc. In providing this information it would be useful to have a breakdown of the relevant parts of the organization—sales staff, after-sales support team, technical support (software and hardware if possible), etc., approximate turnover (machines not finances), customers (numbers—names are not necessary although they might prove useful). The details of, say, three customer reference sites would be helpful.
3. *Company outreach and presence*. An extension to the first two points—what presence do you have overseas—in particular in Europe and the USA—and what, if any, is the size of the current install base outside the UK.

4. *Company areas of expertise*. From the technical perspective, what level of technical expertise do you feel you bring to the system integration market that makes you competitive and a long-term prospect in that marketplace. Please mention any other tie-ins you provide which you feel are relevant.
5. *Company perspective of the marketplace*. How do you feel the cluster marketplace is changing (especially with fEC coming into effect). Any information you can provide as to changes/investments you are making to adapt to this changing climate, for example, requirement to increase skills in middleware, compiler, database, files systems, etc.; need to forge strong links with software companies/interconnect solution; impact of GRID/e-science developments?
6. *Relationship to Tier-1 organizations*. Do you feel your company has the strength and depth to build high-performance, scalable systems which can support tera- and petascale solutions in the not-too-distant future. Are you able to provide this independently, or do you require Tier-1 support in dealing with areas such as risk and liability. Do you feel that the UK integrator providers are competitive with those in the USA?
7. Relationship to other system integrators—Do you perceive any role for the more traditional SIs, for example, EDS, CSC in this marketplace. These are pretty invisible to us, but you may have a different perspective over what appears to be a more expensive alternative—at least in the nonacademic space?
8. The above pointers are clearly not exhaustive, so if you feel that any other information is important in our trying to understand your position in the marketplace, please feel free to mention it. Can you provide a viable cost-effective alternative to the established blue chip/Tier-1 companies when it comes to procuring midrange/high-end systems?

References

[1] Zelkowitz M., 2008. Advances in Computers: High Performance Computing, vol. 72, ISBN 978-0-12-374411-1-1. Academic Press, London.

[2] Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R., and Sunderam V., 2007. PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing. http://www.epm.ornl.gov/pvm/.

[3] Petitet A., Whaley R. C., Dongarra J., and Cleary A., September 10, 2008. HPL—A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. Version 2.0, http://www.top500.org/.

[4] Brown R. G., August 15, 2005. The Future of HPC Clusters, Linux Magazine, http://www.linux-mag.com.

[5] The Fusion and Larrabee processors. see http://www.reghardware.co.uk/2006/10/26/the_story_of_amds_fusion/and http://www.theinquirer.net/gb/inquirer/news/2008/09/12/larrabee-might-great-analyst.

[6] VanCourt T., and Herbordt M. C., 2009. Elements of high performance reconfigurable computing. *In Advances in Computers*, (M. Zelkowitz, ed.), Volume. 75. Academic Press, London.

[7] Boden N., Cohen D., Felderman R. E., Kulawik A. E., Seitz C. L., Seizovic J. N., and Su W., 1995. Myrinet: A gigabit-per-second local-area network. *IEEE Micro*, **15:** 29–36.

[8] http://www.dolphinics.com.

[9] Petrini F., Feng W.-C., Hoisie A., Coll S., and Frachtenberg E., 2002. The quadrics network: High-performance clustering technology. *IEEE Micro*, **22:** 46–57.

[10] Advanced Simulation and Computing Program, http://www.sandia.gov/NNSA/ASC/pubs/pubs.html.

[11] HPCWIRE Mc. Cann T., June 29, 2007. Getting Data Centers to Chill. http://www.hpcwire.com/features/17900829.html.

[12] A Strategic Framework for High End Computing, produced by the UK's High End Computing Strategic Framework Working Group at the request of the High End Computing Strategy Committee. http://www.epsrc.ac.uk/CMSWeb/Downloads/Other/2006HECStrategicFramework.pdf.

[13] International Review of Research Using HPC in the UK, report commissioned by EPSRC. http://www.epsrc.ac.uk/ResearchFunding/FacilitiesAndServices/HighPerformanceComputing/InternationalReview/IntRevReport.htm.

[14] The President's Information Technology Advisory Committee, 'Computational Science: Ensuring America's Competitiveness', http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf.

[15] Report of the National Science Foundation Blue-Ribbon. available at http://www.nsf.gov/od/oci/reports/toc.jsp.

[16] http://www.nsf.gov/pubs/2006/nsf06573/nsf06573.html#pgm_.

[17] http://www.scidac.org/Conference2006/SD06program.html.

[18] Boghosian B. M., Coveney P. V., Dong S., Finn L. I., Jha S., Karniadakis G., and Karonis N., 2006. Nektar, SPICE. and Vortonics: Using federated grids for large scale scientific applications. *Challenges of Large Applications in Distributed Environments* pp. 34–42. IEEE Catalogue Number: 06EX1397, ISBN. 1-4244-0420-7, Library of Congress 2006925560.

[19] Geographically distributed domain decomposition is discussed in Boghosian B. M., Finn L. I., and Coveney P. V., January 2006. Moving the data to the computation: Multi-site distributed parallel computation. available at http://www.realitygrid.org/publications.shtml.

[20] Coupling real-time visualisation and steering with high performance computing Chin J., and Coveney P. V., 2006. Chirality and domain growth in the gyroid mesophase. *Proceedings of the Royal Society of London Series A*, **462:** 3575–3600. DOI: 10.1098/rspa.2006.1741.

[21] Collaborative High. End Computing 2006. *Future Generation Computing Systems*, **22:** 1006, Section 3 on 'Distributed Visualization'.

[22] SRIF Scientific Research Investment Fund, http://www.hefec.ac.uk/research/srif.

[23] The background to the implementation of fEC as the means by which HEIs may apply to Research Councils for project funding may be found at http://www.pparc.ac.uk/jes/DualSupport.asp, with a non-technical introduction to TRAC and a summary of the its main features, available at http://www.jcpsg.ac.uk/downloads/guidance/Overview.pdf.

[24] OSCAR http://svn.oscar.openclustergroup.org/trac/oscar and http://oscar.openclustergroup.org/tiki-index.php.

[25] ROCKS http://www.rocksclusters.org/wordpress/.

[26] HPCWIRE, Feldman M., February 22, 2008. Requiem for a Cluster Vendor. http://www.hpcwire.com/blogs/17909749.html.

[27] The Standard Performance Evaluation Corporation (SPEC). http://www.spec.org/.

[28] SPEC. CPU2006, see http://www.spec.org/cpu2006/press/V1.1release.html; A PDF summary of all 29 benchmarks, as published in September 2006. *ACM SIGARCH Computer Architecture News*, **34**(4); A series of eleven articles analyzing SPEC CPU2006 was published in March 2007. *ACM. SIGARCH Computer Architecture News*, **35**(1).

[29] For example Ashworth M., Bush I. J., and Guest M. F., 2005. Vector vs. scalar processors: A performance comparison using a set of computational science benchmarks. *In Proceedings of the Cray User Group, CUG2005*.

[30] STREAM, Sustainable Memory Bandwidth in High. Performance Computers, http://www.cs.virginia.edu/stream/.

[31] GAMESS-UK is a package of ab initio programs written by Guest M. F., van Lenthe J. H., Kendrick J., Schoeffel K., and Sherwood P., with contributions from Amos R. D., Buenker R. J., Dupuis M., Handy N. C., Hillier I. H., Knowles P. J., Bonacic-Koutecky V., von Niessen W., Harrison R. J., Rendell A. P., Saunders V. R., and Stone A. J. The package is derived from the original GAMESS code due to Dupuis M., Spangler M. D., and Wendoloski J., 1980. NRCC Software Catalog, vol. 1, Program No. QG01 (GAMESS).

[32] Smith W., and Forester T. R., 1996. The DL_POLY molecular simulation package. *Journal of Molecular Graphics*, **14:** 136, http://www.cse.scitech.ac.uk/ccg/software/DL_POLY/index.shtml.

[33] Guest M. F., and Sherwood P., 2002. Computational chemistry applications: performance on high-end and commodity-class computers. In *Proceedings of HPCS'2002*, Moncton, Canada, June 17–19, 2002.

[34] The IMB Parallel Communications Benchmarks, see http://softwarecommunity.intel.com/isn/downloads/softwareproducts/pdfs/219318_relnotes.pdf.

[35] MPT—SGI's Message Passing Toolkit (http://www.sgi.com/products/software/mpt/). Note that appropriate settings of the environment variables—MPI_IB_RAILS, MPI_BUFFER_MAX and MPI_BUFS_PER_HOST—are also required to realize optimal performance.

[36] Hutter J., and Curioni A., 2003. CPMD—Car–Parrinello molecular dynamics, see http://www.cpmd.org; Dual-level parallelism for *ab initio* molecular dynamics: Reaching teraflop performance with the CPMD code. IBM. Research Report, RZ. 3503.

[37] Sandham N. D., Ashworth M., and Emerson D. R., PDNS3D, Direct Numerical Simulation of Shock/Boundary Layer Interaction, http://www.cse.clrc.ac.uk/ceg/sbli.shtml.

[38] Emerson D. R., Nicole D. A., and Takeda K., 1998. ANGUS, direct numerical simulation (DNS) of turbulent flow. An. Evaluation of Cost Effective Parallel Computers for CFD, presented at Parallel CFD'98, Taiwan.

[39] Smith W., Yong C., and Rodger M., 2002. DL_POLY: Applications to molecular simulation. *Molecular Simulation*, **28:** 385.

[40] Guest M. F., Application Performance on High-end and Commodity-class Computers. http://www.ukhec.ac.uk/publications/reports/benchmarking.pdf.

[41] Bush I. J., and Smith W., A Parallel Implementation of SPME. for DL_POLY 3. http://www.cse.clrc.ac.uk/arc/fft.shtml.

[42] Essmann U., Perera L., Berkowitz M. L., Darden T., Lee H., and Pedersen P. G., 1995. A smooth particle mesh Ewald method. *Journal of Chemical Physics*, **103,** 8577.

[43] Allen M. P., and Tildesley D. J., 1987. Computer Simulation of Liquids. Clarendon Press, Oxford.

[44] Frigo M., and Johnson S. G., 1997. The fastest Fourier transform in the West. MIT Technical Report, MIT-LCS-TR-728, http://www.fftw.org/.

Elements of High-Performance Reconfigurable Computing*

TOM VANCOURT†

Department of Electrical and Computer Engineering, Boston University, Boston, Massachusetts 02215

MARTIN C. HERBORDT

Department of Electrical and Computer Engineering, Boston University, Boston, Massachusetts 02215

Abstract

Numerous application areas, including bioinformatics and computational biology (BCB), demand increasing amounts of processing capability. In many cases, the computation cores and data types are suited to field-programmable gate arrays (FPGAs). The challenge is identifying those design techniques that can extract high performance from the FPGA fabric while allowing efficient development of production-worthy program codes. After brief introductions to high-performance reconfigurable computing (HPRC) systems and some applications for which they are well suited, we present a dozen such techniques together with examples of their use in our own research in BCB. Each technique, if ignored in a naive implementation, would have cost at least a factor 2 in performance, with most saving a factor of 10 or more. We follow this by comparing HPRC with an alternative accelerator technology, the use of graphics processors for general-purpose computing (GPGPU). We conclude with a discussion of some implications for future HPRC development tools.

* This chapter is based on two articles previously published by the IEEE: ‘‘Achieving High Performance with FPGA-Based Computing’’ which appeared in IEEE Computer in March 2007, and ‘‘Computing Models for FPGA-Based Accelerators’’ which appeared in IEEE Computing in Science and Engineering in November/December 2008.

† *Currently with Altera, Inc.*

ADVANCES IN COMPUTERS, VOL. 75
ISSN: 0065-2458/DOI: 10.1016/S0065-2458(08)00802-4

1. Introduction

For many years computational scientists could depend on continual access to ever faster computers. In the last few years, however, power concerns have caused microprocessor operating frequencies to stagnate. Moreover, while advances in process technology continue to provide ever more features per chip, these are no longer used primarily to augment individual microprocessors; rather they are commonly used to replicate the CPUs. Production chips with hundreds of CPU cores are projected to be delivered in the next several years. At the same time, however, it has become clear that replicating cores is only one of several viable strategies for developing next-generation high-performance computing (HPC) architectures.

Some promising alternatives are based on field-programmable gate arrays (FPGAs) [25]. FPGAs are commodity integrated circuits (ICs) whose logic can be determined, or *programmed*, in the field. This is in contrast to other classes of ICs (e.g., application-specific integrated circuits—ASICs) whose logic is fixed at fabrication time. The tradeoff is that FPGAs are less dense and fast than ASICs; often, however, the flexibility more than makes up for these drawbacks. Applications accelerated with FPGAs have often delivered 100-fold speedups per node over microprocessor-based systems. This, combined with the current ferment in computer architecture activity, has resulted in such systems moving toward the mainstream, with integration support being provided by the largest vendors [6, 52].

The enormous potential performance derived from accelerating HPC applications with FPGAs (high-performance *reconfigurable* computing—HPRC) comes from two sources (1) parallelism—1000× is possible, especially for low-precision computations and (2) payload per computation—since most control is configured into the logic itself, overhead instructions (such as array indexing and loop computations) need not be emulated. On the other hand, there are significant, inherent, challenges. One is the low operating frequency: an FPGA clocks at one tenth the speed of a high-end microprocessor. Another is simply Amdahl's law: to achieve the speedup factors required for user acceptance of a new technology (preferably 50× [11]) close to 99% of the target application must lend itself to substantial acceleration. As a result, performance of HPC applications accelerated with FPGA coprocessors is unusually sensitive to the quality of the implementation. Put another way, the potential performance of HPRC is tremendous, but what users often find is that it is much easier to get no speedup at all.

The problem described here is a classic one: how to achieve significant speedups on a new architecture without expending exorbitant development effort, and while retaining flexibility, portability, and maintainability. This problem is familiar in porting uniprocessor applications to massively parallel processors (MPPs).

Two differences are as follows (1) FPGAs are far more different from uniprocessors than MPPs are from uniprocessors and (2) the process of parallelizing code for MPPs, while challenging, is still better understood and supported than porting codes to FPGAs. The basic parameters for the ''portability problem'' (whether among MPPs or other non-PC architectures) were stated by Snyder [64]. First, that a parallel solution utilizing P processors can improve the best sequential solution by at most a factor of P. Second, that HPC problems tend to have third- to fourth-order complexity and so parallel computation, while essential, offers only modest benefit. And finally, that therefore ''the whole force of parallelism must be transferred to the problem, not converted to 'heat' of implementational overhead.''

The portability problem has been addressed variously over the last 40 years, with well-known approaches involving language design, optimizing compilers, other software engineering tools and methods, and function and application libraries (see, e.g., last year's *Advances in Computers* for some approaches used in DARPA's HPCS program [21, 24]). It is generally agreed that compromises are required: one can restrict the variety of architectures, or the scope of application; or one can bound expectations in performance, or in ease of implementation. The point in the spectrum we have chosen is as follows. For architecture, we assume a standard PC with FPGA coprocessor on a high-speed bus. For performance, we aim to achieve at least 10× (with 50× the target) to motivate using a nonstandard architecture. For applications, we concentrate on those that are widely used, have high potential parallelism, and, preferably, low precision. And finally, for effort, we consider from a few programmer/designer months to 1 year or 2 (depending on application complexity and potential impact) as being realistic. Our methods follow standard FPGA design procedures, based primarily on the VHDL hardware description language (HDL) augmented with our own LAMP tool suite [68, 69].

We continue this chapter with brief introductions to HPRC systems and some applications for which they are well suited. The central part of this chapter describes 12 methods we have used to avoid ''generating implementational heat'' in our acceleration of several HPRC applications. Each, if ignored, would have cost at least a factor of 2 in performance, with most saving a factor of 10 or more. As many of these methods are well known to those experienced in the HPRC, this survey is particularly targeted to the many newcomers to the field who may wish to augment performance obtained through direct C-to-gates implementations. We follow this by comparing HPRC with an alternative accelerator technology, the use of graphics processors (GPUs) for general-purpose computing (GPGPU). We conclude with a discussion of the implications of these methods to the design of the next generation of design tools. Integration into future HPRC design processes appears to be readily achievable, and will not require HPRC designers to become proficient in an HDL.

2. FPGA Accelerator Architecture and Computing Models

FPGAs have been available since the 1980s, but have only recently started to become popular as computation accelerators. To understand why, it is necessary to understand something about their basic technology. That technology explains the FPGAs' strengths and limitations as application accelerators, shows why FPGA programming is so different from traditional kinds of application programming, and determines the applications likely to benefit from FPGA acceleration.

2.1 Low-Level FPGA Models

FPGAs are reprogrammable chips containing large numbers of configurable logic gates, registers, and interconnections. FPGA programming means defining the bit-level configuration of these resources to create a circuit that implements a desired function. The FPGAs of interest store their configuration data in static RAM cells distributed across the chip, so they can be reused indefinitely for different computations defined by different logic configurations.

While it is possible to program an FPGA by specifying settings of individual components, this is rarely done for HPRC applications; if anything, HPRC developers are more willing to trade off performance for programmability by using the highest possible level of abstraction. Still, the historic computing model is useful: FPGAs are a ''bag of gates'' that can be configured into logic designs using HDLs such as Verilog and VHDL.

In the last few years, high-end FPGAs have come to be dominated by embedded components such as multipliers, independently addressable memories (block RAMs—BRAMs), and high-speed I/O links. Aligned with these changes, a new low-level computing model has emerged: FPGAs as a ''bag of computer parts.'' A designer using this model would likely consider the following FPGA features when designing an application:

- Reconfigurable in milliseconds
- Hundreds of hardwired memories and arithmetic units
- Millions of gate-equivalents
- Millions of communication paths, both local and global
- Hundreds of gigabit I/O ports and tens of multigigabit I/O ports
- Libraries of existing designs analogous to the various system and application libraries commonly used by programmers

As with microprocessors, making FPGAs appropriate for HPC requires added hardware support and this too is part of the low-level model. A sample system is the Wildstar board from Annapolis Microsystems, a facsimile of which is shown Fig. 1. Although now dated, we found this design to be particularly well balanced. Critical are the seven independently addressable memory banks per FPGA (SRAMs and DRAM). Since memory is managed explicitly in HPRC applications, there is no hardware caching support. Communication with the host is over an I/O bus (PCI).

Recently, the trend with HPRC systems is toward tighter integration of the FPGA board into the host system, for example, by making FPGA boards plug-compatible with Intel Front Side Bus slots (see Fig. 2) with offerings from XtremeData and DRC [17, 54, 67, 78]. The effect is to give FPGAs in a standard PC or server access capability to main memory and other system components equal to that of the microprocessors. These architectural developments have been accompanied by a similar level of activity in software integration. For example, Intel has developed QuickAssist to be ‘‘a common software framework that exposes a unified accelerator interface on top of FPGA accelerator modules’’ [52]. This integration support is perhaps the crucial factor in differentiating this from previous generations of accelerators.

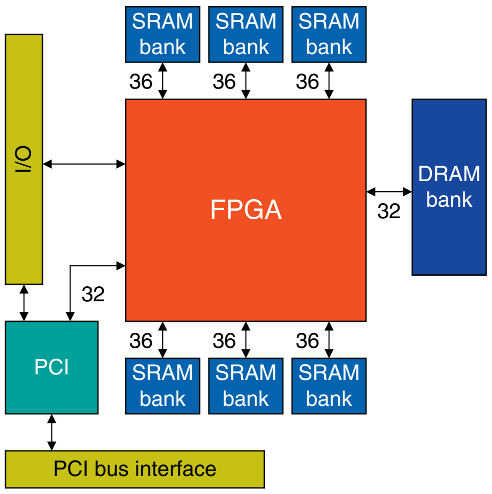


Fig. 1. Typical configuration of an FPGA-based accelerator that could be plug-compatible with a microprocessor. This configuration is based on the Annapolis Microsystems Wildstar II Pro.

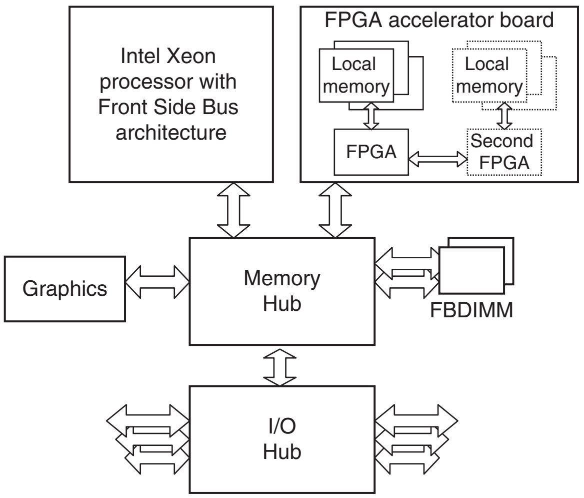


FIG. 2. Current HPRC accelerators are coequals with microprocessors in standard multisocket PCs. This configuration is based on the XtremeData XD1000.

2.2 FPGA Computation Basics

Recall that FPGA-based acceleration is successful when high parallelism and utilization can be achieved. Here we examine FPGA attributes in more detail and see how these translate into that capability. If FPGAs can be viewed in the second order as a configurable bag of computer parts, these parts must still be laid out in two dimensions and in finite space. This puts a premium on (1) connecting computational blocks with short paths, (2) taking advantage of long paths with high fan out, *viz.*, broadcast, and (3) low-precision computation.

Another issue, as with microprocessors, is support for various working set sizes and the bandwidth available to swap those working sets. There are typically several distinct levels in the HPRC memory hierarchy. Most have analogs in a conventional PC, but with somewhat different properties, especially to support fine-grained parallelism:

1. *On-chip registers and lookup tables (LUTs).* The FPGA substrate consists of registers and lookup tables through which logic is generated. These components can be configured into either computational logic or storage, with most designs having some mix. While all register contents can potentially be accessed every cycle, LUTs can only be accessed 1 or 2 bits at a time. For example, the Xilinx Virtex-5 LX330T has 26 KB of registers and 427 KB of LUT RAM; the aggregate potential bandwidth at 200 MHz is 12 TB/s.

2. *On-chip BRAMs*. High-end FPGAs have several hundred independently addressable multiported BRAMs. For example, the Xilinx Virtex-5 LX330T has 324 BRAMs with 1.5 MB total storage and each accessible with a word size of up to 72 bits; the aggregate potential bandwidth at 200 MHz is 1.2 TB/s.
3. *On-board SRAM*. High-end FPGAs have hundreds of signal pins that can be used for off-chip memory. Typical boards, however, have between two and six 32-bit independent SRAM banks, with recent boards, such as the SGI RASC having close to 100 MB. As with the on-chip BRAMs, off-chip access is completely random and per cycle. The maximum possible such bandwidth for the Xilinx Virtex-5 LX330T is 49 GB/s, but a figure between 1.6 and 5 GB/s is more common.
4. *On-board DRAM*. Many boards either also have DRAM or replace SRAM completely with DRAM. Recent boards support multiple GB of DRAM. The bandwidth numbers are similar to those with SRAM, but with higher access latency.
5. *Host memory*. Several recent boards support high-speed access to host memory through, for example, SGI's NumaLink, Intel's Front Side Bus, and Hypertransport used by AMD systems. Bandwidth of these links ranges from 5 to 20 GB/s or more.
6. *High-speed I/O links*. FPGA applications commonly involve high-speed communication. High-end Xilinx FPGAs have up to 24 3 GB/s ports.

The actual performance naturally depends on the existence of configurations that can use this bandwidth. In our own work, we frequently use the entire available BRAM bandwidth, and almost as often use most of the available off-chip bandwidth as well. In fact, we interpret this achievement for any particular application as an indication that we are on target with our mapping of application to hardware.

3. Sample Applications

Any computing platform works better for some applications than for others, partly because of the physical structure of the computing hardware and partly due to the computational characteristics of the application. This section serves two purposes. The first is to put in a single place outlines of the applications used as case studies in our description of implementation methods. The second is to show the characteristics of applications that are good candidates for FPGA acceleration. We first describe a number of applications we have accelerated, and then summarize their applicable characteristics.

3.1 Molecular Dynamics

Molecular dynamics (MD) is an iterative application of Newtonian mechanics to ensembles of atoms and molecules (for more detail, see, e.g., Rapaport [58] or Haile [31]). Time steps alternate between force computation and motion integration. The short- and long-range components of the nonbonded force computation dominate execution. As they have very different character, especially when mapped to FPGAs, we consider them separately. The short-range force part, especially, has been well studied for FPGA-based systems (see, e.g., [1, 3, 28, 41, 61, 76]).

MD forces may include van der Waals attraction and Pauli repulsion (approximated together as the Lennard–Jones, or LJ, force), Coulomb, hydrogen bond, and various covalent bond terms:

$$F^{\mathrm{total}} = F^{\mathrm{bond}} + F^{\mathrm{angle}} + F^{\mathrm{torsion}} + F^{\mathrm{H-bond}} + F^{\mathrm{nonbonded}}. \tag{1}$$

Because the hydrogen bond and covalent terms (bond, angle, and torsion) affect only neighboring atoms, computing their effect is $O(N)$ in the number of particles N being simulated. The motion integration computation is also $O(N)$. Although some of these $O(N)$ terms are easily computed on an FPGA, their low complexity makes them likely candidates for host processing, which is what we assume here. The LJ force for particle i can be expressed as:

$$F_i^{\mathrm{LJ}} = \sum_{j \neq i} \frac{\varepsilon_{ab}}{\sigma_{ab}^2} \left\{ 12 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^{14} - 6 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^{8} \right\} r_{ji}, \tag{2}$$

where ε_{ab} and σ_{ab} are parameters related to the types of particles, that is, particle i is type a and particle j is type b. The Coulombic force can be expressed as:

$$F_i^{\mathrm{C}} = q_i \sum_{j \neq i} \left(\frac{q_j}{|r_{ji}|^3} \right) r_{ji}. \tag{3}$$

In general, the forces between all particle pairs must be computed leading to an undesirable $O(N^2)$ complexity. The common solution is to split the nonbonded forces into two parts: a fast converging short-range part, which consists of the LJ force and the nearby component of the Coulombic, and the remaining long-range part of the Coulombic (which is described in Section 3.2). The complexity of the short-range force computation is then reduced to $O(N)$ by only processing forces among nearby particles.

3.2 Multigrid for Electrostatic Computation

Numerous methods reduce the complexity of the long-range force computation from $O(N^2)$ to $O(N \log N)$, often by using the fast Fourier transform (FFT). As these have so far proven difficult to map efficiently to FPGAs, however, the multigrid method may be preferable [26] (see, e.g., [9, 37, 60, 62] for its application to electrostatics).

The difficulty with the Coulombic force is that it converges too slowly to restrict computation solely to proximate particle pairs. The solution begins by splitting the force into two components, a fast converging part that can be solved locally without loss of accuracy, and the remainder. This splitting appears to create an even more difficult problem: the remainder converges more slowly than the original. The key idea is to continue this process of splitting, each time passing the remainder on to the next coarser level, where it is again split. This continues until a level is reached where the problem size (i.e., N) is small enough for the direct all-to-all solution to be efficient.

The overall multigrid algorithm is shown schematically in Fig. 3 (for details, see, e.g., [79]). Starting at the upper left, the per-particle potentials are partitioned into short- and long-range components. The short range is computed directly as shown in Section 3.1, while the long-range component is applied to the finest grid. Here the force is split again, with the high-frequency component solved directly and the low-frequency passed on to the next coarser grid. This continues until the coarsest level where the problem is solved directly. This direct solution is then successively combined with the previously computed finer solutions (corrections) until the finest grid is reached. Here the forces are applied directly to the particles.

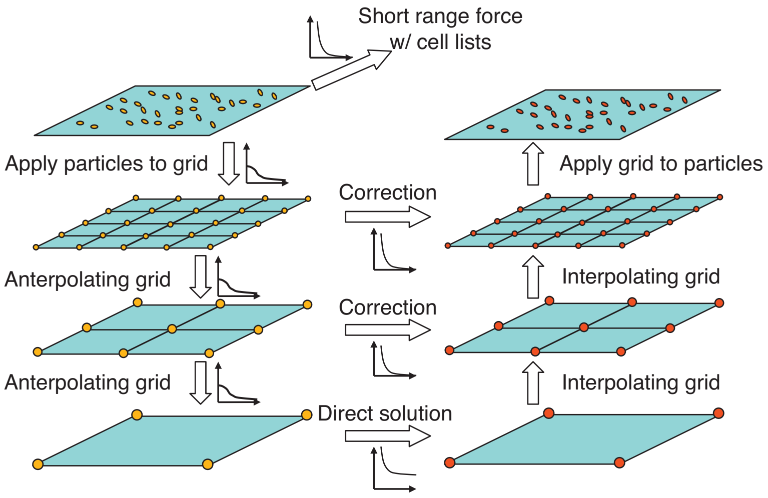


FIG. 3. Schematic of the multigrid method for the Coulomb force.

3.3 Discrete Molecular Dynamics

Increasingly popular is MD with simplified models, such as the approximation of forces with stepwise potentials (see, e.g., [58]). This approximation results in simulations that advance by discrete event rather than time step. The foundation of discrete molecular dynamics (DMD) is intuitive, hypothesis-driven, modeling based on tailoring simplified models to the physical systems of interest [19]. Using intuitive models, simulation length and time scales can exceed those of time step-driven MD by eight or more orders of magnitude [20]. Even so, not only is DMD still compute bound, causality concerns make it difficult to scale to a significant number of processors. An efficient mapping to FPGAs is described in [34, 53].

Discrete event simulation (DES) is sketched in Fig. 4: the primary components are the event queue, event processor, event predictor (which can also cancel previously predicted events), and system state. Parallelization of DES has generally taken one of two approaches (1) conservative, which guarantees causal order, or (2) optimistic, which allows some speculative violation of causality and corrects violations with rollback. Neither approach has worked well for DMD. The conservative approach, which relies on there being a ''safe'' window, falters because in DMD there is none. Processed events invalidate predicted events anywhere in the event queue with equal probability, and potentially anywhere in the simulated space. For similar reasons, the optimistic approach has frequent rollbacks, resulting in poor scaling.

3.4 Modeling Molecular Interactions (Docking)

Noncovalent bonding between molecules, or molecular docking, is basic to the processes of life and to the effectiveness of pharmaceuticals (see, e.g., [42] for a survey and [65, 71, 74] for FPGA implementations). While detailed chemical

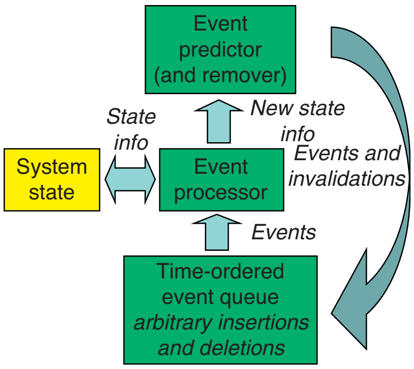


FIG. 4. Block diagram of a typical discrete event simulation.

models are sometimes used, such techniques are computationally exorbitant and infeasible for answering the fundamental question: at what approximate offsets and orientations could the molecules possibly interact at all? Less costly techniques are used for initial estimates of the docked pose, the relative offset and rotation that give the strongest interaction. Many applications assume rigid structure as a simplifying approximation: 3D voxel grids represent the interacting molecules and 3D correlation is used for determining the best fit [39]. This is the specific application we address here.

3.5 Sequence Alignment: Dynamic Programming-Based Methods

A fundamental abstraction in bioinformatics represents macromolecules such as proteins and DNA with sequences of characters. Bioinformatics applications use sequence alignment (approximate string matching) to find similarities among molecules that have diverged through mutation and evolution (for more detail, see, e.g., Durbin *et al.* [22] or Gusfield [30]). For two sequences of length m and n, dynamic programming (DP)-based methods (such as Needleman–Wunsch and Smith–Waterman) build an $m \times n$-table of character–character match scores. The table is then traversed using a recurrence relation where the score of each cell $S_{i,j}$ depends only on the scores of cells $S_{i,j-1}$, $S_{i-1,j}$, and $S_{i-1,j-1}$. The serial complexity is $O(mn)$.

DP-based can be implemented in hardware with a one-dimensional systolic array of processing elements (PEs). In a simple case, the length m-sequence is held in the array, one character per PE, while the length n-sequence streams through. The hardware complexity is $O(n)$; there have been many such implementations [7, 8, 14, 23, 29, 35, 49, 50, 59, 73, 80].

3.6 Sequence Alignment: BLAST

Although $O(mn)$ for sequence alignment is a remarkable improvement over the naive algorithms, which have unbounded complexity, it is still far too great for large databases. A heuristic algorithm, BLAST, generally runs in $O(n)$ time, and is often sufficiently sensitive (see, e.g., [2, 44] for details). BLAST is based on an observation about the typical distribution of high-scoring character matches in the DP alignment table: There are relatively few overall, and only a small fraction are promising. This promising fraction is often recognizable as proximate line segments parallel to the main diagonal.

We now sketch the classic BLAST algorithm [2]. There are three phases: identifying contiguous high scores (parallel to the main diagonal), extending them along the diagonal, and attempting to merge nearby extensions, which may or may not be on the same diagonal. The third phase, which accounts for gaps, is nowadays often replaced by a pass of Smith–Waterman on the regions of the database identified as of possible interest. The $O(mn)$ complexity of Smith–Waterman is not as significant when only run on small parts of the database; for example, Krishnamurthy *et al.* [45] find that, for a set of BLASTn experiments, the final pass accounts for well under 1% of the run time. This (effectively) makes the final pass $O(m^2)$, where $m \ll n$. There have been several FPGA implementations of BLAST (see, e.g., [13, 32, 38, 55]).

3.7 Sequence Analysis Case Study: Finding Repetitive Structures

Another important bioinformatics task is analyzing DNA or protein sequences for patterns that might be indicative of disease or be otherwise fundamental to cell processes. These patterns are typically repetitive structures, such as tandem arrays and palindromes, under various mismatch models [5, 30, 46]. The asymptotically optimal algorithms are often based on suffix trees; practical algorithms often include heuristics. Some of the hardware structures useful for HPRC in finding repetitive structures are either obvious or go back decades; Conti *et al.* [15] describe some of these and several extensions.

3.8 Microarray Data Analysis Case Study: Finding Best Combinations

Microarrays, sometimes called ‘‘gene chips,’’ measure the expression products of thousands of genes in a tissue sample and so are being used to investigate a number of critical biological questions (see, e.g., [4, 43, 77]). Typical questions microarrays are used to answer involve finding relationships among gene expressions, therapeutic agents, and patient outcomes. Although a remarkably powerful tool, the analysis of the resulting data is extremely challenging. Data are low precision (just a few bits), noisy, and sometimes missing altogether. The number of microarrays is invariably much smaller than the data per microarray leading to underconstrained systems not amenable to traditional statistical analysis such as finding correlations. More common are various forms of clustering, inference nets, and decision trees.

In one study, Kim *et al.* [40] would like to find a set of genes whose expression could be used to determine whether liver samples are metastatic on not. For biological reasons, it is likely that three genes is an appropriate number to make this determination. Kim *et al.* further propose that use of linear regression would be appropriate to evaluate the gene subsets over the available samples. Since there are tens of thousands of potential genes, 10^{11}–10^{12} data subsets need to be processed. Although simple to implement, he reported that this computation was intractable even on his small cluster of PCs. We found that this algorithm could be implemented extremely efficiently on an FPGA [75] (see also [57] for the application of FPGAs in using microarray data for learning gene regulatory networks). While this combinatoric algorithm has not found widespread use, the FPGA case study illustrates the methods central to microarray computation, including handling noisy low-precision data, reducing large vector spaces, and applying basic operators in linear algebra.

3.9 Characteristics of Computations Amenable to FPGA Acceleration

We now summarize the characteristics of computations amenable to FPGA acceleration:

- *Massive, open-ended parallelism.* HPRC applications are highly parallel, with the possibility of thousands of operations being executed concurrently. Many HPRC applications also feature open-ended parallelism, in the sense that there is effectively no upper bound on the number of PEs that can be applied to the calculation. These applications map well onto devices with thousands of concurrent PEs. For example, processing of long strings parallelizes at the character level, and grid-based molecule interactions parallelize at the level of grid cells. Many HPRC applications share this level of parallelism, despite their different basic units of computation.
- *Dense, regular communication patterns.* Communication is generally regular and local: on any iteration, data only need to be passed to adjacent PEs. The FPGA's large number of communication paths ensures that all PEs can send and receive data every cycle, while the local communication ensures low latency. For example, string processing, alignment by dynamic programming, 3D correlation, and other applications all meet this description. This allows well-understood hardware techniques to be employed, including systolic arrays and pipelines with hundreds or thousands of steps.
- *Modest working sets and deterministic data access.* Although HPRC data sets can be large, they are often amenable to partitioning and to heavy reuse of data within partitions. When the working sets are too large to fit on-chip, they usually have predictable reference patterns. This allows the relatively high latency of off-chip transfers to be hidden by the high off-chip bandwidth (500 signal pins). In extreme cases, such as when processing large databases,

data can be streamed through the FPGA at multi-Gb rates by using the dedicated I/O transceivers.

- *Data elements with small numbers of bits.* Reducing the precision of the function units to that required by the computation allows the FPGA to be configured into a larger number of function units. Many HPRC applications naturally use small data values, such as characters in the four-letter nucleotide alphabet, or bits and fixed-point values for grid models of molecules. Although standard implementations generally use floating point, analysis often shows that simpler values work equally well.
- *Simple processing kernels.* Many HPRC computations are repetitive with relatively simple processing kernels being repeated large numbers of times. The fine-grained resource allocation within an FPGA allocates only as many logic resources as needed to each PE. Simpler kernels, requiring less logic each, allow more PEs to be built in a given FPGA. This tradeoff of computation complexity versus processor parallelism is not available on fixed processors. As already stated, HPRC calculations often benefit from large computing arrays, and PEs within the arrays are typically simple.
- *Associative computation.* FPGA hardware works well with common associative operators: broadcast, match, reduction, and leader election. In all of these cases, FPGAs can be configured to execute the associative operator using the long communication pathways on the chip. The result is that, rather than being a bottleneck, these associative operators afford perhaps the greatest speedup of all: processing at the speed of electrical transmissions.

Not all problems work well in FPGAs. Those requiring high-precision floating-point calculations often consume so many logic resources that there is little opportunity for on-chip parallelism. In many cases, however, applications implemented in double-precision floating point on standard processor can be reimplemented in reduced precision, fixed point, or other arithmetic, with little or no cost in accuracy.

4. Methods for Avoiding Implementational Heat

These 12 methods were selected for easy visualization; they are neither exhaustive nor disjoint. Also, we have avoided low-level issues related to logic design and synthesis that are well known in electronic design automation, and high-level issues such as partitioning that are well known in parallel processing. The focus is on our own work in bioinformatics and computational biology (BCB), but applies also to other standard FPGA domains such as signal and image processing.

4.1 Use an Appropriate FPGA Computing Model

4.1.1 Overview

In recent work [33], we have addressed the fact that while HPRC has tremendous potential performance, few developers of HPC applications have thus far developed FPGA-based systems. One reason, besides the newness of their viability, is that FPGAs are commonly viewed as hardware devices and thus require use of alien development tools. Another is that new users may disregard the hardware altogether by translating serial codes directly into FPGA configurations (using one of many available tools; see, e.g., [36] for a survey). While this results in rapid development, it may also result in unacceptable loss of performance when key features are not used to their capability.

We have found that successful development of HPRC applications requires a middle path: that the developer must avoid getting caught up in logic details, but at the same time should keep in mind an appropriate FPGA-oriented computing model. There are several such models for HPRC; moreover, they differ significantly from models generally used in HPC programming (see, e.g., [16, 64]). For example, whereas parallel computing models are often based on thread execution and interaction, FPGA computing can take advantage of additional degrees of freedom than available in software. This enables models based on the fundamental characteristics from which FPGAs get their capability, including highly flexible fine-grained parallelism and associative operations such as broadcast and collective response (see DeHon *et al.* [18] for a perspective of these issues from the point of view of design patterns).

Putting this idea together with FPGA characteristics described earlier: A good FPGA computing model is one that lets us create mappings that make maximal use of available hardware. This often includes one or more levels of the FPGA memory hierarchy. These mappings commonly contain large amounts of fine-grained parallelism. PEs are often connected as either a few long pipelines (sometimes with 50 stages or more), or broadside with up to a few hundred very short pipelines.

Another critical factor in finding a good FPGA model is that code size translates into FPGA area. The best performance is, of course, achieved if the entire FPGA is used continuously, usually through fine-grained parallelism as just described. Conversely, if a single pipeline does not fit on the chip, performance may be poor. Poor performance can also occur with applications that have many conditional computations. For example, consider a molecular simulation where determining the potential between pairs of particles is the main computation. Moreover, let the choice of function to compute the potential depend on the particles' separation. For a microprocessor, invoking each different function probably involves little overhead. For an FPGA, however, this can be problematic: each function takes up part of the chip, *whether it is being used or not*. In the worst case, only a fraction of the FPGA is ever in use. Note that all may not be lost: it may still be possible to maintain high utilization by scheduling tasks among the functions and reconfiguring the FPGA as needed.

Finally, while FPGA configurations resemble high-level language programs, they specify hardware, not software. Since good computing models for software are not necessarily good computing models for hardware, it follows that restructuring an application can often substantially improve its performance. For example, while random access and pointer-based data structures are staples of serial computing, they may yield poor performance on FPGAs. Much preferred are streaming, systolic and associative computing, and arrays of fine-grained automata.

4.1.2 Examples

4.1.2.1 Molecular Dynamics: Short-Range Forces.

The short-range force kernel can be mapped into a streaming model [28]; this is illustrated in Fig. 5. Particle positions and types are the input, the accelerations the output. Streams source and sink in the BRAMs. The number of streams is a function of FPGA hardware resources and the computation parameters, with the usual range being from 2 to 8.

The wrapper around this kernel is also implemented in the FPGA: it ensures that particles in neighborhoods are available together in the BRAMs; these are swapped in the background as the computation progresses. The force computation has three parts, as shown in blue, purple, and orange, respectively. The first part checks for validity, adjusts for boundary conditions, and computes r^2. The second part computes the exponentials in r. As is often done even in serial MD codes, these terms are not computed directly, but rather with table lookup followed by interpolation. Third order is shown in Fig. 5. The final part combines the r^{-n} terms with the particle type coefficients to generate the force.

4.1.2.2 Sequence Alignment Using BLAST.

Recall that the BLAST algorithm, which operates in multiple phases. First seeds, or good matches of short subsequences, are determined. Second, these seeds are extended to find promising candidates. The direct mapping of this algorithm onto the FPGA is dominated by the extension phase, which requires many random accesses into

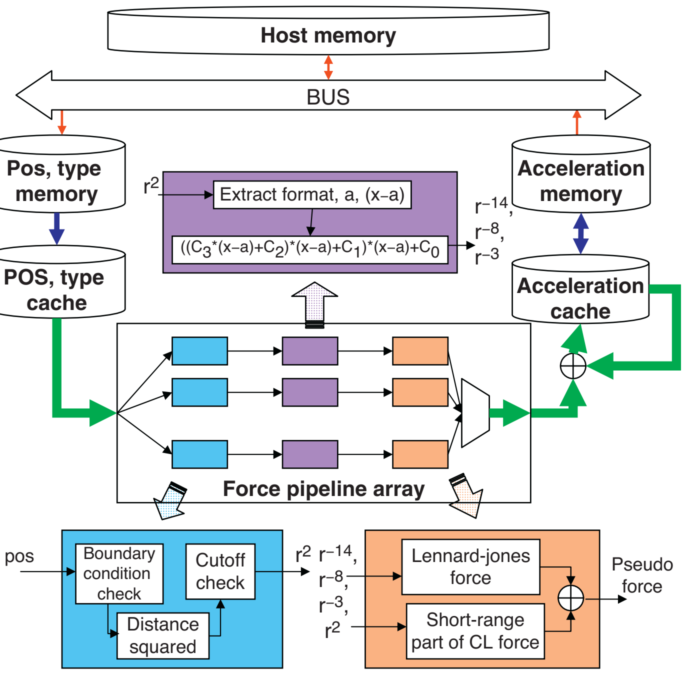


FIG. 5. Pipeline for short-range force computation.

off-chip memory. We found a different approach that avoids random accesses into a large database; rather, the database is streamed through a two-dimensional systolic array. The first dimension generates, on every cycle, the character–character match scores for a particular alignment of the sequence of interest versus the database. The second dimension processes the score sequence to find the maximal local alignment. The tree structure keeps the hardware cost low; pipelining assures that the maximal local alignments are generated at streaming rate (Fig. 6). Multiple streams can operate in parallel.

4.1.2.3 Discrete Event-Based Molecular Dynamics (DMD). For DMD, our approach is based on associative computing [53]. We process the entire simulation a single long pipeline (see right panel of Fig. 7). While dozens of events are processed simultaneously, at most one event is committed per cycle. To achieve maximum throughput, the following must be done within a single cycle (1) update the system state, (2) process all causal event

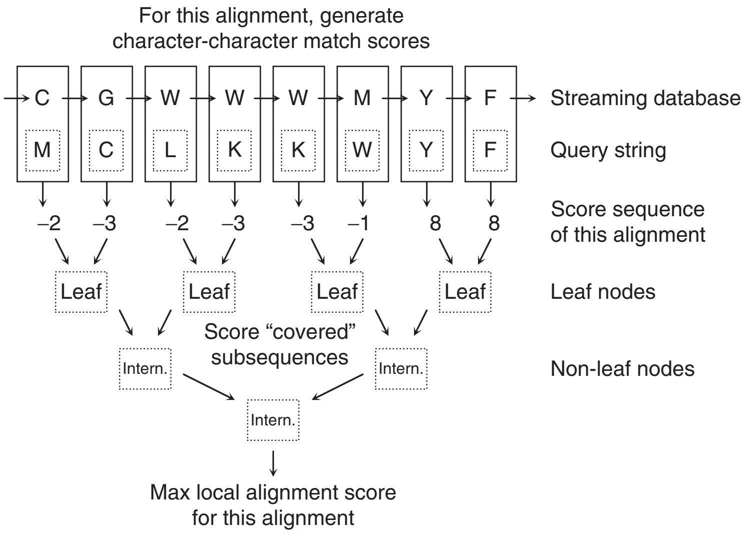

For this alignment, generate character-character match scores
C
G
W
W
W
M
Y
F
Streaming database
M
C
L
K
K
W
Y
F
Query string
–2
–3
–2
–3
–3
–1
8
8
Score sequence of this alignment
Leaf
Leaf
Leaf
Leaf
Leaf nodes
Score "covered" subsequences
Intern.
Intern.
Non-leaf nodes
Intern.
Max local alignment score for this alignment

FIG. 6. BLAST can be restructured so that most of the work is done by a filtering step, resulting from the use of the streaming computing mode.

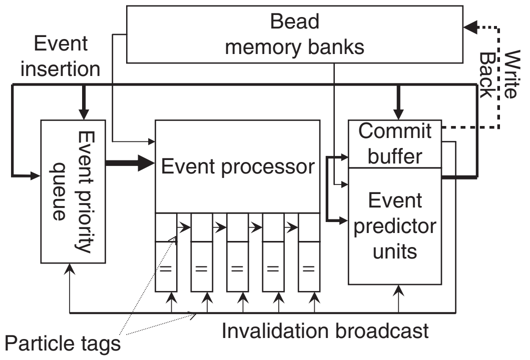


FIG. 7. Block diagram of an HPRC implementation of DMD. System state is the bead memory, updates and invalidations are performed by the broadcast network, and processing is done by the computation pipeline.

cancellations and (3) new event insertions, and (4) advance the event priority queue. This, in turn, uses the associative primitives of broadcast, tag check, and conditional execution. When an event is committed, the IDs of the particles it involves are broadcast to the events in the priority queue. If there is an ID match, the predicted event is cancelled. Similarly, when events are predicted, their timestamp is broadcast throughout the priority queue. Existing events compare their timestamps to that of the new event and it is inserted accordingly.

4.2 Use an Appropriate FPGA Algorithm

4.2.1 Overview

Even with the selection of an appropriate computing model, it is common for there to be multiple plausible algorithms for a given task, with selection based on application and target hardware. Crucial to creating HPRC applications is that, frequently, the optimal algorithm for an FPGA is different from the optimal algorithm for a serial computer or MPP.

4.2.2 Example

In rigid molecule docking, a commonly used technique digitizes each molecule onto a 3D voxel grid, then applies correlations to match the physical shape and chemical affinities of a candidate drug molecule to pockets within a protein or other biomolecule of medical interest. A generalized 3D correlation is then computed using some number of FFTs.

Correlation can of course be performed either directly or by FFTs. Transform-based techniques have better asymptotic complexity than direct summation. Comparisons of polynomial complexity can be deceptive, however, since they apply only to problems so big that low-order terms and scaling constants no longer matter.

When the digitizing grid has edge dimension N, correlation by direct summation has asymptotic complexity $O(N^6)$. FFT-based techniques, however, have complexity $O(N^3 \log N)$. This theoretical advantage has large practical importance on PC implementations of useful sizes, cubical grids of edge dimension around 100. Comparisons of polynomial complexity are only valid for asymptotically large programs, however. When one of the molecules is small, as is often the case in drug design, then technology-dependent coefficients and low-order terms often dominate [65].

So despite having worse asymptotic complexity, the preferred FPGA algorithm—at least for small molecule docking—is based on direct summation. This is due to multiple factors. The first is that small data sizes, such as 1-bit values for representing interior versus exterior information, offer little advantage on a PC processor. On an FPGA, however, smaller PEs allow larger numbers of PEs for a given amount of computing fabric, and products of 1-bit values are trivial to implement. Second, efficient systolic arrays for correlation are well known. The form we chose requires one input value and generates one output value per cycle, while holding partial sums in on-chip registers and RAM-based FIFOs. Hundreds of dual-ported, on-chip RAMs hold intermediate results, eliminating that as a potential bottleneck. Third, our implementation (after a brief setup phase) delivers one multiply-accumulate (MAC) operation per clock cycle per PE, with hundreds to thousands of PEs in the computing array. No additional cycles are required for indexing, loop control, load/store operations, or memory stalls. Despite clock rates at least 10× lower than a PC's, the FPGA executes thousands of times more payload computations per cycle.

As an aside, we observe that direct summation creates research opportunities that were considered infeasible using transform-based techniques. FFTs handle only standard correlation, involving sums of products. Although complex chemical effects are modeled by summing multiple correlations of different molecular features, FFTs require every model to be phrased somehow as sums of $a \times b$. Direct summation makes it easy to perform a generalized sum-of-$F(a, b)$ operation, where F computes an arbitrary and possibly nonlinear score for the interaction of the two voxels from the two molecules. It also allows arbitrary (and possibly different) data types for representing voxels from the two molecules. We commonly represent voxels as tuples with fields for steric effects, short-range forces, Coulombic interaction, or other phenomena.

4.3 Use Appropriate FPGA Structures

4.3.1 Overview

Certain data structures such as stacks, trees, and priority queues are ubiquitous in application programs, as are basic operations such as search, reduction, parallel prefix, and suffix trees. Digital logic often has analogs to these structures and operations that are equally well known to logic designers. They are also completely different from what is obtained by translating the software structures to hardware using an HDL. The power of this method is twofold: to use such structures when called for, and to steer the mapping toward those structures with the highest relative efficiency. One particular hardware structure is perhaps the most commonly used in all HPRC: the systolic array used for convolutions and correlations (see, e.g., [66]).

4.3.2 Examples

4.3.2.1 Multigrid for Electrostatic Computation. When mapping multigrid to an FPGA, we partition the computation into three functions (1) applying the charges to a 3D grid, (2) performing multigrid to convert the 3D charge density grid to a 3D potential energy grid, and (3) applying the 3D potential to the particles to compute the forces. The two particle–grid functions are similar enough to be considered together, as are the various phases of the grid–grid computations. For the 3D grid–grid convolutions we use the well-known systolic array. Its iterative application to build up two- and three-dimensional convolvers is shown in Fig. 8.

In the basic 1D systolic array, shown in Fig. 8A, the kernel $A[0..L]$ is held in the PEs and the new elements of the ''signal'' $B[i]$ are broadcast to the array, one per iteration. At each PE, the $B[i]$ are combined with the $A[k]$; the result is added to the running sum. The running sums are then shifted, completing the iteration. One result is generated per iteration. The same basic procedure is used for the 2D and 3D cases (shown in Fig. 8B and C), but with delay FIFOs added to account for difference in sizes of A and B.

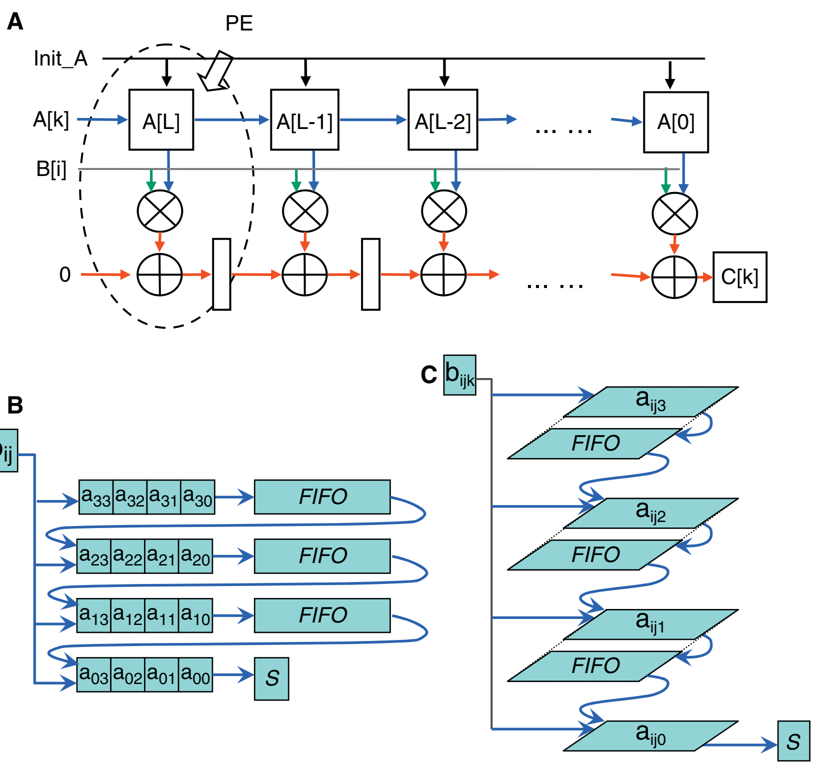


FIG. 8. Shown are (A) a one-dimensional systolic convolver array, and its extension to (B) two, and to (C) three dimensions.

4.3.2.2 Finding Repetitive Structures in Biological Sequences.

The hardware implementation of a palindrome finder is well known. We use it as shown in Fig. 9 to find, at streaming rate, palindromes of various lengths, with arbitrary gap size, and with some number of mismatches. Sequences to be analyzed are input from the left and streamed through the circuit. Palindrome recognition results from character–character comparisons of the folded string. To find the greatest length exact match palindrome at any position (not shown), the matches pass directly through a priority encoder which converts the binary sequence (of *T*/*F*) to the length of the sequence of *T*s from right to left. For a predefined gap size, a delay FIFO can be added.

To account for some number of mismatches, the logic shown is appended. As a string streams through the character comparison portion of the array, it is folded back on itself. The center of the fold is considered to be position 0. Characters at positions $+n$ and $-n$ relative to the fold are stored in the same cell and compared, giving a value of +1 for matching success or 0 for failure. The 1 and 0 outputs are summed across the length of the folded string, starting from position 0. As long as the sum at position n has the value n, then all characters from the fold to that point match and the palindrome is exact. Rather than create a summing chain (and propagation delay)

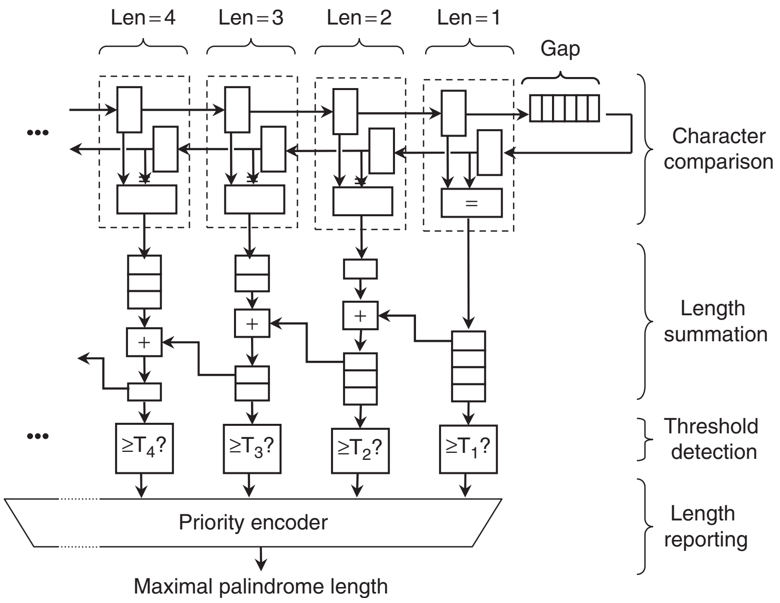


FIG. 9. This structure finds all palindromes in a sequence with some number of errors and a fixed gap.

the full length of the character comparison array, this implementation pipelines summation so that only one addition is performed per clock cycle. Summation results are lagged so that all of the length totals for a single time step exit the length summation section together.

4.4 Mitigate Amdahl's Law

4.4.1 Overview

Amdahl's law states that speeding up an application significantly through an enhancement requires most of the application to be enhanced. This is sometimes difficult to achieve with existing HPC code; for example, profiling has pointed to kernels comprises just 60–80% of execution time when much more could have been expected (as was found, e.g., by Alam *et al.* [1] in their MD application). The problem is especially severe with legacy codes and may require a substantial rewrite. Not all is lost, however. The nonkernel code may lend itself to substantial improvement; as its relative execution time decreases, expending effort on its optimization may become worthwhile. Also, combining computations not equally amenable to FPGA acceleration may have optimized the original code; separating them can increase the acceleratable kernel.

4.4.2 Example

Molecular dynamics codes are often highly complex and often have legacies extending over decades (see, e.g., [10, 12]). While these codes sometimes extend to millions of lines, the acceleratable kernels are much smaller. Various approaches have been used. These include writing the MD code from scratch [61]; using a simplified version of an existing standard, in this case NAMD [41]; accelerating what is possible in an existing standard, in this case AMBER [63]; and using a code already designed for acceleration, for example, ProtoMol [28]. In the last case the ProtoMol framework was designed especially for computational experimentation and so has well-defined partitions among computations [51]. We have found that the acceleratable kernel not only comprises more than 90% of execution time with ProtoMol, but the modularity enables straightforward integration of an FPGA accelerator [28].

4.5 Hide Latency of Independent Functions

4.5.1 Overview

Latency hiding is a basic technique for obtaining high performance in parallel applications. Overlap between computation and communication is especially desirable. In FPGA implementations, further opportunities arise: rather than allocating tasks to processors among which communication is then necessary, functions are simply laid out on the same chip and operate in parallel.

Looking at this in a little more detail, while having function units lying idle is the bane of HPRC, functional parallelism can also be one its strengths. Again, the opportunity has to do with FPGA chip area versus compute time: functions that take a long time in software, but relatively little space in hardware are the best. For example, a simulator may require frequent generation of high-quality random numbers. Such a function takes relatively little space on an FPGA, can be fully pipelined, and can thus provide random numbers with latency completely hidden.

4.5.2 Example

We return to the docking example. There are three independent functions, shown in Fig. 10: rotation, correlation, and filtering. The correlations must be repeated at many three-axis rotations: over 10^4 for typical 10-degree sampling intervals. Implementations on sequential processors typically rotate the molecule in a step separate from the correlation.

Again, the FPGA solution is quite different. Rather than performing an explicit rotation, the pixels are retrieved in ''rotated order.'' The (i, j, k) of each voxel in index space can be expressed as a function of the original (x, y, z) coordinates and the rotation (see Fig. 11). A simplified computation depends on 18 parameters specific to each rotation. One possible FPGA implementation computes the (i, j, k) in series with the pixel fetch, resulting in prohibitive overhead. Another possible solution is to precompute the indices and load them as needed. But since there are typically 10^6

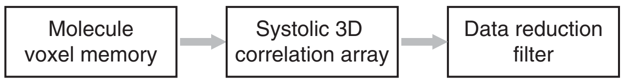


FIG. 10. The HPRC rigid docking application consists of three pipelined, independent functions.

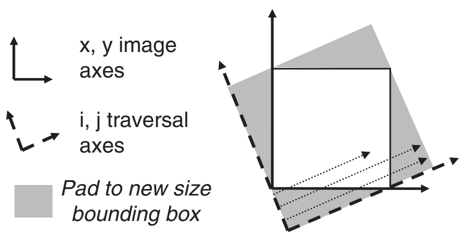


FIG. 11. The rotation function is performed by fetching voxels in rotated order.

voxels and 10^4 rotations, this would require gigabytes of storage, and so not lend itself to rapid retrieval.

The preferred FPGA solution is based on the run-time index calculation, but with two modifications. The first is that the index calculator be a separate hardware module; this only requires a few percent area of a contemporary high-end FPGA. The second is that the calculator be fully pipelined so that the rotation-space coordinates are generated at operating frequency.

4.6 Speed-Match Sequential Computations

4.6.1 Overview

If a computation pipeline has strictly linear structure, it can run only at the speed of the slowest element in the pipeline. It is not always convenient or even possible to insert pipelining registers into a time-consuming operation to increase its clock rate. Instead, the pipeline element can be replicated to operate in parallel and used in rotation to get the desired throughput.

4.6.2 Example

In our microarray case study, we used a data set (from Perou *et al.* [56]) of roughly 100 samples, each of 10^4 gene expressions, with cancerous versus healthy state as the independent variable. We analyzed the samples to find correlations between expression patterns and disease phenomena. Each gene expression is represented as a vector with 100 entries, each corresponding to a sample. The procedure was to examine all three-way combination of the 10^4 expressions, or roughly 10^{11} combinations. Scoring of expression subsets was done by linear regressions of diagnosis against expressions. The kernel operation here is a series of dot products and sums (DPS) feeding covariance, matrix inversion, and regression (CIR) logic.

The problem was that CIR was 10 times faster than DPS. As we have seen, the power of the FPGA comes from the parallel hardware that can be brought to bear on a problem. Usually the solution, as here, involves a very deep pipeline hundreds or even thousands of stages long. When successive functions have different rates of sourcing and sinking data, however, throughput is reduced to that of the bottleneck.

The solution is to rate-match sequential functions by replicating the slower functions, and then using them in rotation to get the desired throughput. In the microarray kernel, the DPS units take about 10 times as long to sum over vectors as the CIR units take to consume DPS results, so DPS are replicated that number of times per CIR: the resulting design is shown in Fig. 12. In this application one more factor contributed to the efficacy of the solution: DPS was much less resource intensive than CIR and so the final design was well balanced between these functions.

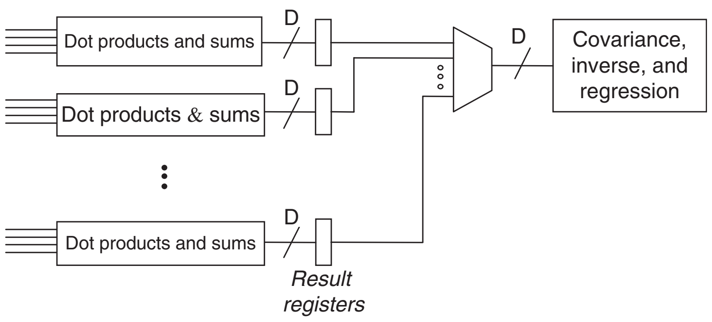


FIG. 12. In the microarray case study, the dot-product bottleneck can be removed by replicating the slow elements.

4.7 High Performance = High-Performance Data Access

4.7.1 Overview

The ''memory wall'' is often said to limit HPC performance. In tuning applications, therefore, a great deal of effort goes into maximizing locality through careful placement of data and ordering of data accesses. With FPGAs, our experience is that there is no one wall; instead, there are many different ones, each characterized by a different pattern of memory access. Data placement and access orchestration are still critical, although with a different memory model. An additional opportunity for optimization results from configuring the multiple internal memory buses to optimize data access for a particular application.

Already mentioned is that if you can use the full bandwidth at any level of the memory hierarchy, the application is likely to be highly efficient. Added here is that on an FPGA, complex parallel memory access patterns can be configured. This problem was the object of much study in the early days of array processors (see, e.g., [48]): the objective was to enable parallel conflict-free access to slices of data, such as array rows or columns, followed by alignment of that data with the correct processing elements. With the FPGA, the programmable connections allow this capability to be tailored to the application-specific reference patterns (see, e.g., [70]).

4.7.2 Examples

For our first application, we continue with the microarray analysis case study. The kernel computation is as before; here we add a communication network to route triplets of input vectors to the DPS units. The FPGA used has enough computing fabric to instantiate 90 DPS pipelines. Each DPS processes three vectors of measurement data (*X*-vectors) plus one vector of diagnosis values (the *Y*-vector). Diagnoses are the same in all cases, but each DPS must process a different set of three *X* values, or 270 in all. At 4 bits per *X* value, that would have required over 1000 bits of memory data per cycle.

Although feasible within chip resources, system considerations showed that our host and backplane would not be able to supply data fast enough to keep the computation units busy. Instead of accessing 270 values from memory, our implementation accesses only nine, as shown in Fig. 13. This bus subsets the nine *X* values into all possible subsets of size three—84 subsets in all. Although the data bus reads only nine *X* values from RAM, the FPGA's high capacity for fan out turns that into 252 values supplied to computation units, a 28$\times$ boost at essentially no hardware cost.

For the second example, we complete our discussion of the multigrid application. The first and third phases are transformations from particle to grid and grid to particle representations, respectively. Since atoms almost never align to the grid points on which the field is computed, tricubic interpolation uses the 64 grid points nearest the atom to determine field strength. Figure 14 illustrates the computation, simplified to the bilinear case in two dimensions.

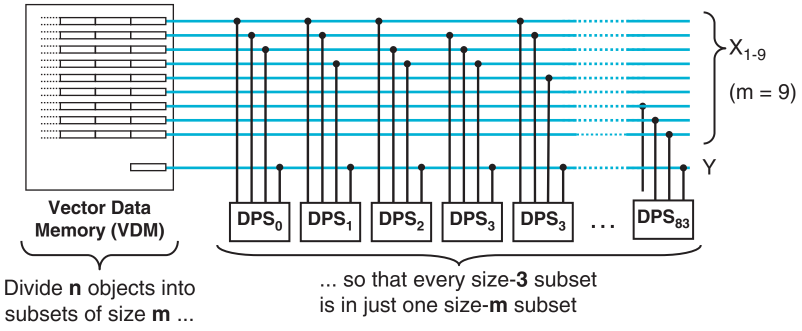

$X_{1\text{-}9}$
(m = 9)
Y
Vector Data Memory (VDM)
DPS_0
DPS_1
DPS_2
DPS_3
DPS_3
...
DPS_{83}
Divide **n** objects into subsets of size **m** ...
... so that every size-**3** subset is in just one size-**m** subset

FIG. 13. In the microarray case study, vectors are routed into combinations.

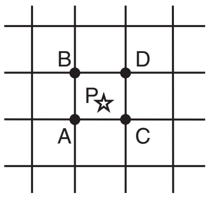


FIG. 14. In this simplified drawing, point *P* is shown along with its nearest, 2D grid points.

Atom positions are not wholly predictable, so there is no opportunity for common memory optimizations based on predicting access patterns. Instead, field values for all 64 of the atom's neighboring grid points must be fetched to compute field strength. In a PC, this would require 64 separate memory access operations. Even if the computation itself had no cost, it would still require a minimum of 64 cycles to complete. The FPGA solution's goal is to create a structure that computes forces at a rate of one per cycle, accounting for unpredictable sequences of atom positions.

Our FPGA implementation starts (for the simpler trilinear eight-point case) with the observation that of the *X*-axis points around the atom, one always has an even value and the other is odd. The same is true along the *Y*- and *Z*-axes. Of the eight grid points neighboring the atom, one has an (even, even, even) index triplet, one is (odd, odd, odd), and so on for all eight of the possible index combinations. That makes it possible to create a memory specifically for this application with eight interleaved memory banks, using the FPGA's internal RAMs. One memory bank holds only words with (even, even, even) indices, and so on for the other seven index combinations and memory banks. It should be clear that every bank is required for one trilinear interpolation, and that no atom position can cause an access collision at any memory bank. In other words, this interleaving allows one eight-point interpolation to be started at each clock cycle, an improvement over one every 8 cycles. Some logic is required for handling the differences between (even, odd) and (odd, even) pairs along each axis. For current purposes, it is enough to say that the necessary logic took only a tiny percentage of the FPGA's logic resources. If that logic had been on the computation's critical path, it could easily have been pipelined.

4.8 Use Appropriate Arithmetic Precision

4.8.1 Overview

With high-end microprocessors having 64-bit data paths, it is often overlooked that many BCB applications require only a few bits of precision. In fact even the canonically floating-point MD has often been implemented with substantially reduced precision, although this remains controversial. In contrast with microprocessors, FPGAs allow data paths to be configured into arbitrary sizes. This offers at least two kinds of potential performance improvement. The smaller effect comes from shorter propagation delays through narrower adders or multipliers. The bigger opportunity, though, comes from the ability to reallocate resources trimmed out of one PE into another one. If one data path is cut from 8 to 4 bits, it may be possible to create a second data path from the resources saved. Resource conservation does not just optimize PEs, it can change the size and degree of parallelism in the array of PEs.

4.8.2 Examples

All applications described here benefit substantially from the selection of nonstandard data type sizes. Microarray values and biological sequences require only 4–5 bits, shape characterization of a rigid molecule only 2–7. While MD probably requires more than the 24 bits provided by single-precision floating point, double precision (53 bits) may not be required [27].

The tradeoff between PE complexity and degree of parallelism was made clear in the docking case study [71]. There we examined six different models describing intermolecular forces. Molecule descriptions range from 2 to 7 bits per voxel, and scoring functions varied with the application. Fitting the various maximum-sized cubical computing arrays into a Xilinx XC2VP70, the number of PEs ranged from 512 to 2744. Since clock speeds also differed for each application-specific accelerator, they covered a 7:1 performance range. If we had been restricted to, say, 8-bit arithmetic, the performance differential would have been even greater.

Similar, though less dramatic, results appeared in a case study that accelerated the computation core of the ProtoMol molecular dynamics code [27]. There we presented a careful examination of computation quality, measured as numerical stability, as a function of the number of bits used for computation. We observed that, after about 35 bits of precision in the accumulators, there was little additional gain in the quality measure. That allowed eight force pipelines to be instantiated rather than four. Because of the difficulty in routing the larger design, only a small performance gain was observed, however.

4.9 Use Appropriate Arithmetic Mode

4.9.1 Overview

Microprocessors provide support for integer and floating point data types, and, depending on multimedia features, 8-bit saturated values. In digital signal processing systems, however, cost concerns often require DSPs to have only integers. Software can emulate floating point, when required; also common is block floating point. FPGA's analogous situation is that, although plausible, single-precision floating point remains costly and should be avoided if possible, with well-tuned libraries available. Alternatives include the block floating point, log representations, and the semifloating point.

4.9.2 Example

The MD computation's inner kernel operation requires computing r^{-14} and r^{-8} for the radius r between atoms, over a wide range, usually with a table lookup. We would generally use double-precision floating point for further computations.

Careful analysis shows that the number of computed distinct alignments is quite small even though the range of exponents is large. This enables the use of a stripped-down floating-point mode, particularly one that does not require a variable shift. The resulting force pipelines (with 35-bit precision) are 25% smaller than ones built with a commercial single-precision (24-bit) floating-point library.

4.10 Minimize Use of High-Cost Arithmetic

4.10.1 Overview

The relative costs of arithmetic functions are very different on FPGAs than they are on microprocessors. For example, FPGA integer multiplication is efficient in comparison with addition, while division is orders of magnitude slower. Even if the division logic is fully pipelined to hide its latency, the cost is still high in chip area, especially if the logic must be replicated. On an FPGA, unused functions need not be implemented; recovered area can then be used to increase parallelism. Thus restructuring arithmetic with respect to an FPGA cost function can result in substantial performance gain.

A related tradeoff involves the general observation that these differences encourage careful attention to the way in which numbers are represented and the ways in which arithmetic operations are implemented, decisions that often go together.

4.10.2 Example

The microarray data analysis kernel as originally formulated requires division. Our solution is to represent some numbers as rationals, maintaining separate numerator and denominator, replacing division operations with multiplication. This doubles the number of bits required, but rational values are needed only in a short, late occurring segment of the data path. As a result, the additional logic needed for the wider data path is far lower than logic for division would have been.

We also turn to the microarray application for an example of where rewriting expressions can be helpful. This application originally involved a matrix inversion for each evaluation.

We initially expressed the problem as a 4×4-matrix, which would then need to be inverted to find the final result. This, however, led to an unacceptable complexity in the hardware computation. After additional research, we found an equivalent way to phrase the problem. It required more computation in setting up the problem, and more bits of precision in the intermediate expressions, but allowed us to reduce the system to a 3×3-matrix. The net effect was to reduce overall computing complexity at the cost of some increase in the number of bits needed in the intermediate results.

At this point, Cramer's rule became an attractive solution technique. The algorithm is notoriously unstable, and has polynomial complexity of *N*! in the size of the matrix. At this small array size, however, *N*! is still small enough for the algorithm to be feasible and it does not have enough terms for the instabilities to accumulate. We also knew that the matrix (and therefore its inverse) was symmetric, so some result terms could be copied rather than recomputed. As a result, the inverse could be expressed in closed form using a manageable number of terms. Since we knew that the input values had only 4-bit precision, we were able to reduce the precision of some intermediate expressions—4-bit input precision hardly justifies 64-bit output precision.

4.11 Support Families of Applications Rather Than Point Solutions

4.11.1 Overview

HPC applications are often complex and highly parameterized: this results in the software having variations not only in data format, but also in algorithm to be applied. These variations, including parameterization of functions, are easily supported with contemporary object-oriented technology. This level of parameterization is far more difficult to use in current HDLs, but enables higher reuse of the design. Amortization of development cost is over a larger number of uses, and there is less reliance on skilled hardware developers for each variation on the application.

4.11.2 Example

Essential methods for searching biological databases are based on dynamic programming (DP). Although generally referred to by the name of one variation, Smith–Waterman, DP-based approximate string matching is a large number of related algorithms, which vary significantly in purpose and complexity. Achieving high performance in HPRC requires careful tuning to the specifics of the application, which limits a component's reusability. General, programmable PEs rarely approach the speed or resource efficiency of tuned applications. Reusable HPC/FPGA applications must resolve the conflicting requirements of generality and customization.

We start with two observations on component reuse in FPGA systems. The first is that, in traditional hardware design systems, components are black boxes with limited parameterization of their internals. Reuse consists largely of creating communication and synchronization structures between them, and connecting them to the memory subsystems. The second observation is that, in HPRC, it is often the leaf data types and arithmetic expressions that change between applications, that is, the innermost components. The FPGA system's performance depends on its memory, synchronization, and communication, which are the parts most unfamiliar to programmers with traditional skills. As with the standard C library's qsort(), control and communication are the reusable parts; inner function blocks and data types are the customizations—the opposite of what typical design tools support.

We use the term application family to describe a computation that matches this description, and offer a family of approximate string-matching algorithms as an example [73]. These are dynamic programming algorithms for computing edit distances between strings—weighted scores representing the number of insertions, deletions, and character replacements needed to convert one string into another. These algorithms all have the same general kind of recurrence relation, but differ from each other at several levels, as shown in Fig. 15.

The lowest level of variation is Fig. 15's character rule (CharRule) abstraction. This defines the data type of a single character in one of the strings to be compared. In biological applications, these commonly represent DNA strings (2-bit encodings), proteins (5 bits), or codons (6 bits). The character rule also defines a matching function, such as an exact equality test, wildcard match, or graded goodness-of-match scores based on parameterized BLOSUM or PAM substitution matrices.

Character rule components written to the common interface are interchangeable in the matching cell (MatchCell in Fig. 15). This implements the recurrence relation in the dynamic programming algorithm. Matching cells are somewhat different for end-to-end (Needleman–Wunsch) string comparisons than for best-substring (Smith–Waterman) matching. Both use character rule instances in the same way,

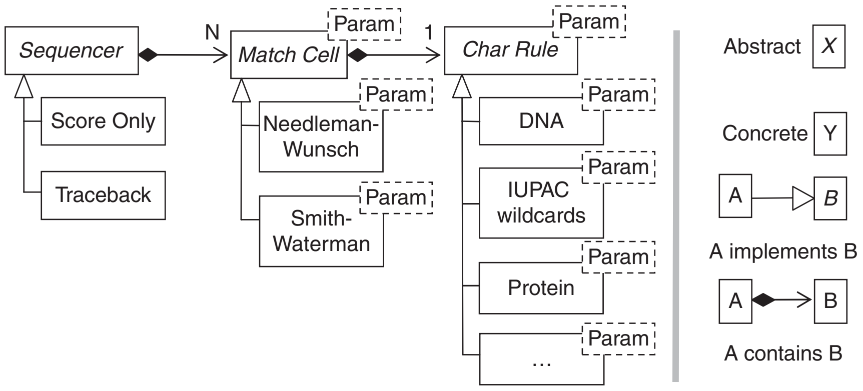


FIG. 15. Logical structure of application family of DP-based approximate string matching. Each level of the design hierarchy has fixed interfaces to the components above and below that hierarchy. Within a hierarchical level, each component type has several possible implementations, which the fixed interface hides from other design layers.

and assume the same communication paths between matching cells. The similarities are captured in their shared abstraction; their differences in algorithm and in bookkeeping data are hidden in their private implementation details.

At the third level, data are sequenced through the matching cells in one of two ways: in a single matching pass, producing only an integer similarity score, or with a second traceback pass that records the character by character relationships that yield the highest score.

Our initial implementation using the LAMP tool suite [68, 69] allowed over 200 combinations of the three component types, with many more variations possible through parameter settings. This structure was quite natural in the object-oriented algorithms we used but required more configurability than VHDL features provide.

4.12 Scale Application for Maximal Use of Target Hardware

FPGA-based computations very often differ from standard logic applications in one critical feature. Logic designs typically have specific performance goals. Success is binary: a design does or does not meet its timing requirements. There is little or no benefit in exceeding requirements by a larger margin. In computing applications, however, faster is better. Since computation accelerators are very often arrays of PEs, and since performance is typically dominated by the degree of parallelism, part of accelerator design consists of instantiating as many PEs as the FPGA's computing fabric will support. The number of PEs depends on three sources of information: the FPGA's capacity, the geometry of the array defined by the application family, and the sizes of PEs defined by a specific member of the application family [72].

FPGA capacity has several terms, according to the hardware resources available: hard multipliers and block RAMs as well as general-purpose logic elements. Even at that, capacity is hard to abstract across different families of FPGAs. For example, block RAMs in Xilinx Virtex-II FPGAs are 18 Kbit units, but Altera's Stratix family offers a combination of 512-bit, 4 Kbit, and 512 Kbit units. The Altera FPGA family that corresponds most closely to V5 is Stratix IV. In that family, the Ram sizes are 640 b, 9k b, 144k b. Multiple units can be combined, in both systems, to create a wide variety of logical memory structures, but the Virtex and Stratix resources are not interchangeable in all cases.

The application family defines the geometry of the computation array. As shown in Fig. 16A, arrays can be simple linear structures. Other geometries are possible, however, and present different problems for designers optimizing the performance of computing arrays. Figure 16B illustrates a rectangular array. It is characterized by two different architectural parameters (N_1 for height and N_2 for width) rather than just one. Architectural parameters need not be numbers of PEs; they can abstract

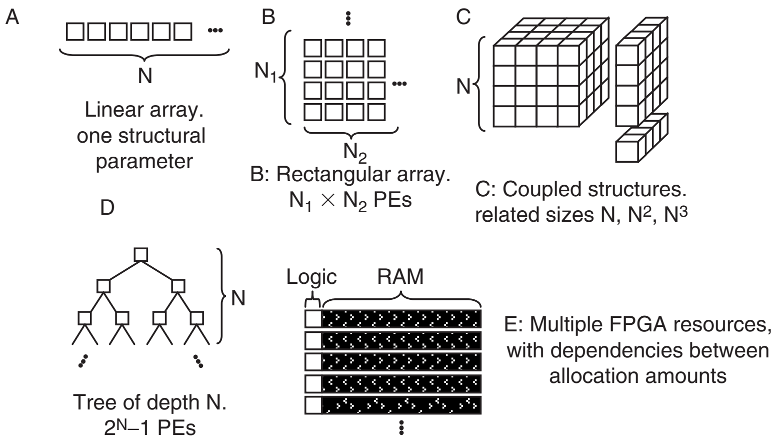


Fig. 16. Growth force laws for computing arrays specified in terms of architectural parameters. (A) Linear array—one structural parameter. (B) Rectangular array—$N_1 \times N_2$ PEs. (C) Coupled structures—related sizes N, N^2, N^3. (D) Tree of depth $N - 2^N - 1$ PEs. (E) Multiple FPGA resources, with dependencies between allocation amounts.

aspects of the design in arbitrary ways. They can also introduce new kinds of constraints, for example, when a rectangular array requires width greater than or equal to its height.

Figure 16D illustrates a tree-structured computing array, showing just one of the ways that arrays can grow according to exponential or other nonlinear law. We have also observed computing arrays like, that in Fig. 16C, composed of multiple coupled subsystems. In many cases, the different subsystems grow according to different algebraic growth laws, at the same time that they remain coupled to each other. Because different subsystems often use different FPGA resources (as in Fig. 16E), either resource can be the one that limits eventual growth of the computing array. Of course, computing arrays can combine these features. The growth law can include multiple architectural parameters, nonlinear growth patterns, coupled subsystems that grow according to different algebraic laws, and use of multiple resource types.

The form of the computing array is defined by the application family, but the size of each PE in the array depends on the specific member of the application family. In string applications, PE size depends on the number of bits in the string element (e.g., 2 bits for DNA or 5 bits for proteins) and on the kinds of comparison being performed. In the drug-docking application, PE size depends on the number of bits per voxel and on the function used to score juxtaposition of the two voxel values.

In this view, the size of an FPGA-based computing array is not a design parameter but a natural result of other design features. The array's architectural parameters are chosen to maximize some application-specific function that represents the value of different configurations. The ‘‘best’’ architectural parameter values are the ones that define the array of highest value, as long as the array’s structure is valid for that application family, and as along as the family member’s PEs live within the FPGA’s resource budget. Automating this kind of optimization is possible within the experimental LAMP design system [68, 69, 72]; it cannot be expressed in mainstream design automation tools or methodologies.

5. GPGPU An Alternative Approach to Acceleration

5.1 Overview

Graphics processing units (GPUs), such as those from Nvidia and ATI/AMD, present another emerging technology in HPC. By intent, GPUs address the seemingly insatiable need for graphics performance in gaming—a market of millions of units annually. Because their architecture optimizes a few basic features of real-time graphics computing, they offer a huge performance boost over standard processors for typical graphics applications. First, the rendering pipeline has a fixed, widely accepted structure, so the processing pipeline can be tuned to that sequence of operations. Second, pixels in different parts of a scene tend to have few dependencies between them, allowing them to be computed independently of pixels elsewhere in the image. Third, pixels close to each other, for example, on the same side of some geometric figure, can often be processed using the same set of operations on slightly different pixel values, allowing a high degree of SIMD parallelism.

Despite the relatively specialized goals of GPUs, their programming model has been adapted for a wide range of applications. As with HPRC, successful acceleration depends on aligning the implementation to the assumptions built into the underlying hardware. Although recent generations of GPUs have additional features that support general programming with GPUs (GPGPU), the features that both enable and limit performance have remained constant. These fall into several major categories, including:

- Data path structure
- Memory access
- Scheduling of processing tasks

Although somewhat intertwined, these need to be considered individually, and in contrast to FPGA programming.

5.2 Data Path Structure

Figure 17 illustrates the logical structure of GPU acceleration hardware. There has been rapid advancement in years. The biggest changes, however, have been in capacity, rather than in drastic architectural updates.

Pixel processing operations have traditionally been characterized by arithmetic intensity (including floating point, 2D interpolations, exponents, and trigonometry operations), but simple control structures and short code fragments. Although capacities keep improving, the code fragments (sometimes called ''shaders'' because of their graphics heritage) have been strictly limited in length. Early GPUs allowed only as few as 64 assembly-level operations. Sizes of code buffers have increased, however, and virtualization allows effectively unlimited numbers of instructions—but with a high penalty when the hardware's code buffer size is exceeded.

Early GPUs used a strict SIMD model, performing the same calculation on a large set of pixels. This made conditional logic difficult, when possible at all, because all of the SIMD PEs would have to execute both the ''then'' and ''else'' branches of a conditional, selectively quashing results from one branch or the other according to each PE's condition settings. More recent GPUs allow greater flexibility in conditional execution, including looping constructs, but good GPU performance often argues against using these features.

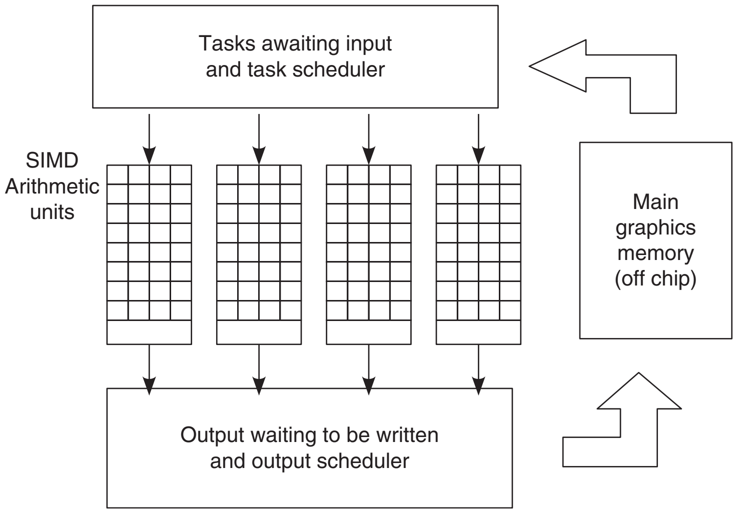


FIG. 17. Logical structure of a GPU.

Although the arithmetic units in a GPU have fixed structure, there is some flexibility in scratch storage. A fixed amount of space for holding the PEs' operands exists, but it can be partitioned differently across the whole ensemble of PEs. Of course, larger allocations to some PEs mean smaller allocations to others, possibly preventing those other PEs from being used at all.

Also, the GPU supports little to no communication between PEs. Recent generations allow some sharing of data between PEs via on-chip buffers, but the main memory remains the only way to exchange large amounts of data between peer PEs or between successive stages of a processing pipeline. As we describe in the next sections, this represents a challenge in exploiting the GPU's processing power.

5.3 Memory Access

The GPU has some amount of on-chip scratch memory that can be accessed by many PEs concurrently, and often has modest memory resources of other kinds. Bulk data storage, however, relies entirely on the GPU's on-board memory. GPUs typically cannot access host memory on their own; the CPU must initiate all data transfers. As a result, even on-board memories of 1 GB or more can become a limiting resource when shared among input data, output data, and intermediate results.

GPU memory is famous for raw bandwidth. Some current models transfer 320 or more bits per clock cycle at rates over 1.5 GHz. Used properly, a GPU can sustain impressive data transfer bandwidth to and from the on-board RAM. This model presents two obvious constraints, however. First, the pipelining in the memory itself, plus additional delays into and out of the GPU, can result in latencies of tens of memory cycles. To sustain throughput, the GPU must have enough tasks available so that many of them can wait for input or output while others occupy the SIMD processors. Second, wide transfers necessarily carry many data words at each memory cycle. Bandwidth is wasted if some of those words are not used.

These features of the GPU memory hierarchy have potentially substantial implications for the application programmer: poor use results in order-of-magnitude performance loss. Wide accesses mean that sparse data structures (like C structs in which not all elements are used) waste bandwidth. Long latency combined with limited communication paths between PEs penalize data dependencies, beyond those within a single code fragment. Also, subtle effects arise, for example, when computing a vector sum "$x = a + b$" on operands aligned differently.

5.4 Task Scheduling

At first glance, the programming model might seem daunting. It requires hundreds to thousands of concurrent tasks to use the hardware effectively, since large numbers of tasks must wait for operands to load from memory or results to store into memory at any given time. Then, memory reads and writes must be scheduled to maximize the utilization of different operands or result slots within each memory transfer. If scheduling needed to be done by the application programmer, for example, by using common system primitives, then this task would be infeasible for all but a very few skilled programmers.

Instead, GPUs move scheduling responsibilities into hardware. In some ways, this resembles data flow computing. As a task (or SIMD task group) has all its dependencies met, it moves to the arithmetic units. On the other side, and somewhat symmetric to the input operations, some number of completed tasks awaits retirement of results into main memory. Here, the scheduler's job is to bundle results from different tasks into a single memory transfer, so that the largest possible number of 16- or 32-bit words in a transfer carry payload data.

An application developer must understand at least this much about the GPU's task scheduler to use the GPU's parallelism effectively. It is not enough to keep all of the PEs busy. The programmer must create a large factor more threads than PEs so that not only are those used, but so that tasks waiting to load operands or store results can use the whole width of each memory transfer effectively. A GPU's potential performance is rarely realized until the application makes hundreds or thousands of tasks available to the scheduler.

Since GPU computations work best as sequences of relatively small computations, a second level of scheduling also becomes important. The host CPU must synchronize and schedule macroevents, such as major phases of execution, while the GPU itself schedules microevents, such as fetching of operands or selection of individual tasks to run. Starting and flushing the GPU's execution pipeline represents more potential loss of efficiency.

5.5 Comparison with HPRC

Although broad generalizations always have exception, a few basic factors tend to distinguish FPGA-based from GPU-based computing. These include:

- *Fixed versus configurable data path*. GPUs use ASIC optimizations to tune performance for data types and operations supported. While FPGAs also have ASIC optimizations, these are at a lower level. FPGAs thus allow arbitrary data types and operations, enabling detailed tradeoffs between numbers of bits per operation and numbers of hardware operators instantiated.
- *Fixed versus configurable memory structure*. GPUs achieve high potential bandwidth using memory and cache structures optimized for rendering operations. FPGAs offer flexibility in the number and width of off-chip memories,

and in arrangement of their hundreds to thousands of independently addressable on-chip RAM blocks.

- *Organization of parallelism*. GPUs offer a fixed parallelism model built around SIMD processing and limited dependencies among data elements. FPGAs allow arbitrary combinations of pipelined and broadside parallelism, and can handle complex communication among processing elements.
- *Familiarity of programming model*. In GPUs, operations on individual data elements can be programmed in languages close to familiar C or assembler. Efficient use of FPGAs usually requires unfamiliar languages, like Verilog or VHDL.
- *Host communication*. GPUs have limited kinds of interactions with host memory. Different FPGA accelerators have different host interaction capabilities. For example, direct Hypertransport connection offers the possibility of access to all of host memory under FPGA control.
- *Floating-point support*. GPUs have traditionally had better floating-point support, up to the set of hardware primitives supported. Larger FPGAs and recent tool innovations [47] have reduced their performance differences.

Each does well at a specific class of operations. If you can cast your application into the GPU's computing model and create enough parallelism (hundreds- to thousands-way), GPUs can do very well, and with widely accessible programming skills. FPGAs do better with idiosyncratic control, data types, concurrency models, and degrees of parallelism. Even with the complexities of GPU memory optimization, FPGAs often require unfamiliar programming skills to reach their performance potential.

6. Conclusion

FPGA-based computation is just now emerging into the main stream of computing practice. Practitioners do not yet have the experience, programming idioms, and tool support needed for pushing FPGAs to their performance potential, and experience from the logic design community has limited value for HPC application. Good tool support will come from a good understanding of the general principles and techniques of FPGA-based computation. General principles can only come from experience with real applications.

We present our case studies, with the lessons learned from them, as a step toward wider use of FPGAs in HPC. Those lessons include intangible advice to application designers. Other lessons have enabled construction of specific design tools, including experimental tools that automate creation of memory structures with application-specific interleaving strategies. At the highest level, we present the concepts of application families and scaling of computation arrays. Although these ideas have been put to use in experimental design tools, they will be most useful as the basis for future experimentation and understanding.

High-performance computers in general remain challenging to program with the result that high-performance programmers are a highly sophisticated but scarce resource. These programmers can therefore be expected to readily use new technology, but not have extended time available to learn a completely new skill such as logic design. As a result, much effort has been expended in developing design tools that translate high-level language programs to FPGA configurations, but with modest expectations even from their developers.

In this chapter, we have described a number of methods that must be applied for HPRC to obtain more than a small fraction of its potential. The critical question is whether these goals are compatible. In other words, what support would enable an HPC programmer to use these methods? We are encouraged that all of the methods we have described appear to be within reach of the HPC programming community. The 12 methods for avoiding implementational heat can be divided into three categories (as shown in Table I): augmenting existing design automation tools in plausible ways (some have already been implemented in some systems), creating function libraries, and modest programmer training.

Table I
A Classification of Design Methods for HPRC

Type of support required	Methods supported
EDA: languages and synthesis	speed-match sequential functions
	high-performance memory access
	select optimal arithmetic precision
	create families of applications
	scale applications
Function/arithmetic libraries	select appropriate HW structures
	select appropriate arithmetic mode
Programmer/designer FPGA awareness	select appropriate computing mode
	select appropriate algorithm
	hide latency of independent functions
	optimize arithmetic with respect to operation cost
None	deal with Amdahl's law

Acknowledgments

This work was supported in part by the NIH through awards #R01 RR023168-01 and #R21 RR020209-01 by an IBM Faculty Award, and facilitated by donations from Altera Corporation, Xilinx Corporation, SGI, and XtremeData, Inc.

References

[1] Alam S., Agarwal P., Smith M., Vetter J., and Caliga D., 2007. Using FPGA devices to accelerate biomolecular simulations. *Computer*, **40**(3): 66–73.

[2] Altschul S., Gish W., Miller W., Myers E., and Lipman D., 1990. Basic local alignment search tool. *Journal of Molecular Biology*, **215,** 403–410.

[3] Azizi N., Kuon I., Egier A., Darabiha A., and Chow P., 2004. Reconfigurable molecular dynamics simulator. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 197–206.

[4] Baldi P., and Hatfield G., 2002. DNA Microarrays and Gene Expression. Cambridge University Press, Cambridge, UK.

[5] Benson G., 1999. Tandem repeats finder: A program to analyze DNA sequences. *Nucleic Acids Research*, **27**(2): 573–580.

[6] Bhatt A., 2007. Accelerating with many-cores and special purpose hardware. Keynote Talk, Field Programmable Logic and Applications.

[7] Bluethgen H.-M., and Noll T., 2000. A programmable processor for approximate string matching with high throughput rate. In *Proceedings of ASAP*, pp. 309–316.

[8] Borah M., Bajwa R., Hannenhalli S., and Irwin M., 1994. A SIMD solution to the sequence comparison problem on the MGAP. In *Proceedings of ASAP*, pp. 336–345.

[9] Briggs E., Sullivan D., and Bernholc J., 1996. Real-space multigrid-based approach to large-scale electronic structure calculations. *Physical Review B*, **54,** 14362–14375.

[10] Brooks B., Bruccoleri R., Olafson B., States D., Swaminathan S., and Karplus M., 1983. Charmm: Program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, **4,** 187–217.

[11] Buell D., July 2006. Reconfigurable systems. Keynote Talk, Reconfigurable Systems Summer Institute.

[12] Case D., Cheatham T., III, Darden T., Gohlke H., Luo R., Merz K., Jr., Onufriev A., Simmerling C., Wang B., and Woods R., 2005. The Amber biomolecular simulation programs. *Journal of Computational Chemistry*, **26,** 1668–1688.

[13] Chang C., 2004. BLAST Implementation on BEE2. Electrical Engineering and Computer Science. University of California, Berkeley.

[14] Chow E., Hunkapiller T., and Peterson J., 1991. Biological information signal processor. In *Proceedings of the International Conference on Application Specific Systems, Architectures, and Processors*, pp. 144–160.

[15] Conti A., VanCourt T., and Herbordt M., 2004. Flexible FPGA acceleration of dynamic programming string processing. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications.*

[16] Culler D., Singh J., and Gupta A., 1999. Parallel Computer Architecture: A Hardware/Software Approach. Morgan-Kaufmann, San Francisco, CA.

[17] D'Amour M., 2008. Reconfigurable computing for acceleration in HPC. *FPGA and Structured ASIC Journal*. http://www.fpgajournal.com/articles_2008/20080226_drc.htm.

[18] DeHon A., Adams J., DeLorimier M., Kapre N., Matsuda Y., Naeimi H., Vanier M., and Wrighton M., 2004. Design patterns for reconfigurable computing. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 13–23.

[19] Ding F., and Dokholyan N., 2005. Simple but predictive protein models. *Trends in Biotechnology*, **3**(9): 450–455.

[20] Dokholyan N., 2006. Studies of folding and misfolding using simplified models. *Current Opinion in Structural Biology*, **16,** 79–85.

[21] Dongarra J., et al., 2008. DARPA's HPCS program: History, models, tools, languages. In *Advances in Computers, v72: High Performance Computing*, M. Zelkowitz, ed. pp. 3–102. Academic Press, London.

[22] Durbin R., Eddy S., Krogh A., and Mitchison G., 1998. Biological Sequence Analysis. Cambridge University Press, Cambridge, UK.

[23] Dydel S., and Bala P., 2004. Large scale protein sequence alignment using FPGA reprogrammable logic devices. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*.

[24] Ginsberg M., 2008. Bibliographic snapshots of high performance/high productivity computing. In *Advances in Computers, v72: High Performance Computing*, M. Zelkowitz, ed. Academic Press, London.

[25] Gokhale M., Rickett C., Tripp J., Hsu C., and Scrofano R., 2006. Promises and pitfalls of reconfigurable supercomputing. In *Proceedings of the 2006 Conference on the Engineering of Reconfigurable Systems and Algorithms*, pp. 11–20.

[26] Gu Y., and Herbordt M., 2007. FPGA-based multigrid computations for molecular dynamics simulations. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 117–126.

[27] Gu Y., VanCourt T., and Herbordt M., 2006. Improved interpolation and system integration for FPGA-based molecular dynamics simulations. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 21–28.

[28] Gu Y., VanCourt T., and Herbordt M., 2008. Explicit design of FPGA-based coprocessors for short-range force computation in molecular dynamics simulations. *Parallel Computing*, **34**(4–5): 261–271.

[29] Guccione S., and Keller E., 2002. Gene matching using JBits. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 1168–1171.

[30] Gusfield D., 1997. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge, UK.

[31] Haile J., 1997. Molecular Dynamics Simulation. Wiley, New York, NY.

[32] Herbordt M., Model J., Sukhwani B., Gu Y., and VanCourt T., 2007. Single pass streaming BLAST on FPGAs. *Parallel Computing*, **33**(10–11): 741–756.

[33] Herbordt M., Gu Y., VanCourt T., Model J., Sukhwani B., and Chiu M., 2008. Computing models for FPGA-based accelerators with case studies in molecular modeling. *Computing in Science & Engineering*, **10**(6): 35–45.

[34] Herbordt M., Kosie F., and Model J., 2008. An efficient $O(1)$ priority queue for large FPGA-based discrete event simulations of molecular dynamics. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines*.

[35] Hoang D., 1993. Searching genetic databases on SPLASH 2. In *Proceedings of FCCM*, pp. 185–191.

[36] Holland B., Vacas M., Aggarwal V., DeVille R., Troxel I., and George A., 2005. Survey of C-based application mapping tools for reconfigurable computing. In *Proceedings of the 8th International Conference on Military and Aerospace Programmable Logic Devices*.

[37] Izaguirre J., Hampton S., and Matthey T., 2005. Parallel multigrid summation for the *n*-body problem. *Journal of Parallel and Distributed Computing*, **65,** 949–962.

[38] Jacob A., Lancaster J., Buhler J., and Chamberlain R., 2007. FPGA-accelerated seed generation in mercury BLASTP. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines.*

[39] Katchalski-Katzir E., Shariv I., Eisenstein M., Friesem A., Aflalo C., and Vakser I., 1992. Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, **89,** 2195–2199.

[40] Kim Y., Noh M.-J., Han T.-D., Kim S.-D., and Yang S.-B., 2001. Finding genes for cancer classification: Many genes and small number of samples. 2nd. In *Annual Houston Forum on Cancer Genomics and Informatics.*

[41] Kindratenko V., and Pointer D., 2006. A case study in porting a production scientific supercomputing application to a reconfigurable computer. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 13–22.

[42] Kitchen D., Decornez H., Furr J., and Bajorath J., 2004. Docking and scoring in virtual screening for drug discovery: Methods and applications. *Nature Reviews. Drug Discovery*, **3,** 935–949.

[43] Kohane I., Kho A., and Butte A., 2003. Microarrays for an Integrative Genomics. MIT Press, Cambridge, MA.

[44] Korf I., Yandell M., and Bedell J., 2003. BLAST: An Essential Guide to the Basic Local Alignment Search Tool. O'Reilly & Associates, Sebastopol, CA.

[45] Krishnamurthy P., Buhler J., Chamberlain R., Franklin M., Gyang K., and Lancaster J., 2004. Biosequence similarity search on the Mercury system. In *Proceedings of the International Conference on Application Specific Systems, Architectures, and Processors*, pp. 365–375.

[46] Landau G., Schmidt J., and Sokol D., 2001. An algorithm for approximate tandem repeats. *Journal of Computational Biology*, **8**(1): 1–18.

[47] Langhammer M., 2008. Floating point data path synthesis for FPGAs. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 355–360.

[48] Lawrie D., 1975. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, **C-24**(12): 1145–1155.

[49] Liptov R., and Lopresti D., 1986. Comparing long strings on a short systolic array. In *Systolic Arrays*, W. Moore, A. McCabe, and R. Uquhart, Eds. Adam Hilger, Boston, MA.

[50] Lopresti D., 1987. P-NAC: A systolic array for comparing nucleic acid sequences. *IEEE Computer*, **20**(7): 98–99.

[51] Matthey T., 2004. ProtoMol, an object-oriented framework for prototyping novel algorithms for molecular dynamics. *ACM Transactions on Mathematical Software*, **30**(3): 237–265.

[52] McCallum I., 2007. Platform-Level Services for Accelerators: Intel QuickAssist Technology Accelerator Abstraction Layer (AAL). Intel Corporation, Santa Clara, CA.

[53] Model J., and Herbordt M., 2007. Discrete event simulation of molecular dynamics with configurable logic. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 151–158. http://www.fpgajournal.com/articles_2007/20070710_cots.htm.

[54] Morris K., 2007. COTS supercomputing. *FPGA and Structured ASIC Journal*. http://www.fpgajournal.com/articles_2007/20070710_cots.htm.

[55] Muriki K., Underwood K., and Sass R., 2005. RC-BLAST: Towards an open source hardware implementation. In *Proceedings of the International Workshop on High Performance Computational Biology.*

[56] Perou C., et al., 2003. Prediction of clinical outcome with microarray data: A partial least squares discriminant analysis (PLS-DA) approach. *Human Genetics*, **112,** 581–592.

[57] Pournara I., Bouganis S.-S., and Constantinides G., 2005. FPGA-accelerated Bayesian learning for reconstruction of gene regulatory networks. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications.*

[58] Rapaport D., 2004. The Art of Molecular Dynamics Simulation. Cambridge University Press, Cambridge, UK.

[59] Roberts L., 1989. New chip may speed genome analysis. *Science*, **244**(4905): 655–656.

[60] Sagui C., and Darden T., 2001. Multigrid methods for classical molecular dynamics simulations of biomolecules. *Journal of Chemical Physics*, **114,** 6578–6591.

[61] Scrofano R., Gokhale M., Trouw F., and Prasanna V., 2006. A hardware/software approach to molecular dynamics on reconfigurable computers. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 23–32.

[62] Skeel R., Tezcan I., and Hardy D., 2002. Multiple grid methods for classical molecular dynamics. *Journal of Computational Chemistry*, **23,** 673–684.

[63] Smith M., Alam S., Agarwal P., Vetter J., and Caliga D., 2006. A task-based development model for accelerating large-scale scientific applications on FPGA-based reconfigurable computing platforms. In *Proceedings of the Reconfigurable Systems Summer Institute*.

[64] Snyder L., 1986. Type architectures, shared memory, and the corollary of modest potential. *Annual Review of Computer Science*, **1,** 289–317.

[65] Sukhwani B., and Herbordt M., 2008. Acceleration of a production rigid molecule docking code. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 341–346.

[66] Swartzlander E., 1987. Systolic Signal Processing Systems. Marcel Dekker, Inc.New York, NY.

[67] Urban K., June 2008. In-socket accelerators: When to use them. *HPC Wire*, http://www.hpcwire.com.

[68] VanCourt T., 2006. LAMP: Tools for Creating Application-Specific FPGA Coprocessors. Ph.D. Thesis, Department of Electrical and Computer Engineering, Boston University.

[69] VanCourt T., and Herbordt M., 2005. LAMP: A tool suite for families of FPGA-based application accelerators. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications.*

[70] VanCourt T., and Herbordt M., 2006. Application-dependent memory interleaving enables high performance in FPGA-based grid computations. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 395–401.

[71] VanCourt T., and Herbordt M., 2006. Rigid molecule docking: FPGA reconfiguration for alternative force laws. *Journal on Applied Signal Processing*, **v2006,** 1–10.

[72] VanCourt T., and Herbordt M., 2006. Sizing of processing arrays for FPGA-based computation. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 755–760.

[73] VanCourt T., and Herbordt M., 2007. Families of FPGA-based accelerators for approximate string matching. *Microprocessors and Microsystems*, **31**(2): 135–145.

[74] VanCourt T., Gu Y., and Herbordt M., 2004. FPGA acceleration of rigid molecule interactions. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications.*

[75] VanCourt T., Herbordt M., and Barton R., 2004. Microarray data analysis using an FPGA-based coprocessor. *Microprocessors and Microsystems*, **28**(4): 213–222.

[76] Villareal J., Cortes J., and Najjar W., 2007. Compiled code acceleration of NAMD on FPGAs. In *Proceedings of the Reconfigurable Systems Summer Institute*.

[77] Wit E., and McClure J., 2004. Statistics for Microarrays. Wiley, New York.

[78] XtremeData, Inc., 2007. XD1000 Development System. http://www.xtremedata.com.

[79] Yavneh I., 2006. Why multigrid methods are so efficient. *Computing in Science & Engineering*, **8,** 12–22.

[80] Yu C., Kwong K., Lee K., and Leong P., 2003. A Smith–Waterman systolic cell. In *Proceedings of the IEEE Conference on Field Programmable Logic and Applications*, pp. 375–384.

Models and Metrics for Energy-Efficient Computing

PARTHASARATHY RANGANATHAN

Hewlett-Packard Labs

SUZANNE RIVOIRE

Sonoma State University

JUSTIN MOORE

Google

Abstract

Over the past decade, however, power and energy have begun to severely constrain component, system, and data center designs. When a data center reaches its maximum provisioned power, it must be replaced or augmented at great expense. In desktops, power consumption and heat contribute to electricity costs as well as noise. Better equipment design and better energy management policies are needed to address these concerns. This chapter will detail current efforts in energy-efficiency metrics and in power and thermal modeling, delving into specific case studies for each. Various benchmarks are explained and their effectiveness at measuring power requirements is discussed.

ADVANCES IN COMPUTERS, VOL. 75
ISSN: 0065-2458/DOI: 10.1016/S0065-2458(08)00803-6

1. Introduction

Energy efficiency in computing has historically improved much more slowly than performance or cost [8]. Over the past decade, however, power and energy have begun to severely constrain component, system, and data center designs. In data centers, power and energy consumption are becoming pressing concerns. In the United States, the power consumed by volume servers doubled between 2000 and 2006, and the Environmental Protection Agency predicts that it will double again between 2006 and 2011 [97]. Furthermore, the cooling infrastructure in a data center consumes half a Watt to a Watt of power for every Watt consumed by the computing equipment. By 2009, according to the Uptime Institute, the 3-year energy cost of a server will exceed its purchase cost [12].

In addition to cost, the scalability of data centers is affected by power consumption and heat dissipation. The power density of servers and racks has increased dramatically over time, meaning that data centers may exceed their power budgets with a small number of servers and with plenty of floor space available. When a data center reaches its maximum provisioned power, it must be replaced or augmented at great expense. The Gartner firm estimates that half of the world's data centers will reach this point by the end of 2008 [39].

Finally, data center power consumption has implications for local utility companies and for the environment. Data centers are contributors to electrical transmission congestion in several major American cities, including the New York and Los Angeles metropolitan areas [97]. Generating this electricity also results in increased greenhouse gas emissions.

In the mobile and desktop domains, energy efficiency is also an important issue. In laptop computers, processor power is increasing much more rapidly than battery capacity, requiring advanced power management solutions to maintain usability. In desktops, power consumption and heat contribute to electricity costs as well as noise. Better equipment design and better energy management policies are needed to address these concerns.

Underlying almost any energy-efficiency optimization are a *metric* and a *model*. Metrics are needed to evaluate proposed solutions, whether those solutions involve designing individual components, assembling complete systems or data centers, or dynamically adjusting a component or system's power consumption. These metrics should correlate strongly with the concerns of end users, while also being understandable, general, and practical to calculate.

Models are needed for several distinct purposes. In the initial design of systems and components, it is impractical to build working prototypes of every possible idea or point in the design space; therefore, models of power, heat, or other metrics of interest play a crucial role in evaluating potential designs. In the day-to-day operation of components, systems, and data centers, models are necessary to predict the effects of policy decisions such as putting a processor in a low-power state or migrating workloads to avoid thermal failures in a data center. Insight into how resource usage in a system affects its power and thermal properties is indispensable in energy-aware scheduling and can only come from models.

This chapter will detail current efforts in energy-efficiency metrics and in power and thermal modeling, delving into specific case studies for each. Section 2 describes the challenges of developing energy-efficiency metrics and summarizes previously proposed benchmarks and metrics. Section 3 presents the JouleSort energy-efficiency benchmark in more detail, explaining both the benchmark specification and the insights into energy-efficient system design that this benchmark provides.

The remainder of the chapter deals with power and thermal modeling. Section 4 is an overview of various approaches to power and thermal modeling, while Section 5 presents Mantis, a framework for generating and evaluating a family of high-level generic full-system power models. Section 6 details a suite of tools for data center thermal modeling and optimization: Splice, an infrastructure for aggregating sensor data; ConSil, a platform for inferring inlet temperature data from temperature sensors internal to a server; and Weatherman, a method of generating the thermal topology of a data center. Finally, Section 7 discusses future challenges in models and metrics for energy efficiency and concludes the chapter.

2. Energy-Efficiency Metrics

Energy-conscious users, as well as computer manufacturers and researchers, need to be able to assess and compare the energy efficiency of computer systems to make purchasing decisions or to identify promising ideas and technologies. Well-defined benchmarks are needed to provide standardized and fair comparisons of computers' energy efficiency. While many benchmarks exist to assess performance, benchmarks for energy efficiency are just beginning to be developed. This section addresses some of the challenges of benchmarking in general and energy-efficiency benchmarking in particular, and then it reviews current energy-aware benchmarks and standalone metrics. Section 2.1 presents a detailed case study of the JouleSort benchmark, which was the first full-system energy-efficiency benchmark to be proposed [79, 80].

2.1 Benchmarking Challenges

A complete benchmark specifies three things: a *workload* to run, which should represent some real-world task of interest; a *metric* or score to compare different systems; and operational *rules* to ensure that the benchmark runs under realistic conditions. Creating a benchmark for energy efficiency shares some challenges with creating any benchmark. There is the question of what to benchmark, for example, components, single machines, or data centers, as well as the question of what application domain(s) to represent. Benchmarks exist for almost every conceivable class of workload and for every class of machine, from small embedded processors [22] to large clusters [94] and supercomputers [73].

There is also the question of workload choice: not only what program(s) to run, but also how many. A suite of programs may provide better coverage of a domain of interest than a single program, but the metric becomes less straightforward and the results become more difficult to interpret. The choice of workload also effectively determines the class of machines to which the benchmark can be applied; for example, a supercomputing benchmark probably could not run on handheld devices, and would not be a representative workload if it could.

The benchmark metric must also be determined. Even when the goal is to measure pure performance, the decision of whether to use a metric based on the time to execute a fixed-size workload or the throughput in a fixed amount of time can bias the metric toward certain types of machines or exclude them entirely. When the goal is to balance performance with another concern, such as cost, the question of how to weigh and combine the two metrics in the final benchmark score adds another element of complexity.

Finally, the benchmark specification must include rules to ensure that the benchmark runs under fair and realistic conditions. For performance-oriented benchmarks, these rules often constrain the types of compiler optimizations that can be applied to the benchmark source code to preclude benchmark-specific compiler optimizations of dubious general correctness. The rules may also constrain the type of hardware, operating system, or file system on which the benchmark is run.

In addition to these typical concerns, benchmarking energy efficiency presents some unique challenges. The choice of workload is complicated by the fact that the desired operating point(s) of the system must be identified; in particular, since lightly utilized systems are currently highly inefficient [6], benchmark designers may want to target or avoid this operating point. The choice of metric also becomes more complex, since it must resolve the question of how to weigh performance against power consumption. However, the benchmark rules are the largest source of increased complexity.

First, the definition of the system and its environment becomes more complex. The ambient temperature around the system affects its power consumption, so benchmark designers may want to regulate it. They also must decide whether or not to include the cooling systems of both the machine and the building housing it, a significant decision since cooling can consume up to 1 W for each Watt of power consumed by the computing equipment [71]. Additionally, energy-efficiency benchmarks require standards to govern the accuracy and sampling rates of the power and temperature instrumentation.

2.2 Current Energy-Efficiency Metrics

A variety of standalone metrics and a few full benchmarks for energy efficiency have emerged in the past few years. This section describes some of these previously proposed energy-efficiency benchmarks and metrics. Table I shows the target system classes of these metrics, and Table II summarizes their specifications.

TABLE I
SUMMARY OF THE TARGET DOMAINS OF DIFFERENT ENERGY-EFFICIENCY BENCHMARKS AND METRICS

Benchmark	Level	Domain
EnergyBench	Processor	Embedded
SWaP	System(s)	Enterprise
Energy Star certification	System	Mobile, desktop, enterprise
SPECpower_ssj	System	Enterprise
Compute power efficiency	Data center	Enterprise
Green Grid metrics	Data center	Enterprise

TABLE II
SUMMARY OF THE SPECIFICATIONS OF DIFFERENT ENERGY-EFFICIENCY BENCHMARKS AND METRICS

Benchmark	Workload	Metric
SWaP	Unspecified	Performance/(space $\times$ watts)
EnergyBench	EEMBC benchmarks	Throughput/Joule
Energy Star: workstations	Sleep, idle, standby, Linpack, SPECviewperf	Certify if ''typical'' power $<$ 35% of max. power
Energy Star: other systems	Sleep, idle, standby modes	Certify if each mode's power $<$ predefined threshold
SPECpower_ssj	Server-side Java under varying loads	Operations/watt averaged over all loads
Green Grid DCD	Unspecified	Equipment power/floor area (kW/ft^2)
Green Grid DCiE	Unspecified	% of facility power reaching IT equipment
Compute power efficiency	Unspecified	IT equipment util. $\times$ DCiE
Green Grid DCeP	Unspecified	Work done/facility power

2.2.1 Component-Level Benchmarks and Metrics

Several standalone metrics have been proposed at the processor level. In 1996, Gonzalez and Horowitz [28] advocated using the energy-delay product to compare processor designs. The rationale was that chips' performance was directly proportional to clock frequency, while power consumption increased as the square of frequency. Therefore, decreasing a processor's clock frequency by a factor of x would result in performance degradation proportional to x and a decrease in power consumption proportional to x^2. Since energy is the product of execution time and average power, the net effect would be an energy decrease by a factor of x. Comparing processors based on energy would therefore motivate processor designers to arbitrarily reduce clock frequency and sacrifice performance.

The energy-delay product, which weighs power against the square of execution time, would show the underlying design's energy efficiency, independent of clock frequency. This metric is a specific case of the $MIPS^{\gamma}$ per Watt metric [103], where the choice of γ reflects the desired balance between performance and power. In any case, these metrics are focused on the processor and do not provide a suggested workload.

For embedded processors, the Embedded Microprocessor Benchmark Consortium (EEMBC) developed the EnergyBench benchmarks [23]. EnergyBench consists of a standardized data acquisition infrastructure that allows processor power to be measured; the idea is to measure power while running one of EEMBC's performance benchmarks. The EnergyBench metric is ''Netmarks per Joule'' for networking benchmarks and ''Telemarks per Joule'' for telecommunications benchmarks. This benchmark is focused solely on the processor and on the embedded domain.

2.2.2 System-Level Benchmarks and Metrics

Several metrics and benchmarks have been proposed at the single-system level. Performance per Watt is a common standalone metric for servers [49]. Performance is typically specified using either MIPS or the rating from peak-performance benchmarks like SPECint [88] or TPC-C [94]. Since both energy and space efficiency are important in data centers, Sun Microsystems has proposed the SWaP (Space, Watts, and Performance) metric to reflect power density as well as power consumption [92].

In addition to these standalone metrics, complete system-level energy-efficiency benchmarks are now emerging. These include the United States government's Energy Star certification guidelines for computers and the SPECpower_ssj benchmark, as well as the JouleSort benchmark, which the next section will discuss in detail.

Energy Star is a U.S. government certification for highly energy-efficient household products, including computers [24]. For desktops, desktop-derived servers, notebooks, and game consoles, the Energy Star certification is awarded to systems with idle, sleep, and standby power consumptions below certain specified thresholds. For workstations, however, Energy Star certification requires that the ''typical'' power (a weighted function of the idle, sleep, and standby power consumptions) not exceed 35% of the ''maximum power'' (the power consumed during the Linpack and SPECviewperf benchmarks, plus a factor based on the number of installed hard disks). Energy Star certification also requires that a system's power supply efficiency exceed 80%. Energy Star certification thus depends mainly on a system's low-power states and does not include a measure of the system's performance. Its metric is coarse-grained: a system is either certified, or it is not.

The SPECpower_ssj benchmark, released in December 2007, is designed to assess the energy efficiency of servers under a wide variety of loads [89]. Data center servers usually operate far below peak utilization, which creates inefficiencies, since peak utilization is the most efficient operating point for modern servers [6]. Therefore, SPECpower_ssj uses a CPU-intensive server-side Java workload and scales it to run at 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and peak utilization. The SPECpower_ssj score is the overall number of operations per Watt across all of these utilization modes. The benchmark also specifies a minimum ambient temperature and standards for the power and temperature sensors used to collect the data. This benchmark is CPU- and memory-centric, and both its workload and metric are tailored to the data center domain.

2.2.3 Data Center-Level Benchmarks and Metrics

Many metrics have been proposed to quantify various aspects of data center energy efficiency, from the building's power and cooling provisioning to the utilization of the computing equipment. The Uptime Institute identified a variety of metrics contributing to data center ''greenness,'' including measures of power conversion efficiency at the server and data center levels, as well as the utilization efficiency of the deployed hardware [90]. To optimize data center cooling, Chandrakant Patel and others have advocated a metric based on performance per unit of *exergy* destroyed [68]. Exergy is the available energy in a thermodynamic sense, and so exergy-aware metrics take into account the conversion of energy into different forms. In particular, exergy is expended when electrical power is converted to heat and when heat is transported across thermal resistances.

The Green Grid, an industrial consortium including most major hardware vendors, has proposed several metrics to quantify data center power efficiency over both space and time. To quantify space efficiency, they define the data center density (DCD) metric as the ratio of the power consumed by all equipment on the raised floor to the area of the raised floor, in units of kW/ft^2 [31]. To quantify time efficiency (i.e., energy efficiency), they propose the data center infrastructure efficiency (DCiE) metric [30]. DCiE is defined as the percentage of the total facility power that goes to the ''IT equipment'' (primarily compute, storage, and network). Since IT equipment power is not necessarily a proxy for performance, two extensions of this metric have been proposed. Compute power efficiency (CPE), proposed by Malone and Belady [53], scales the DCiE by the IT equipment utilization, a value between 0 and 1. With this metric, the power consumed by idle servers counts as overhead rather than as power that is being productively used. Similarly, the Green Grid has introduced the data center energy productivity metric (DCeP), which is the useful work divided by the total facility power [32]. This metric can be applied to any data center workload. None of these data center metrics specifies a workload, and most do not take any measure of performance into account.

2.3 Summary

The field of energy-efficiency benchmarking is still in its infancy. While benchmarks and standalone metrics have been proposed for contexts from embedded processors to underutilized servers to entire data centers, energy-efficiency metrics for many important computing domains have not been methodically addressed. Completely specified full-system benchmarks for energy efficiency did not exist until 2007, when JouleSort and SPECpower_ssj were proposed. Section 3 presents the details of the JouleSort energy-efficiency benchmark, which was the first completely specified full-system energy efficiency benchmark, and which remains the only energy-efficiency benchmark for data-intensive computing.

3. Case Study: The JouleSort Benchmark

JouleSort, initially presented in [79], is an I/O-centric, system-level energy-efficiency benchmark that incorporates performance, power, and some cooling costs. The benchmark is designed to be balanced, portable, representative, inclusive, and simple. It can be used to compare different existing systems, to evaluate the energy-efficiency balance of components within a given system, and to evaluate different algorithms that use these components. These features make it possible to chart past trends in energy efficiency and can help to predict future trends.

This section presents the JouleSort benchmark design goals and specification, followed by an analysis of the energy efficiency of the historical winners of performance- and cost performance-oriented sort benchmarks. The lessons learned in this evaluation are used to design the CoolSort machine, a fileserver built from low-power mobile components, which is over 3.5 times more energy-efficient than any previous sort benchmark winners. Finally, the JouleSort benchmark's metric is compared to metrics weighing different combinations of performance, price, and power. This analysis shows that systems designed around the JouleSort metric also perform well when cost and performance are weighted more heavily than in the JouleSort metric.

3.1 Energy-Efficiency Benchmark Goals

The JouleSort benchmark was created with the goal of providing a fully specified energy-efficiency benchmark to identify trends and inspire improvements in energy efficiency. This section describes the design criteria that the JouleSort specification seeks to balance.

Power-Performance Tradeoff. The benchmark's metric should capture a system's performance as well as some measure of power use. Peak or average power would be an impractical metric, since neither includes a measurement of performance; the benchmark should not reward a system that consumes almost no power and completes almost no work. We chose energy, which weighs performance and power equally, as the metric; we prefer it to the energy-delay product, which privileges performance, because many performance benchmarks already exist and JouleSort should therefore emphasize power consumption.

Peak Efficiency. Most computer systems are most energy-efficient at peak utilization [6] and less efficient at low utilization. For simplicity of benchmarking and clarity of the benchmark score, our benchmark does not specify an operating point and will therefore likely be run at the most energy-efficient operating point. The peak utilization point influences design and provisioning constraints for data centers as well as mobile devices. Furthermore, peak utilization is the most common operating point in some computing domains, such as enterprise environments that use server consolidation to improve energy efficiency, as well as scientific computing.

Holistic and Balanced. A full-system benchmark should reflect the performance and power characteristics of all core components, showing how an improvement in a single component translates to end-to-end energy-efficiency gains. The workload and metric should therefore exercise and measure all core components.

Inclusive and Portable. The benchmark should ideally be able to assess the energy efficiencies of a wide variety of systems, from PDAs to supercomputers. The workload, metric, and rules should be meaningful and unbiased for all of these platforms.

History Proof. The benchmark should remain relevant as technologies evolve, and the metric should allow meaningful comparisons across different generations of systems.

Representative. The benchmark's workload should represent an important class of workloads for the systems being benchmarked.

Simple. The benchmark should be as simple as possible to set up and administer, and the score should be easy to understand.

3.2 Benchmark Definition

3.2.1 Workload

JouleSort's workload is external sort, as specified by the Sort Benchmark Web site [87]. External sort has been used by the database community since 1985 [2], and researchers have used it to understand the performance and cost performance of system-level effectiveness of algorithm and component improvements, as well as to identify promising technology trends. Previous sort benchmark winners have foreshadowed the transition from supercomputers to shared-memory multiprocessors to commodity clusters, and have recently demonstrated the promise of general-purpose computation on graphics processing units (GPUs) [29]. The sort benchmarks have historically been used as a bellwether to illuminate the potential of new technologies, rather than to guide purchasing decisions.

The sort benchmarks' workload can be summarized as follows: sort a file consisting of randomly permuted 100-byte records with 10-byte keys. The input file must be read from, and the output file written to, external nonvolatile storage. The output file must be newly created rather than reusing the input file, and all intermediate files used by the sort program must be deleted.

This workload is *representative* because most platforms must manage an ever-increasing supply of data [52] and thus all perform some type of I/O-centric task. Since the sort benchmarks have been implemented on clusters, supercomputers, multiprocessors, and personal computers [87], sort is *portable* and *inclusive*. It is a *simple* workload to understand and implement. It is also *holistic* and *balanced*, stressing the core components of I/O, memory, and the CPU, as well as the interfaces that connect them. Finally, the sort workload is relatively *history proof*. While the size of the data set has changed over time, the fundamental sorting task has been the same since the original sort benchmark was proposed in 1985 [2].

3.2.2 Metric

The benchmark's metric must be a fair comparison across systems and must be free of loopholes that obviate the benchmark's intent. Since the JouleSort benchmark's metric should give power and performance equal weight (see Section 3.1), there are three ways to define the benchmark score:

- Within a fixed energy budget, compare systems based on the number of records sorted.
- Within a fixed time budget, compare systems based on the ratio of records sorted to energy consumed, expressed in sorted records per Joule.
- Using a fixed workload size, compare systems based on the amount of energy consumed while sorting.

This section examines these three possibilities in detail and explains the decision to choose a fixed workload size for JouleSort.

Fixed Energy Budget. Using a fixed energy budget is an intuitive extension of the current sort benchmarks. However, since the power consumption of current platforms varies by orders of magnitude, this approach would require many benchmark classes with different energy budgets appropriate to different classes of systems, and these classes would need to be updated as technology changes.

This decision would limit the benchmark's ability to be *inclusive* and *history proof*. Furthermore, this metric makes administering the benchmark more complicated: since energy is the product of power and time, it is susceptible to variations in both quantities, making it difficult to maximize the budget.

Fixed Time Budget. Specifying a fixed time budget for the sort is a superficially attractive idea because it does not require separate categories for different classes of systems and would not need to change with technology. However, two issues prevent it from being a fair metric of comparison. These issues are illustrated in Fig. 1. This figure shows the benchmark score in sorted records per Joule for varying input sizes (N) evaluated on the winning JouleSort system, which is described in detail in Section 3.4.

As the figure shows, the benchmark score varies considerably with N. The initial steep climb in energy efficiency at the leftmost data points occurs because the smallest data sets take only a few seconds to sort and thus poorly amortize the startup overhead of the sorting program. As the data sets grow larger, this overhead is better amortized and energy efficiency increases, up to a data set of 15 million records. This is the largest data set that can be sorted completely in this machine's memory. For larger data sets, the system cannot perform the entire sort in memory

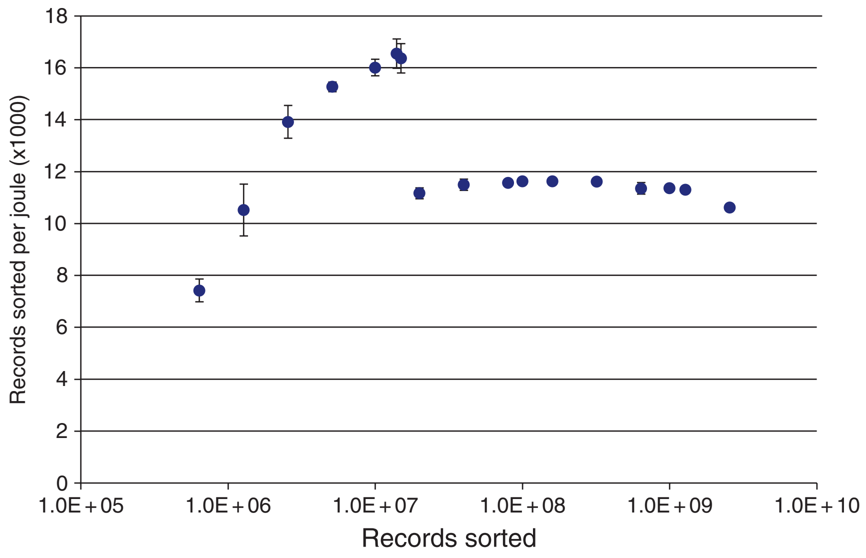


FIG. 1. The measured energy efficiency of the current JouleSort-winning system at varying input sizes.

and must temporarily write data to disk, necessitating a second pass over the data that doubles the amount of I/O and dramatically decreases the performance per record. After this transition, energy efficiency stays relatively constant as N grows, eventually trending slowly downward.

The first problem with this metric occurs at the transition from one-pass to two-pass sorts. To maximize benchmark scores, systems will continue sorting only if the marginal energy cost of sorting an additional record is lower than the cost of sleeping for the remaining time. For this system, the best strategy with a time budget of 1 min would be to sort 1.5×10^7 records, which takes 10 s, and sleep in a low-power state for the remaining 50 s. Benchmarking a sleeping system violates the benchmark design goal of balancing power and performance.

The second problem with this metric is due to the $(N \log N)$ algorithmic complexity of sort, which causes the downward trend for larger N. Total energy is a complex function of many performance factors that vary with N: the amount of I/O, the number of memory accesses, the number of comparison operations, CPU utilization, and the amount of parallelism. Figure 1 shows that once the sort becomes CPU-bound ($N \geq 8 \times 10^7$ records), the sorted records per Joule score trends slowly downward as total energy increases superlinearly with N. This decrease occurs in part because the number of comparisons done in sorting is $O(N \log N)$, and the constants and lower-order overheads hidden by the O-notation are no longer obscured when N is sufficiently large. This effect implies that the metric is biased toward systems that sort fewer records in the allotted time. That is, if two fully utilized systems A and B have the same energy efficiency for a fixed number of records, and A can sort twice as many records as B in a minute, the metric of sorted records per Joule will unfairly favor B.

Fixed Input Size. In light of the problems with metrics based on a fixed energy budget or a fixed time budget, a metric based on fixed input size was chosen for the benchmark. This decision necessitates multiple benchmark classes, similar to the TPC-H benchmark's scale factors [95], since different workload sizes are appropriate for different classes of systems. Three JouleSort classes were chosen, with data set sizes of 100 million records (about 10 GB), 1 billion records (about 100 GB), and 10 billion records (about 1 TB). For consistency, MB, GB, and TB will henceforth be used to denote 10^6, 10^9, and 10^{12} bytes, respectively.

JouleSort's metric of comparison then becomes the energy to sort a fixed number of records, which is equivalent to the number of records sorted per Joule when the number of records is held constant. The latter metric is preferred for two reasons: first, it makes the power/performance balance more clear, and second, it allows rough comparisons across different benchmark classes, with the caveats described in the discussion of the fixed time budget.

One disadvantage of the fixed time budget is that as technologies improve, benchmark classes may need to be added at the higher end and deprecated at the lower end. Since comparisons across benchmark classes are not perfectly fair, this approach is not fully *history proof*. However, since even the best-performing sorts are improving more slowly than the Moore's law rate, these benchmark classes should be reasonable for at least 5 years.

3.2.3 Measuring Energy

The most important rules for energy measurement govern the boundaries of the system to be measured, constraints on the ambient environment, and acceptable methods of measuring power consumption. First, we specify that the energy measurements must capture all energy consumed by the physical system executing the sort. Power must be measured from the wall, and any change in the system's potential energy (e.g., using batteries) must be accounted for.

The next question is which cooling devices to account for; air conditioners and other cooling devices consume significant energy in data centers, but it would be unfair to include air conditioners in the energy measurement for a desktop. Therefore, the only cooling costs included in the JouleSort metric are those incurred by devices directly attached to the system being benchmarked. To provide fair comparisons between systems, the benchmark requires an ambient temperature between 20 and 25 °C to be maintained at the system's inlets.

Finally, the power meter used for the measurement must meet the accuracy requirements defined by SPECpower_ssj [89], and three consecutive readings must be reported to account for noise.

3.3 Energy Efficiency of Past Sort Benchmark Winners

This section estimates the energy efficiency of previous sort benchmark winners and examines the question of whether any of the existing sort benchmarks can serve as a surrogate for an energy-efficiency benchmark.

Although previous sort benchmark winners were not configured with power consumption in mind, they roughly reflect the power characteristics of desktop and higher-end systems in their day. Figure 2, which compares the energy efficiency in sorted records per Joule of previous sort benchmark winners, supports a few qualitative observations about the relative improvements in performance, price–performance, and energy efficiency over the last decade.

First, the PennySort winners, which were optimized for price–performance, are clearly more energy-efficient than the winners of the MinuteSort and Terabyte Sort benchmarks, which were optimized for pure performance. There are two reasons for

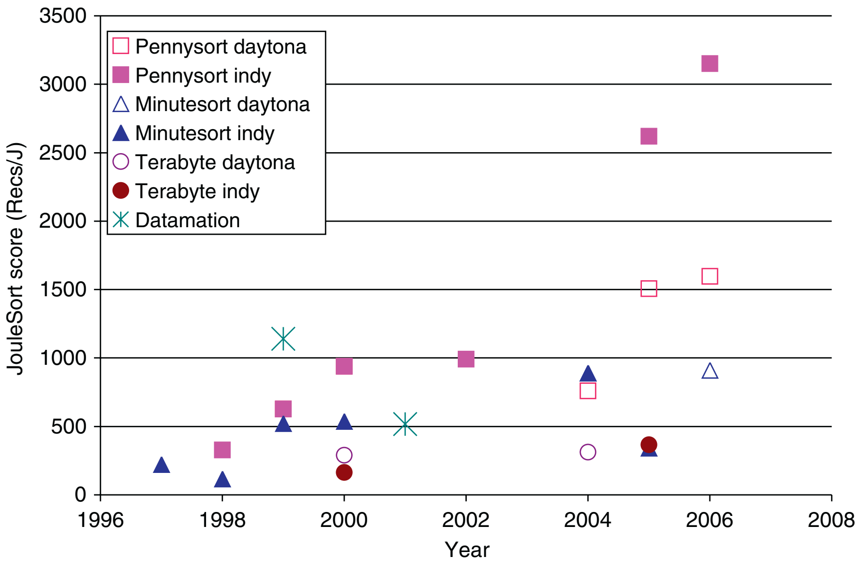


FIG. 2. Estimated energy efficiency of previous winners of sort benchmarks.

this effect. First, the price–performance metric provides an incentive for system designers to use fewer components, and thus less power. Second, it rewards the use of cheaper commodity components, which, for a given performance point, traditionally have used less energy than expensive, high-performance components.

In addition to being more energy-efficient than performance-optimized systems, cost-conscious systems have also shown much more improvement in energy efficiency over time. This lack of energy-efficiency improvement over time for cluster hardware was also noted by Barroso in 2005 [5].

Much of the energy-efficiency improvement among the PennySort winners is due to the last two winners in the Indy category. The 2005 winner, Sheenk Sort [100], benefited from algorithmic improvements and a minimal hardware configuration—but most importantly, trends in CPU design had finally swung toward energy efficiency. The processor used in Sheenk Sort had six times the clock frequency of the processor used by the previous PennySort winner, while only consuming twice the power. Overall, Sheenk Sort had triple the performance of the previous winner, while consuming only twice the power.

The 2006 PennySort winner, GPUTeraSort [29], improved upon its predecessors' energy efficiency by introducing a GPU to provide high streaming memory bandwidth. The chosen GPU, the NVidia 7800 GT, is comparable in estimated power consumption (57 W) to the system's (80 W).

TABLE III
ESTIMATED YEARLY IMPROVEMENT IN PURE PERFORMANCE (SRECS/S), PRICE–PERFORMANCE (SRECS/$), AND ENERGY EFFICIENCY (SRECS/J) OF PAST SORT BENCHMARK WINNERS

Benchmark	SRecs/s	SRecs/$	SRecs/J
PennySort	51%/year	58%/year	25%/year
Performance sorts	38%/year	N/A	13%/year

Performance sorts include MinuteSort, Terabyte Sort, and *Datamation* Sort.

Table III compares the growth rates of previous sort benchmark winners along three dimensions: performance (sorted records per second), price–performance (sorted records per dollar), and energy efficiency (estimated sorted records per Joule). The benchmarks are divided into two categories according to the benchmark's goal: price–performance and pure performance. Table III shows that PennySort systems are improving almost at the pace of Moore's law along the performance and price–performance dimensions. The pure-performance systems, however, are improving much more slowly.

This analysis also shows much slower growth in estimated energy efficiency than in the other two metrics for both benchmark categories. Therefore, either energy efficiency is improving much more slowly than the other metrics, or the current benchmarks are not capturing the most energy-efficient systems. The 2006 PennySort winner sorts an estimated 3200 records per Joule, but the 2007 JouleSort winner sorts 11,600 records per Joule (see Section 3.4), rather than the 4000 expected from extrapolating the yearly trends. This result suggests that a benchmark focused on energy efficiency is necessary to track trends and to promote development of energy-efficient systems and technologies, independent of cost considerations.

3.4 Design of the JouleSort Winner

After estimating the energy efficiency of previous sort benchmark winners and experimentally evaluating the benchmark on commodity hardware, the next goal was to create an energy-efficient machine that convincingly overtook the other measured and estimated systems. For simplicity, the system was composed of commercially available components, and Nsort was used as the software. The strategy for building this machine was to create a balanced sorting system out of low-power components. The manufacturer specifications of a variety of low-power x86 processors and mobile disks were examined to estimate the sorting efficiency of potential systems, resulting in the configuration shown in Table IV. This machine, nicknamed CoolSort, is over 3.5 times more energy-efficient than any of the previously measured or estimated systems.

This system uses a high-end mobile CPU with five frequency states and a TDP of 34 W for the highest frequency state. The choice of motherboard was somewhat constrained; few boards support both a mobile CPU and enough I/O bandwidth to achieve a balanced sort. The chosen motherboard, the Asus N4L-VM DH, has two SATA connectors on the motherboard and two PCI-Express slots: one 1-channel and one 16-channel. To fill those slots, CoolSort uses two RAID controllers, one of which holds four disk drives and one of which holds eight. Connected to those controllers and to the motherboard are 13 low-power SATA laptop disks. According to their manufacturer specifications, their average seek time is approximately 11 ms [36], and their sequential bandwidth through the XFS file system was measured to be 45 MB/s in experiments with CoolSort. The manufacturer specifications list an average power consumption of 1.8 W when reading and writing and 0.85 W in the active idle state (idle but not sleeping) [36]. For memory, CoolSort uses two 1 GB DIMMs that each consume 1.9 W of power, according to the manufacturer specifications [46] (Table V).

TABLE IV
COMPONENTS OF THE COOLSORT MACHINE AND THEIR RETAIL PRICES AT THE TIME OF PURCHASE

Component	Model	Price ($)	Power (W)
CPU	Intel Core 2 Duo T7600	639.99	34 (TDP)
Motherboard	Asus N4L-VM DH	108.99	
Case/PSU	APEVIA X-Navigator ATXA9N-BK/500	94.99	
Eight-disk RAID card	HighPoint Rocket RAID 2320	249.99	9.5
Four-disk RAID card	HighPoint Rocket RAID 2300	119.99	2.0
Memory (2)	Kingston 1 GB DDR2 667	63.99	1.9 W (spec)
Disk (13)	Hitachi TravelStar 5K160, 5400 rpm, 160 GB	119.99	Active: 1.8 Idle: 0.85
Adapters		130.25	
Total		3032.05	Active: 100 Idle: 59

TABLE V
POWER AND PERFORMANCE OF COOLSORT SYSTEM

Recs	SRecs/J	Energy (kJ)	Power (W)	Time (s)
10^8	11,628 ± 41	8.6 ± 0.03	99.3 ± 0.2	86.6 ± 0.4
10^9	11,354 ± 29	88.1 ± 0.23	100.0 ± 0.1	880.8 ± 1.5

The optimal configuration uses 13 disks because the PCI-e RAID cards hold a maximum total of 12 disks, and the I/O performance of the motherboard controller with more than one disk is poor. The input and output files are striped across a six-disk array configured via LVM2, and the remaining seven disks are independent for the temporary data. In the idle state at the lowest CPU frequency, this system consumes 59.0 ± 1.3 W of power.

3.5 Other Energy-Efficiency Metrics

Although JouleSort addresses a computer system's energy efficiency, energy is just one piece of the system's total cost of ownership (TCO). From a system purchaser's perspective, a TCO-Sort would be the most desirable sort benchmark; however, the components of TCO vary widely from user to user. Combining JouleSort and PennySort to benchmark the costs of purchasing and powering a system is a possible first step, although the current sort benchmark metrics omit reliability, manageability, and security issues. This section examines alternative possible metrics for an energy-aware sort benchmark using various weighings of price, performance, and power, and compares the types of systems that each favors. Because these systems were benchmarked using different data set sizes, comparisons between them are not precise.

The machines compared in this section are the historical sort benchmark winners discussed in Section 3.3, a standard fileserver configured specifically for JouleSort [79], the CoolSort machine discussed in Section 3.4, and three additional ultralow-power sorting systems designed by Meza et al. [56]. Table VI summarizes these systems and their JouleSort results. The first is an ARM-based Gumstix device, typically used in embedded devices. The second is an AMD Geode-based Soekris board, designed for routers and networking equipment. The final machine is a VIA picoITX-embedded multimedia machine with flash hard drives.

Figures 3 and 4 show the PennySort and JouleSort scores, respectively, of all of these systems. Note that the ultralow-power machines were only able to sort very small data sets (less than 10 GB) with the amount of storage they had; for purposes

TABLE VI
LOW-POWER MACHINES BENCHMARKED BY MEZA ET AL. [56, 80]

Name	Description	Price ($)	Avg Pwr (W)
Gumstix	Used in embedded devices	376.50	2
Soekris	Used for networking apps	477.50	6
VIA	Multimedia machine with flash drives	1156.60	15

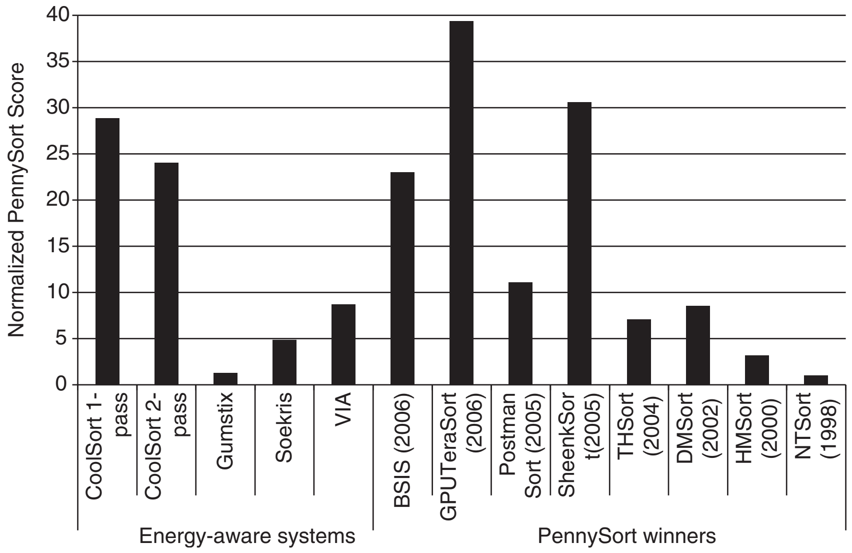


FIG. 3. PennySort scores of energy-aware systems and previous PennySort benchmark winners, normalized to the lowest-scoring system.

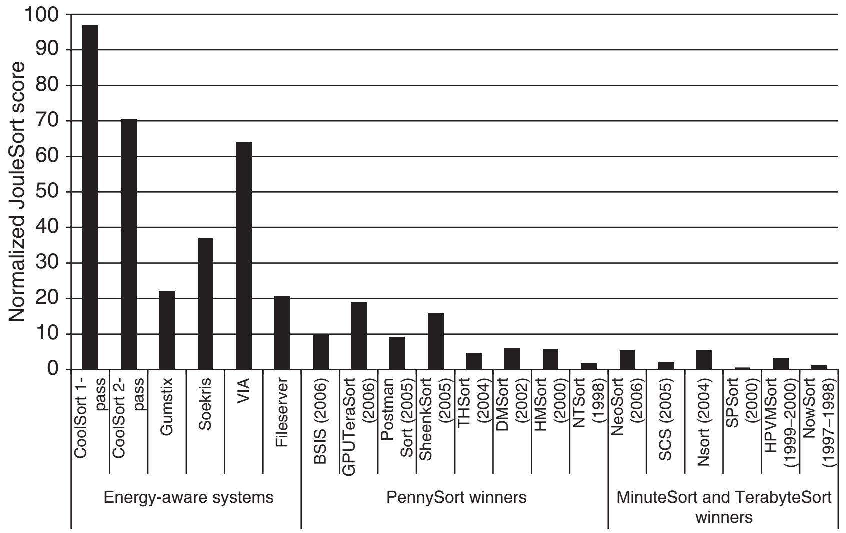


FIG. 4. JouleSort scores of energy-aware systems and previous PennySort, MinuteSort, and Terabyte Sort benchmark winners, normalized to the lowest-scoring system.

of comparison, CoolSort's performance on a similarly sized data set (CoolSort-1 pass) is included. All of these scores are normalized to the lowest-performing system to facilitate comparisons among the different metrics in addition to the different machines. Note that the balanced fileserver and the winners of the purely performance-based sort benchmarks are not included in the comparisons of metrics involving cost, such as PennySort.

The PennySort results are dominated by GPUTeraSort, followed by two recent PennySort winners (BSIS [38] and Sheenk Sort) and CoolSort. Since mobile components, such as those used in CoolSort, usually cost a premium compared to their desktop equivalents, this result is somewhat surprising. This cost premium also penalizes the ultralow-power components used in the other energy-aware systems, resulting in their low rankings according to this metric.

The JouleSort results in Fig. 4 show that all of the energy-aware machines outperform all of the historical PennySort winners, although the Gumstix, the fileserver, and GPUTeraSort are roughly equivalent. Among the ultralow-power machines, the VIA in particular comes close to the energy efficiency of CoolSort's two-pass sorts. This result illustrates the energy-efficiency potential of ultralow-power, flash-based systems.

Figures 5 and 6 combine performance, price, and power in two different ways. Figure 5 evaluates systems based on their JouleSort rating per dollar (records per Joule per dollar), using the list price of the system. Compared to the JouleSort and PennySort metrics, this metric gives extra weight to resource constraints: the PennySort and JouleSort metrics can be abstracted as performance divided by a resource constraint (price or power), while this metric is equivalent to performance divided by the product of two resource constraints. Since price and power consumption somewhat correlate, as explained in Section 3.3, this metric should favor smaller systems with lower power and lower price. Indeed, this metric is where the ultralow-power systems shine, far outstripping the historical PennySort winners and the other energy-efficient systems. CoolSort and the minimally configured Sheenk Sort come in a distant fourth and fifth. In a situation where resource constraints are the primary concern, these ultralow-power machines are the best choice.

Figure 6, on the other hand, keeps PennySort and JouleSort's one-to-one balance between performance and resource constraints by multiplying the PennySort and JouleSort ratings to get a combination metric in Records^2/(J $\times$ s), which is normalized to the lowest-scoring system in the figure. Because of the great disparities in this rating among the systems compared, Fig. 6 uses a logarithmic scale. According to this metric, CoolSort gives the best balance of performance and price/power, followed distantly by the VIA, the Soekris, and the recent PennySort winners. This combined metric shows that the additional cost of high-performance, low-power

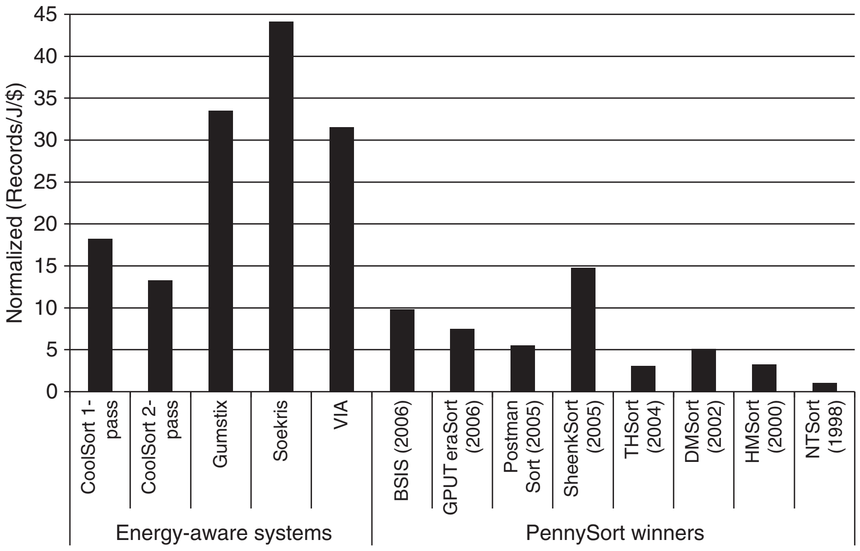


FIG. 5. Records sorted per Joule per dollar of purchase price of energy-aware systems and previous PennySort winners, normalized to the lowest-scoring system.

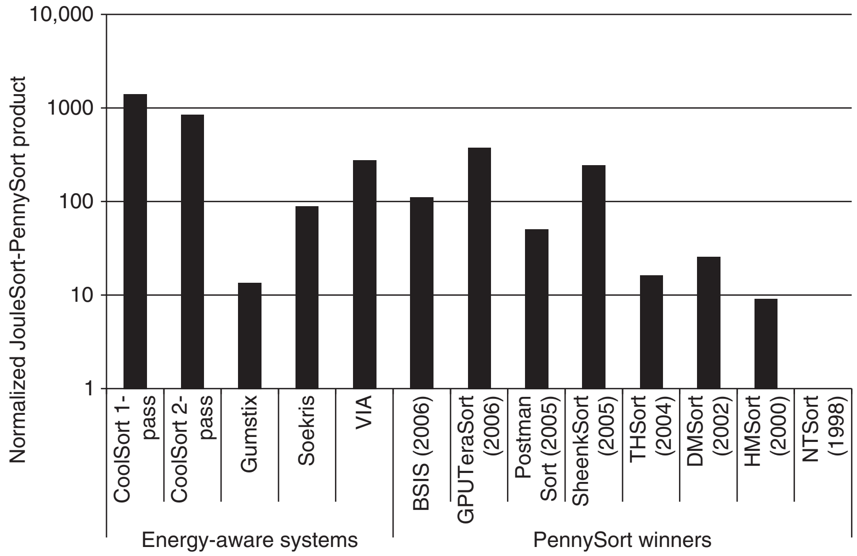


FIG. 6. Product of JouleSort and PennySort scores of energy-aware systems and previous PennySort winners, on a logarithmic scale, normalized to the lowest-scoring system.

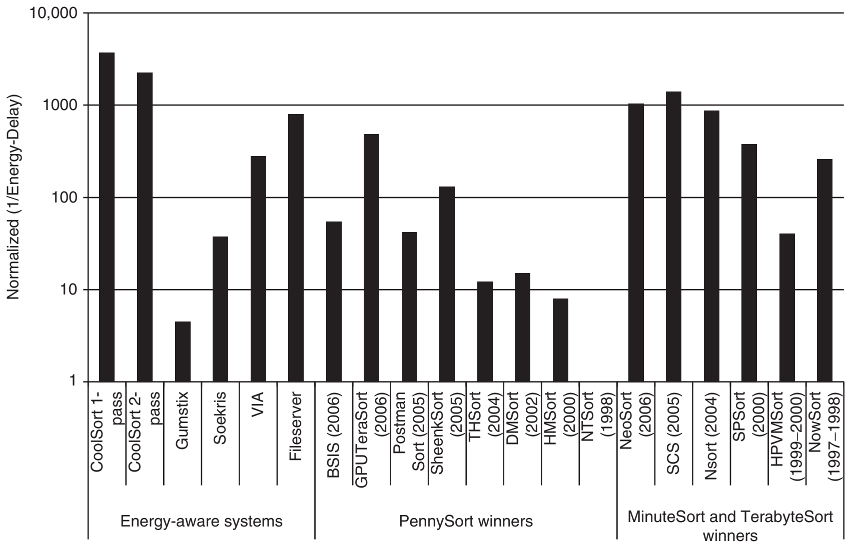


FIG. 7. Reciprocal of energy-delay product of energy-aware systems and previous sort benchmark winners, on a logarithmic scale, normalized to the lowest-scoring system.

components does not prohibit them from being good choices when price is taken into consideration.

Figure 7 does not consider price, but instead provides a different weighing of performance and power than the JouleSort metric. Rather than weighing performance and power equally, it uses the energy-delay product, which privileges performance. The metric used is (Records/J) × (Records/s); because the machines being compared sorted varying numbers of records, execution time must be normalized to the workload size to provide a fairer comparison. Note that this metric is the inverse of the energy-delay product, so higher scores are better. Again, because of the great disparities in this metric among the different systems, Fig. 7 uses a logarithmic scale.

Because of the additional weight given to performance, the MinuteSort and Terabyte Sort winners fare much better with this metric than with the original JouleSort metric, as does the balanced fileserver. In general, this metric clearly favors larger systems than the JouleSort metric. However, CoolSort remains the highest-ranked system according to this metric, and the ultralow-power VIA is surprisingly competitive at this metric that should be biased against it.

Finally, Fig. 8 attempts to synthesize a TCO estimate and compare the systems based on their sorting performance per dollar of TCO. The performance metric used is the number of records sorted per unit time. The metric used to approximate TCO is

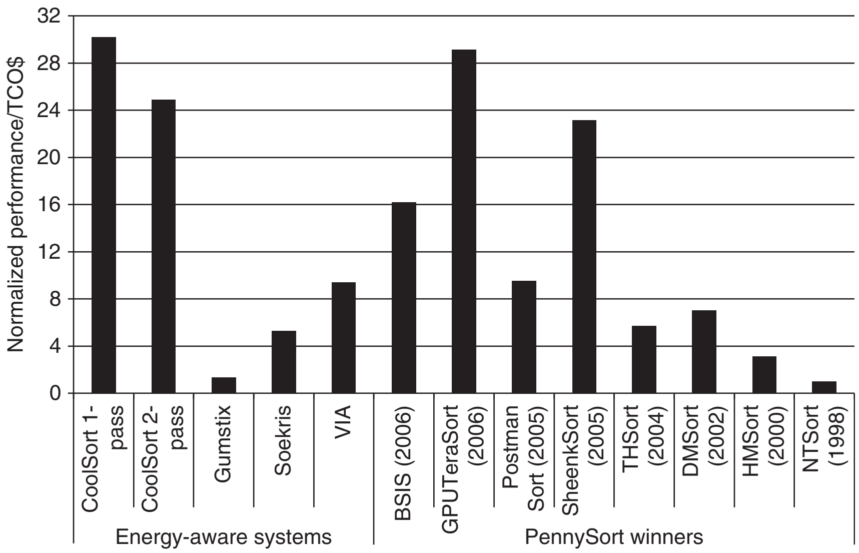


FIG. 8. Performance–TCO ratio for energy-aware systems, normalized to the lowest-scoring system.

the initial cost of the hardware plus an estimate of the energy cost of running the sort for 3 years at a cost of \$100/MW h [69]. The TCO estimates in the graph are computed according to Equation 1. This equation does not include cooling or burdened costs:

$$\begin{aligned}\mathrm{TCO}(\$) &= \mathrm{Price} + \mathrm{AvgPwr}(\mathrm{W}) \times \frac{24\mathrm{h}}{\mathrm{day}} \times \frac{365\mathrm{days}}{\mathrm{year}} \times 3\mathrm{year} \times \frac{\$100}{\mathrm{MW\,h}} \times \frac{\mathrm{MW}}{1\times10^{6}\ \mathrm{W}} \\ &= \mathrm{Price} + 2.628 \times \mathrm{AvgPwr}. \qquad (1)\end{aligned}$$

Over 3 years, the TCO is thus the initial cost of hardware plus 2.6 times the average power in Watts. For all of the machines in Fig. 8, the initial purchase price is a larger factor in this TCO metric than the estimated cost of energy; the initial cost is particularly dominant for the energy-aware machines and the early PennySort winners. Using this TCO-aware metric, GPUTeraSort fares best among the two-pass sorting configurations, followed by CoolSort and Sheenk Sort. The overall graph is similar to the PennySort graph (Fig. 3), with slightly lower disparities between the best- and worst-scoring systems. If cooling costs are factored in, the contribution of power consumption to the TCO rises, and the graph begins to look more like the JouleSort graph (Fig. 4). CoolSort passes GPUTeraSort as the highest scorer when cooling costs reach 0.8 W per Watt of power consumed by the computing equipment; in data centers, the cost of cooling is 0.5–1 W for each Watt of compute power [71].

Comparing these alternative metrics leads to several conclusions. First, low-power components are excellent choices for energy efficiency even when cost is taken into account. Second, although different weighings of performance and power privilege different system classes, the CoolSort mobile fileserver and the VIA multimedia machine are fairly insensitive to changes in these weighings, remaining at or near the top tier of systems for all of these metrics. These results indicate that systems with high JouleSort benchmark scores do not become impractical when cost or performance concerns are emphasized.

3.6 Conclusions

This section presented the specification of the JouleSort energy efficiency benchmark and the JouleSort scores of a variety of systems, including the estimated scores of previous sort benchmark winners and the measured scores of specially designed machines. It also presented CoolSort, the machine which achieves the highest known JouleSort score, with over 3.5 times the energy efficiency of previous sort benchmark winners. CoolSort is a fileserver built from a mobile processor and 13 laptop disks connected through server-class I/O interfaces, an unusual design that highlights the potential of low-power mobile components in the data center. While CoolSort was optimized specifically for the JouleSort benchmark, this type of design has the potential to be broadly useful. Finally, this section examined alternative benchmark metrics weighing different combinations of price, performance, and power. Different combinations of these metrics privilege different types of systems, but CoolSort and two ultralow-power sorting systems scored well across several different metrics.

These benchmark results show that JouleSort continues the Sort Benchmark's tradition of identifying promising new technologies. CoolSort's benchmark scores illustrate that a server composed of mobile-class components is highly energy-efficient, without being prohibitively costly or low performance. The VIA system's success demonstrates the potential of ultralow-power processors and flash storage for energy-efficient execution of data-intensive applications. Finally, GPUTeraSort's success among the historical sort benchmark winners shows that the high performance of GPUs comes at a relatively small energy cost, although this observation may not continue to hold as GPUs grow ever more power hungry.

JouleSort does not address all possible energy-efficiency concerns. It targets data-intensive applications running at peak energy efficiency, which is equivalent to peak utilization for today's technologies. This design choice differs from the full-system SPECpower_ssj benchmark, which is CPU-intensive and compares the average energy efficiency across the utilization spectrum. As a single-system benchmark, JouleSort also omits some metrics of importance in the data center. It does not account for losses in power delivery at either the data center or the rack level, nor does it account for the building cooling used to maintain the ambient temperature around the system. However, the JouleSort benchmark is relevant to an important class of systems and has already led to innovations and insights in energy-efficient system design.

4. Power and Thermal Modeling Challenges

The previous sections explained the benefits of paying careful attention to the *metrics* that power- and energy-efficiency optimizations seek to maximize. Implementing these energy-efficiency solutions requires *models* of how these metrics respond to changes in a system's design or use. Sections 4.1–4.3 discuss a variety of proposed power and thermal models, as well as the energy-efficiency optimizations that they enable.

Just like the different energy-efficiency metrics described in Section 2, energy-efficiency solutions can target the component, single system, or data center. These solutions may take the form of choices made at design time or policies for day-to-day operation. Design-time optimizations include clock gating in processors, which minimizes the power dissipated by unused parts of the chip; at the data center level, they include such practices as alternating hot and cold aisles to avoid wasting energy when cool air mixes with hot exhaust. Day-to-day policies include consolidating data center workloads on as few machines as possible and putting the others to sleep, since lightly utilized machines consume almost as much power as fully utilized machines [6]. Some optimizations combine design choices with usage policies, such as processors that are designed with multiple possible clock frequency states, among which the operating system selects based on real-time utilization.

While the discussion in this chapter has been concerned with the tradeoff between performance and average power, another important class of solutions is focused on avoiding the reliability consequences of exceeding recommended power and heat levels. These solutions may, for example, spatially distribute the workload on a chip [86] or within a data center [63] so as to avoid a particular area becoming dangerously hot.

All of these different types of solutions require some model of the system's current performance, power consumption, and/or temperature, as well as a model of how the quantity of interest will respond to changes in the system's design or operation. With hardware instrumentation becoming more prevalent, particularly in server systems, information about the current power or temperature of the system can potentially be obtained without models, although not necessarily at the desired temporal or spatial granularity. However, hardware instrumentation yields no insight into the relationship between system usage and temperature or power consumption, nor can it predict the effects of a policy on future power or temperature.

The desired properties of a model vary depending on its intended use, as Section 4.1 explains. The remainder of the section details examples of three distinct types of models, from increasing to decreasing levels of detail and accuracy. Sections 4.1 and 4.2 present case studies of power and heat models: Section 5 evaluates the portability and accuracy of a family of high-level real-time power models, while Section 6 describes several proposed real-time temperature models for clusters and data centers as well as the optimizations they enable.

4.1 Model Properties

The ideal properties of a power or thermal model depend greatly on the optimization that the model is designed to enable. The most important distinction is between models used in the design phase and models used in day-to-day operation; speed and simplicity are generally paramount considerations in the latter case, while they are often sacrificed for accuracy in the former case. Design-time models tend to be slower and more accurate than online, real-time models. This section enumerates the most important considerations in creating a model.

Accuracy. First and foremost, the model must be sufficiently accurate to enable the desired optimization; otherwise, it is useless. For optimizations designed to cap peak power or heat below a certain level, the model's error must be strictly bounded; for optimizations that simply save energy, average accuracy may be the only metric of concern.

Granularity. The granularity at which predictions are made depends upon the application. To avoid thermal emergencies, predictions may need to be made for subsections of a processor that are just a few square millimeters in area [86]. Other optimizations may require predictions at the level of a component [26], a system [34], an enclosure or rack [25, 78], or a section of a data center [63].

Speed. The necessary speed of generating model predictions also depends upon the application. For models used by real-time policies, predictions must occur sufficiently quickly to enable good choices or to prevent power or thermal emergencies. This constraint means that the input data must be sampled at a certain rate and that the model must transform the input data into a power or thermal prediction within a certain amount of time. In simulation-based models used by system designers, the speed constraint is much more flexible, and the desired balance between speed and accuracy is entirely up to the designers.

Portability and Generality. Ideally, the model could easily be adapted to changes in the system. The modeling method could ideally work for different classes of systems and components (mobile, desktop, and enterprise); different architectures and implementation technologies; different power footprints and dynamic ranges; and different balances of components, rather than assuming that any particular component will dominate the system. It should ideally be easy to generate models for a new system, and doing so should require neither exhaustive tuning nor extensive design space exploration.

Affordability. The model should ideally not require expensive equipment, particularly in the deployment stage. Models that are generated using sophisticated hardware instrumentation or that require the deployment of expensive sensors are less likely to be widely adopted than models with fewer infrastructure requirements.

Nonintrusiveness. Ideally, generating and using the model should not require intrusive adjustments to the system hardware, such as cutting into a system's power planes; simulation-based models should ideally not require fundamental changes to the system's performance simulator. Deploying the model should also have minimal software overhead.

Simplicity. The model should be as simple as possible, both in terms of the number of inputs and the complexity of generating predictions.

4.2 Simulation-Based Models

One approach to creating power and thermal models is to integrate them into software that simulates the execution of programs on a particular system or component. This approach is typically used by system designers who have already written performance simulators of their prospective systems and wish to understand their power and thermal characteristics as well. These models range in detail and accuracy from fine-grained, low-level models that use detailed circuit knowledge to models that use a combination of architectural information and physical parameters.

Simulation-based models generally sacrifice the other considerations described above for accuracy, and they generally model components rather than systems or collections of systems, since it is difficult to obtain detailed knowledge of the many components in a full system. These models tend to be slow, since making predictions requires fully simulating the component or system. They are typically neither generic nor portable, relying on the specific implementation details of a particular component and requiring specialized knowledge even to be ported to components in the same family. They are not necessarily low overhead, relying on knowledge of the specific instructions being executed on a machine. Finally, they are not simple: they require a great deal of information about the system architecture and state, and require simulation to relate that information to the power consumption or thermal state. These models are primarily useful during the design phase; however, despite their shortcomings for real-time use, their high accuracy makes them worth examining as possible upper bounds on accuracy and for their insights about component or system behavior. This section describes some representative simulation-based models, from component level to full-system level.

One of the first processor power models to use architectural information rather than detailed circuit knowledge was Wattch, proposed by Brooks et al. [13]. Wattch integrates into the widely used SimpleScalar performance simulator [14], adding power predictions to that simulator's performance information. Wattch abstracts the structures on a processor into four main categories: array structures such as caches and register files; fully associative content-addressable memories such as TLBs; combinational logic such as functional units; and clock-related circuitry. It uses the CACTI cache models [93] to estimate the power usage of each structure based on that structure's size and number of ports, the process technology, the amount of switching activity, the supply voltage, and the clock frequency; this approach yields processor power estimates with an average error of less than 15% (30% at the high end of the power range). The initial Wattch study used it to evaluate the power and performance of microarchitectural design choices such as instruction window and data cache sizing and result memoization, as well as the compiler optimization of loop unrolling. Subsequent microarchitectural studies have used Wattch to evaluate innovations in cache structure [75] and chip multiprocessors [48]. Wattch is sufficiently accurate to enable these optimizations and is considerably less complex than circuit- or register-transfer-level models. It has been ported to a variety of processors, although these ports did require expert architectural knowledge. Generating Wattch models requires no sophisticated equipment or instrumentation.

To avoid processor thermal emergencies, separate thermal models are needed; Skadron et al. [86] showed that time-averaged power measurements alone are insufficient, since a microarchitectural unit's temperature also depends on the temperature and material properties of units that are physically nearby. To address this problem, they developed the HotSpot compact thermal model, which adds information about the chip's layout and packaging to Wattch's power predictions to predict temperature. Using HotSpot, they evaluated the performance cost of several techniques to respond to thermal emergencies: lowering clock frequency; scaling the duty cycles of architectural units; and migrating activity to cooler parts of the chip. The increased accuracy of this thermal model comes at the price of more model inputs as well as more complex model equations.

A higher-level approach was used in the Tempo simulator by Shafi et al. [84] to simulate the power consumption of the PowerPC 405GP embedded system-on-a-chip. The model associates energy consumption with individual architectural events, such as data cache misses or TLB reads, and then predicts the energy consumption by counting these events: that is, $E_{\text{pred}} = E_{\text{idle}} + (e_i \times n_i)$, where each i is an architectural event whose energy e_i is multiplied by the number of times it occurs, n_i. To generate this model, the authors used a data acquisition system to measure the voltage drops across different parts of the chip at a 10 KHz frequency while running 300 detailed microbenchmarks to isolate the architectural events of interest. This approach is accurate within 5% on average and generates simpler models than Wattch, but it requires detailed knowledge of the system both in the hardware wiring and in the creation of microbenchmarks. The infrastructure for generating the model is also more intrusive and expensive.

The SoftWatt power simulator proposed by Gurumurthi et al. [33] is unusual in that it models not only the CPU, but also the memory hierarchy and disk. The CPU and memory structures are categorized and modeled similarly to Wattch. In SoftWatt, however, the model postprocesses simulation traces rather than being incorporated into the simulator (SimOS [81]). This approach is faster and less intrusive, at the expense of coarser time granularity and some accuracy. The disk model, on the other hand, is a simple state machine based on the manufacturer's specifications of the power and energy costs of transitioning to and from low-power states. To capture these transitions properly, the disk model is incorporated into the simulator.

Finally, the Mercury infrastructure developed by Heath et al. [35] emulates temperature sensors within a system or cluster, with accuracies within 1 °C. The inputs to Mercury's finite-element solver are an intercomponent heat-flow graph; an intramachine airflow graph; an optional intermachine airflow graph for clusters; constants describing components' physical properties such as heat transfer coefficient and surface area; and component utilization data. After an initial calibration phase to discover the relationship between component utilization and power consumption, Mercury requires no hardware instrumentation to generate thermal predictions. The authors use Mercury to emulate thermal emergencies to develop a dynamic thermal management policy.

4.3 Detailed Analytical Models

Power and thermal models can be constructed without simulation by periodically collecting hardware and software metrics at runtime. These models often rely on processor *performance counters*, which are hardware registers that can be configured to count various kinds of microarchitectural events, such as instructions retired or branch mispredictions. The number of performance counter registers and the events that can be counted vary among manufacturers and processor families, but these registers are present in all major processors, and there is a great deal of overlap in the types of events that can be counted. In general, the number of countable events exceeds the number of performance counter registers, so that only a small subset of the possible events can be counted at any one time. Azimi et al. [4] proposed a methodology, which is used in many of these models, for time-multiplexing different sets of events on the performance counter registers. This approach allows many more events to be monitored, at the price of increased overhead and lower accuracy, since the counts for a particular event are sampled rather than continuously monitored.

Joseph and Martonosi adapted the Wattch processor power models [13] to develop runtime, performance counter-based power models for the Compaq Alpha 21264 and the Intel Pentium Pro [43]. Using the circuit-level and microarchitectural information from the Wattch models, they correlated performance counter events with inputs to the Wattch models, resulting in a model based on nine performance counters for the 21264 and 12 performance counters for the Pentium Pro. To capture the dynamic power due to data-dependent switching, they periodically sampled the population counts of registers to get an idea of the amount of switching taking place. This work thus allows the Wattch models to be used in online policies, and it can be applied to different processor families, although it does require specialized hardware knowledge.

Isci and Martonosi [42] developed real-time models for the Pentium 4 based on performance counters. Because their goal was to facilitate thermal optimizations, they divided the processor into 22 units based on physical components on the die, and sought to estimate the overall power as well as the power of each unit. Their final model used 15 performance counters, which required time-multiplexing into four event sets. Sixteen of the 22 units used linear models; the other six, all of which were issue-logic units, used piecewise linear models. The model is real-time and highly accurate, and it yields relatively simple equations for each of the models. However, it necessarily relies on detailed microarchitectural and layout knowledge for a particular processor, and it models only the processor component.

Kadayif et al. [45] used performance counters to model the energy consumption of the Sun UltraSPARC memory hierarchy, with the goal of synthesizing the manufacturer-provided performance counters into ''virtual'' counters for energy events. They used eight performance counters and combined them with knowledge about the size, design, and technology of the caches and memory to provide energy estimates. These energy estimates for the memory hierarchy are real-time and fairly simple, particularly for the end user.

These detailed real-time analytical models provide a middle ground between the simulation-based models discussed in Section 4.2 and the high-level black-box models covered in Section 4.4. Because they are essentially more complex and specialized versions of the models covered in Section 4.4, they illuminate the tradeoff between portability and accuracy.

4.4 High-Level Black-Box Models

A third approach to power and thermal modeling is to construct a real-time model based solely on fitting a model to the real-time metrics collected and the corresponding temperature or AC power measurements, without relying on implementation knowledge. The general procedure is to calibrate the model by running a suite of synthetic benchmarks designed to generate a range of values for each metric; for example, if the number of floating-point instructions is one of the metrics collected, the synthetic benchmarks should stress the floating-point unit at various intensities. A variety of these black-box models have been proposed, usually in the context of facilitating energy-efficiency optimizations on a particular configuration. This section examines the types of models that have been used in previous studies. Section 5 presents an evaluation of several of these models over a wide range of systems, illuminating the tradeoffs among model simplicity, accuracy, and portability.

4.4.1 Processor and Memory Models

At the processor level, Bellosa's [9] power models for the Pentium II were some of the first to correlate power consumption with performance counter events. He found linear correlations between power consumption and each of four quantities measured by performance counters: integer micro-operations retired, floating-point operations, second-level cache references, and main memory references.

Several studies have used performance counters to understand when an application is processor-bound versus memory-bound to optimize the power budgets for processor and memory, usually by dynamically scaling the processor frequency. Weissel and Bellosa [99] used the performance counters for memory requests per cycle and instructions retired per cycle to inform frequency scaling choices; Kotla et al. [47] used the first-level and third-level cache hit rate counters and the number of memory references to enable similar optimizations on a quad-core PowerPC. In a server environment, Felter et al. [26] proposed dynamic power budgeting between processor and memory to enable less conservative static power budgeting; they showed that processor power linearly varied with the number of instructions dispatched and memory power with memory bandwidth for a simulated IBM POWER4 processor and its memory system.

Finally, Contreras and Martonosi [19] created a highly accurate linear model of the power consumption of an Intel XScale processor and its memory system by using performance counters for data dependency stalls, instruction- and data-TLB misses, data cache misses, and instructions executed. All of these component models are simple, fast, low overhead, and accurate enough to enable energy-saving optimizations; however, their generality and portability have not been tested.

4.4.2 Single-System Models

At the full-system level, Li and John [50] estimated the power consumption of operating system routines by creating performance counter-based linear models. They examined per-routine models using the performance counters for cycles and graduated instructions, and they contrast them with models using up to seven performance counters but using no knowledge of the software routines running. They concluded that only the per-routine models are consistently accurate.

Cignetti et al. [16] developed a full-system energy model for the Palm IIIE personal digital assistant. They divided the device into eight major power-consuming components: the CPU, the LCD, the backlight, the pen, the keypad, the serial port, the infrared port, and the sound system. They then modeled power for each as a constant function of its power state, and they instrumented system calls to convey information about the power state transitions of these components. Both the Cignetti and Li models are simple and highly effective for their target systems; however, their generality is limited. In the case of the Cignetti model for the Palm, it is unlikely that accurate power models for larger systems can be generated based solely on component power states. The Li model, on the other hand, relies on previous profiling of the routines being run.

4.4.3 Server Models

High-level power models have proven useful to enable energy-efficiency optimizations in server environments. Fan et al. [25] showed that over a large group of homogeneous servers, full-system models based on OS-reported CPU utilization alone proved highly accurate. They investigated two types of models: one was linear in CPU utilization, and the other was an empirical model of the form $P = C_0 + C_1 \times u + C_2 \times u^r$, where C_0, C_1, C_2, and r are experimentally determined model parameters and u is the CPU utilization.

Ranganathan et al. [78] implemented similar dynamic power budgeting optimizations at the blade-enclosure level. They created lookup tables correlating system resource utilization to performance and power and a methodology for porting a model generated on one machine to another in the same family by examining the different relationships between performance and resource utilization [77].

Heath et al. [34] used a similar approach to full-system power modeling to enable energy-aware server consolidation in clusters. Using OS-reported CPU, memory, and disk utilization, they developed linear models for two different types of servers that were sufficiently accurate to enable energy savings.

Finally, we used both OS-reported component utilizations and CPU performance counters to model a blade and a four-way Itanium 2 server [21]. Using the same inputs for both systems, we generated linear models that typically yielded errors of less than 10%. Given the disparities in power footprint, component balance, and component architectures between the two systems, this result is strong evidence for the generality of this approach; however, the tradeoffs between simplicity and accuracy were not examined.

4.5 Summary

Energy-efficiency optimizations from the chip to the data center rely on accurate power and thermal models. This section outlined the often-conflicting goals that models must balance, and it described three different approaches to power and thermal modeling: a simulation-based approach, detailed analytical real-time modeling, and high-level black-box modeling. Sections 5 and 6 examine case studies in power and thermal modeling in more detail.

5. Power Modeling Evaluation

As a case study in power modeling, this section describes the Mantis model development infrastructure, which can be used to generate several of the real-time black-box models described in the previous section. It also presents an evaluation of the accuracy of these models over a wide range of hardware systems and software workloads, giving insight into the relationships among simplicity, accuracy, and generality. First, we describe the process of formulating and applying power models, including the physical instrumentation setup and the software model generation process. Then, we describe the choices of model inputs and the particular models studied. Finally, we describe the evaluation framework and results.

5.1 Model Development Process

5.1.1 Overview

Figure 9 illustrates the Mantis model development and application process. First, we run a calibration suite on the system being modeled and simultaneously collect measurements of the system's total AC power as well as software utilization metrics. This calibration suite, described in more detail in Section 5.2, consists of programs that stress each individual component of the system at varying utilization levels. The outcome of this first stage is a vector of utilization metrics for each sampling interval

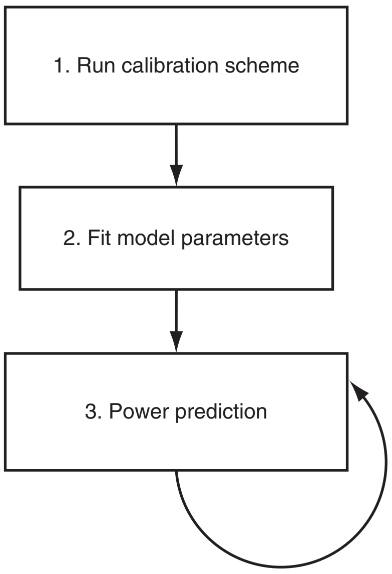


FIG. 9. Overview of Mantis model generation and use.

(a 1-s interval is used throughout this work), correlated to an AC power measurement for that interval.

The second stage of the process is to fit models to these utilization and power measurements. Section 5.3 describes the models we fit in more detail. The result of model generation is an equation or set of equations that relates the metrics collected to the full-system AC power. No information about the system other than the data collected by the calibration process is used to generate the models. Steps 1 and 2 need to be done *exactly once* for a system configuration, potentially by its vendor.

The third stage of the process is using the Mantis models to predict power. In this stage, a low-overhead dæmon collects the utilization metrics and predicts power based on the models developed in the second stage, without the need for an AC power meter. However, to evaluate the models' accuracy in Section 5.4, we both measure AC power and use the models to predict power; we compute the models' accuracy by comparing the predicted power to the actual measured power for each sampling interval.

Figure 10 shows the measurement infrastructure for the calibration stage. The system under test is plugged into a power meter, in this case the Brand Electronics 20-1850CI, which in turn plugs into the wall. The power meter measures the AC power of the system under test. Another computer, the control and measurement

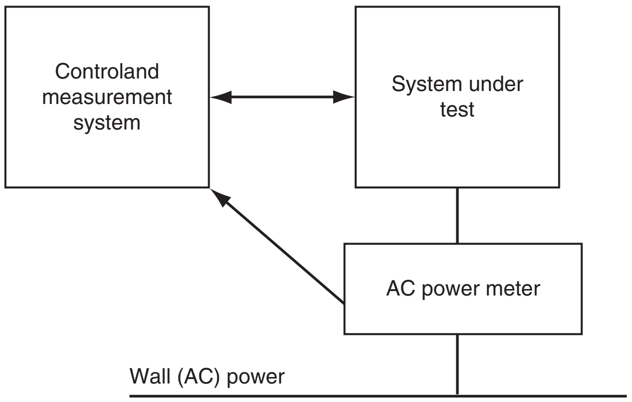


FIG. 10. Mantis instrumentation setup.

system, reads the AC power measurements from the meter via serial cable and initiates the calibration suite and the metric-collecting dæmon via Ethernet connection with the system under test. To generate models, the timestamped metrics are correlated with the timestamped power measurements. Occasionally, samples will be dropped due to drift in the timing routines of the metric-collecting dæmon or the power meter interface; in this case, the missing value is replaced by a linear interpolation of the preceding and succeeding samples.

5.2 Calibration Process

The goal of the calibration process is to generate a data set that captures the relationship between high-level software metrics and full-system AC power. This goal is achieved by collecting software metrics and AC power measurements while running a suite of programs that exercises each of the core components at varying levels of utilization, from idle to peak. This section describes the calibration suite used to generate this data, including a discussion of its portability and its limitations.

5.2.1 Calibration Software Suite

The calibration software suite has four different components: a brief ‘‘baseline’’ test to establish the idle power, a CPU test, a memory test, and a disk test; similar tests could be developed for other components if needed. The memory and disk tests use the gamut system stress software [57]. To stress the memory system, gamut creates worker threads that allocate chunks of memory and write random values to each page. Gamut allows the user to specify varying working set sizes, access rates, and ratios of sequential to random accesses. The Mantis memory calibration workload consists of successive 55-s runs for all combinations of two workload sizes (a maximal amount of memory and half of that number), sequential versus random access patterns, and a variety of access rates.

To stress the disk subsystem, gamut creates worker threads, each of which performs reads and writes to a file. Gamut allows the user to specify the file location, the file size, the size of each access, the I/O rate, and the mix of read, write, and seek commands. The Mantis calibration suite varies the I/O rate and the read/write mix. For systems with multiple disks, the suite is run multiple times; the first run exercises one disk, and each run adds an additional disk to the workload.

To stress the CPU(s), we wrote a program that performs integer or floating-point matrix multiplication on user-specified matrix sizes at user-specified levels of utilization; the idea is to exercise each level of the on-chip memory hierarchy as well as the functional units. This program uses a similar approach to gamut's CPU tests: it defines an ''epoch'' as 1/25th of a second, and runs an initial test to determine the maximum number of inner loop iterations per epoch to scale the utilization. The Mantis calibration suite runs this program at five different utilization points for varying matrix sizes and both integer and floating-point data types. For systems with multiple processors, the suite is run multiple times, each time on an increasing number of processors (i.e., first with one processor with varying utilization points, matrix sizes, and data types, then with two, and so on).

In this study, no program was used to selectively stress the network subsystem. Our initial research [21] showed that high-level network utilization data did not affect the power models generated; the network terms simply dropped out of the models. Power management is not yet widely employed in the network subsystem, either in the form of sleep states or in the form of scaling power down in response to low utilization [65]. At the time of this work, network subsystems thus do not have high enough dynamic power variation to add any useful information to full-system power models.

5.2.2 Portability and Limitations

In order for Mantis to be portable, the calibration suite must be portable as well. Currently, the suite requires some manual tuning, beyond simply specifying system properties, to be sure that resources are adequately utilized. For the CPU test, the matrix sizes and number of iterations must be checked to ensure that they remain in cache and that the test takes a reasonable amount of time. The memory and disk tests also require some experimentation to discover the access rates that saturate these components. Most of this parameter space exploration could potentially be automated in future versions of the software.

Beyond portability concerns, the calibration suite has some other caveats and limitations. First, the approach of stressing each component independently runs the risk of not capturing correlations between components. The models studied in this work assume that the components can be approximated as linearly independent; for more complicated models, the calibration suite may need to be re-evaluated.

Second, the calibration suite may not be able to fully stress all components for several reasons. Since it runs on top of the operating system, it has imperfect control of the hardware, which is an issue for the memory and disk tests in particular. In addition, program overheads may prevent it from maximizing the bandwidth to a subsystem. Finally, maximally stressing a particular unit may require architecture-specific knowledge; in the case of a CPU, for example, knowledge of the hardware functional units and the amount of parallelism available would help to ensure the maximum dissipation of power.

Finally, the calibration suite attempts to stress each component in isolation, which means that if we assume that the CPU, memory, and disk tests take equal amounts of time, each component will be idle for two-thirds of the duration of the calibration suite. Because the model generation process attempts to minimize error over all data points without correcting for this bias, models may be biased toward the low-utilization case.

5.3 Generating Models

This section describes the possible inputs for high-level black-box power models and then discusses the five types of models evaluated in this study.

5.3.1 Model Inputs

Full-system power consumption can be divided into two components: the *static* or idle power, which is independent of system activity, and the *dynamic* power, which varies with utilization. The static power is easily obtained by measuring the system's power consumption when it is idle. Utilization metrics, however, are necessary to understand the dynamic power. The utilization metrics used as model inputs should therefore be those that best correlate to the dynamic power consumption of the overall system. This section describes in detail the utilization metrics used in the prior modeling work described in Section 4, including their relationship to dynamic power and how they can be collected. Section 5.3.2 will describe the specific models evaluated in this work.

OS-reported CPU utilization is often used as a first-order proxy for dynamic power, as in the models by Fan et al. [25]. The reason is that the CPU has historically dominated systems' dynamic power. OS-reported CPU utilization corresponds to the percentage of cycles in which the CPU is not in the idle state. This statistic captures the dynamic power variation brought about when CPUs transition to a low-power mode when idle. Its weakness, however, is that it does not capture how fully the CPU is utilized: the CPU counts as nonidle as long as any of its pipeline stages or functional units are active. CPU utilization is an easily available statistic on most platforms, via tools such as sar from the Linux sysstat package.

The dynamic power of hard disks can be approximated to a first order by the disk power state and secondarily by the balance between seeking and transferring data [33]. Programs like Linux's iostat provide information about the disk utilization (which is defined analogously to CPU utilization), the number of read and write accesses, the number of blocks or sectors read and written, and the occupancy of the disk queues. Disk utilization provides the first-order information about whether the disk is active or idle, without the caveats of the CPU utilization metric since disks lack CPUs' parallelism. The number of accesses combined with the number of blocks or sectors written provides some information about the balance between random and sequential I/O activity.

Network utilization can also be captured by sar. However, our preliminary work found very little correspondence between network utilization and dynamic power, as explained in Section 5.2.

Finally, a wealth of information about the behavior of the CPU and memory can be obtained from the processor's hardware performance counters, as described in Section 4.3. Monitoring performance counters usually requires modifying the operating system; on Linux, it requires recompiling the kernel. A number of interfaces to the performance counters exist; we use the Perfmon2 suite in this work, because it provides a high-level interface to system-wide performance counter sampling and because it has been ported to a wide range of processor families [72]. Performance counters that are found on many systems and that capture significant dynamic power variation (see Section 4) are those corresponding to the memory bandwidth, the amount of instruction-level parallelism, the activity of the cache hierarchy, and the utilization of the floating-point unit.

5.3.2 Models Studied

We examine five different types of models, which vary in the inputs used and the complexity of the model equation. This section describes these five models, which are evaluated for a variety of machines and benchmarks in Section 5.4.

The first model simply uses a constant C_0 as the predicted power, irrespective of utilization. This C_0 is obtained by measuring the average power consumption during the calibration suite. Using a constant power model is valuable for two reasons.

First, it provides a baseline against which to evaluate the utilization-based models. Second, it represents the practice of estimating power by looking at the manufacturer-specified or ''nameplate'' power, although this model will probably be more accurate than the ''nameplate'' power since it represents the average power during use rather than a conservative worst-case estimate.

The next two models use only CPU utilization to predict power; these are the two models used by Fan et al. [25]. The first model, described by Equation 2, predicts power as a linear function of the CPU utilization u_{CPU}. The C_0 term represents the power consumption that is invariant with CPU utilization, and the C_1 term represents the contribution of CPU utilization to dynamic power. C_0 and C_1 are obtained by simple linear regression over the timestamped CPU utilization and AC power measurements collected during the calibration suite. The second model, described by Equation 3, adds empirical parameters C_2 and r:

$$P_{\mathrm{pred}} = C_0 + C_1 \times u_{\mathrm{CPU}}, \tag{2}$$

$$P_{\mathrm{pred}} = C_0 + C_1 \times u_{\mathrm{CPU}} + C_2 \times u_{\mathrm{CPU}}^{r}. \tag{3}$$

The fourth model, similar to that proposed by Heath et al. [34], uses disk metrics from iostat in addition to the CPU utilization. This model, of the form shown in Equation 4, is linear in CPU and disk utilization (u_{disk}), with coefficients C_0, C_1, and C_2 derived from simple linear regression over the calibration data:

$$P_{\mathrm{pred}} = C_0 + C_1 \times u_{\mathrm{CPU}} + C_2 \times u_{\mathrm{disk}}. \tag{4}$$

The final model uses performance counters in addition to OS-reported utilization data, as we proposed in Economou et al. [21]. The exact performance counters used vary with the systems being modeled, but the basic form of the model is shown in Equation 5, where the coefficient C_i is derived for each performance counter p_i sampled. For this family of models, we use only as many performance counters as can be sampled at one time, in the interest of low overhead and model simplicity; for most of the systems we study, four performance counters can be sampled simultaneously:

$$P_{\mathrm{pred}} = C_0 + C_1 \times u_{\mathrm{CPU}} + C_2 \times u_{\mathrm{disk}} + \sum (C_i \times p_i). \tag{5}$$

5.4 Evaluation Setup

To investigate the generality and portability of these models, and to understand when to choose one type of model over another, we evaluate them on a variety of machines running a variety of benchmarks. During these experiments, the machines are plugged into an AC power meter, and both software utilization data and AC power measurements are collected once per second. The power predicted by the models that were developed using the calibration data is then compared to the AC power, and the percent error of each prediction is noted. This section describes the machines and benchmarks used to evaluate the models.

5.4.1 Machines

We tuned and evaluated the Mantis-generated models on a diverse group of machines, which varied in several important characteristics. Their processors span three different processor families (Core 2 Duo/quad-core Xeon, Itanium, and Turion) and two different manufacturers (Intel and AMD). Their memories include mobile memory, desktop/server DDR2 memory, and fully buffered DIMM technology, while their disks include both mobile and enterprise disks. At the full-system level, the machines vary in the balance of power among their components and in the amount of variation in their dynamic power consumption. This section describes these machines in detail, while Table VII summarizes their characteristics.

The first machine studied is an HP Proliant DL140 G3 server that was purchased in 2008. This machine has eight processor cores, split across two quad-core Intel Xeon processors with clock frequencies of 2.3 GHz. Its memory consists of eight 4 GB FBDIMMs, for a total of 32 GB of memory. Finally, it has two 500 GB, 7200 rpm disks. This machine's power consumption is approximately 220 W when idle and up to 340 W when the processor and memory are highly utilized. This high dynamic range is historically unusual for a server [6].

The other enterprise-class machine studied is a 2005-era HP Itanium server prototype. This prototype is heavily unbalanced in favor of the CPU, with four 1.5 GHz Itanium 2 processors but only 1 GB of memory and one relatively low-capacity (36 GB) hard disk. The dynamic range of this system is quite low, with power consumption ranging between 630 and 680 W.

TABLE VII
SUMMARY OF MACHINES USED TO EVALUATE MANTIS-GENERATED MODELS

Machine	Description
Xeon server	8-core server with 32 GB FBDIMM
Itanium server	Compute-optimized server with four Itanium 2 CPUs
CoolSort-13	The CoolSort machine described in Section 3, using all 13 disks
CoolSort-1	CoolSort with only one disk
Laptop	Laptop with AMD Turion processor and 384 MB memory

The CoolSort machine described in Section 3.4 was used in four different configurations: with all 13 disks (CoolSort-13) at its highest and lowest frequencies, and with just one disk (CoolSort-1) at its highest and lowest frequencies. The dynamic power of the 13-disk configuration is dominated by the CPU and disks. In the one-disk configuration, CPU and memory dominate the overall power consumption.

The final machine is a 2005-era laptop, the HP Special Edition L2000. Its processor is an AMD Turion 64; models were developed for the highest (1800 MHz) and lowest (800 MHz) processor frequencies. It has 384 MB of DDR memory, and one 60 GB disk. Its dynamic range, the highest of any of the systems studied, extends from a minimum of 14 W at its lowest frequency to a maximum of 42 W at its highest frequency. Its battery was removed and the display closed for this study.

The five machine configurations studied span a wide range of components and system balances, which makes them a useful collection for testing the generality and portability of the different models. Table VIII summarizes their component classes, processor families, component balances, and power footprints and ranges.

5.4.2 Benchmarks

To evaluate the predictions of the Mantis-generated models, we chose a set of benchmarks that exercise various combinations of a system's core components. Because the calibration suite was biased toward the low-utilization case, the typically high component utilizations of these benchmarks present a worst-case challenge for the models. Table IX lists the benchmarks used.

First, the benchmarks comprised by the integer and floating-point SPEC CPU suites all stress the processor(s), but differ in the amounts of instruction-level parallelism present, the stress to the memory hierarchy, and in their memory access patterns [101]. Using these benchmarks will help to determine the benefit,

TABLE VIII
SELECTED PROPERTIES OF MANTIS EVALUATION MACHINES

Machine	Class	Processor	Dominant component(s)	Dynamic range (W)
Xeon server	Enterprise	4-core Intel Xeon	CPU, memory	220–340
Itanium server	Enterprise	Itanium 2	CPU	630–680
CoolSort-13	Hybrid	Mobile Intel	CPU, disk	58–108
CoolSort-1		Core 2 Duo	CPU, memory	45–85
Laptop	Mobile	AMD Turion	CPU	14–42

TABLE IX
DESCRIPTIONS OF BENCHMARKS SELECTED TO EVALUATE MANTIS MODELS

Name	Description
SPECint	SPEC CPU2006 integer benchmarks
SPECfp	SPEC CPU2006 floating-point benchmarks
SPECjbb	SPECjbb2005 server-side Java benchmark
stream	The STREAM memory stress benchmark
clamAV	Antivirus scanner
Nsort	Sorting program
SPECweb	SPECweb2005 Web server benchmark

if any, of performance counter-based processor information over CPU utilization alone. Next, the SPEC JBB benchmark emulates a server-side Java workload, focusing on the application tier. It stresses both the processor(s) and the memory hierarchy, with little or no I/O. The benchmark duration consists of several short runs, with the number of threads increasing in each run. Third, the STREAM benchmark is a synthetic memory benchmark that attempts to maximize memory bandwidth [54]. Thus, it heavily stresses the memory without exerting commensurate pressure on the CPU. As with the preceding benchmarks, STREAM has very little I/O activity.

Three benchmarks were used to stress the I/O subsystem. On most machines, the Clam antivirus scanner [17] was used, with multiple copies instantiated if necessary to exercise multiple disks. On these machines, the Clam program utilized CPU, memory, and disk. Two of the machines necessitated using different programs to stress the I/O system. On CoolSort-13, Clam was CPU-bound and barely utilized the disk subsystem; therefore, the Nsort program [66] was substituted. As described in Section 3.4, CPU and I/O utilization are both maximized during the first pass of the sort, with CPU utilization dropping in the second pass due to the bottleneck created by the increased amount of random I/O. On the Itanium server, on the other hand, SPECweb2005 was used to provide a combination of disk and network I/O.

Table X presents the typical component utilizations of these benchmarks. An unusually configured machine may have bottlenecks that result in different utilization characteristics, as with ClamAV on CoolSort-13. In general, however, these benchmarks stress different combinations of components to varying degrees, providing a spectrum of test cases for the Mantis models. The next section presents the results of these tests.

TABLE X
COMPONENT UTILIZATIONS OF MANTIS EVALUATION BENCHMARKS

	Component Utilization		
Name	CPU	Memory	Disk
SPECint	Very high	High	Very low
SPECfp	Very high	High	Very low
SPECjbb	Very high	Very high	Very low
stream	Medium	Very high	Very low
clamAV	Medium	Medium-low	Medium-high
Nsort	High	Medium	Very high
SPECweb	Medium-low	Low	Medium

5.5 Results

Figures 11 and 12 show the mean and 90th percentile errors, respectively, for each model and benchmark across all of the machine configurations tested. Each cluster of columns represents one of the benchmarks described in Section 5.4.2, except for the leftmost cluster, which represents the calibration suite used to fit the models. Within each cluster, the leftmost bar shows the error from the constant power model defined in Section 5.3, and the next two bars represent the two CPU utilization-based models defined by Equations 2 and 3, respectively. The fourth bar from the left is the model defined by Equation 4, which is based on the CPU and disk utilizations. Finally, the rightmost bar shows the error from the model defined by Equation 5, which includes performance counters as well as the CPU and disk utilizations.

These graphs support some high-level observations. First, all of the utilization-based models clearly outperform the constant model, showing that hardware resource utilizations do correlate to power consumption. In addition, the performance counter-based model, which uses the most information, is overall the best model for every benchmark, with the lowest mean and 90th percentile absolute errors across the board. Finally, all of the utilization-based models predict power within 10% accuracy (mean) and 12% accuracy (90th percentile) for each benchmark, averaged over all configurations.

Figure 13 shows the mean absolute percentage error for the benchmarks that make the strongest case for the empirical CPU utilization-based model. These benchmarks are the CPU-intensive SPECint, SPECfp, and SPECjbb benchmarks running on the Xeon server. For each of these benchmarks, the empirical CPU utilization-based model far outperforms the other models, and only this model and the performance counter-based model meet the goal of 10% absolute error or less.

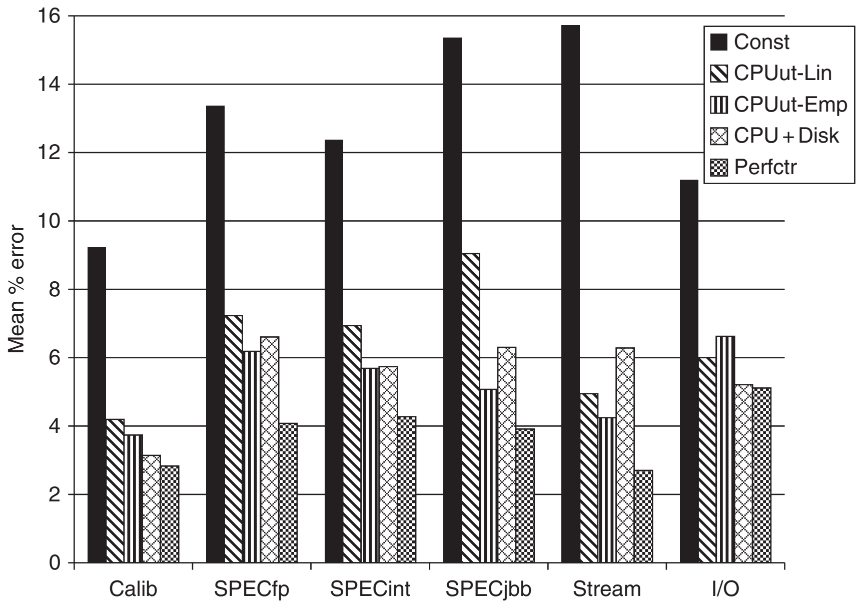


FIG. 11. Overall mean absolute error for Mantis-generated models over all benchmarks and machine configurations.

Figure 14 shows the power predicted by this model for different values of CPU utilization, while Fig. 15 shows the actual power during the calibration suite. The curve is close to linear for CPU utilizations between 0% and 500% (note that the maximum CPU utilization is 800%, since this machine has 8 cores). The curve then levels off at higher utilizations, which is much closer to the actual behavior of the server than a linear model. This effect may be attributable to the shared resources on the quad-core Xeon chips used in this server; for example, the maximum number of references to the shared L2 cache increases very slowly at high utilization, since this resource can be fully utilized even when not all of the cores are active. This empirical model was originally proposed in the context of multicore servers [25], so it may capture a fundamental behavior in server-class multicores with shared on-chip resources.

Figure 16 shows the best case for the performance counter-based power model. The leftmost cluster of columns shows the mean absolute percentage error for the stream benchmark on the Xeon server, which has 32 GB of FBDIMM memory. Stream is a memory-intensive but not particularly CPU-intensive benchmark, and so the high dynamic power of this machine's memory is only captured by metrics specifically based on memory utilization. The only model that uses such a metric is

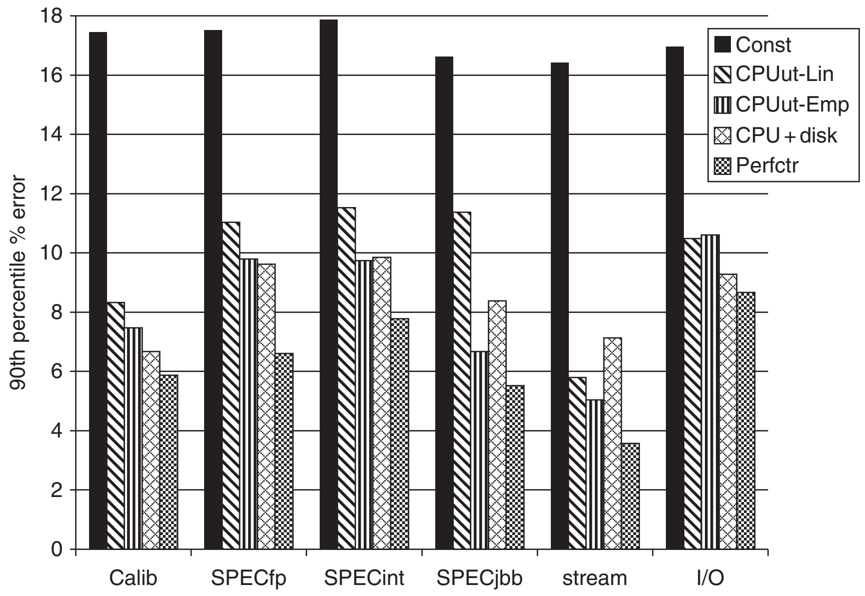


FIG. 12. Overall 90th percentile absolute error for Mantis-generated models over all benchmarks and machine configurations.

the performance counter-based model, which includes the performance counter for memory bus transactions as a parameter. This information results in a much more accurate power prediction.

The other three clusters of columns in Fig. 16 show the mean absolute percentage errors for the models on the CPU-intensive benchmarks on CoolSort-13 at the highest frequency. The reason that the performance counter model improves upon the others is that the CPU uses aggressive clock gating to shut down unused units, even at high utilization [41], so power depends not only on *whether* the CPU is utilized, but upon *how* it is utilized. In particular, the SPECjbb benchmark has much lower instruction-level parallelism than the SPECcpu benchmarks, even though it runs on both cores and the SPECcpu benchmarks run on just one.

5.6 Summary

This section presented the Mantis infrastructure for generating a family of high-level black-box full-system power models. Evaluating these models on a wide variety of systems shows that even the simplest utilization-based model, a linear

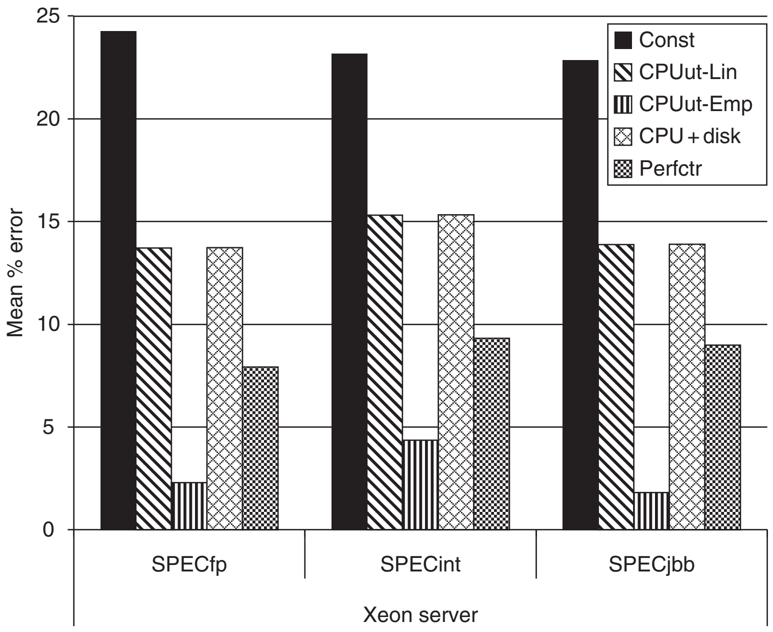


FIG. 13. Best case for the empirical CPU utilization-based model: CPU-intensive benchmarks on Xeon server.

model based on CPU utilization, is much more accurate than a constant prediction and may be sufficient for some energy-efficiency optimizations. However, this model breaks down in multicores with shared resources, in systems and workloads that are not CPU-dominated, and in processors with aggressive power management. Since these three situations are likely to become more prevalent given hardware trends, the more accurate performance counter-based model may be increasingly necessary in the future.

6. Online Thermal Modeling and Management

Infrastructure traditionally maintained by a facilities management team—such as cooling and the room's power grid—is now an integral part of data center design [83] due to the cost and reliability implications of thermal management. Current-generation 1U (1.75″) servers consume over 350 W at peak utilization, releasing

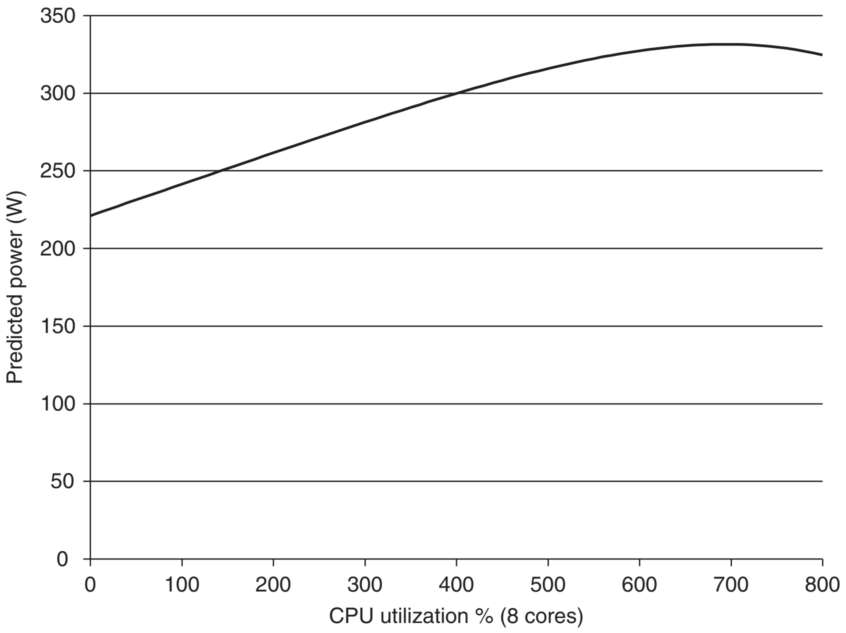


FIG. 14. Power predicted by the empirical CPU utilization-based model versus CPU utilization for the Xeon server.

much of this energy as heat. A standard 42U (73.5″) rack of servers consumes over 15 kW. Barroso et al. [7] estimate that the power density of the Google data center is 3–10 times that of typical commercial data centers. Their data center uses commodity midrange servers; that density is likely to be higher with newer, more power-hungry server choices [64].

Current data centers overprovision cooling and power by planning for the worst-case scenario [70]. While overprovisioning reduces the risk of hardware failure, it leads to excessive capital and recurring operational costs.

A thermal management policy that considers facilities components, such as computer room air-conditioning (CRAC) units and the physical layout of the data center, and temperature-aware IT components, can:

- *Decrease cooling costs*. In a 30,000 ft^2 data center with 1000 standard computing racks, each consuming 10 kW, the initial cost of purchasing and installing the CRAC units is \$2–\$5 million; with an average electricity cost of \$100/MW h, the annual costs for cooling alone are \$4–\$8 million [71].

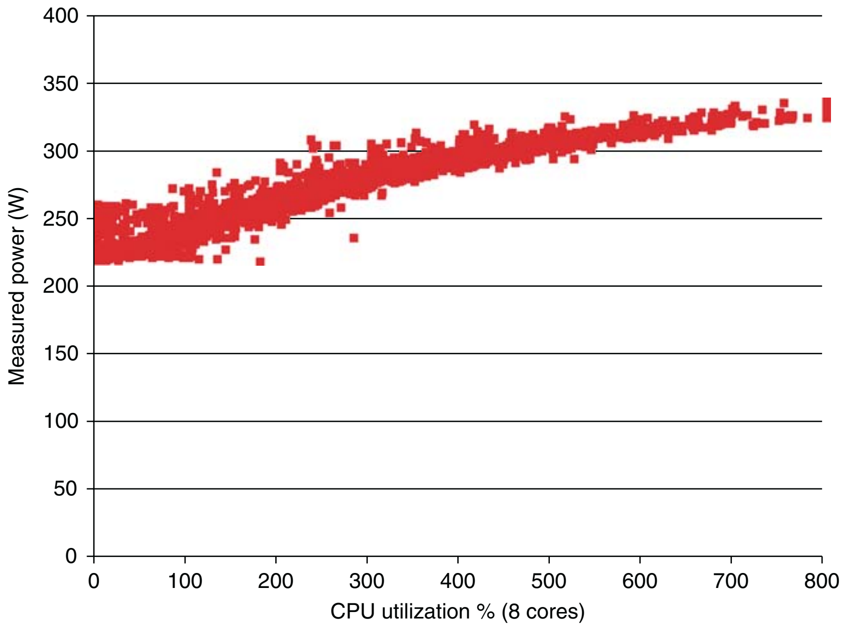


FIG. 15. Measured power versus CPU utilization for the Xeon server during the calibration suite. Note that memory and disk utilization also varied over this data.

- *Increase hardware reliability*. An Uptime Institute study [91] indicated that to avoid thermal redlining, a typical server should have the air temperature at its front inlets be in the range of 20–30 °C. Every 10 °C increase over 21 °C decreases the long-term reliability of electronics, particularly disk drives, by 50% [1, 18, 91].
- *Decrease response times to transients and emergencies*. Data center conditions can change rapidly. Sharp transient spikes in server utilization [3, 44] or the failure of a CRAC unit can upset the current environment in a matter of minutes or even seconds.
- *Increase densities and improve operational efficiencies*. A high ratio of cooling power to compute power limits the compaction and consolidation possible in data centers, correspondingly increasing the management costs.

Much of the work in automated data center thermal management focuses on formulating effective thermal management policies; for example, several projects reduced data center cooling costs using different approaches, such as optimizing

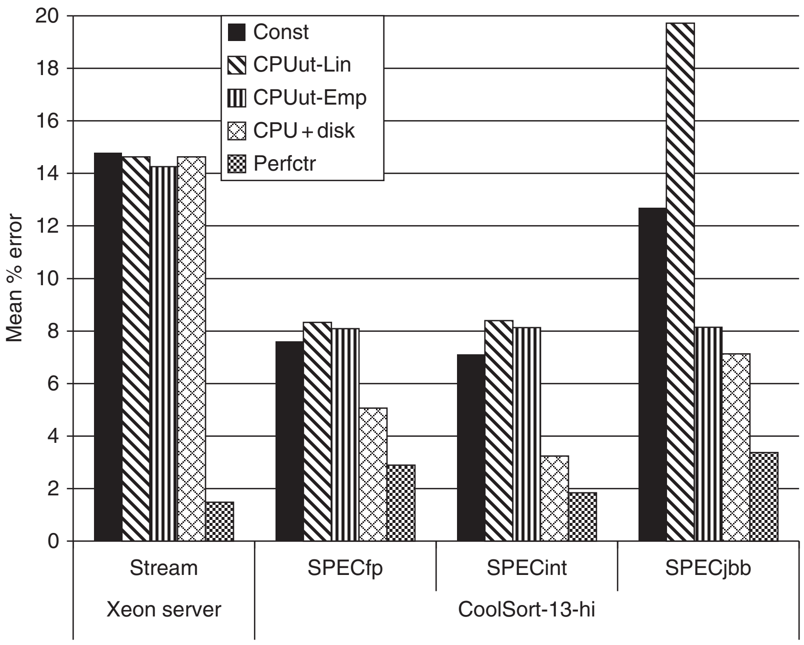


FIG. 16. Best case for the performance counter-based model: Selected benchmarks on the Xeon server and on CoolSort-13 at the highest frequency.

cooling delivery [71], minimizing global power consumptions [15, 76, 102], and distributing heat efficiently [11, 61, 74, 91].

These projects depend on an underlying instrumentation layer to obtain and aggregate the necessary power and temperature data to produce the thermal map. The ability to use internal platform sensors reduces the need for external sensors, and makes dynamic thermal management practical and easy to set up. Moreover, it improves the quality of the information driving the thermal control policy. In the absence of fine-grained thermal instrumentation, these policies must rely on simplistic heuristics, such as minimizing server power consumption or CRAC return temperature, or generating a uniform exhaust profile with minimal mixing of hot and cold air.

This section explores these server- and data center-level thermal modeling questions. It explains the necessity for instrumentation that captures the physical relationship between data center components and presents *Splice*, a location-aware instrumentation infrastructure [59, 60]. It also presents *ConSil*, which improves temperature sensor coverage by combining ubiquitous motherboard sensor information with workload data and machine learning methods [62]. Finally, it describes *Weatherman*, a data center thermal mapping prototype [63]. The combined infrastructure of Splice, ConSil, and Weatherman has been used to enable data center workload placement policies that nearly halve data center cooling costs compared to a thermally unaware distribution scheme.

6.1 Location-Aware Instrumentation with Splice

The first step in implementing automated thermal management is to instrument the data center to collect relevant metrics, such as server power consumption, airflow information, and CRAC unit settings. Much of this information is only available using *ad hoc* methods and at coarse granularities. For example, there are several different brands of sensors available to measure ambient air temperature, but they lack a standard application programming interface (API) to collect this information, or even a standard physical link layer.

Together, these trends form the basis of a ‘‘knowledge plane’’ [98]. This chapter explores new dimensions of a knowledge plane: the role of physical location, spatial and topological relationships, and environmental sensors with respect to facilities infrastructure. A comprehensive monitoring system needs to integrate this information with conventional metrics such as system utilization and performance. Location knowledge is useful to isolate and localize problems that require manual intervention. It can also help to predict failures resulting from environmental conditions, and to improve availability by selecting resources to avoid common failures.

This section explores the role of physical location and object relationships. A key element of our work is a database engine to filter and index sensor data. We store this data in an archive supporting an SQL query interface optimized for selected spatial queries. Our prototype, *Splice*, combines environmental sensor readings with performance measures collected from a standard instrumentation framework such as Ganglia [82], and normalizes readings to a common spatial frame of reference.

6.1.1 Location-Aware Infrastructure Properties

An automated, facilities-aware instrumentation infrastructure must support *location-aware* sensors, represent *object relationships*, store *attribute history*, and permit *data filtering*. These properties allow management components to build optimizing feedback control loops, while minimizing storage space.

Location-aware sensors are necessary because of the physical nature of facilities management attributes. For example, concentrations of hot air near server inlet fans can reduce long-term hardware reliability. Location-aware sensor readings do not have to come from a sensor that is itself location-aware; it is sufficient for a *conduit*—an intermediary that collects and forwards data—to tag each reading it aggregates with location metadata.

The ability to represent *object relationships* is crucial as data centers continue to grow. The infrastructure supports arbitrary relationships between objects, including the power grid, network topology, and server location within racks. For example, if a given power circuit overloads due to server utilization, we can attempt to select replacement servers on the same network segment, but on different power circuits. Even a simple query as to the location and connectivity of a server is useful in large data centers.

Timestamped *attribute history* is a fundamental requirement for management components that construct models of attribute behavior. The manager learns how control settings affect the attribute value, either by perturbing the control settings in a fixed pattern, observing previous behavior, or a combination of the two methods.

Finally, *data filtering* allows managers to establish data retention policies. We enable two classes of filtering: online and postprocessing. Online filtering allows conduits to restrict the amount of data that initially enters the database. Postprocessing can use statistical models to better distinguish between ‘‘normal’’ sensor readings—ones that can be discarded or compressed—and outliers of interest. Combined, these capabilities enable the site to balance the detail and granularity of stored data with the amount of storage space required.

6.1.2 Splice Architecture

Two emerging data center trends guide the design of Splice. First, data centers are increasingly dynamic; equipment is added, reconfigured, or removed. Similarly, each successive generation of equipment offers new capabilities and features. Second, the drive toward larger data centers and consolidation increases the number of measurement points and objects within data centers. Both types of changes also include adjustments to the supporting power, cooling, and network infrastructure.

Therefore, the goals of the Splice data model are to:

- Support multiple data sources, objects, and object properties
- Archive a history of changes to the state of the data center over time
- Scale to large numbers of objects and attributes, and long histories

Figure 17 depicts the structure of Splice. Data sources (or *sensors*) export current values for named metrics of interest, while consumers (or *sinks*) import sensor readings. A network of *conduits* disseminate streams of sensor readings to sinks. The sensor set includes physical sensors—temperature or other environmental conditions that might affect equipment functioning, such as dust, humidity, or electromagnetic noise—and computer system performance metrics.

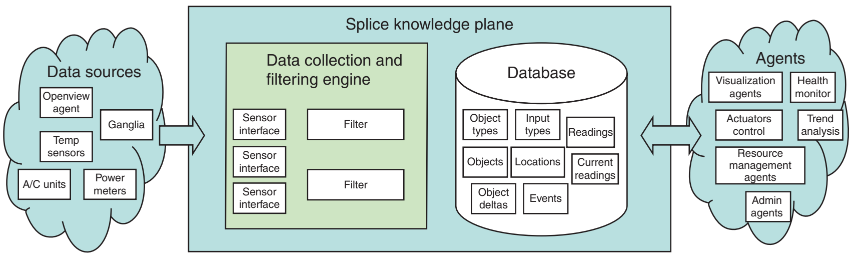


FIG. 17. Splice consists of a data aggregation and filtering engine that interfaces with a dynamic collection of data sources (sensors). It archives a filtered history of sensor readings and other object attributes in a database, and exports an SQL query interface to management and analysis tools (agents).

Splice associates each sensor with an *object* occupying a specific *location* in a three-dimensional coordinate space, and each object with an extensible set of named *attributes* with string values, similar to LDAP or SNMP. Object attributes include a type, a location, and dynamic attributes derived from the sensors currently bound to that object. Administrative tools (agents) may define or update additional object attributes. For example, an administrative agent may populate the database with a data center configuration specified externally using Data Center Markup Language [37].

The Splice data model does not distinguish between the dynamic attributes and configuration attributes normally considered to be fixed, such as hardware specifications, physical relationships among components, and properties related to location (e.g., the power circuit feeding a given system). Splice represents and processes these attributes in the same way as dynamic sensor readings. This principle allows the system to record a history of the evolution of the data center over its lifetime.

Splice exports an SQL query interface to higher-level tools or *agents*. Agents may use the query interface to perform analysis and management tasks. For example, they may monitor status in real time for health monitoring and anomaly determination, or to drive control policies. Agents may query the data along multiple dimensions, involving the history of object sets defined by arbitrary collections of attributes, or activity within specified regions and locations over specified time intervals. Splice is built above a relational database package to handle these queries efficiently; our prototype uses *MySQL*.

6.1.3 Implementation Results

We illustrate the power of incorporating location-awareness and environmental data into a knowledge plane. The HP Labs Data Center (HPLDC) contains approximately 320 servers and 40 TB of storage. The HPLDC is set up to run the entire production load of HP Labs and the computing needs of the individual research groups. The Smart Data Center group [71] instrumented the power and cooling infrastructure, as well as the configuration parameters of all the components.

The HPLDC has four primary data sources:

1. *Power meters*. An OLE for Process Control (OPC) server [67] maintains the 1-min average power consumption for each rack. The server updates power values once every 60 s.
2. *Temperature sensors*. A separate OPC infrastructure monitors temperature through a series of wired sensors attached to the racks.
3. *Utilization metrics*. HP Open View's Service Reporter polls the clients periodically and updates its MS-SQL database with performance measurements, including load averages, memory use, and disk utilization.
4. *Facility configuration descriptions*. We loaded location information into the database from an XML description of the room; the XML document came from a manually constructed and annotated figure of the room.

A workload that consumes large amounts of power at the end of a row of racks can cause the inlet temperatures of some of the other machines in its row to increase. This effect results from hot exhaust air drawn around the end of the row and into these machines; such recirculation is less likely to occur for machines in the middle of the row. Figure 18 illustrates this recirculation effect by showing the inlet temperatures for six racks (F2–F7) arranged in a row. The servers are idle until the 60-min mark. Over the course of 30 min—starting at the 80-min mark—we turn off 10 of the 20 servers in rack F4; this decreases the power consumption of the equipment in rack F4 by approximately 1 kW. We maintain this configuration for 10 min, allowing the data center to achieve thermal steady state.

At the 2-h mark we use Gamut [57] to deploy a CPU-bound workload on the 10 previously off machines in rack F4. This workload consumes well over 1 kW, causing power consumption in the rack to approach 3.5 kW. Ten minutes later, the data center reaches a new equilibrium as the heat from the workload propagates through the room. However, the observed effects from this workload are not constrained to rack F4; racks F2, F3, and F4 register a temperature increase of approximately 1.5 °F, and F5 measures a 0.75 °F increase. While these four racks span a distance of nearly 3 m and the sensors that detect a rise in heat are several meters from the exhaust fans of rack F4, Splice enables us to correlate these observations.

6.2 Temperature Modeling with ConSil

A fundamental prerequisite of integrating power and thermal concerns into a feedback control loop is accurate and complete information to drive the management policy. A crucial component is a detailed *thermal map* of the data center, which contains

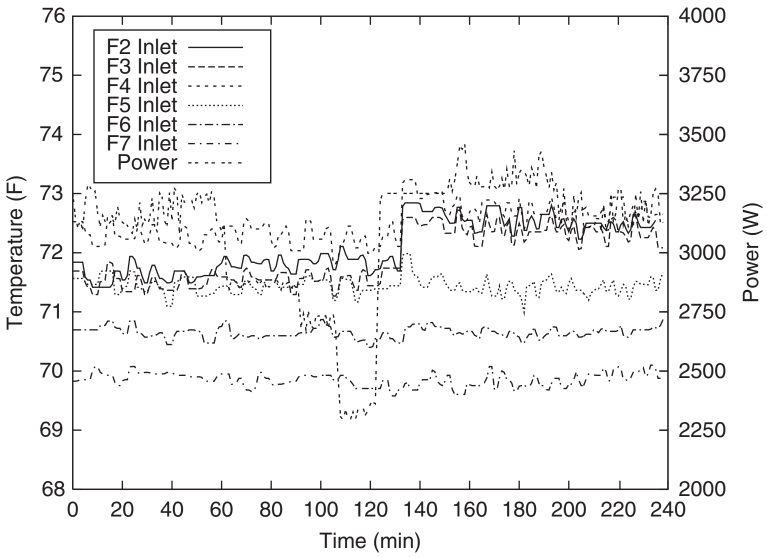


FIG. 18. The inlet temperatures of six racks arranged in a row when we deploy a high-power workload at the 120-min mark on 10 of the 20 machines in rack F4. Within 10 min the heat from this workload recirculates within the data center and causes a 0.75 °F–1.00 °F increase in the air temperature at the inlets of servers in four adjacent racks.

temperature and airflow information at a fine-grained resolution. For example, recent work in data center thermal management reveals that maintaining a low inlet temperature at each server is more important than creating a ‘‘balanced’’ thermal profile or avoiding the creation of localized concentrations of heat [61, 85]. Implementing these policies requires accurate knowledge of the inlet temperature for every server.

The amount and type of thermal data available through conventional sensors is coarser-grained than that available for application and system performance. Performance data is available from processors, memory subsystems, network devices, and storage devices, in addition to application-level metadata from batch queues, Web servers, and other data center applications. A management agent attempting to control thermal conditions, however, must rely upon sparse or ineffective sensors. The number of ambient temperature sensors is small compared to the number of servers, much less the number of running processes or jobs. Common thermal management practices involve placing at most two or three sensors on the front and rear of each rack. This results in a thermal map of fewer than 150 data points providing information for over 1000 servers. Furthermore, the total cost of deploying these additional sensors can be prohibitive, up to $100 per sensor [40, 71].

Ideally, fine-grained thermal instrumentation would be possible without widespread installation of external sensors. Servers have some internal sensors, most notably those on motherboards that measure the temperature at selected points within the server (Linux has included device drivers for these sensors since 1998 [51]). However, these sensors do not provide a good proxy for ambient air temperature since their values are influenced heavily by local thermal conditions, such as recent processor utilization. Even though some servers contain a temperature sensor near the front inlet, data center owners should not be forced to limit their purchasing options based on this single factor.

This section describes how to synthesize per-server inlet temperature estimates by modeling and then ‘‘masking out’’ local thermal conditions within each server. While local thermal conditions may dominate a single temperature sensor—such as those on top of each processor reflecting recent processor utilization—the readings from multiple sensors over time allow us to model and quantify the effects of any server workload on the sensors.

We use statistical analysis and machine learning techniques to combine existing workload data with multiple internal temperature readings to infer the current server inlet temperature. We demonstrate the effectiveness of this approach by implementing *ConSil*, our prototype modeling software, and building thermal models of two types of servers. With a few hundred data points per server, our models are capable of inferring inlet temperatures within 1.0 °C of hardware-based sensors over 80% of the time, and within 2.0 °C over 98% of the time. This level of accuracy is similar to that of off-the-shelf temperature sensors [20].

Figure 19 depicts the role ConSil fills in our cost-aware data center management architecture. ConSil analyzes data from internal and external thermal sensors and produces an accurate and current thermal map. The map serves as an input to the control policy. ConSil models heat flow within a server, inferring the current inlet temperature based on the amount of heat generated within the server (due to workload) and the amount of heat extracted from the server (due to cooling fans). Splice, presented in the previous section, aggregates these inferences from each server to construct the thermal map.

6.2.1 Problem Statement

At a high level, the heat measured within a server (Q_{measured}) is a function of the heat at the server inlet (Q_{inlet}) and the heat generated by the server's workload (Q_{workload}):

$$Q_{\text{measured}} = f(Q_{\text{inlet}}, Q_{\text{workload}}).$$

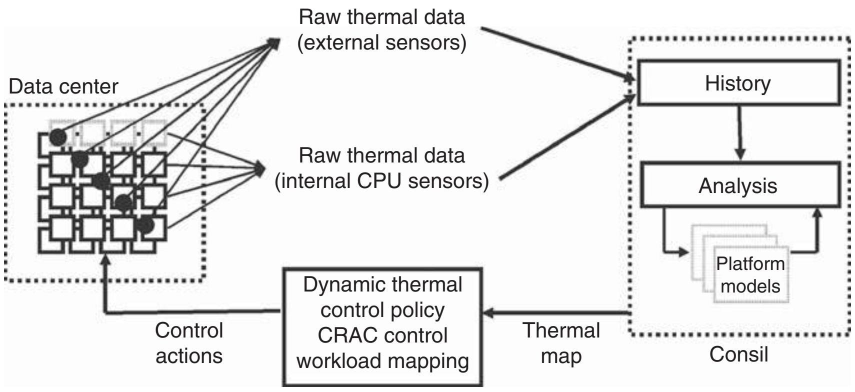


FIG. 19. ConSil combines readings from internal sensors in each server platform with other instrumentation data to produce more detailed thermal maps. The use of internal sensors reduces the need to instrument the data center with external thermal sensors.

This equation omits several details. For example, most servers have multiple internal sensors. The amount of heat generated by the workload and measured by these sensors varies significantly within the server. For example, the values reported by a sensor near a processor are influenced heavily by the activity of that processor over the last few minutes.

Each server has X internal temperature sensors. At each sensor i, the measured heat M_i is the sum of the inlet heat and the amount of workload-generated heat present at that location within the server, H_i. Representing H and M as a vector, we have

$$[M_1 M_2 \ldots M_X] = f(Q_{\mathrm{inlet}}, [H_1 H_2 \ldots H_X]).$$

If we invert this function, we can ‘‘mask out’’ the thermal effects of the workload on that particular server. This enables us to infer the inlet temperature—a value we cannot easily measure directly—from internal sensors and performance data, both of which are easy to obtain from existing server instrumentation:

$$Q_{\mathrm{inlet}} = f'([M_1 M_2 \ldots M_X], [H_1 H_2 \ldots H_X]).$$

To estimate the amount of heat injected into the system, the model must include information about the system utilization and/or the workload. As Section 5 showed, not all power consumption correlates to processor utilization. Therefore, the model may include a variety of system metrics, such as processor performance counters, memory access rates, and disk utilization and throughput. We update the model to include Y of these system utilization metrics. Combining the current measured internal temperatures with the current workload characterization parameters gives us D, the complete set of instrumentation values used as inputs to the model:

$$D = \{[W_1 W_2 \ldots W_Y],[M_1 M_2 \ldots M_X]\}.$$

Finally, the model must address the time dependence of heat flow. Unlike power, which is an instantaneous property, temperature is a function of the current temperature as well as the amount of heat being generated or extracted. Therefore, the model must include recent data in addition to current data in order to accurately infer a workload's effect on internal measurements. The final description of the model thus includes the N most recent data sets at time t, where N represents a time window large enough to encompass lingering thermal effects:

$$Q_{\text{inlet}} = f(D_t, D_{t-1}, \ldots, D_{t-N}).$$

6.2.2 Implementation

This section discusses the sensor infrastructure, machine learning methods, and software libraries used to implement ConSil, our prototype model construction application. At a high level, we are dealing with a model that has $N \times (X + Y)$ inputs—our workload and instrumentation data for each epoch—and one output—the inferred ambient air temperature at the server inlet.

The first step is to collect the data necessary to construct the ConSil model. Splice provides the necessary flexibility and features. Since the model is constructed offline, it is not necessary to aggregate data in real time; it is sufficient to timestamp readings as they pass through a conduit. The input data, workload and internal sensor readings, are available through a variety of standard monitoring infrastructures.

The output data come from sensors that measure ambient temperature at server inlets. While using these sensors for every server is cost-prohibitive complex, this method requires only 10 or 15 sensors per *type* of server. A data center on a three-tiered upgrade cycle thus only needs 30–45 sensors to provide sufficient data for model construction.

The method we select to model heat flow and infer ambient air temperature must have certain properties. It must:

- Produce an output that falls within a continuous range of values
- Represent complex relationships, since the equations governing heat flow and transfer are nonlinear
- Handle a large amount of input data; accurate models require a sufficient variance of input values in the $[N \times (X + Y)]$-dimensional parameter space

- Make ‘‘live’’ inferences quickly; approximate solutions generated in 1 or 2 s are superior to more accurate solutions that take longer

Neural nets are one machine learning method that satisfies the necessary criteria. In essence, training a neural net is how ConSil ‘‘learns’’ the relationship between the collection of input parameters and heat flow, allowing us to infer the ambient air temperature from the given workload instrumentation and internal temperature readings. The strength of this approach is that it allows us to add observations to our model during normal operation of our servers. Furthermore, the more often we run a given workload, and the more unique internal temperature sensor combinations we capture during that workload, the better the model learns the server inlet temperatures during that workload. For example, a host that operates as an application server in a three-tiered Web workload can collect a significant number of unique internal sensor combinations. In turn, the model uses these combinations to infer server inlet temperatures for a wide range of ambient air temperatures without the need to observe every possible inlet temperature.

A model using the N most recent epochs, with X internal temperature sensors and Y system workload metrics, will have $N \times (X + Y)$ inputs to our system. The output is the inferred ambient air temperature.

Between the input layer and the output layer, there are L *internal* or *hidden* layers. Each layer contains a set of elements known as *neurons*. Each neuron i accepts N_i inputs from the previous layer, applies a weighting factor $w_{i,a}$ to each input x_a, and uses the sum of the weighted inputs as the x-value for its activation function, g. The result of this function, y_i is passed to neurons in the next layer:

$$y_i = g\left(\sum_{a=0}^{N_i} w_{i,a} x_a\right).$$

To implement the neural network, we selected the Fast Artificial Neural Net (FANN) off-the-shelf development library [96], which implements standard neural net training and execution functions. Of the three activation functions implemented in the FANN library—threshold, sigmoid, and hyperbolic tangent—only the sigmoid activation function meets the criteria of allowing only positive output values and outputting contiguous values:

$$g(x) = \frac{1}{1 + \mathrm{e}^{-xs}}.$$

To construct the neural net, we select values for the model and implementation parameters and calculate the weights for each input to each neuron to optimize the mean squared error (MSE). The training process continues until the MSE reaches a user-defined minimum threshold or the training process has executed a specified number of iterations. Therefore, MSE is an implementation parameter of interest.

Another implementation parameter of interest is the sigmoid parameter s, which controls the *steepness* of the sigmoid function. An overly steep sigmoid function requires precise inputs at all stages of the neural net to produce accurate outputs; small errors grow as they pass through the network, producing incorrect outputs. A sigmoid function that is ''flat'' may result in an overly trained network. In other words, it can make accurate inferences for inputs similar to previously seen data, but is not general enough to provide accurate answers for new input sets.

The second stage in constructing a single neural net is testing the network. Testing involves using the neural net to infer the outputs for a given set of inputs that were not present in the training data. Testing examines to what extent the neural net is generally applicable, and checks that the training session did not create a net that is overly trained to inputs it has already seen.

The final stage employs *fivefold crossvalidation* (FFCV), a standard statistical analysis technique, to assess the suitability of a model-building process and parameters. In FFCV, the (input, output) tuples are divided into fifths. The learner is trained on four-fifths of the tuples, and the final fifth is used as a test set. This process is repeated five times with a different fifth used for testing each time. For example, if we have data from 10 servers, we break them into five groups of two servers each. The first neural net is trained using data from servers 1–8, and tested on data from servers 9 and 10. The second neural net is trained using data from servers 1–6 and servers 9 and 10; this net is then tested using data from servers 7 and 8, and so on.

6.2.3 Results

We created models for two standard 1U servers, the HP DL360G3 and the Dell 1425. For each type of server we collect data from external temperature sensors, and internal temperature and workload data from those servers whose inlets are adjacent to the external sensors. With this data, we train the neural networks to infer the external temperatures using only internal temperature and workload data. For a detailed sensitivity analysis of the model and implementation parameters' effect on accuracy and training time, see [58].

HP DL360G3. The first model was developed in a data center containing several hundred HP DL360 servers. We identified a dozen servers with external temperature sensors situated directly in front of their front air inlet panels. For a period of 45 h, we collected CPU data at 1 s granularities, internal temperature data at 5 s granularities, and external temperature data when provided by the external sensor infrastructure.

At the time of observation the data center was in heavy use running large computational batch jobs. This provided for moderate variation in both processor utilization and ambient air temperature. Temperatures at the server inlets varied between 20 and 28 °C.

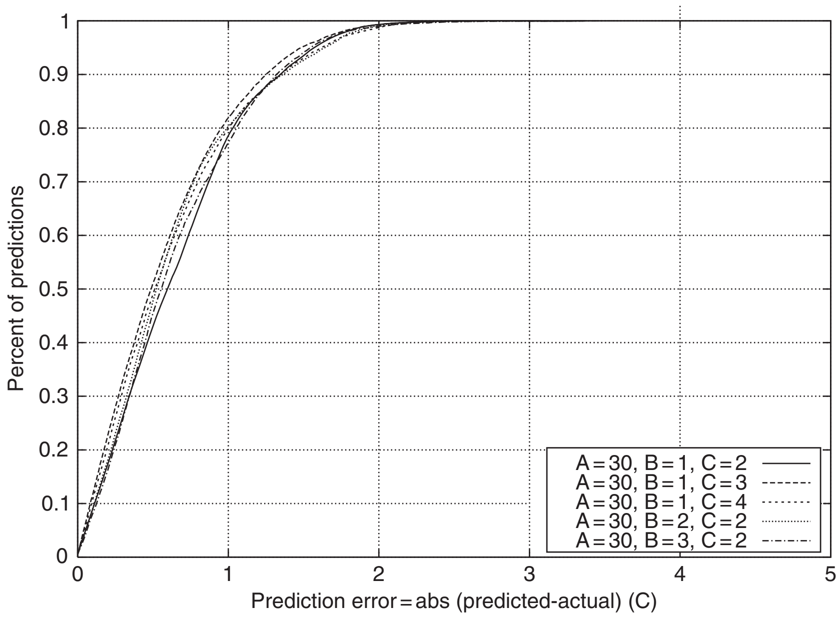


FIG. 20. This graph plots the cumulative distribution function of inference error for five different combinations of workload epochs and internal temperature epochs on the HP DL360 server; the epoch time is 30 s. In each case over 80% of the inferences are within 1 °C of the actual server inlet temperature.

Figure 20 shows the cumulative distribution function of the model's inference accuracy for five combinations of workload epoch length and internal temperature epoch length. The *x*-axis is the absolute value of the difference between the inferred value of ambient air temperature and the actual temperature. In each case, over 80% of the inferences are within 1 °C of the correct value. Over 95% of the inferences are within 1.5 °C of the correct value.

Dell 1425. The second data set comes from a data center containing approximately 500 Dell 1425 servers. We identified 10 servers with external temperature sensors situated adjacent to their front air inlet panels. For a period of 36 h, we collected all data at 30 s granularities. This experiment was performed after the completion of the DL360 results, allowing us to apply the lessons learned from analysis of that data.

Figure 21 shows the CDF of this model's inference accuracy for five combinations of the model parameters. The *x*-axis is the absolute value of the difference between the inferred value of ambient air temperature and the actual temperature.

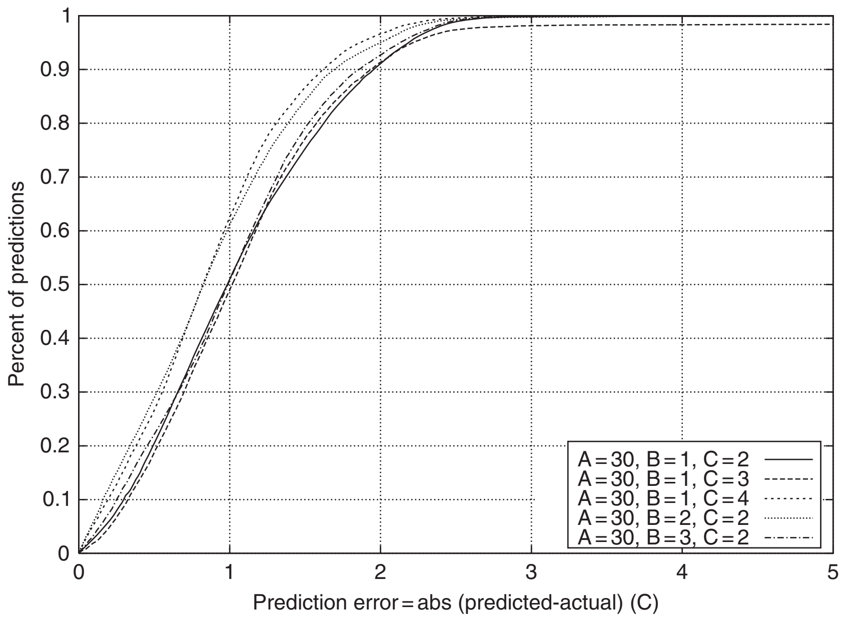


FIG. 21. This graph plots the cumulative distribution function of inference error for five different combinations of workload epochs and internal temperature epochs on the Dell 1425 server; the epoch time is 30 s. In each case over 80% of the inferences are within 1.5 °C of the actual server inlet temperature.

In each case, over 80% of the inferences are within 1.5 °C of the correct value. Over 90% of the inferences are within 2 °C of the correct value.

These results demonstrate that it is possible to deploy a software solution to augment existing coarse-grained hardware temperature sensors. The software solution requires minimal time to train, and produces accurate models that apply to all servers of that type. With widespread deployment of these models, it is possible to construct an accurate thermal map of the entire data center.

6.3 Data Center Thermal Modeling: Weatherman

A key challenge in optimizing data center operating costs is the need to not simply infer, but *predict* the thermal map. Once the thermal map for a configuration is known, it can be used to determine properties such as cooling costs, cooling efficiency, long-term component reliability, and the number of individual servers in danger of triggering their internal thermal ‘‘kill’’ switch.

Predicting a data center’s thermal map relies on understanding the *thermal topology*, which is the description of how and where heat flows through the data center. The thermal topology and its relationship to the thermal map are often complex and nonintuitive. The thermal topology is a function of several factors, including the physical topology of the room, the distribution of cooling, and the heat generated by the individual servers. Furthermore, many of these parameters change during the day-to-day operation of the data center and have nonlinear effects on the thermal topology. Past work on thermal optimizations laid the foundation for thermal management through the use of simple methods. These include using either proxies or heuristics—that is, using the overall power consumption [15] or a single-point temperature [85]—to characterize the ‘‘goodness’’ of the solution, running time-consuming thermodynamics simulations, or conducting elaborate calibration experiments requiring the entire data center to be taken offline to evaluate the thermal map for each configuration [61]. However, as optimizations focus on power and cooling control at a finer granularity [10], it becomes important to formulate better models of the data center thermal topology, predicting the thermal map in real time and at low cost.

This section presents Weatherman, an implementation of our automated, online, predictive approach to thermal modeling. Weatherman can learn the complexities of the thermal topology and predict the thermal map of a 1000-plus-node data center using measurements from day-to-day operations. In experiments, over 92% of Weatherman’s real-time predictions were within 1.0 °C of the actual temperature.

6.3.1 Problem Statement

Before selecting an appropriate technique to model data center thermal topology, we must formalize our problem statement. In this section we define the relevant *model parameters*; that is, parameters that are necessary to construct any thermal topology, independent of the method chosen to implement that model. Section 6.3.2 discusses our specific implementation.

The thermal topology is a function by which we predict the thermal map that will result from a given set of input factors:

$$M = T(I)$$

To formulate a problem statement, we must enumerate the variables in I that affect the thermal topology, and what instrumentation values are sufficient to provide a useful M.

There are three primary input factors, each represented as a vector of values:

1. *Workload distribution* (W), which includes utilization data for any hardware that produces measurable amounts of heat. Servers, storage, network switches, and other hardware fall into this category. In practice, we can obtain this data—including, but not limited to, CPU utilization, disk I/O rates and rotate speed, memory I/O rates, and network activity—from any number of available instrumentation infrastructures.
2. *Cooling configuration* (C) of the room, including the number and distribution of CRAC units, their airflow velocity, and the temperature of the air they supply to the data center. This configuration also includes non-CRAC factors that affect airflow in a data center, including fan speeds of the servers.
3. *Physical topology* (P). The physical topology consists of the objects in the room, including the locations of server racks, walls, doors, and slotted floor tiles.

We make a similar generalization for the thermal map, specifying a set of instrumentation values that provide an accurate representation of the map. This results in our formal problem statement:

$$M = T(W,C,P).$$

The set of values contained in W, C, and P are the input to our function, and the set of values contained in M are the output.

6.3.2 Implementation

The first step in implementing Weatherman is to collect the input data to construct the model. The data include server utilization metrics and CRAC data such as fan speeds and temperature, which is available through instrumentation infrastructures such as OPC [67]. The output data are ambient air temperature measurements from external sensors. All of the data must be tagged with metadata to indicate the object of origin. For input data, this object is the server or CRAC from which the readings came. For output data, it is the server that the external temperature sensor is located directly in front of.

Exact solutions using computational fluid dynamics methods are too complex and time-consuming for online scheduling. As with ConSil, we instead use machine learning techniques. The properties of the desired technique are the same as those described for ConSil, and so we select neural nets as the modeling technique and FANN to develop the models. For Weatherman, the data sets are pairs of power and thermal maps, taken while the data center is at a temporary steady state. For a data center with X workload parameters, Y cooling settings, and Z room layout

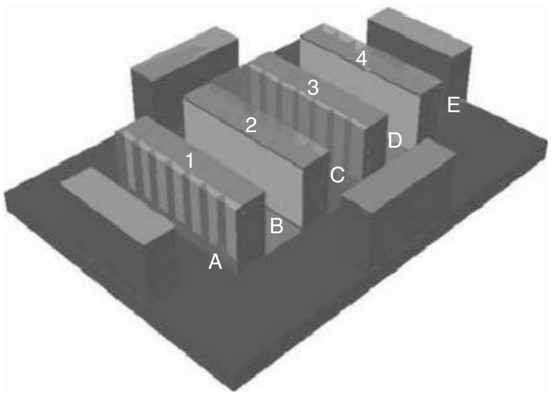


FIG. 22. Data center layout, containing 1120 servers in four rows of seven racks. The racks are arranged in a standard hot-aisle/cold-aisle configuration. Four CRAC units push cold air into a plenum, which then enters the room through floor vents in aisles B and D. Servers eject hot air into aisles A, C, and E.

measurements, there are $N = X + Y + Z$ inputs to the model. The outputs of the model to be constructed are the M measurements that comprise our thermal map.

6.3.3 Results

This section presents the results using Weatherman to learn a thermal topology, in which we demonstrate Weatherman's ability to predict the thermal map resulting from new workload distributions.

We study a typical medium-sized data center, as shown in Fig. 22. The data center contains four rows of seven racks, containing a total of 1120 servers. The data center has alternating ‘‘hot’’ and ‘‘cold’’ aisles. The cold aisles, B and D, have vented floor tiles that direct cold air upward toward the server inlets. The servers eject hot air into the remaining aisles: A, C, and E. The data center also contains four CRAC units. Each CRAC pushes air chilled to 15 °C into the plenum at a rate of 10, 000 ft^3 / min. The CRAC fans consume 10 kW each.

Each 1U server has a measured power consumption of 150 W when idle and 285 W with both CPUs at 100% utilization. The total power consumed and heat generated by the data center is 168 kW while idle and 319.2 kW at full utilization. Percent utilization is measured as the number of machines that are running a workload. For example, when 672 of the 1120 servers are using both their CPUs at 100% and the other 448 are idle, the data center is at 60% utilization.

TABLE XI
THE LIST OF MODEL PARAMETERS (*A*) AND IMPLEMENTATION PARAMETERS (*B*, *C*, AND *D*), AND THE LIST OF POSSIBLE VALUES WE ASSIGN TO THEM DURING TRAINING

ID	Parameter	P_1	P_2	P_3
A	Block size	4	10	20
B	KW scale	200	300	400
C	Target MSE	10^{-5}	5×10^{-5}	2.5×10^{-4}
D	Sigmoid steepness	1×10^{-4}	5×10^{-4}	2.5×10^{-3}

Ideally, to validate accuracy, we would like to compare the thermal map from our model with that from instrumentation of a real data center. Given the costs and difficulties of instrumenting and performing experiments on this sized data center, we instead used the CFD approach discussed earlier with Flovent [27], a commercially available simulator. At the conclusion of each simulation, Flovent provides the inlet and exhaust temperature for each object in the data center. Previous work validated the accuracy of Flovent-based simulations with experiments on a real data center [85].

Table XI specifies the model and implementation parameters we explored in creating Weatherman models; in all, we trained 81 models. For each parameter we attempted to select one value that was overly aggressive and likely to result in an overly trained net, one value that would result in a significantly less accurate net, and one ''ideal'' target value. If our assumptions regarding target accuracy, block size, or scaling were invalid, an analysis of the results would indicate a statistically significant difference in the accuracy of the nets that were trained using those parameters. For a discussion of the sensitivity of these parameters, see [58].

The model we ultimately selected has a 4U block size, a 200 KW scaling value, and small MSE and sigmoid values. This produces a model that is accurate and able to learn how subtle differences in the input values affects the thermal map. Figure 23 shows a scatter plot of predicted temperature value distribution versus the actual distribution for our 280 test experiments (a total of 313,600 data points), while Fig. 24 shows a CDF of the accuracy of our predictions. Over 75% of our predictions are within 0.5 °C, and 92% are within 1.0 °C.

Given that the accuracy of most hardware temperature sensors is within 1.0 °C [20], this demonstrates that it is possible to construct thermal topology models whose accuracy is within the margin of error for off-the-shelf temperature sensors. To our knowledge, ours is the first work to prove that such an approach is feasible, using data available from day-to-day instrumentation.

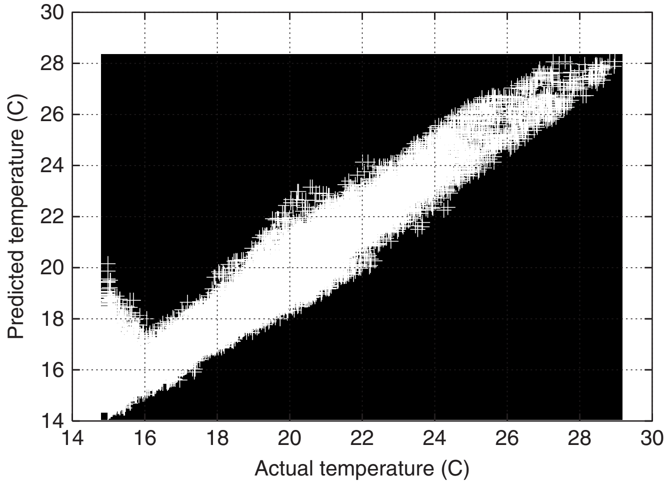


FIG. 23. Scatter plot of predicted values versus actual values. A perfect model would create a straight line with a slope of 1. Our predictions are accurate across the 15 °C range of values.

6.4 Summary

This section explored methods of modeling a complete data center thermal topology. We demonstrate a method by which one may construct these models using existing instrumentation culled from the day-to-day operation of a representative data center. The software used to construct these models uses simple, off-the shelf machine learning modules. The resulting accuracy of these models—predictions were within 1.0 °C of actual values over 92% of the time—shows that even a naive approach is capable of yielding accurate predictions.

Overall, this work demonstrates that it is possible to have accurate, automated, online, and cost-effective thermal topology prediction. Most importantly, it provides the ability to make quantitative predictions as to the results of workload distribution and cooling decisions. As the problem of heat management becomes more and more critical, these and more sophisticated models will be an integral part of cost-aware management.

7. Conclusions

Metrics and models for energy-efficient computing have only recently begun to receive systematic attention, and so the work described in this chapter leaves some questions unaddressed. This section discusses some of these questions and concludes the chapter.

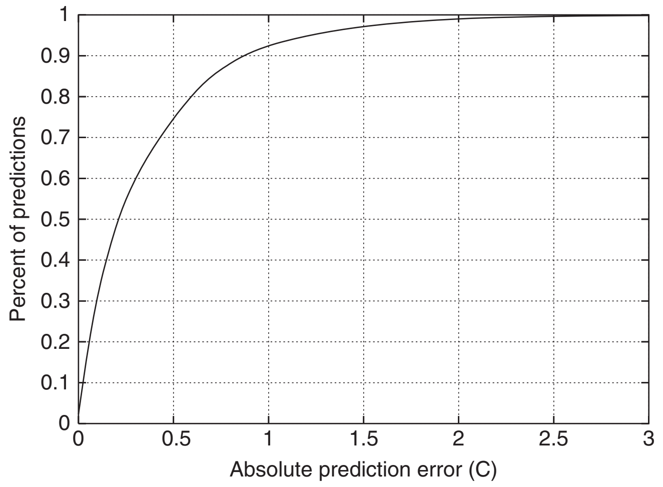


FIG. 24. CDF of the error between actual and predicted values. Over 90% of predictions are accurate within 0.87 °C; the median error is 0.22 °C.

7.1 Future Work

7.1.1 Metrics

First, the question of energy-efficient system design using the JouleSort benchmark has not been fully explored. In designing the CoolSort system for the JouleSort benchmark, we focused primarily on the 100 GB benchmark class. The most energy-efficient systems for the 10 GB and 1 TB benchmark classes may be constructed very differently from CoolSort's mobile fileserver design. At the 10 GB class, ultralow-power components and flash drives are promising technologies, although the question of how best to connect them remains open [80]. For the 1 TB class, a more traditional category of server may be best. The Sun UltraSPARC T1 and T2 processor designs, which maximize memory bandwidth and thread-level parallelism at the cost of processor complexity that is largely unneeded by sort, seem ideal as sorting processors [55]. The question of how to build an energy-efficient sorting system around such a processor, and what class of components to use, has yet to be explored.

Second, since the JouleSort benchmark does not address every possible domain of interest for energy efficiency, additional energy-efficiency benchmarks must be formulated for different domains of interest. The JouleSort workload is I/O-intensive, making it a less useful benchmark in situations where I/O bandwidth is not important. JouleSort also allows systems to run at their most energy-efficient operating point, which is currently peak utilization. However, many data center and server machines are underutilized [6]. The SPECpower_ssj benchmark addresses this area to some extent, but its CPU- and memory-intensive workload is very different from JouleSort's; in addition, it summarizes energy efficiency at 10 different operating points with one number, meaning that the energy efficiency at any particular utilization is unclear from the benchmark score. Finally, JouleSort is not a data center-level benchmark, since it does not include power and cooling; developing a fair workload, metric, and rules for a building-scale energy-efficiency computing benchmark is a challenge that has yet to be addressed.

7.1.2 Power Models

The models investigated in this work were very simple, not only in the number of inputs sampled, but also in the complexity of the equations used to obtain power predictions from utilization metrics. As Section 5 showed, these simple, mostly linear models may not adequately express the behavior of some systems, multicores in particular. Combining the empirical power model for CPU power with the information provided by disk utilization and CPU performance counters is one obvious extension.

Next, the models we investigated assume that the dynamic power correlates to the utilization of CPU, memory, and disk. Systems with other components, such as graphics processors or power-aware networking equipment, would require adjustments to the calibration suite. Furthermore, future power optimizations are likely to pose modeling challenges: in particular, the dynamic power consumption of the cooling system and aggressive power management policies in individual components would have to be visible to the OS and incorporated in the model.

We also observed that these high-level power models are less accurate for machines whose dynamic power consumption is not CPU-dominated. Since the CPU is likely to be a less dominant component in the future [6], it is important to understand how to develop accurate power models for other components. Part of the solution may be to offer high-level interfaces to detailed metrics for these components, analogous to CPU performance counters and their interface libraries.

Finally, while several of the models evaluated in Section 5 have been employed in data center energy-efficiency optimizations, the performance counter-based model has not yet been incorporated into a data center scheduler. Evaluating it in this context would help to quantify the energy-efficiency improvements of increased model accuracy.

7.1.3 Thermal Models

While ConSil's initial results are promising and approximately equivalent to hardware sensors, there is room for improvement along three axes. First, a more complete definition of server workload—beyond CPU utilization—has the potential to better capture the sources of heat within a server. Second, a larger sampling of servers and more training data could provide ConSil with a more thorough representation of the relationship between workload and internal temperatures. Finally, although FANN was used for rapid and easy prototyping, more emphasis on machine learning techniques may improve ConSil's accuracy and thus the accuracy of the thermal map.

Next, the initial work on Weatherman focused on predicting the effects of workload distribution on the thermal map. The models did include CRAC supply temperature and fan speed data as part of the training set, but did not use any input data for which these values varied. Predicting the effects of cooling changes on the thermal map will be difficult for two reasons. First, the magnitude of the effects the CRACs have on the thermal map is significantly greater than even a whole rack of servers. Second, air velocity—and therefore fan speeds—have a nonlinear effect on temperature distribution. More experiments and data will be needed to model these large, nonlinear factors.

Previous Weatherman-based workload placement policies used a relatively simple heuristic search through the possible space of workload placement candidates. In reality, this is an *N*-dimensional space, where *N* is the number of possible servers on which to place workload. Given that this search is complex for even a moderately sized data center of a few hundred servers, the task of finding the global minimum is challenging. One potential research avenue is to explore the application of powerful search and approximation algorithms in an attempt to navigate the search space. This challenge will only grow in importance as both the dimensionality of the problem and the set of possible values for each dimension increases as a result of improved per-server power control features, such as processor voltage scaling, multiple cores, and blade centers.

7.2 Summary

This chapter examined two questions fundamental to energy-efficiency optimizations: the metrics to be optimized, and the models to be used in the design and implementation of these optimizations. In the area of metrics, the JouleSort benchmark and SPECpower_ssj are the first full-fledged system-level benchmarks for energy efficiency; more such benchmarks and metrics are sure to be created and refined as energy efficiency receives increasing attention. To optimize these metrics, models that are accurate and general will be of great interest, and approaches similar to Mantis, ConSil, Splice, and Weatherman will facilitate energy savings in the data center and beyond.

References

[1] Anderson D., Dykes J., and Riedel E., March 2003. More than an interface: SCSI vs. ATA. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*.

[2] Anonymous, April 1985. A measure of transaction processing performance. *Datamation*, **31**(7): 112–118.

[3] Arlitt M., and Jin T., 1999. Workload characterization of the 1998 World Cup Web site. Technical Report HPL-1999-35R1, Hewlett-Packard Laboratories.

[4] Azimi R., Stumm M., and Wizniewski R. W., June 2005. Online performance analysis by statistical sampling of microprocessor performance counters. In *Proceedings of the International Conference on Supercomputing (ICS)*.

[5] Barroso L. A., September 2005. The price of performance. *ACM Queue*, **3**(7): 48–53.

[6] Barroso L. A., and Hölzle U., December 2007. The case for energy-proportional computing. *IEEE Computer*, **40**(12): 33–37.

[7] Barroso L. A., Dean J., and Hölzle U., March–April 2003. Web search for a planet: The Google cluster architecture. *IEEE Micro*, **23**(2): 22–28.

[8] Belady C. L., February 2007. In the data center, power and cooling costs more than the IT equipment it supports. *Electronics Cooling Magazine*, **13**(1): 24–27.

[9] Bellosa F., June 2000. The benefits of event-driven energy accounting in power-sensitive systems. In *Proceedings of the ACM SIGOPS European Workshop*.

[10] Bianchini R., and Rajamony R., November 2004. Power and energy management for server systems. *IEEE Computer*, **37**(11): 68–74.

[11] Bradley D. J., Harper R. E., and Hunter S. W., September/October 2003. Workload-based power management for parallel computer systems. *IBM Journal of Research and Development*, **47**(5/6): 703–718.

[12] Brill K. G., 2007. The invisible crisis in the data center: The economic melt-down of Moore's law. Online: http://www.uptimeinstitute.org/wp_pdf/(TUI3008)Moore'sLawWP_080107.pdf.

[13] Brooks D., Tiwari V., and Martonosi M., June 2000. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*.

[14] Burger D., and Austin T. M., June 1997. The SimpleScalar tool set, version 2.0. *ACM SIGARCH Computer Architecture News*, **25**(3): 13–25.

[15] Chase J. S., Anderson D. C., et al., October 2001. Managing energy and server resources in hosting centers. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP)*.

[16] Cignetti T. L., Komarov K., and Ellis C. S., 2000. Energy estimation tools for the Palm™. In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*.

[17] Clam AntiVirus. Online: http://www.clamav.net/.

[18] Cole G., 2000. Estimating drive reliability in desktop computers and consumer electronics. Technology Paper TP-338.1, Seagate Technology.

[19] Contreras G., and Martonosi M., August 2005. Power prediction for Intel XScale® processors using performance monitoring unit events. In *Proceedings of the International Symposium on Low-Power Electronics and Design (ISLPED).*

[20] Dallas Semiconductor, November 2005. DS1822 Econo 1-Wire digital thermometer, http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2795.

[21] Economou D., Rivoire S., et al., June 2006. Full-system power analysis and modeling for server environments. In *Proceedings of the Workshop on Modeling, Benchmarking, and Simulation (MoBS).*

[22] Embedded Microprocessor Benchmark Consortium (EEMBC). Online: http://www.eembc.org.

[23] Embedded Microprocessor Benchmark Consortium (EEMBC). Energy-Bench™ version 1.0 power/energy benchmarks. Online: http://www.eembc.org/benchmark/power_sl.php.

[24] Energy Star Program Requirements for Computers: Version 4.0, 2007. Online: http://www.energystar.gov/ia/partners/prod_development/revisions/downlo%ads/computer/Computer_Spec_Final.pdf.

[25] Fan X., Weber W.-D., and Barroso L. A., June 2007. Power provisioning for a warehouse-sized computer. In *Proceedings of the International Symposium on Computer Architecture (ISCA).*

[26] Felter W., Rajamani K., et al., June 2005. A performance-conserving approach for reducing peak power consumption in server systems. In *Proceedings of the International Conference on Supercomputing (ICS).*

[27] Flomerics Ltd, 1999. *Flovent. Version 2.1.*

[28] Gonzalez R., and Horowitz M., September 1996. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, **31**(9): 1277–1284.

[29] Govindaraju N., Gray J., et al., June 2006. GPUTeraSort: High performance graphics coprocessor sorting for large database management. In *SIGMOD Conference on Management of Data.*

[30] Green Grid, 2007. The Green Grid data center power efficiency metrics: PUE and DCiE. Online: http://www.thegreengrid.org/gg_content/TGG_Data_Center_Power_Efficiency%_Metrics_PUE_and_DCiE.pdf.

[31] Green Grid, 2007. The Green Grid opportunity: Decreasing datacenter and other IT energy usage patterns. Online: http://www.thegreengrid.org/gg_content/Green_Grid_Position_WP.pdf.

[32] Green Grid, 2008. A framework for data center energy productivity. Online: http://www.thegreengrid.org/home/White_Paper_13_-_Framework_for_Data_Ce%nter_Energy_Productivity5.9.08.pdf.

[33] Gurumurthi S., Sivasubramaniam A., et al., February 2002. Using complete machine simulation for software power estimation: The SoftWatt approach. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA).*

[34] Heath T., Diniz B., et al., June 2005. Energy conservation in heterogeneous server clusters. In *Proceedings of the Symposium on Principles and Practice of Parallel Programming (PPoPP).*

[35] Heath T., Centeno A. P., et al., October 2006. Mercury and Freon: Temperature emulation and management for server systems. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).*

[36] Hitachi Travelstar 5K160 Datasheet. Online: http://www.hitachigst.com/hdd/support/5k160/5k160.htm.

[37] Howes T., and Thomas D., 2003. Gaining control of complexity: The DCML standard for the data center. Online: http://www.dcml.org/.

[38] Huang X., Huang B., and Song B., 2006. Bytes-split-index sort (BSIS). Online: http://research.microsoft.com/barc/SortBenchmark/BSIS-PennySort_2006.pd%f.

[39] Gartner, Inc., press release, *Gartner says 50 percent of data centers will have insufficient power and cooling capacity by 2008*, November 2006.

[40] Intanagonwiwat C., Govindan R., and Estrin D., August 2000. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Conference on Mobile Computing and Networking (MobiCom)*.

[41] Intel® Core™ 2 Duo Mobile Processor Product Brief. Online: http://www.intel.com/products/processor/core2duo/mobile_prod_brief.htm.

[42] Isci C., and Martonosi M., December 2003. Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proceedings of the International Symposium on Microarchitecture (MICRO-36)*.

[43] Joseph R., and Martonosi M., August 2001. Run-time power estimation in high performance microprocessors. In *Proceedings of the International Symposium on Low-Power Electronics and Design (ISLPED)*.

[44] Jung J., Krishnamurthy B., and Rabinovich M., May 2002. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In *Proceedings of the International World Wide Web Conference*.

[45] Kadayif I., Chinoda T., et al., June 2001. vEC: Virtual energy counters. In *Proceedings of the ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering (PASTE)*.

[46] Kingston Memory Module Specification: KVR667D2N5/1G. Online: http://www.valueram.com/datasheets/KVR667D2N5_1G.pdf.

[47] Kotla R., Devgan A., et al., October 2004. Characterizing the impact of different memory-intensity levels. In *Proceedings of the IEEE Annual Workshop on Workload Characterization (WWC-7)*.

[48] Kumar R., Farkas K. I., et al., December 2003. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[49] Laudon J., November 2005. Performance/Watt: The new server focus. *SIGARCH Computer Architecture News*, **33**(4): 5–13.

[50] Li T., and John L. K., June 2003. Run-time modeling and estimation of operating system power consumption. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*.

[51] LM Sensors Development Team, November 2005. *Hardware Monitoring for Linux*, http://secure.netroedge.com/~lm78/.

[52] Lyman P., Varian H. R., et al., 2003. How much information? Online: http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/.

[53] Malone C., and Belady C., July 2007. Metrics and an infrastructure model to evaluate data center efficiency. In *Proceedings of the Pacific Rim/ASME International Electronic Packaging Technical Conference and Exhibition (IPACK)*.

[54] McCalpin J. D. 1995. STREAM: Sustainable memory bandwidth in high performance computers. Online: http://www.cs.virginia.edu/stream/.

[55] McGhan H., November 1, 2006. Niagara 2 opens the floodgates: Niagara 2 is the closest thing yet to a true server on a chip. *Microprocessor Report*, **20**(11): 1–12.

[56] Meza J., Shah M. A., and Ranganathan P., 2008. An energy-efficient approach to low-power computing. Technical Report HPL-2008-67, Hewlett-Packard Laboratories.

[57] Moore J., 2005. COD: Cluster-on-demand. Online: http://issg.cs.duke.edu/cod/.

[58] Moore J., June 2006. In *Automated Cost-Aware Data Center Management*. Ph.D. Thesis, Duke University.

[59] Moore J., Chase J., et al., 2004. A sense of place: Toward a location-aware information plane for data centers. Technical Report HPL-2004-27, Hewlett-Packard Laboratories.

[60] Moore J., Chase J., et al., February 2005. Data center workload monitoring, analysis, and emulation. In *Proceedings of the Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*.

[61] Moore J., Chase J., et al., April 2005. Making scheduling ‘‘cool’’: Temperature-aware workload placement in data centers. In *Proceedings of the USENIX Annual Technical Conference*.

[62] Moore J., Chase J., and Ranganathan P., June 2006. ConSil: Low-cost thermal mapping of data centers. In *Proceedings of the Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*.

[63] Moore J., Chase J. S., and Ranganathan P., June 2006. Weatherman: Automated, online, and predictive thermal mapping and management for data centers. In *Proceedings of the International Conference on Autonomic Computing (ICAC)*.

[64] Mouton J., 2004. Enabling the vision: Leading the architecture of the future. Keynote speech, Server Blade Summit.

[65] Nedevschi S., Popa L., et al., April 2008. Reducing network energy consumption via sleeping and rate-adaptation. In *Proceedings of the USENIX Symposium on Networked System Design and Implementation (NSDI)*.

[66] Nyberg C., and Koester C., 2007. Ordinal technology—Nsort home page. Online: http://www.ordinal.com.

[67] OPC Foundation, October 1998. *OLE for Process Control Overview. Version 1.0*, http://www.opcfoundation.org/.

[68] Patel C. D., December 2003. A vision of energy-aware computing from chips to data centers. In *Proceedings of the International Symposium on Micro-Mechanical Engineering (ISMME)*.

[69] Patel C. D., and Shah A. J., June 9, 2005. Cost model for planning, development and operation of a data center. Technical Report HPL-2005-107(R.1), Hewlett-Packard Laboratories.

[70] Patel C. D., Sharma R. K., et al., May 2002. Thermal considerations in cooling large scale high compute density data centers. In *Proceedings of the Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*.

[71] Patel C. D., Bash C. E., et al., July 2003. Smart cooling of data centers. In *Proceedings of the Pacific RIM/ASME International Electronics Packaging Technical Conference and Exhibition (IPACK)*.

[72] Perfmon2. The hardware-based performance monitoring interface for Linux. Online: http://perfmon2.sourceforge.net/.

[73] Petitet A., Whaley R. C., et al., 2004. HPL—A portable implementation of the high-performance Linpack benchmark for distributed-memory computers. Online: http://www.netlib.org/benchmark/hpl/.

[74] Pinheiro E., Bianchini R., et al., September 2001. Load balancing and unbalancing for power and performance in cluster-based systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP)*.

[75] Powell M. D., Agarwal A., et al., December 2001. Reducing set-associative cache energy via way-prediction and selective direct-mapping. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[76] Rajamani K., and Lefurgy C., March 2003. On evaluating request-distribution schemes for saving energy in server clusters. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*.

[77] Ranganathan P., and Leech P., February 2007. Simulating complex enterprise workloads using utilization traces. In *Proceedings of the Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*.

[78] Ranganathan P., Leech P., et al., June 2006. Ensemble-level power management for dense blade servers. In *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*.
[79] Rivoire S., Shah M. A., et al., June 2007. JouleSort:A balanced energy-efficiency benchmark. In *SIGMOD Conference on Management of Data*.
[80] Rivoire S., Shah M. A., et al., December 2007. Models and metrics to enable energy-efficiency optimizations. *IEEE Computer*, **40**(12): 39–48.
[81] Rosenblum M., Herrod S. A., et al., Winter 1995. Complete computer system simulation: The SimOS approach. *IEEE Parallel & Distributed Technology*, **3**(4): 34–43.
[82] Sacerdoti F. D., Katz M. J., et al., December 2003. Wide area cluster monitoring with Ganglia. In *Proceedings of the IEEE Cluster Computing Conference*.
[83] Schmidt R. R., Cruz E. E., and Iyengar M. K., 2005. Challenges of data center thermal management. *IBM Journal of Research and Development*, **49**(4–5): 709–723.
[84] Shafi H., Bohrer P. J., et al., September/November 2003. Design and validation of a performance and power simulator for PowerPC systems. *IBM Journal of Research and Development*, **47**(5–6): 641–651.
[85] Sharma R. K., Bash C. E., et al., January 2005. Balance of power: Dynamic thermal management for Internet data centers. *IEEE Internet Computing*, **9**(1): 42–49.
[86] Skadron K., Stan M. R., et al., June 2003. Temperature-aware microarchitecture. In *Proceedings of the 30th International Symposium on Computer Architecture (ISCA)*.
[87] Sort Benchmark Home Page. Online: http://research.microsoft.com/barc/sortbenchmark/.
[88] Standard Performance Evaluation Corporation (SPEC). 2006. SPEC CPU2006. Online: http://www.spec.org/cpu2006/.
[89] Standard Performance Evaluation Corporation (SPEC). 2008. SPECpower_ssj2008. Online: http://www.spec.org/specpower/.
[90] Stanley J. R., Brill K. G., and Koomey J., 2007. Four metrics define data center ''greenness''. Online: http://uptimeinstitute.org/wp_pdf/(TUI3009F)FourMetricsDefineDataCenter%.pdf.
[91] Sullivan R. F., 2000. Alternating cold and hot aisles provides more reliable cooling for server farms. Uptime Institute.
[92] Sun Microsystems, 2007. SWaP (Space, Watts and Performance) Metric. Online: http://www.sun.com/servers/coolthreads/swap/.
[93] Tarjan D., Thoziyoor S., and Jouppi N. P., 2006. CACTI 4.0. Technical Report HPL-2006-86, Hewlett-Packard Laboratories.
[94] Transaction Processing Performance Council. TPC-C. Online: http://www.tpc.org/tpcc/.
[95] Transaction Processing Performance Council. TPC-H. Online: http://www.tpc.org/tpch/.
[96] University of Copenhagen Department of Computer Science, May 2005. *The Fast Artificial Neural Network Library*, http://leenissen.dk/fann/.
[97] U.S. Environmental Protection Agency, ENERGY STAR Program, August 2006. Report to Congress on server and data center energy efficiency. Online: http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Da%tacenter_Report_Congress_Final1.pdf.
[98] Wawrzoniak M., Peterson L., and Roscoe T., November 2003. Sophia: An information plane for networked systems. In *Proceedings of ACM HotNets*.
[99] Weissel A., and Bellosa F., October 2002. Process cruise control: Event-driven clock scaling for dynamic power management. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*.
[100] Yang L., Huang H., et al., March 2003. SheenkSort: 2003 performance/price sort and PennySort. Online: http://research.microsoft.com/barc/SortBenchmark/SheenkSort.pdf.

[101] Ye D., Ray J., et al., October 2006. Performance characterization of SPEC CPU2006 integer benchmarks on x86-64 architecture. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*.

[102] Zeng H., Fan X., et al., October 2002. ECOSystem: Managing energy as a first class operating system resource. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[103] Zyuban V., and Kogge P., August 2000. Optimization of high-performance superscalar architectures for energy efficiency. In *Proceedings of the IEEE Symposium on Low Power Electronics and Design*.

The Emerging Landscape of Computer Performance Evaluation

JOANN M. PAUL

Department of Electrical and Computer Engineering, Virginia Tech, National Capital Region, 7054 Haycock Road, Falls Church, Virginia

MWAFFAQ OTOOM

Department of Electrical and Computer Engineering, Virginia Tech, National Capital Region, 7054 Haycock Road, Falls Church, Virginia

MARC SOMERS

Department of Electrical and Computer Engineering, Virginia Tech[1]

SEAN PIEPER

Department of Electrical and Computer Engineering, The University of Wisconsin-Madison[2]

MICHAEL J. SCHULTE

Department of Electrical and Computer Engineering, The University of Wisconsin-Madison, 1415 Engineering Drive, Madison, Wisconsin

[1] *Current address: United States Patent and Trademark Office, 401 Dulany St., Randolph Building 5A30, Alexandria, Virginia*

[2] *Current address: Viable Inc., 5320 Marinelli Road, Rockville, Maryland*

ADVANCES IN COMPUTERS, VOL. 75
ISSN: 0065-2458/DOI: 10.1016/S0065-2458(08)00804-8

Abstract

Since the dawn of computing, performance has been the dominant factor driving innovation. The underling hypothesis is that there is always more computation that can be done if the computer would be made faster in performing some application or set of applications. Latency and throughput are the two metrics commonly used to model performance. Lower latency for a given application means that the application will execute faster from beginning to end, while higher throughput for a set of applications means that the set will execute faster, again, from beginning to end. Computer architects and designers focus on techniques that reduce latency and increase throughput at all levels of computer design, from the instruction level to the multiapplication level. In this chapter we illustrate how the applications and architectures of emerging mobile, personal computer devices call this focus into question. A sea change is occurring in performance evaluation which requires re-evaluation of computer performance from the perspective of the end user. We develop a taxonomy and include examples to motivate future directions for computer evaluation and design.

1. User—Application Taxonomy

Emerging computers have a categorically different relationship with users now than at any other time in history, affecting not only performance evaluation but the organizational and design principles that go with it. The foundation for this observation can be established by development of a new taxonomy that we will call the U-A (User-Application) taxonomy. The most widely known taxonomy in computing is Flynn's taxonomy [35], which categorizes computer architecture into single instruction, single data stream (SISD), single instruction, multiple data stream (SIMD), multiple instruction, single data stream (MISD), and multiple instruction, multiple data stream (MIMD). Flynn's taxonomy is included in virtually every textbook on computer architecture as a way of introducing how computers split the data and instruction streams as compared to the conventional SISD computer. We further discuss the categories of Flynn's taxonomy in a later section, where we propose a new architecture that falls into the previously limited class, MISD. For our current purposes, we note that while Flynn's taxonomy is a powerful first-order concept for illustrating parallelism, the focus is on the structure of the computer and not the objectives of the computer. Put another way, Flynn's taxonomy is formed from the architecture outward, instead of the end-usage inward. By far, the most common computing structures from Flynn's taxonomy that are realized are that of SISD and MIMD. However, the distinction between the SISD and MIMD architectures is great, and an additional classification scheme for computing seems necessary. Thus, we propose the U-A taxonomy as a way of distinguishing the objective of the computer from its structure. After defining the axes for the U-A taxonomy, we define four new classes of computing and further subdivide them according to their realization as SISD or MIMD computers.

We define the elements around which the U-A taxonomy is formed as:

- Single user (SU)—a computer designed for use by a single individual at a time
- Multiple user (MU)—a computer designed for use by multiple individuals at a time
- Single application (SA)—a computer designed to execute a single application at a time
- Multiple application (MA)—a computer designed to execute multiple applications at a time

We discuss these in more detail before forming the categories of our taxonomy, and then further classify computers that implement the U-A taxonomy as SISD and MIMD structures.

An SU computer is designed with the presumption that only one person will be using the computer at any point in time. By contrast, an MU computer is designed to satisfy the needs of multiple users who share the computer services at the same time. An SA computer is designed to execute a single application at a time. When the single application is complete, the computer moves on to the next application, batch style. By contrast, an MA computer is designed to execute multiple applications at the same time—multiple applications are considered to be executing in a MA system concurrently, even though the concurrency can be achieved a variety of ways.

The objective of SUSA (single user, single application) computers is to process single applications, one at a time, for a single user at a time. Examples of SUSA computers are the very earliest computers which processed individual jobs, batch style, as well as the earliest personal computers which had operating systems (such as DOS) that did no timesharing and did not support multiple windows. These early SUSA computers are SUSA–SISD computers. However, another category of SUSA computers is that of supercomputers. These are SUSA–MIMD computers. Most supercomputers still execute simulations batch style, in which individual users have time on the machine, but they have the entire machine, all of its processing capability, for the duration of their time on the computer. The largest-scale simulations require the largest-scale computers to be dedicated to processing a single application at a time without incurring the overhead of swapping out applications for other users. Interestingly, the SUSA class of computer covers two seemingly opposite ends of the computer architecture spectrum—it categorizes the oldest, simplest kind of computer and the highest-performance parallel computer of any era.

The objective of MUSA (multiple user, single application) computers is to service multiple users who utilize the same application at the same time. The earliest MUSA computers were those used for record keeping, where records were stored in files that were accessed by a common application, for example, database processing. MUSA computers tend to be transaction-based, where the transactions are implemented by a single application that ensures atomic access to the common database. The earliest MUSA computers were database machines where the user's interface was a dumb terminal. These MUSA–SISD machines were dedicated to a single application (like modern-day embedded systems) in that the custom application that they ran was dedicated to the database processing of some organization such banking, accounting, or inventory management. More modern-day MUSA machines utilize Webpage-style access to common information. Applications such as Gmail and Facebook are simultaneously executed by multiple users, but with data exchanged in a more sophisticated client–server model than database systems of the past. For this reason, Web servers are MUSA–MIMD.

The objective of MUMA (multiple user, multiple application) computers is to permit computing resources to be simultaneously shared by multiple users with potentially entirely different computing objectives. The earliest MUMA computers were realized by time-sharing operating systems on SISD computers, thereby alleviating the need for users to enter jobs into a batch queue. Time-sharing operating systems on MUMA–SISD computers provide fair access to limited computing resources at the expense of the additional overhead incurred by swapping out multiple jobs (processes) executing on the system concurrently. Unix-style operating systems permit time-slicing jobs on workstations, personal computers, and even laptops, so that multiple users with multiple login IDs can share the same resources. Modern MUMA–MIMD computers arise when users store their data remotely on a server that is shared by multiple users. For example, Google now permits users to store not only mail but files and pictures on their servers, where the sorting and editing of the files and pictures is done by remote processing on the server.

The objective of SUMA (single user, multiple application) computers is to service a single user who simultaneously executes a variety of applications. The earliest SUMA–SISD computers were realized by the backgrounding of jobs executing on Unix-style operating systems. Later, windows-based operating systems permitted the user to simultaneously process multiple applications on their personal computing device with a much friendlier user interface. Underneath the windows interface, however, the processes still needed to be time-sliced on the computer.

A new category of computing is emerging, that of a SUMA–MIMD computer. The SUMA–MIMD computer has the objective to execute multiple applications concurrently on a multiprocessor in the service of a single user. Examples of SUMA–MIMD machines include personal computers and laptops with multiple cores. However, current two and four core computers have yet to realize the full potential of SUMA–MIMD computing because they have not broken free of traditional programming models—and performance evaluation. Because personal computing devices are increasingly designed to meet portability demands (on size and power consumption), SUMA–MIMD computers are starting to be realized as heterogeneous multiprocessors. For example, the iPhone contains a variety of ARM processors [9, 10, 76]. As SUMA–MIMD computers increase in computational capacity they tend to do different things at different times and take on postdesign time functionality—and so they are not pure embedded computers, nor are they general-purpose programmable computers.

SUMA–MIMD computers represent a class of computer for which performance is not always dominated by latency and throughput over the application set. The most obvious way to see this is that the user's perceptions are inherently limited, so that more computing power does not always impact the user's ability to perceive it. For example, graphics and audio each have points of diminishing returns, because human beings cannot perceive the effects of any additional quality. Furthermore, when humans juggle sets of applications where data is arriving in real time, they can only pay attention to so much information in a fixed amount of time. As a result, faster processing of some tasks is wasted on the user—who is the ultimate judge of performance.

Latency over one or more job types is an important performance metric for SUSA and MUSA computers, while throughput is important for MUMA computers. For SUMA–SISD computers, latency and throughput of the single CPU are also presumed to be the dominant factors limiting computer performance, simply

TABLE I
EXAMPLES OF THE U-A TAXONOMY WITH SISD VERSUS MIMD COMPARISONS

		SU	MU
SA	SISD	PCs, laptops running DOS	Early database systems
	MIMD	Supercomputers	Web servers
MA	SISD	PCs, laptops running windows	Timeshare systems
	MIMD	Emerging mobile computers	General-purpose servers

because the single CPU is a factor in every job the system executes. For SUSA and MUSA systems, lower latency and higher throughput translate to improved performance because their performance is correctly evaluated using an aggregate model of their workload.

By contrast, the variety of applications that execute on SUMA–MIMD need not execute on the same CPU, nor even the same CPU type; aggregate models of workloads are not appropriate. At the same time, workloads on SU machines are becoming more situational, defined less by keeping up with a constant workload, and more by responding to a variety of situations that arise from the intersection of user preferences and the data arrival from the Internet. SUMA–MIMD computers are an emerging category, with implications on performance evaluation as well as computer design and organization.

Table I shows the U-A taxonomy with the examples previously discussed. While dominant examples exist, each category represents a broad subclass of computing. Similarly, SUMA–MIMD computing is a broad subclass of computing. An exhaustive examination of all types of SUMA–MIMD computers, and especially the broader class of SUMA computers, is beyond the scope of this chapter. Here, we seek to illustrate, through examples, how SUMA computing results in novel organizational principles for performance evaluation and design.

2. Perspective

In his bestselling book, *The World Is Flat: A Brief History of the Twenty-first Century* [36], Thomas Friedman draws the conclusion that we are at the *beginning* of the IT revolution. He further implies that developing nations are adopting newer technologies faster than developed nations, because they do not have the burden of existing infrastructure. An analogy can be drawn to existing research communities in computing [67]. An ability to re-examine existing and develop new foundations is perhaps more important in computing than in any other field, since computing is at the foundation of virtually all of our information exchange, while at the same time being the single most rapidly changing technology. Perhaps the most fundamental of all computing foundations is that of performance evaluation.

The most fundamental of all speedup observations is Amdahl's law [5] which states that the performance improvement realized by using a faster mode of execution is limited by the fraction of time the faster mode can be used [41]. Amdahl's law is one of the few, fundamental laws of computing, taught in every undergraduate computer architecture course. Amdahl's law is a truly elegant law that seems inviolate. However, it also contains a key assumption: that a performance improvement in one part of a computer system will not negatively affect the performance of some other part. However, as Fig. 1 shows, narrowly focusing on latency (and throughput) improvements cannot only be wasteful, but harmful to overall performance [68]. The leftmost bar of Fig. 1 shows three regions of execution that are executing serially from top to bottom for a total latency of L_0. The regions are R_1, f, and R_2. f is the sequential fraction described by Amdahl. According to Amdahl, if the sequential fraction goes to zero, the performance of the system is limited to an improvement of $L_A = L_0 - f$, which is shown in the middle of the figure. The clear implication is that any performance improvement in the system that will lead to the improvement of the execution time of the sequential fraction will not negatively impact the execution time of the regions, R_1 and R_2.

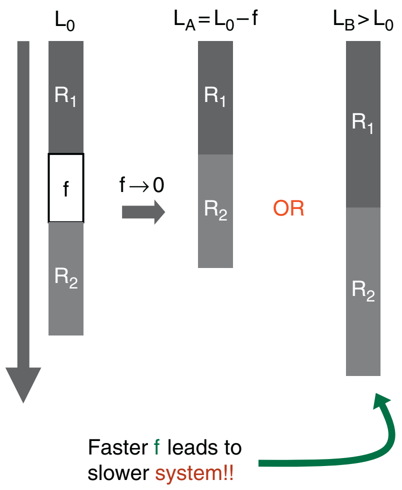


FIG. 1. Second-order effects contradict Amdahl's law.

However, it is relatively simple to show this is not always the case. Significantly, the implication is that the slowing down of one part of a computer system can actually speed up a computer system over the theoretical limit of Amdahl's law. If the performance improvement of the fraction f comes at the expense of the performance of regions R_1 and R_2, the performance improvement of the fraction f can actually slow the system down. This situation is shown in the bar on the right of Fig. 1 where a slower f would have made for a faster system. This counterintuitive observation also runs counter to the way computer design is approached where faster is always better. We encountered this on a heterogeneous multiprocessor when tasks were vying for the resource upon which they would execute the fastest when instead tasks could execute on other processor resources. This effect can also be seen in microarchitecture. For example, when the size of a register file is sacrificed for a floating-point unit that speeds up a fraction of a computation, it can be better to slow down the floating-point operations by providing them less hardware support, so that the size of the register file is not decreased too much.

The observation that secondary effects can negate Amdahl's law is increasingly important as we need to begin to learn how to understand how to effectively design a chip where the design elements are concurrent software tasks executing on concurrent hardware resources. The migration to single-chip heterogeneous multiprocessors (SCHMs) with 10–100 cores will occur over the next few years and is expected to allow exponential increases in performance to continue with minimal reliance on clock scaling [48]. Concurrent software executing on concurrent hardware is not simply ''more of the same'' of existing design principles. Computer design has been dominated by two main models for the past 40 years: instruction set architecture (ISA) for programmable designs and components for VLSI (very large-scale integration) and embedded system design. Neither of these models works well for SCHMs.

The ISA model permits software to be designed separately from hardware because each can be optimized for performance separately. For example, removing lines of execution from a software program for the purpose of making it more efficient (presuming it does not lose functionality in the process), tends to make programs run faster. On the hardware side, making the clock cycle faster or the processor more efficient tends to make programs run faster, overall. By contrast, process or thread models executing on either networked or shared memory multiprocessors provide the same kind of layering as an ISA, but not the same ability to optimize software and hardware independently. Adding or removing either processes (or threads) or processors to a concurrent machine does not give insight as to whether the system will run faster or not.

The component model permits pure hardware to be designed in a hierarchical manner where the level of abstraction can be raised and performance is preserved. This is because hardware is a dataflow paradigm, enforced by wire-like interconnect, resulting in true encapsulation. Consider two interconnected hardware components: A and B. Their behavior—functionality over time—can be specified at the ports independent of the remainder of the system to which they are connected. For hardware components there is no global affectation on the individual components that results from their being interconnected. While hardware underlies all computation, software permits dynamic resource sharing—which is the foundation of great efficiency, especially in meeting a wide variety of situations that cannot be anticipated at design time or for which the worst case design would make the system infeasible. Packet switching is an example of dynamic resource sharing, as is the ISA foundation of computer architecture. Software is layered, with global affectation (resource sharing) of the underlying layer, while hardware is spatially interconnected, permitting partitioning and isolation [70].

Something new is clearly required for SCHM designs which will be at the heart of future mobile computer systems. We have previously described how modern computer usage must be described in terms of *scenarios* consisting of numerous I/O streams, timing information, and parallel tasks that enter and leave the system, rather than in terms of programs executing in isolation from the physical world and each other [71, 73]. Scenario-oriented (SO) computing is in contrast with other well-established styles such as general purpose (GP) and application specific (AS). Table II compares these styles in more detail. Scenarios impact programming models, orientation for design, performance evaluation, and even the structure of simulation and execution models, since the nature of the relationship of inputs to the machine is fundamentally different from that of either general-purpose or application-specific computing [69]. Performance is more about *enabling*, rather than accelerating.

TABLE II
COMPARISON OF APPLICATION-SPECIFIC (AS), SCENARIO-ORIENTED (SO), AND GENERAL-PURPOSE (GP) COMPUTING

	Application specific	Scenario oriented	General purpose
User programmability	Limited or no programmability	Can install software for new functionality	Complete programmability
Design	Excellent performance for a single application	Variety of uses, but performance of some is emphasized	Balanced performance
Performance evaluation	Compared against known requirements	Holistic evaluation of scenario components and their interactions	Each application evaluated individually
Inputs	Timed to external reference	Both timed to external reference and sequenced by applications	Sequenced by application

We have previously described usefulness and timeliness as high-level terms for the new performance evaluation of SO computing [73]. Usefulness indicates the degree to which the device helps the user perform the computing scenario, while timeliness indicates the ability of the device to perform the scenario in a timely manner. Usefulness captures the tradeoffs in complexity of single applications as well as the number of applications that execute in the system at a given time. Most applications have varying degrees of complexity, which translates to overall application quality. Simple speech recognition might have a very limited vocabulary, which is only useful in a given context. Very complex speech recognition is capable of natural language processing. In between is a wide range of algorithmic complexity for speech processing.

Many applications exhibit a wide range of complexity; even email has a simple core set of functionality which can grow in complexity with added features such as search and sort. For computing devices bound by constraints on area and power, a wide range of complexity for a given set of applications is an important consideration in evaluation of overall performance. Simpler application instances can sometimes afford the possibility that the device can integrate more applications of different kinds in a size-constrained device. For example, a traveler might want very simple speech recognition to be incorporated with a more complex email program that uses search when they are in the airport, but prefer to combine more complex speech recognition with simple email when they are in their hotel room. Both systems are doing speech processing and email. However, the form the applications take on is very different.

Timeliness weights the relative importance of the applications as they execute in concert on the machine—it merges the SU property of SUMA machines with the finite nature of the underlying machine. Usefulness models the set of applications the computer carries out in support of the user's mission—each application in that set as well as its overall complexity. Timeliness considers how well that set executes on the machine and can even provide the ability to trade off the performance of some applications in certain situations, compared to that of others. As with usefulness, timeliness considers the behavior of a set of applications, in concert. But timeliness also considers the net effect on the end user. Faster search in email does the user no good if speech recognition is of paramount importance to the user. In addition, different users may prefer to emphasize the response time of different applications. For SUMA machines, the overall experience of the user is what counts—the complexity of the set of applications as well as how they respond to the user's needs on the space- and power-constrained computing device.

In the remaining sections of this chapter, we illustrate two types of SUMA machines. Later, we will speculate about the next frontier in computing by considering a SUMA–MISD architecture which we call spectroprocessing. But first, we develop a type of SUMA–MIMD architecture by examining Webpages as one usage scenario. In each case we consider performance from the outside in—from the single user inward.

3. Workload-Specific Processors

We sought to develop a type of SUMA–MIMD computer by considering the set of applications important for next-generation computing. In so doing, we developed a new type of SUMA–MIMD computer, that of workload-specific processors (WSPs), motivated by the scenario of individual users accessing Webpage-based workloads. Webpages are becoming a *de facto* standard for information exchange and human communication. Mobile device users have incorporated the Internet into their daily lives [61], implying that future mobile device designs need to consider accessing Websites and executing their contents as a central aspect of performance evaluation.

In January 2007, Apple launched its breakthrough iPhone; a cell phone that also serves as an Internet communicator and iPod [7]. Google has updated their mobile homepage for iPhone users for quick and easy access to all of Google apps. In November 2007, Google launched Android, a software stack for mobile devices that includes an operating system, middleware, and key applications. The Google mobile phone (gPhone) is expected to be in the market in the fourth quarter of 2008 [44]. Intel Corporation President and CEO Paul Otellini said that the world is ''going ultra-mobile'' with smaller, more powerful, connected mobile devices that ''deliver a no-compromise Web experience in an ultra-low power device small enough to fit in your pocket or purse.'' He predicted that mobile Internet devices will be the ''next big thing in computing'' [45].

Currently, emerging mobile computing devices are considered ''embedded'' or ''application-specific'' systems. And yet, a mobile device that accesses Webpages is very different from the embedded systems model. When a Webpage is accessed, the tasks are decomposed into smaller task types such as processing of JPEG, Flash, and GIF files.

Figure 2 portrays the input from Webpage applications over time. It shows a collection of jobs of different types that arrives simultaneously as the user accesses a Webpage. The vertical bars in the figure reflect a collection of jobs of different sizes—different working set size or different complexity—but of the same type.

Figure 3 relates the very high-level view of Webpages as benchmarks with the design (the chip). We show processors of different types in each bar on the chip, with the numbers of each projected into the chip. Almost all mobile computing devices have a communication bus connecting the different internal components

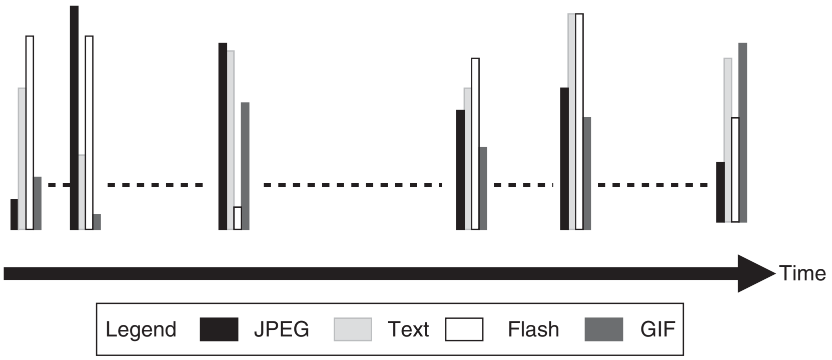


FIG. 2. Bursty nature of Webpage access.

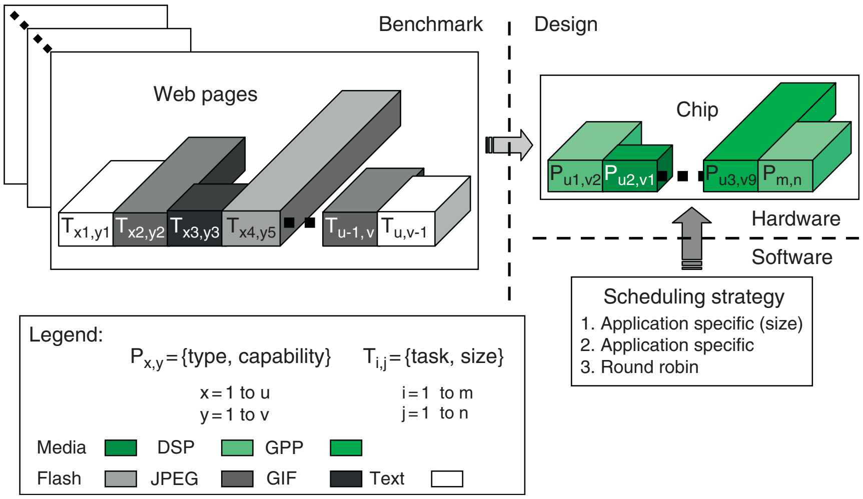


FIG. 3. Webpage interaction with chip architecture.

together [14]. The resultant design, the chip, may be considered a ‘‘processor of processors’’ instead of a processor of register files and functional units. The combination of scheduling strategy, processors, and communications results in an SCHM. The applications are the Webpages. Like the design undergoing evaluation, the Webpages contain different types, again represented as bars, and different numbers and complexities of jobs of the same type.

3.1 Development of WSPs

The concurrent heterogeneity represented by Webpages is well suited for SUMA–MIMD computing, where tasks are already parallel. This is in contrast to the SUSA–MIMD computers, where applications must be parallelized for performance. Significantly, individual tasks need not be parallelized in the processing of a Webpage—tasks come in parallel form, based solely on Webpage content. The performance evaluation of Webpages on SUMA–MIMD computers is not well modeled by existing categories of computer design. We develop the WSP model by contrasting it to that of existing computing models and illustrating its impacts through examples.

3.1.1 Computer Architecture—General-Purpose Processors

General-purpose processors (GPPs) are designed to execute a wide variety of applications well. The reason for this is that the system is intended for general use. Computer architects, therefore, often evaluate system performance by executing individual software programs as they are executed one at a time on the computer architecture. The inputs are not presented to a system in a timed fashion. Rather, they are sequenced by the processor. This is illustrated in Fig. 4, where a downward arrow indicates the beginning of an application and an upward arrow indicates the end of an application.

Each application App$_i$ represents a different type of application. These applications are executed individually—there is no contention between any of App$_1$, App$_2$, and App$_3$. The GPP can be thought of as $P_i = \text{Gpp}(\text{App}_i(D_i))$, or as a function Gpp with an individual inputs App$_i$(D_i), where D_i is in the input data for the application App$_i$, and producing as output P_i. Evaluating for all i yields a set of performance

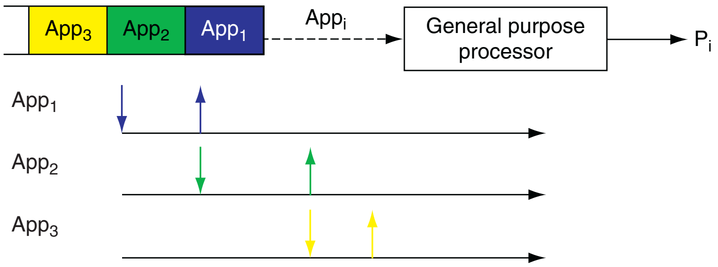


FIG. 4. Computer architecture design.

values, where P_i is typically a cumulative (average) latency value; only recently have power consumption and heat dissipation become issues [16]. Note that speedup is the performance metric commonly used by computer architects, but it is derived directly from individual application latency. The result is one that strikes a balance between selected benchmark applications. In the uniprocessor world, SPEC [84] is a popular set of benchmark applications; SPLASH-2 [91] is the analog in the multiprocessor world.

3.1.2 Real-Time Embedded Systems and Application-Specific Processors (ASPs)

The embedded systems approach is focused on the design of an instance of a computer system given an upfront specification of the application(s) it will carry out. These systems are designed to meet real-time demands of processing an I/O stream, placing emphasis not only on the data received, but on the timing of the data as well. The specified applications are executed repeatedly for long periods of time, conceptually forever, processing many different input sets to produce many output sets.

As Fig. 5 illustrates, with real-time embedded systems usage, a group of applications, App_1, enters the system, executes for some period of time, and completes and restarts. (The figure shows the applications as taking up the execution time of the entire period for simplicity—most real-time systems would be designed with slack.) Embedded systems are designed for periodic and persistent relationships with their environment. P_1 is the execution time of App_1 and so long as the periodic deadline is met (App_1 completes before it is periodically triggered to execute again) it does not matter if P_1 is made faster or not—unless the designer wishes to add to the set, App_1. Performance of embedded computers is specified, rather than optimized.

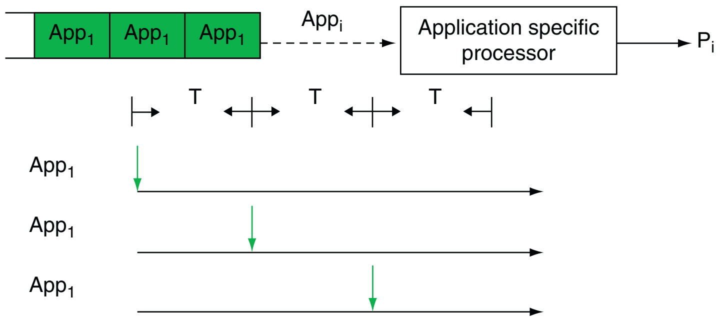


FIG. 5. Real-time embedded systems.

3.2 Workload-Specific Processors

Modern computer usage, where individual users access Webpages, can more accurately be described in terms of *workloads* consisting of numerous I/O streams, timing information, and parallel applications that enter and leave the system, rather than in terms of programs executing in isolation from the physical world and each other [71, 73].

Figure 6 illustrates a WSP. At any given time there is a set of n-applications. Each application potentially executes concurrently. The set of applications that is executing at any point of time, set S_1 at time t_1 has n_1-applications, S_2 at time t_2 has n_2-applications, and so on. Arrival time, missing from the computer architecture approach to system evaluation, is an important aspect of WSP performance evaluation. The loading of the system is a function of time. When executed, each application in an application set S_i processes associated input data from the set. Input data D_j is relevant to application App$_j$. Some applications may be in the system because they were triggered by previous inputs. Each application has input data sets that also change with time $D_j(t_i)$.

Figure 7 depicts a workload example and also different time granularities associated with Webpage contents. A user opens a Webpage, the Webpage has three application types: App$_1$, App$_2$, and App$_3$. The Webpage is loaded at time t_0. Each application has different number of instances; and each instance has different input data. The instances of App$_1$ in this Webpage are four times that of App$_3$, while App$_3$ is, overall, 10 times more complex than App$_1$. Each instance of each application has different time granularities. For example, App$_2$ instances are updated *on the order of* every *day*, while App$_1$ is updated *on the order of* every *second*. The arrival and execution times of these applications may lead to overlap. For example, the second instance of the first application arrived in the system after one second while the same instance (with different input data) is still executing from the first time t_0. Different instances of different applications also overlap in execution.

The key implications of WSP are that:

- Time is an important input to the system; most inputs in modern computer systems arrive from the Internet as a complex byproduct of the user's direction of the computer.

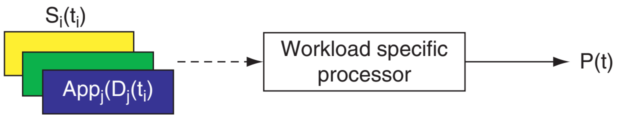


FIG. 6. Workload-specific processor.

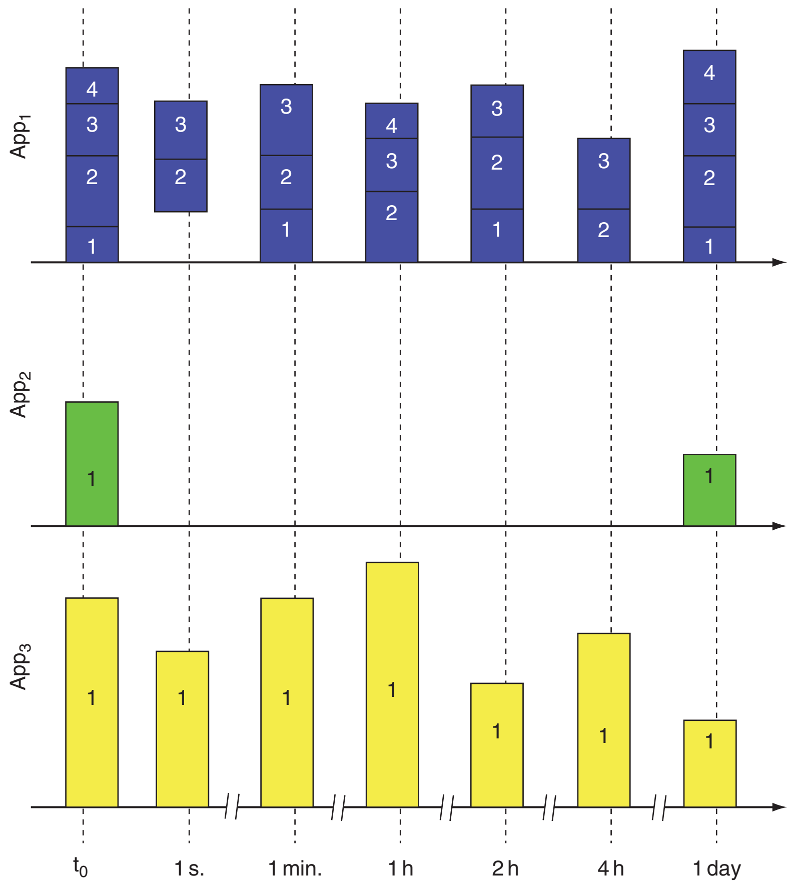


FIG. 7. Workloads and time granularity of App_1, App_2, and App_3, from top to bottom.

- Time is becoming continuous; although distinct execution intervals are easily discerned from the GPP and ASP models, no such intervals exist in WSPs.
- Execution duration can cause overlap; some jobs may persist in the system indefinitely while others may not complete before the next input arrives.
- Performance evaluation is situational; some applications may be more important to process at some times, depending upon both the user's and the processor's context.
- Benchmarks are more accurately modeled by user preference; in contrast to GPP which represents a broad set of applications and ASP which is application

specific, the benchmarks of WSPs are more tuned to individual user preferences, that is, SUMA.

While Webpage-based workloads represent more complexity in terms of content and time in contrast to GPP and ASP, possibilities exist to leverage the SUMA nature of the processing of Webpages.

3.3 Personalized Architectures

SUMA-computing points to a new orientation, away from the design of computers according to the size and speed of their contents (e.g., more memory, larger/faster disk, more processing speed). Instead, computers may well be designed according to the way individuals use them, resulting in a range of computers at the ‘‘top end.’’ For example, a high-end sports car is different from a high-end sports utility vehicle—the costs for each may be the same, but the contents vary greatly according to the way the end result is anticipated to be used. Computers have yet to reach that kind of separation, but we motivate that categorization in what we refer to as ‘‘personalized architectures,’’ a concept for simplifying the broad design benchmark space that results from WSPs.

A number of existing benchmark suites exist, all focused on the evaluation of a single program at a time, whether for general-purpose programming, embedded systems, or even the evaluation of a single application executed concurrently on a parallel computer. The SPEC CPU [84] is the most widely used benchmark by computer architects [40]. Examples of ‘‘embedded system’’ benchmark suites include MiBench [40], MediaBench [55], and EDN Embedded Microprocessor Benchmark [31]. MediaBench focuses on complete applications for multimedia and communications systems [15, 55]. Most of the previous work in constructing workloads has focused on MUMA–MIMD, server workloads [2, 20, 21, 32, 34, 43, 54], or else targeted SISD–MIMD supercomputers [23, 28, 29, 49]. By contrast, SUMA–MIMD workloads have distinct arrival times focused on individualized user preference. Several experts observe that what is extremely valued by one group of Web users is not valued by others [51].

We focused on a preliminary investigation of how individualized user access to Webpages can affect computer architecture. As we will show later, virtually all Websites have dramatically increased in complexity and speed of update over the past several years with this trend expected to continue. However, differentiation can also be expected to result between classes of Websites and this differentiation can also be expected to increase as users increasingly use computers as their personalized interface to the outside world. We developed a model of Webpages, architectures, and schedulers in Somers and Paul [83] to fill out the major elements of Fig. 3 and examine the impact of categorizing Webpage-based benchmarks by user preference on WSP designs.

3.3.1 Modeling

Most Webpages are composed of three basic elements: text/scripts, images, and movie/animated Flash applications. A Webpage also consists of links to many objects that need to be downloaded and processed to view a Webpage correctly. Consider the news Websites for BBC.com and CNN.com. Despite providing similar information, these Websites differ in the content that each Webpage provides. Both of these Websites, however, are more text-based oriented than the other Websites being analyzed. Next, consider sporting news Websites such as ESPN.com and MLB.com. ESPN is a general sporting news Website while MLB is more specialized for a particular sport and organization. ESPN tends to provide headline stories on its main page while MLB provides links to stories and allows users to follow baseball games in progress. Both have approximately the same number of image files but ESPN has more Flash files, although they are smaller sized Flash files.

There are three types of movie/animated Flash applications. The first type is an MPEG type that the Website provides a link to so users can download the packets and view the movie file. The second type is a Flash movie very similar to the MPEG-type movie with frames and audio, but instead it is built with Flash animation and is typically executed when a user visits a Website. The last type is a still image Flash frame. These still images tend to be in a rotation pattern beginning with a still image and then proceeding to the next image of a series of images after short delays. The still image Flash files are similar to regular images in processing. Instead of multiple frames and audio, these files tend to have a singular frame with no audio simplifying the task of processing these types of files. MPEG and animated Flash files take hundreds of times longer to process than all other files on a Website collectively [88].

Table III shows a summary of job types and sizes by Webpage category that we collected for our model of Webpages. The majority of Webpage content tends to be media oriented as in image or movie/animation files, a trend which can be expected to continue.

Table IV depicts an approximation of the various file types and sizes with the Webpages. The elements in the Webpages provide insight into categorizing the Webpages to aid in the Webpage utilization portion of the experiments, discussed later. Notice the differences between the Webpages that provide similar services.

3.3.1.1 Architectures.

Our objective for WSP design is to investigate the impact of single-chip, MIMD architectures in the context of SUMA-style workloads. Thus, two key properties of our architecture investigation are that the chip is bound in overall area and that power is an important consideration in addition to performance. Based upon the job types that dominate virtually all Webpages, it seems reasonable to select three categories of processor types to simulate: a media

TABLE III
STATISTICAL DATA OF WEBSITES (COMPILED USING [88])

	BBC	CNN	ESPN	MLB	EDU
Total objects	214	414	92	75	124
Total size (KB)	269	438	343	116	394
Total JPEG	10	8	10	5	24
Total GIF	193	390	61	57	86
Total Flash	0	0	10	3	1
Total Text	11	16	11	10	13
Media content (%)	94.9	96.1	88.0	86.7	89.5

TABLE IV
WEBPAGE CATEGORIZATION

File types/sizes	BBC	CNN	ESPN	MLB	EDU
JPEG > 50 KB	0	0	0	0	2
JPEG > 10 KB	0	1	2	0	2
JPEG < 10 KB	10	7	8	5	20
GIF > 50 KB	0	0	0	0	0
GIF > 10 KB	0	1	1	1	0
GIF < 10 KB	200	380	60	56	16
Flash > 50 KB	0	0	1	2	1
Flash > 10 KB	0	0	4	1	0
Flash < 10 KB	0	0	5	0	0
Text > 50 KB	1	0	0	0	0
Text > 10 KB	2	6	6	1	2
Text < 10 KB	8	10	5	9	11

processor, a digital signal processor (DSP), and a general-purpose processor (GPP). GPPs have the largest variety of individual chip makers and types ranging from a few GHz to low 100 MHz clock speeds. DSPs are second in terms of choice selection followed by the media processor. The processors considered were determined by the availability of benchmarked data [31]. Still, each benchmarking organization uses their own in-house scoring scheme and not all processors can be compared against all attributes. The processors we used in our experiments are referred to as media, DSP, and GPP processors where Philips PNX1700 is the media processor, Analog Devices ADSP Blackfin533 is the DSP, and the AMD K6-2E+ is the GPP [4, 6, 64]. All processors were geared toward an embedded system, thus making these exact processors inadequate for use in a mobile device such as a cell phone due to size and power requirements. Detailed information that would allow comparison of the range of processor types used in actual cell phones is proprietary, and our goal was to cover a heterogeneous design space. However, relative performance of these processors can be related to the relative performance of processors used in emerging mobile devices. Table V has the relative performance for each task on every processor normalized to the processor with the lowest performance for that task, while Table VI has the overall relative performance encompassing all the tasks normalized to implementing MPEG on the AMD K6-2E+ processor. Note that higher numbers mean faster execution.

The media processor is the best for images and movies, especially when those media files need to be processed in large quantities. Although consuming more power, the GPP is the second best choice for image files, interesting since the DSP barely outperforms the GPP for movie files. Both the DSP and GPP are significantly faster for text processing than the media processor, the GPP is the fastest.

The processors are compared in terms of relative area and power in Table VII. The GPP is the worst processor in terms of size and power consumption. It is three times larger than the media processor and two times larger than the DSP processor. In terms of power consumption, the GPP consumes five times more power than the media processor and almost 20 times more power than the DSP processor. Since the GPP processor is the best processor with respect to text processing by being twice as fast as the DSP processor and five times as fast as the media processor, the GPP processor is needed to achieve the best system runtime performance. However, the quantity of GPP processors available in the system and the tasks to be scheduled for the processor should be limited to consume less energy. The tables also suggest that

TABLE V
RELATIVE PERFORMANCE FOR EACH TASK

Task type	ADSP-BF533	PNX1700	AMD K6-2E+
JPEG	1	8.93	1.81
Text	2.936	1	5.015
MPEG	1.28	4.383	1

TABLE VI
OVERALL RELATIVE PERFORMANCE

Task type	ADSP-BF533	PNX1700	AMD K6-2E+
JPEG	14.294	127.642	25.868
Text	17.536	5.973	29.952
MPEG	1.28	4.383	1

the media processor is essential for a mobile device due to its small size and excellent performance for image and movie files. The only disadvantage for the media processor is its inability to deal well with text processing. However, supplementing a media processor with DSP and GPP processors could prove invaluable in creating an architecture that achieves great performance while minimizing area and power.

Table VIII displays the comparative values of the different SCHM architectures we tested. The architectures were created based on an equivalent size of the video iPod architecture, which encompasses approximately eight relative size units. Thus, we normalized our processors to what we believe would be one eighth of an iPod for one size unit. Also shown are relative and absolute power values. Thus we investigated a total of 3 homogeneous architectures, 7 two processor-type architectures, and 2 three processor-type architectures. We presumed shared memory in all cases—further experimentation could differentiate on the basis of communications as well.

TABLE VII
RELATIVE AREA AND POWER CONSUMPTION COMPARISON

Processor	Relative area	Relative power
AMD-K6E (500 MHz)	3.3998	19.8571
PNX1700 (500 MHz)	1.0000	4.0286
ADSP-BF533 (594 MHz)	1.5772	1.0000

TABLE VIII
ARCHITECTURAL COMPARISONS

Architecture	Power (W)	Relative power	Relative size	Area (mm^2)	Relative area
2 GPP	27.8	7.94	6.80	4957	1.04
5 DSP	3.5	1	7.89	5749	1.20
8 Media	22.6	6.45	8	5832	1.22
2 GPP, 1 Media	30.6	8.75	7.80	5686	1.19
1 GPP, 2 DSP	15.3	4.37	6.55	4778	1
1 GPP, 2 DSP, 1 Media	18.1	5.18	7.55	5507	1.15
1 GPP, 1 DSP, 3 Media	23.1	6.59	7.98	5815	1.22
1 GPP, 4 Media	25.2	7.19	7.40	5394	1.13
4 DSP, 1 Media	5.6	1.61	7.31	5328	1.12
3 DSP, 3 Media	10.6	3.02	7.73	5636	1.18
2 DSP, 4 Media	12.7	3.62	7.15	5216	1.09
1 DSP, 6 Media	17.6	5.03	7.58	5524	1.16

3.3.1.2 Schedulers.

A total of four scheduling strategies were employed. The schedulers were round robin (RR), static (SS), application-specific scheduler that knows job sizes (ASJ), and application-specific scheduler that does not know job sizes. For the application-specific scheduler that does not know job sizes, the order that threads (representing jobs) were created influenced the placement in the job queue. The application-specific scheduler, where threads were created from biggest to smallest was called ''application-specific big'' (ASB) while the scheduler with threads created from smallest to biggest was called ''application-specific normal'' (ASN).

More sophisticated scheduling requires additional cost/overhead. Since program memory is typically much smaller than data memory for these kinds of systems, a conservative estimate would be that an architecture with three different processor types uses approximately three times the memory of a homogeneous architecture with the same number of processors. Other work indicates that the cost of storage for more complex scheduling strategies can be ignored [22]. Most authors that have covered scheduler overhead agreed that the scheduler's computational overhead is insignificant versus the computational demands of the applications. We model the overhead associated with the application-specific schedulers as estimated from [93]. Table IX shows the scheduler overhead.

The concept of overhead, in general, is interesting to consider in the context of a new category of computing. Previously, architecture has been the art of the effective placement of overhead—caches and branch predictors can be considered to be overhead in microarchitecture, but it is overhead that greatly facilitates the performance model of microarchitecture, which is to optimize a set of programs in a benchmark suite for overall latency and throughput. Newer classes of processors may well result in new ways to consider the placement of overhead to facilitate overall performance in the context of SUMA computing for the processing of Webpage-based workloads.

3.2 Experimental Results

Throughout this chapter, we use MESH (modeling environment for software and hardware) because it permits performance and power evaluation when threads execute on sets of heterogeneous resources under a variety of custom scheduling [62].

TABLE IX
SCHEDULER PERFORMANCE OVERHEAD

Scheduler	RR	ASJ	ASB	ASN	SS
Cycles	159	4770	1590	1590	159
Time (s)	5.06×10^{-5}	1.52×10^{-3}	5.06×10^{-4}	5.06×10^{-4}	5.06×10^{-5}

Power is calculated as the per-cycle wattage times the number of cycles the processor executes, averaged over time. MESH is capable of simultaneously modeling and simulating all of the design elements of Fig. 3 at a high level, and also evaluating the timed nature of job arrival as in Fig. 2.

To establish a baseline, we established performance for all architecture candidates by averaging performance over all Webpages being considered. The best performing designs, even for this baseline, have heterogeneous architectures and application-specific schedulers [83]. Next we considered the SUMA-oriented design of WSPs according to individual user preference.

Table X defines various user profiles and their respective Webpage usage frequency. User profiling has been studied extensively in the area of recommendation systems and information filtering systems [85]. A number of approaches have been developed dealing with specific aspects of Web usage mining for the purpose of automatically discovering user profiles. For example, Perkowitz and Etzioni [72] proposed the idea of optimizing the structure of Websites based on co-occurrence patterns of pages within usage data for the site. Schechter *et al.* [77] have developed techniques for using path profiles of users to predict future HTTP requests, which can be used for network and proxy caching. Spiliopoulou *et al.* [86], Cooley *et al.* [24], and Buchner and Mulvenna [18] have applied data mining techniques to extract usage patterns from Web logs, for the purpose of deriving marketing intelligence. Shahabi *et al.* [80], Yan *et al.* [92], and Nasraoui *et al.* [63] have proposed clustering of user sessions to predict future user behavior. Many techniques have been investigated for discovering various Web access patterns from Web usage logs for Web personalization. These include rule-based filtering approaches, content-base filtering approaches [56], collaborative filtering approaches [53], and hybrid approaches [81]. Our profiles, shown in Table X, were developed intuitively. For our experimental purposes it is sufficient that a separation of Webpage access patterns based upon individual user interest exists.

TABLE X
WEBPAGE UTILIZATION (IN %) FOR VARIOUS USER PROFILES

Type of person	BBC	CNN	ESPN	MLB	EDU
International political junkie	90		10		
Political junkie	75	20		5	
Web surfer	20	20	20	20	20
Political and college sports enthusiast	25	20	20		35
Sports fanatic	0	0	75	15	10
Typical college student		65	25		10

Figure 8 shows performance for different Webpage utilizations. We normalized all performance results against the performance of a homogeneous multiprocessor, using GPPs and averaged overall performance (unit performance is for a homogeneous multiprocessor, and smaller numbers mean faster time to process the entire content of a Webpage—here, lower numbers mean faster performance).

Different utilization patterns favor different architectures. While the ''2 GPP, 1 Media'' architecture might seem to be the best overall performer, it performs 20% worse than the next two best performers for pure CNN users and almost 50% worse than the best architecture for pure EDU users. (This architecture also has the highest power consumption.) Even considering only modern-day content in Webpages, personalized architectures can improve performance up to 70% over a homogeneous multiprocessor composed of GPPs with 25% additional improvement over the next best architecture when individual user profiles are also considered.

We next considered the performance-energy product for the static scheduler when the device is constantly in operation, that is, accessing a Webpage frequently enough so that it is never worth incurring the penalty of restart. In this situation, the ''4 DSP, 1 Media,'' is the best performer. However, that same architecture is no better in terms of performance energy than the baseline case of only GPPs for pure BBC users, and is almost the same as the baseline case for the user profile which we dubbed, ''political junkies.'' If a news-oriented person did not have their device on all of the time, unlike a sports enthusiast who follows multiple games, they would do better with a different architecture.

Finally, Fig. 9 explores the performance of the same architectures when using the ASN scheduler. While the ASN schedulers tend to favor, overall, the architecture with ''2 GPP and 1 Media processor,'' the curves intersect the closer a user gets to one who largely visits the MLB or EDU Websites. Significantly, aggressive schedulers that execute jobs on different processor types can offset the performance differentiation seen in Fig. 8. Performance differentiation of designs based upon Webpage access patterns exists, and this differential can be expected to grow as Webpage content becomes more diverse and complex.

3.4 Workload Time Granularity

Personalized architectures address the overall complexity of benchmarks for WSPs by narrowing user preference. A second major implication of WSPs is that time, specifically the arrival time of the workloads, is becoming complex—virtually continuous. We investigated trends in Webpage content, to establish more insight into SUMA workloads and determine how time might be simplified—and leveraged—for new organizing principles in Otoom and Paul [65].

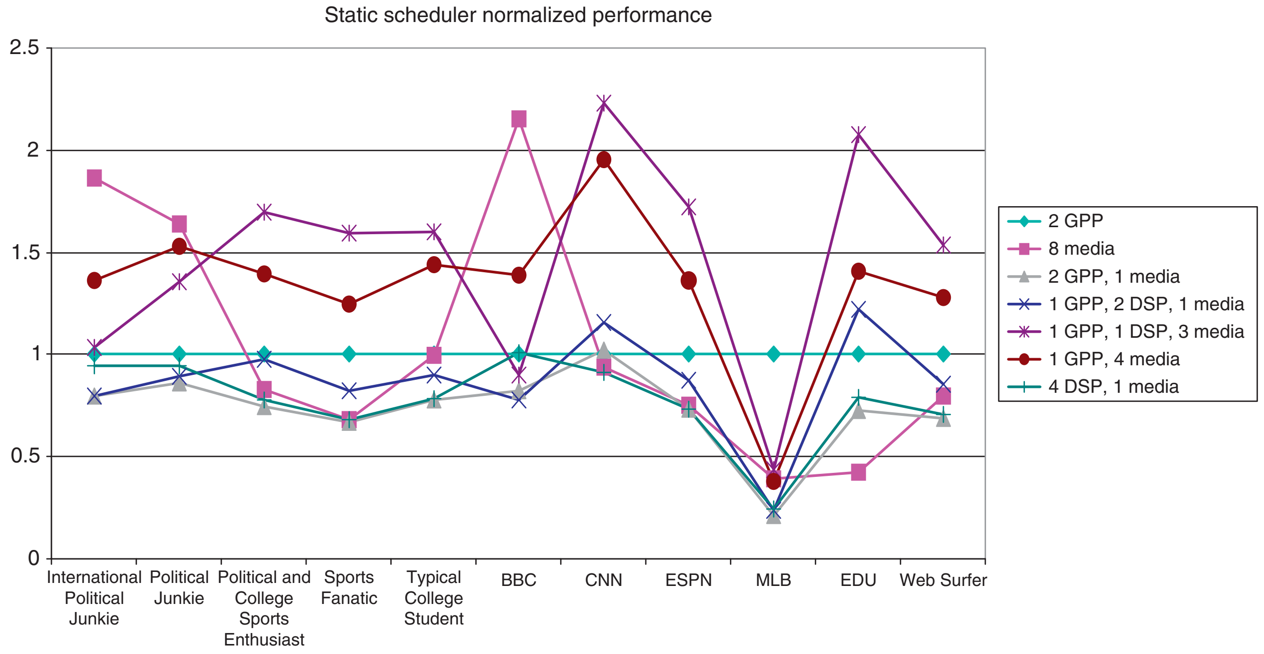


FIG. 8. Performance for webpage utilization.

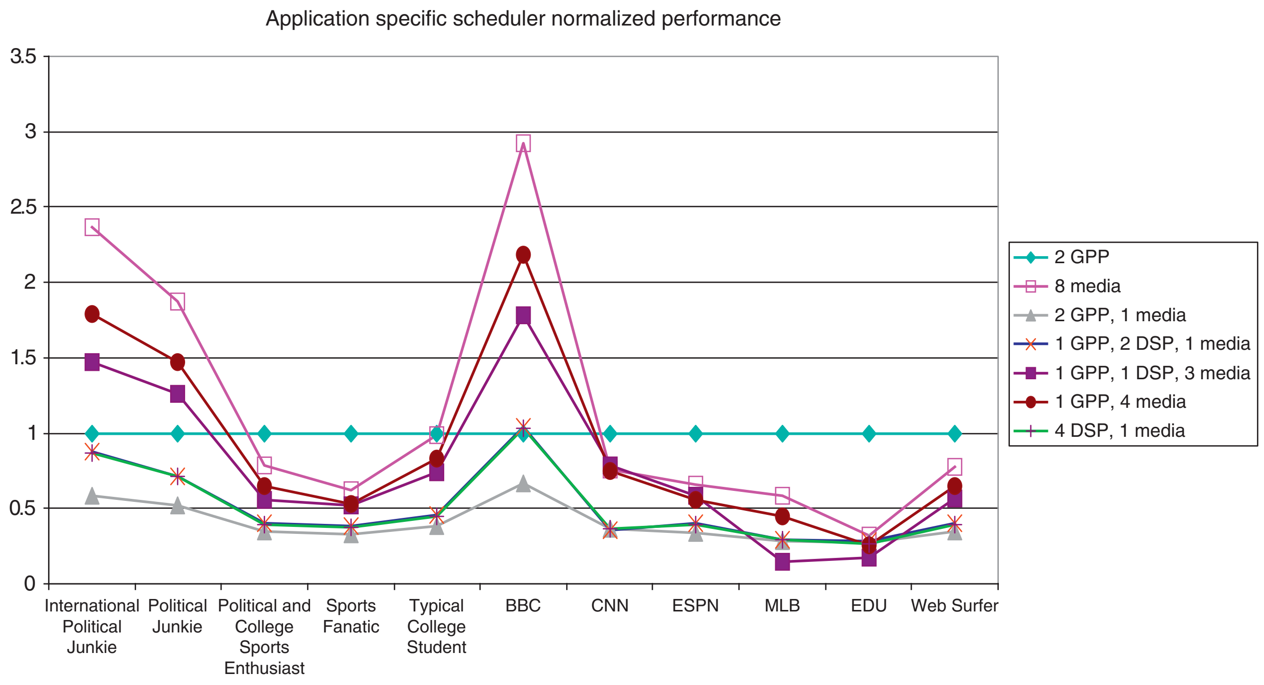


FIG. 9. ASN performance over webpage utilization.

3.4.1 Webpage Trends Analysis

This investigation focuses on one particular Website on the Internet, that of http://www.mlb.com, chosen for three reasons. First, sports is proving to be a leader in creating demand for real-time information exchange in consumer electronics. Second, sports has an interesting combination of news and entertainment, where information changes rapidly, but formatting of information is important to the consumer. Finally, we chose to focus on a single Website rather than many, because past observations led us to believe that a few Websites are harbingers of trends. Our particular focus on http://www.mlb.com was the Flash file used for following a game in progress—which is only available during baseball season when there are games to follow. http://www.mlb.com is a popular Webpage that represents 0.023% of the total daily Internet traffic in the whole world [3]. This Webpage has rich content in terms of diversity, complexity, usability, and usefulness. In our analysis, Internet archive [46] was used to obtain historical Webpages (2001 and 2004). The 2010 projections were extrapolated as a trend from the last three time periods. By investigating trends on http://www.mlb.com, and using conservative estimates, the work of Otoom and Paul [65] is summarized as:

- Webpage contents are increasingly becoming more complex. This is not surprising, as the Internet is increasingly becoming the standard of information exchange and users have incorporated it into their daily lives. The total complexity of 2007 Webpages is more than five times that of 2001 Webpages.
- Multimedia content (still images and movies) is becoming more dominant than text. In 2007, 76% of Webpage content is multimedia which represents 1.5 times the 2001 percentage. In 2007, 57% of the multimedia content was Macromedia Flash (which we refer to simply as Flash in this chapter), but more importantly, that Flash can be expected to dominate the nontext formatting of Webpages even further, as demand for other kinds of nontextual information on Webpages is projected to flatten out by comparison. Image data extracted directly from the outside world, such as pictures and movies, is slower to generate than forms of raw data which can be computer generated or manipulated.
- The number of blocks (typically files of data) on Webpages is decreasing while the number of elements *within* a single Flash file is increasing—Webpage elements like text, links, images, etc., are becoming increasingly encapsulated in Flash formats.
- Webpage contents are becoming more dynamic, overall.
- However, while 42% of the 2007 Webpage is Flash, only 17% of the content of Flash is dynamic.

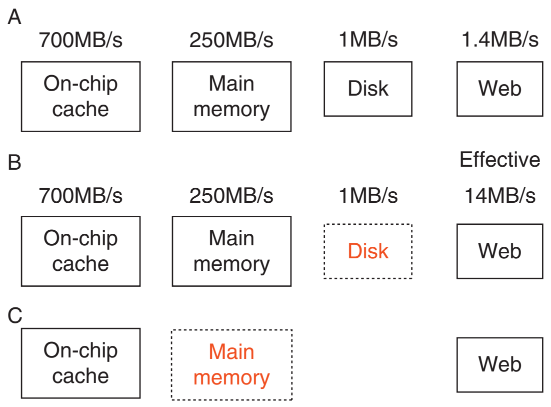


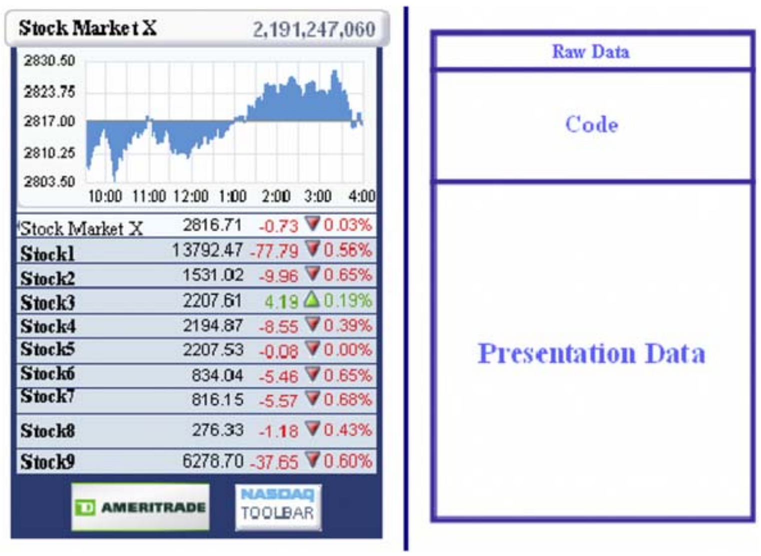
FIG. 10. Memory hierarchy.

Our trends analysis shows the increase in information transmitted across the Internet and processed by mobile computing devices is largely due to formatting—and this trend will likely increase.

Figure 10A shows the current memory hierarchy and the corresponding theoretical bandwidth [17] of Web data access; the chip is on the left and the Web is on the right. Based on the observation that the dynamic part of the Flash file is on average 10% of the total file size, if a scheme can be introduced to reduce the amount of information communicated on the Web by 90%, the *effective* bandwidth of the Web is increased an order of magnitude. This is shown in Fig. 10B. Previously, any caching of Flash content was done on a computer's disk because it posed no bottleneck to performance. With a more sophisticated caching scheme, the hard disk will become the bottleneck in the memory hierarchy as shown in Fig. 10B. Since hard disks are not found in many mobile devices, a caching scheme that eliminates the need for storage to disk while effectively improving Internet bandwidth has twofold value. Also, since Web bandwidth is improving faster than memory bandwidth [66], we are expecting that the main memory will be the bottleneck in the next a few years as shown in Fig. 10C.

3.4.2 Flash Structure

Macromedia Flash Player is distributed among over 99% of Web browsers, exceeding that for other media players [1]. A typical Flash file contains heterogeneous media ingredients (graphics, images, sounds, text, etc.). Figure 11 shows an example of a Flash object (on the left), a stock ticker that shows stock prices both numerically and graphically.



Stock Market X	2816.71	-0.73	▼0.03%
Stock1	13792.47	-77.79	▼0.56%
Stock2	1531.02	-9.96	▼0.65%
Stock3	2207.61	4.19	▲0.19%
Stock4	2194.87	-8.55	▼0.39%
Stock5	2207.53	-0.08	▼0.00%
Stock6	834.04	-5.46	▼0.65%
Stock7	816.15	-5.57	▼0.68%
Stock8	276.33	-1.18	▼0.43%
Stock9	6278.70	-37.65	▼0.60%

FIG. 11. Our Flash animation file structure.

Conventionally, the client browser pulls the Flash file from the server using the refresh command. This means that for the stock market example, the browser will be busy pulling the *entire* Flash file every second to keep the user updated, even though only a small amount of actual content will have changed. By contrast, our proposed breakdown of the structure of the Flash file is shown on the right. The separation of Flash content into the three parts is informed by the analysis of Webpage trends, and also the model of WSPs, specifically Fig. 7. Our Flash content consists of raw data (stock prices), presentation data (graphics), and code to process the two (program) in order to form the image on the mobile computer. Since presentation data and code have approximately the same time bins—1 day or more—we will define the two of them, together, as *packaging*. *Raw data*, by contrast, has a time bin more on the order of 1 s. Conceptually, we leveraged knowledge about how users access this Webpage (load it once then view it for a long time) to move the workload from the 1-s time bin to the 1-day time bin—enough to potentially dramatically affect performance.

Figure 12 shows sample Webpage object structure format with the suggested tag. In addition to the tag, a simple modification is also required in the Flash protocol between client and server. In our proposed model, the client generates the Flash file locally by executing the code on both the raw data and the cached presentation data. Many surveys about Web caching schemes exist [26, 87, 89]. Current Web caching

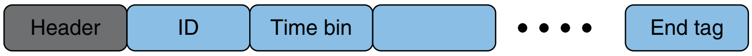


FIG. 12. Webpage object structure.

schemes assume that most Web access patterns are HTML text and images [8, 25, 58], none of them have effectively explored Web multimedia caching techniques, especially Flash. Conventional page-level caching cannot effectively address our observations about Flash content, because they do not break up the content of blocks—a small change in raw data will still lead to renewed Web content generation and transmission [79].

The cost of caching schemes is in the increased complexity of protocol and the cost of storage. It is a classic example of the inclusion of overhead into the system in such a way that overall performance is optimized. Our caching scheme is applied at the system level, derived by analysis of how the relationships between the applications, users, and architectures are changing with the advent of SUMA–MIMD computing.

3.4.3 Experiments

Once again, our target implementation is an SCHM with a fixed area budget. We divided the chip into four regions, to be populated by four categories: media processors (M), digital signal processors (D), general-purpose processors (G), and chip-level cache (C); where the cache is intended to support the Flash content. Sets A and B consider processor arrangements that fit on a chip when there is no cache on chip. Set C considers processor arrangements that fit on a chip with a 1024 K cache set aside for Flash content. These sets produce a total of 41 different architectures, as described in Table XI.

Architectures in the table are differentiated by the numbers and types of processors they contain. Once again, the three different processors chosen for our experiments are the Philips PNX1700 as the media processor (M), Blackfin533 as the DSP (D), and the AMD K6-2E+ as the GPP (G).

Our experiments are broken down into three parts. These three parts correspond to the three parts of Fig. 10, except that we do not include a hard disk, since we presume that many future mobile computing devices will not have them. In part A, we consider traditional Web caching only. An object with a time bin tag equal to or greater than 1 day is saved in the main memory. Based on the data which was gathered from http://www.mlb.com, 345 KB of the entire Webpage content (all of the content on the Website, not just Flash) can be cached for 1 day, main memory

Table XI
Processing Elements for Sets A, B, and C

Architecture	Sets A and B	Architecture	Set C
1	4G	26	3G
2	8D	27	6D
3	16M	28	12M
4	4M, 3G	29	8M, 1G
5	8M, 2G	30	4M, 2G
6	12M, 1G	31	4D, 1G
7	6D, 1G	32	2D, 2G
8	4D, 2G	33	6M, 1D, 1G
9	2D, 3G	34	4M, 2D, 1G
10	2M, 5D, 1G	35	2M, 3D, 1G
11	4M, 4D, 1G	36	2M, 1D, 2G
12	6M, 3D, 1G	37	10M, 1D
13	8M, 2D, 1G	38	8M, 2D
14	10M, 1D, 1G	39	6M, 3D
15	2M, 3D, 2G	40	4M, 4D
16	4M, 2D, 2G	41	2M, 5D
17	6M, 1D, 2G		
18	2M, 1D, 3G		
19	2M, 7D		
20	4M, 6D		
21	6M, 5D		
22	8M, 4D		
23	10M, 3D		
24	12M, 2D		
25	14M, 1D		

size is sufficient to host hundreds of times this amount. Currently, Apple's iPhone has 128 MB of main memory [78]. This produced 25 different architectures to consider as shown in Sets A and B of Table XI.

In part B, we combine the Web caching with our suggested Flash file structure where a Flash file is broken into raw data and presentation data. The caching is again done in the main memory. Based on the collected data from http://www.mlb.com, 640 KB is the data which can be cached in main memory for 1 day. In this part, we are able to cache the packaging content of the Flash file. The same architecture set is produced as in part A since the caching is done in main memory.

Finally, in part C, we again combine the Web caching with our suggested Flash file structure. In this part, the caching is done on chip. We anticipate that if caching can effectively be done on the chip, we may be able to find additional power savings.

This results in 16 additional architectures to consider as shown in Set C of Table XI. The 1024 K cache was chosen to fit the contents collected from http://www.mlb.com homepage, where each Flash file can represent a different game in progress. The data which can be cached for 1 day is 640 KB—the same data as in Part B.

eCacti [60] was used to determine the cache area and cache power consumption based on the selected cache size, block size, and technology. We assumed a 0.13 μm manufacturing technology. Also, we assumed that the power consumption when a processor is in an idle state is 20% of its active power consumption [11]. Furthermore, we assumed a perfect memory system, that is, caches always have the requested data. In other words, experiments show the performance of the second and subsequent requests. This is reasonable, since we are interested in the many applications which persist in the system for hours or even days, constantly updating information in real time. While we do not consider the initial cost of loading the packaging data for the Flash, it would only be significant if the entire working set of Flash content did not fit into the device's cache. For simulation, we once again used MESH. Our scheduling strategy is the best available resource scheduler—each task is scheduled on the best available resource by order of appearance.

In Figs. 13 and 14, the best performer is normalized at 1; thus lower numbers are always poorer performers. Table XI architectures are on the independent axes with sets and experiments grouped as A, B, and C. Figure 13 shows the normalized Webpage loading time. Normalized speed is the reciprocal of dividing the architecture speed by the lowest speed among all architectures. Parts B and C, which are our proposed approach for caching Flash content, show a significant improvement over that of part A. It also shows the effects of picking an optimal architecture even within the optimized Flash caching structure.

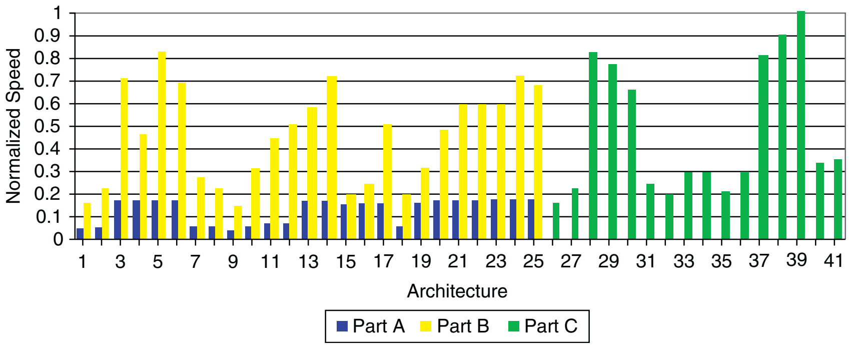


FIG. 13. Loading time.

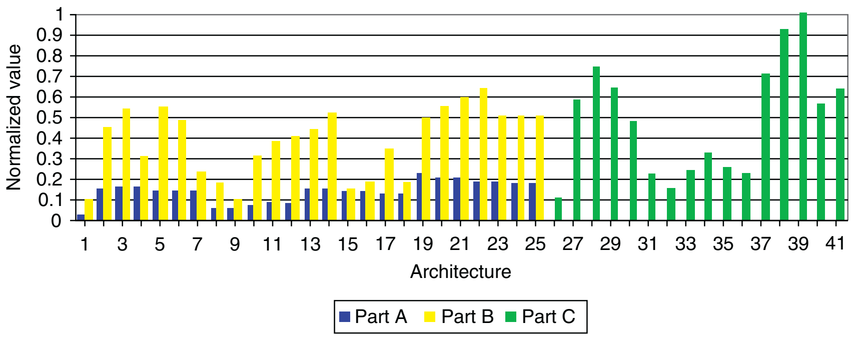


FIG. 14. Loading latency and energy consumption.

Figure 14 shows the normalized value of speed and energy consumption or $\mathrm{NV} = N(\mathrm{NS} + \mathrm{NE})$, where NV is the normalized value resulting from the two values: NS which is the normalized speed and NE which is the normalized energy. This graph shows an even more significant advantage for our approach—especially Part C—as well as interesting variation between architectures that utilize our system-level caching scheme.

While we focused on http://www.mlb.com for the reasons previously described, we also found that these results are applicable to more than just this Web domain. We found similar results for the stock ticker example of Fig. 11. There, the raw data is 0.2 KB and is changing every minute or even every second. The presentation data is 4 KB and it changes every week or more. The code is 6 KB and it rarely changes. Our application-level caching scheme for Flash resulted from analyzing Webpage content in consideration of user access patterns, that is, by considering SUMA–MIMD computers that are modeled as WSPs. In so doing, we found a performance increase of 84%, a decrease in power consumption by 71%, and an order of magnitude savings in communications bandwidth for mobile computing devices that process Webpages.

3.5 Discussion

WSPs are a type of processor class that arises when the emerging landscape of SUMA–MIMD computing is considered. The benchmarks and organizing principles of WSPs are fundamentally different from that of general-purpose or application-specific processing, as illustrated by the concepts of personalized architectures and the time granularity used to develop a system-level caching scheme for Flash.

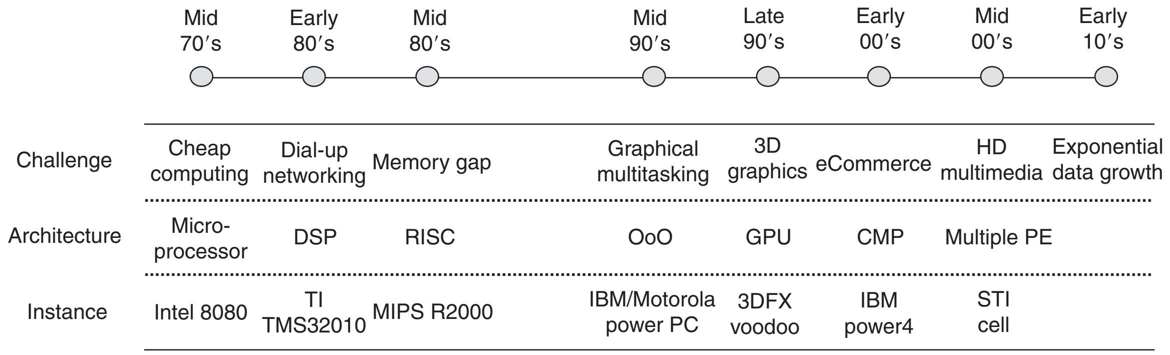
By illustrating these concepts, we intend to motivate how the SUMA concept, viewed from the lens of a new era of performance evaluation, can motivate computer architects and designers to break free of historical frameworks. Beyond SUMA–MIMD, we next discuss what could be the next frontier in computing, a class of architecture that can be considered to fill out Flynn's taxonomy with a SUMA–MISD architecture.

4. Spectroprocessing for Qualitative Computation

Beyond Webpages, what might be next on the horizon for SUMA computers? Figure 15 illustrates how computer architecture has responded to challenges in computation, with the result that new categories of computing have been created every few years. In the past few years, the amount of personal data generated and transmitted by individual users has grown exponentially and is fast outpacing the ability of traditional computing structures to deal with it. If computers are to facilitate humans in dealing with data recognition, classification, and representation, they must take on new forms of computation that break away from quantitative models of computation to *qualitative* models of computation.

Since the beginning of electrical and electronic computers, virtually all computation has been quantitative in nature. The goal of the computation has been to functionally map inputs from one mathematical space to another, using functional transformations, or $b = F(a)$, where a is the input, b is the output, and the function F is a mapping of all possible points from space A to all possible points in space B. Note that the computation is still quantitative even if the result is intended to include some uncertainty or if the computation uses heuristics or nondeterminism in the technique. When uncertainty is involved, the computation can be thought of as $b' = F(a)$, where b' is considered to be a sufficient approximation of the accurate and objective output b. Note also that functional transformations typically carried out by digital computers transform points from spaces with more dimensions to those with fewer dimensions. This mathematical foundation applies even to compilation and other language-based input transformations, because the language representation of the computer input relates directly to structures—or tuples, which are the binary representation of the inputs. The important point is that accurate outputs, or points in the transformational space, are known and can be modeled mathematically.

In qualitative computing, by contrast, no mathematical transformation is possible, because the mathematical basis for evaluating the key features of input data is subjective. This subjectivity is a necessary element of data classification—even



	Mid 70′s	Early 80′s	Mid 80′s	Mid 90′s	Late 90′s	Early 00′s	Mid 00′s	Early 10′s
Challenge	Cheap computing	Dial-up networking	Memory gap	Graphical multitasking	3D graphics	eCommerce	HD multimedia	Exponential data growth
Architecture	Micro-processor	DSP	RISC	OoO	GPU	CMP	Multiple PE	
Instance	Intel 8080	TI TMS32010	MIPS R2000	IBM/Motorola power PC	3DFX voodoo	IBM power4	STI cell	

FIG. 15. Architectural responses to new challenges.

humans cannot agree on an accurate interpretation of the nature of the content of information. For example, while it may be easy to think in terms of color encoding in a computer, when can an image be classified as being ''red'' in nature? Some images are more red than others—the threshold of a red image is subjective, and will vary considerably according to need. Much more precision in the classification of an image as having a ''red'' quality will be desired by an artist than by someone seeking to separate sports teams by uniform color.

Rather, the objective of qualitative computing is to establish if a given input has a certain quality, where the presence of the quality is established according to its likely (probabilistic) belonging to a space which is not completely defined—and which is ever evolving based upon user preferences and past qualitative computing experiences. Even as humans can never completely define qualities in the same objective way quantities can be established, qualitative computation can never result in a perfectly accurate answer that is not subject to evolution and interpretation. Accordingly, the objective of qualitative computing is to identify how well some input a has the qualities $\{A', B', C', D',\ldots\}$ where those qualities are represented in imperfect mathematical spaces. Significantly, since qualities are defined in a highly personal manner to begin with, SUMA machines may be the optimal way to differentiate user preference while preserving performance—or making qualitative computing tractable at all.

4.1 A Universal Problem

Bush's seminal article, ''As We May Think'' [19], eloquently describes the potential for machines to eventually store an enormous portion of the human experience, and the corresponding necessity for powerful search functionality. This vision has inspired Dumais' ''Stuff I've Seen'' system to assist users in rediscovering digital information [30] and Bell's ''MyLifeBits'' project which is testing the practicality of recording all data one comes in contact with Bell [13] and Gemmell *et al.* [38]. Bell describes the special difficulties in handling multimedia data (''My database colleagues have yet to convince me that they can do better than 'grep' searching the free text or viewing thumbnails''). As the amount of information stored grew, however, the capacity of ''ordinary indexing and searching'' proved insufficient. As of 2006, MyLifeBits uses a database with some basic metadata. The challenges of automatically extracting metadata beyond time and location and effectively searching image data have not been solved [38].

A recent study projects that the amount of new digital information generated will increase by 57% per year between 2006 and 2010 [37]. In 2010, the amount of new digital information generated will exceed 988 billion gigabytes [37]. In Fig. 16, we compare the rate at which new information is generated with advances in silicon

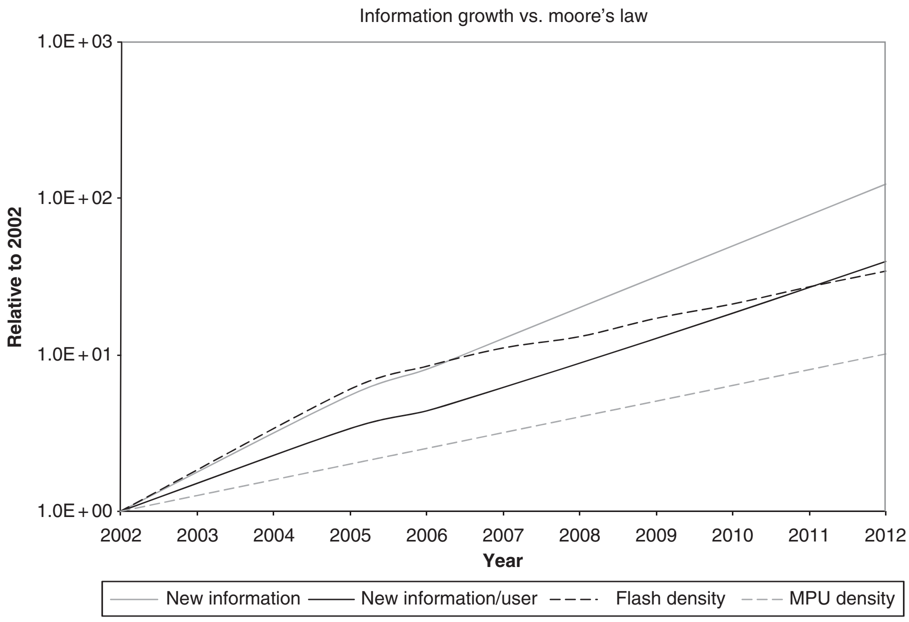


FIG. 16. Relative growth of digital information and transistor density.

technology, drawing upon data and projections from recent studies [37, 47, 48, 57]. Because some projections are worldwide, increased adoption of digital technology plays a role in the overall rate of growth. To decouple this effect, we approximate the number of users of digital content by the number of people with Internet connectivity. As shown in Fig. 16, which uses a logarithmic scale for the vertical axis, between 2002 and 2012, both total information and information per user are expected to greatly outpace the capabilities of digital technology. This represents all who use digital data.

The personal information management (PIM) community provides a foundation for understanding the difficulties *individual users* face in finding and filing their data. PIM encompasses an extensive range of problems. Barreau, for example, defines PIM as a system that ''includes a person's methods and rules for acquiring the information which becomes part of the system, the mechanisms for organizing and storing the information, the rules and procedures for maintaining the system, the mechanisms for retrieval, and the procedures for producing the various outputs required'' [12]. Research into PIM has touched such varied issues as human–computer interface [30, 59], practices for filing paper documents [59], and psychological concerns [52], in addition to techniques for digital content management

[13, 30, 38]. The underlying computation necessary to manage the continued explosion of digital content, however, has a unique set of challenges that are not captured by Barreau's definition of PIM or others described in Jones' overview of the field [50].

We project the growing difficulty individual users face in managing their digital data in Fig. 17. We assume that by the end of each year, three quarters of existing data has been archived or deleted, and the rest must still be managed. To account for the increasing pervasiveness of the Internet, we also assume that the number of times that people need to share or receive data will grow in proportion to the total number of Internet users. Finally, we attempt to account for the difference in difficulty between searching data in textual form and that of multimedia data. We assume that the growth of multimedia data will be exponential, driven primarily by video, and represent an increasing fraction of the ''digital universe.'' This is supported by IDC's projections about the growth of digital video that predicts a doubling of camcorders and tenfold increase in digital security cameras between 2006 and 2010 [37]. A combination of technical factors including advances in Flash memory, adoption of H.264, and increased availability of cellular broadband, along with social factors such as the rising popularity of video blogs, may accelerate this boom even further.

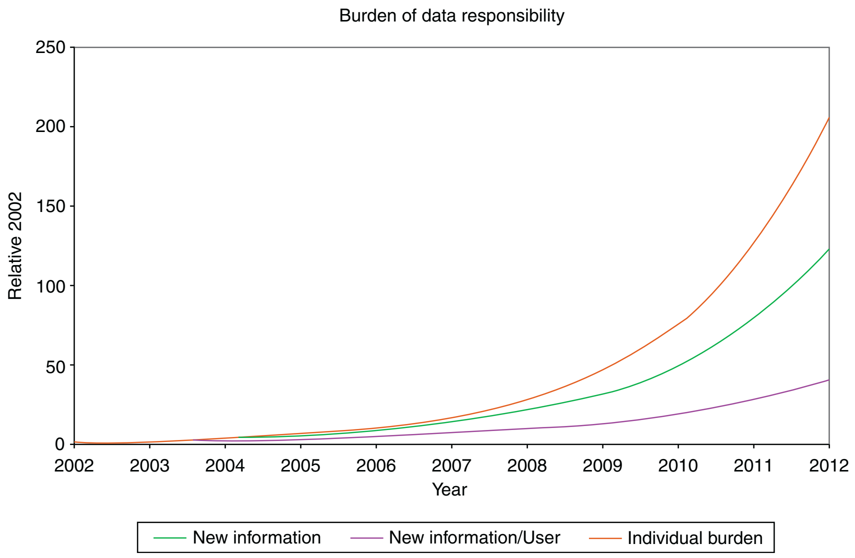


FIG. 17. Effort required to find, file, and share data grows exponentially.

Figure 17 shows that without major improvements in digital content management, the difficulty faced by an average individual in sharing and organizing their digital content will rapidly spiral out of control. IDC reports that locating information already requires 9.6 h a week for knowledge workers and projects that information missed during this search typically costs a company with 1000 knowledge workers $5.3M per year [37]. In the absence of more effective tools, we expect the financial impact to increase either through more hours spent looking for information or more lost revenue from missed information. The human costs are significant too—Kirsh [52] describes a ‘‘cognitive overload’’ resulting from the huge amount of information workers encounter on a daily basis. In Hobbs *et al.* [42], it was found that ‘‘one third of managers suffer from ill health as a direct consequence of stress associated with information overload.’’ And that was 12 years ago.

Because the difficulty of managing personal content is growing faster than process technology, and because algorithmic improvements have not been able to compensate, solutions must come from advances in both algorithms and computer architecture, developed in concert. The goal is to enable constant effort access—which means that the effort to access data will not grow with the data, but will remain constant.

4.2 Enabling Constant Effort Access

We observe that many different algorithms, which currently exist, can be employed in concert, but that their effectiveness has been limited by a failure to leverage:

- *Personalization*—optimizing data management by tuning to individual preferences, enabled by the ability for data management to take place on personal (and mobile) computers
- *Multi-interpretation*—simultaneously applying different techniques, enabled by the underlying architecture
- *Integration*—effective coupling of global data and processing enabled by novel SCHM architectures

Figure 18 shows that high-level descriptions are computationally difficult to extract, but require relatively little storage, making it possible to store them locally (on a chip). This is convenient for index structures since it enables sort to avoid the latency of wireless communication. The figure is loosely based on speech recognition, with the second point representing the output of the front end (2% of the total computation) and the third a set of 10 sentences. The challenge lies in finding ways to increase the effectiveness of computation.

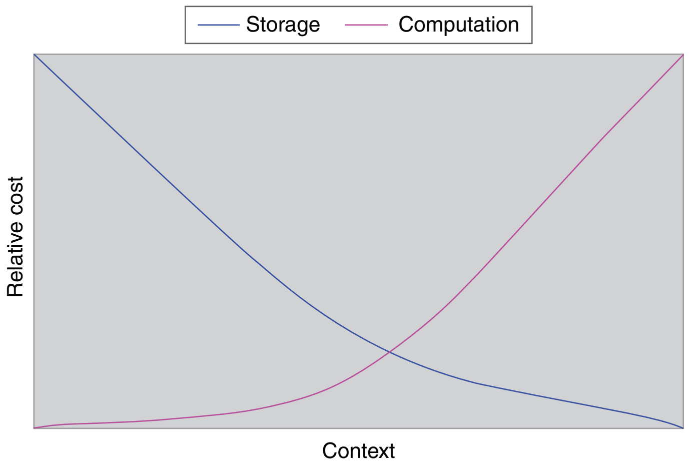


FIG. 18. Storage and computational requirements versus recovered context.

We believe that both information and knowledge retrieval can benefit from the same approach—that of a SUMA architecture which leverages customization, multi-interpretation, and integration of approaches because multiple algorithmic approaches have different strengths and weaknesses, and they all can benefit from a narrowing of context [27, 39, 42, 74, 75, 82].

However, we believe that the SUMA machine best suited to quantitative computing is not SISD or MIMD, but MISD, thus filling out Flynn's taxonomy of multiprocessors [35]. In that taxonomy, MIMD is by far the most common structure of multiprocessors. The remainder of Flynn's classification includes SISD, or conventional uniprocessors, and SIMD which are best represented by the heavily pipelined systolic arrays researched in the early 1980s, suitable for the regular computation of signal processing, and by recent instruction set extensions for vector and multimedia processing. Previously, the fourth category of Flynn's classification MISD made no sense—what does it mean to take a single data stream and process it over multiple instructions? The MISD classification for quantitative computation does not make sense because the same input would result in different outputs, unless the outputs were completely decoupled, in which case the computer would not be logically unified.

However, the objective of qualitative computing is to evaluate the set of qualities a given datum possesses—and for this objective, it makes sense to process the datum in concert simultaneously along multiple, heterogeneous computational fronts according to a common objective. The simultaneous, heterogeneous processing is unified not only by the common objective, but it is also informed by a common set of past experiences, captured in a global database. The result is an entirely new class of computer architecture, which we refer to as a spectroprocessor because of the image of breaking a single input down into heterogeneous composite streams. A spectroprocessor replicates an input datum, which is a single stream of information, for simultaneous, heterogeneous processing. This structure uniquely fills out Flynn's classification and is shown in Fig. 19.

The figure shows how a single input datum is simultaneously and heterogeneously processed with respect to the global database (and yet local and custom to the device and the individual user's preferences), labeled as ``{definition, history}'' that loosely couples the processing elements as shown by the grey line which extends upward from the database. The ``definition'' represents the individualized, user-defined qualities (e.g., the meaning of ``red'') and the ``history'' represents a record of past successes. The result of the computation is a set of probabilities for a set of qualities, $Q_1...Q_n$. For example the presence of a complex quality may be broken down into subqualities and therefore processed as the set $Q_1...Q_n$. Or, each of the qualities may represent the result of a different algorithmic approach. The user will utilize the results of the computation to reiterate (narrow qualities or change algorithms) or accept a result, thereby modifying the {definition, history} database local to their custom device.

The challenges in developing a spectroprocessor include the logical representation of the qualities, the physical challenges of providing a balanced set of processing elements (the circles shown in the figure) which are tuned to likely algorithms,

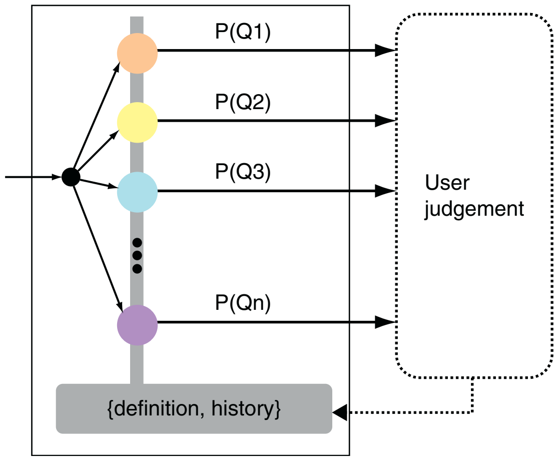


FIG. 19. Spectroprocessing.

and the establishment of effective cooperation between the processing elements and the global database, so that levels of integration and customization currently possible on single chips can be utilized.

5. Conclusions

The goals of this chapter are to categorically, exemplary and experimentally motivate how the landscape of performance evaluation in computing is changing—in turn motivating the dawn of a new era in computing. We developed the U-A taxonomy, intended to view computers from the outside in rather than the inside out, and at the same time motivating an important new category of computing, that of SUMA. Combined with Flynn's taxonomy, a SUMA–MIMD computer is one that is a parallel processor, executing a collection of applications that support a single user's mission. The performance evaluation of the collection of applications for the benefit of a single user is fundamentally different from that of parallel processing of old, and so opens up new possibilities for entirely new organizing principles and frameworks in computer design. To illustrate this, we developed the concept of a WSP, suited to the processing of Webpages on a SUMA–MIMD computer, and showed how it enabled new concepts for design, such as Personalized Architectures and Workload Time Granularity. Finally, we utilized the SUMA concept, in conjunction with Flynn's taxonomy, to look even farther ahead, into what is potentially the next frontier of computing, that of personalized data access and management. The concept of Spectro-processing for Qualitative Computation is one that could, potentially, become the focus of the computing industry for years to come.

Acknowledgments

This work was supported in part by the National Science Foundation under grant 0607934. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Adobe, 2008. Flash Player Statistics, http://www.adobe.com/products/player_census/flashplayer/, accessed 2007.

[2] Agrawala A. K., Mohr J. M., and Bryant R. M., June 1976. An approach to the workload characterization problem. *Computer*, **9**(6): 18–32.

[3] Alexa—The web Information Company. http://www.alexa.com, accessed 2008.

[4] AMD, n.d. AMD-K6 Series, http://www.amd.com/epd/processors/6.32bitproc, accessed 2007.

[5] Amdahl G. M., 1967. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the AFIPS Spring Joint Computer Conference*, p. 30, Atlantic City, NJ.

[6] Analog Devices, 2008. ADSP-BF533, http://www.analog.com/en/epProd/0,ADSP-BF533,00.html, accessed 2007.

[7] Apple, 2008. iPhone, http://www.apple.com/iphone/, accessed 2008.

[8] Arlitt M., and Willliamson C., 1996. Web server workload characterization: The search for invariants. In *ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 126–137.

[9] ARM, 2008. ARM7TDMI, http://www.arm.com/products/CPUs/ARM7TDMI.html, accessed 2008.

[10] ARM, 2008. ARM1176JZF, http://www.arm.com/products/CPUs/ARM1176.html, accessed 2008.

[11] Babighian P., Benini L., and Macii E., 2004. Sizing and characterization of leakage-control cells for layout-aware distributed power-gating. *DATE*, **1**: 720–721.

[12] Barreau D. K., 1995. Context as a factor in personal information management systems. *Journal of the American Society for Information Science*, **46**(5): 327–339.

[13] Bell G., January 2001. A personal digital store. *Communications of the ACM*, 86–91.

[14] Benini L., and De Micheli G. D., 2002. Networks on chips: A new SoC paradigm. *IEEE Computer Magazine*, **35**(1): 70–78.

[15] Bishop B., Kelliher T., and Irwin M., 1999. A detailed analysis of Mediabench. In *Proceedings of the IEEE Workshop on Signal Processing Systems*, pp. 448–455.

[16] Brooks D. M., Bose P., Schuster S. E., Jacobson H., Kudva P. N., Buyuktosunoglu A., Wellman J.-D., Zyuban V., Gupta M., and Cook P. W., November/December 2000. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *Proceedings of the IEEE Micro*, **20**(6): 26–44.

[17] Brown R. G., May 2004. Engineering a Beowulf-style compute cluster, http://www.phy.duke.edu/~rgb/Beowulf/beowulf_book/beowulf_book/, accessed 2008.

[18] Buchner A., and Mulvenna M. D., 1999. Discovering Internet marketing intelligence through online analytical web usage mining. *SIGMOD Record*, **27**(4): 54–61.

[19] Bush V., 1945. As we may think. *The Atlantic Monthly*, **176**(1): 101–108.

[20] Calzarossa M., and Serazzi G., August 1993. Workload characterization: A survey. *Proceedings of the IEEE*, **81**(8): 1136–1150.

[21] Chiang S.-H., and Vernon M. K., 2001. Characteristics of a large shared memory production workload. In *Job Scheduling Strategies for Parallel Processing*, pp. 159–187, LNCS 2221. Springer-Verlag, London.

[22] Cho Y., Yoo S., Choi K., Zergainoh N., and Jerraya A., 2005. Scheduler implementation in MP SoC design. In *Proceedings of the Design Automation Conference*, pp. 151–156.

[23] Cirne W., and Berman F., April 2001. A model for moldable supercomputer jobs. In *Proceedings of the IPDPS 2001—International Parallel and Distributed Processing Symposium (IPDPS)*, p. 59.

[24] Cooley R., Mobasher B., and Srivastava J., 1999. Data preparation for mining World Wide Web browsing patterns. *Journal of Knowledge and Information Systems*, **1**(1): 5–32.

[25] Cunha C., Bestavros A., and Crovella M., 1995. Characteristics of WWW Client-Based Traces, TR-95-010. Boston University, CS, Boston.

[26] Davison B. D., 1999. A survey of proxy cache evaluation techniques. In *Proceedings of the 4th International Web Caching Workshop*, pp. 67–77.

[27] Deerwester S. D., Dumais S. T., Landauer T. K., Furnas G. W., and Harshman R. A., 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, **41**(6): 391–407.

[28] Downey A., August 1997. A parallel workload model and its implications for processor allocation. In *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing (HPDC'97)*, p. 118.

[29] Downey A., and Feitelson D., March 1999. The elusive goal of workload characterization. *Performance Evaluation Review*, **26**(4): 14–29.

[30] Dumais S., Cutrell E., Cadiz J., Jancke G., Sarin R., and Robbins D. C., July 2003. Stuff I've seen: A system for personal information retrieval and re-use. In *SIGIR 2003: Proceedings of the 26th Annual International ACM*, pp. 72–79.

[31] EEMBC—The Embedded Microprocessor Benchmark Consortium, 2008. http://www.eembc.org, accessed 2007.

[32] Feitelson D. G., September 2003. Metric and workload effects on computer systems evaluation. *Computer*, **36**(9): 18–25.

[33] Feitelson D. G., and Tsafrir D., March 2006. Workload sanitation for performance evaluation. In *2006 IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 221–230.

[34] Ferrari D., July/August 1972. Workload characterization and selection in computer performance measurement. *Computer*, **5**(4): 18–24.

[35] Flynn M., 1972. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, **C21**: 948.

[36] Friedman T., 2005. The World Is Flat: A Brief History of the Twenty-first Century. Farrar–Straus–Giroux, New York.

[37] Gantz J. F., Reinsel D., Chute C., Schlichting W., McArthur J., Minton S., Xheneti J., Toncheva A., and Manfrediz A., March 2007. The expanding digital universe, a forecast of worldwide information growth through 2010. White Paper sponsored by EMC. IDC. [Online]. Available:http://www.emc.com/about/destination/digital_universe/pdf/Expanding Digital Universe IDC White Paper 022507.pdf.

[38] Gemmell J., Bell G., and Leuder R., January 2006. MyLifeBits: A personal database for everything. *Communications of the ACM*, **49**(1): 89–95.

[39] Golub G. H., Luk F. T., and Overton M. L., June 1981. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software*, **7**(2): 149–169.

[40] Guthaus M., Ringenberg J., Ernst D., Austin T., Mudge T., and Brown R., 2001. MiBench: A free, commercially representative embedded benchmark suite. In *Proceedings of the 4th Workshop on Workload Characterization*, pp. 1–12.

[41] Hennessy D., and Patterson D., 2003. Computer Architecture: A Quantitative Approach, 3rd edition, pp. 40–41. Morgan Kaufmann, San Francisco, CA.

[42] Hobbs J. R., Appelt D., Bear J., Israel D., Kameyama M., Stickel M., and Tyson M., 1996. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In *Finite-State Devices for Natural Language Processing*, MIT Press, Cambridge, MA.

[43] Hotovy S., 1996. Workload evolution on the Cornell Theory Center IBM SP2. In *Job Scheduling Strategies for Parallel Processing*, pp. 27–40. LNCS 1162. Springer-Verlag, London.

[44] iGoogle—Google's personalized start page service. http://www.igoogle.com, accessed 2008.

[45] Intel, 2008. Pressroom, http://www.intel.com/pressroom/archive/releases/20080107corp.htm, accessed 2008.

[46] Internet Archive, 2008. http://www.archive.org, accessed 2008.

[47] ITRS—International technology roadmap for semiconductors, 2002. SEMATECH, http://www.itrs.net/Links/2002Update/Home.htm, accessed 2007.

[48] ITRS—International technology roadmap for semiconductors, 2005. SEMATECH, http://www.itrs.net/Links/2005ITRS/Home2005.htm, accessed 2007.

[49] Jann J., Pattnaik P., Franke H., Wang F., Skovira J., and Riordan J., 1997. Modeling of workload in MPPs. *In Job Scheduling Strategies for Parallel Processing*, (D. Feitelson and L. Rudolph, Eds.), Lecture Notes in Computer Science, Vol. 1291, Springer-Verlag, New York.

[50] Jones W., November 2005. Personal information management, University of Washington Information School, Technical Report IS-TR-2005-11-01, November 2005, http://hdl.handle.net/1773/2155, accessed 2007.

[51] Kauneckis D., 2000. Preferences on the Landscape: How Much Do Individual Values Matter? Paper presented at the International Society for the Study of Common Property, Bloomington, IN.

[52] Kirsh D., 2000. A few thoughts on cognitive overload. *Intellectica*, **30,** 19–51.

[53] Konstan J. A., Miller B. N., Maltz D., Herlocker J., Gordon L., and Riedl J., 1997. Grouplens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, **40**(3): 77–87.

[54] Law A. M., and Kelton W. D., 2000. Simulation Modeling and Analysis, 3rd edition. McGraw Hill, New York.

[55] Lee C., Potkonjak M., and Mangione-Smith W., 1997. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. *International Symposium on Computer Architecture (ISCA)*, 330–335.

[56] Lieberman H., 1995. Letizia: An agent that assists web browsing. In *Proceedings of the 14th International Joint Conference on AI*, pp. 924–929.

[57] Lyman P., and Varian H. R., 2003. How much information. White Paper. University of California at Berkeley, http://www.sims.berkeley.edu/how-much-info-2003/, retrieved March 24, 2007.

[58] Mahanti A., Williamson C., and Eager D., 2000. Traffic analysis of a web proxy caching hierarchy. *IEEE Network*, **14**(3): 16–23.

[59] Malone T. W., January 1983. How do people organize their desks? Implications for the design of office information systems. *ACM Transactions on Office Information Systems*, **1**(1): 99–112.

[60] Mamidipaka M., and Dutt N., 2004. eCacti: An enhanced power estimation model for on-chip caches. Technical Report #04-28, UCI.

[61] Meeker M., Pitz B., Fitzgerald B., and Ji R., 2005. Internet Trends, http://www.morganstanley.com/institutional/techresearch/pdfs/Internet_Trends1005.pdf, accessed 2007.

[62] Meyer B. H., Pieper J. J., Paul J. M., Nelson J. E., Pieper S. M., and Rowe A. G., June 2005. Power-performance simulation and design strategies for single-chip heterogeneous multiprocessors. *IEEE Transactions on Computers*, **54**(6): 684–697.

[63] Nasraoui O., Frigui H., Joshi A., and Krishnapuram R., August 1999. Mining web access logs using relational competitive fuzzy clustering. In *Proceedings of the 8th International Fuzzy Systems Association World Congress*, Vol (2), pp. 17–20.

[64] NXP, 2008. PNX17xx Series, http://www.nxp.com/pip/PNX17XX_SER_N_1.html, accessed 2007.

[65] Otoom M., and Paul J. M., October 2008. Holistic design and caching in mobile computing. *CODES+ISSS: International Conference on Hardware/Software Codesign and System Synthesis*, pp. 115–120.

[66] Patterson D., 2005. Latency lags bandwidth. In *International Conference on Computer Design*, p. 3.

[67] Paul J. M., March 2006. What's in a Name? *IEEE Computer*, **39**(3): 87–89.

[68] Paul J. M., and Meyer B. H., April 2007. Amdahl's law revisited for single chip systems. *International Journal of Parallel Programming*, **35**(2): 101–123.

[69] Paul J. M., Thomas D. E., and Bobrek A., 2004. Benchmark-based design strategies for single chip heterogeneous multiprocessors. In *Proceedings of the International Conference on Hardware/Software Codesign*, pp. 54–59.

[70] Paul J. M., Thomas D. E., and Cassidy A. S., July 2005. High-level modeling and simulation of single-chip programmable heterogeneous multiprocessors. *ACM Transactions on Design Automation of Electronic Systems*, **10**(3): 431–461.

[71] Paul J. M., Thomas D. E., and Bobrek A., August 2006. Scenario-oriented design for single-chip heterogeneous multiprocessors. *IEEE Transactions on VLSI*, 868–880.

[72] Perkowitz M., and Etzioni O., 1998. Adaptive web sites: Automatically synthesizing web pages. In *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, WI, pp. 727–732.

[73] Pieper S. M., Paul J. M., and Schulte M. J., September 2007. A new era of performance evaluation. *IEEE Computer*, **40**(9): 23–30.

[74] Porter M. F., 1980. An algorithm for suffix stripping. *Program*, **14**(3): 130–137.

[75] Robertson S., and Jones K. S., May 1997. Simple, proven approaches to text retrieval, Cambridge Computer Laboratory, Technical Report UCAM-CL-TR-356.

[76] Scemama D., 2007. Three ARM processor cores per iPhone, http://www.eetimes.com/news/design/showArticle.jhtml?articleID=196900827, accessed 2008.

[77] Schechter S., Krishnan M., and Smith M. D., 1998. Using path profiles to predict HTTP requests. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, pp. 457–467.

[78] Semiconductors, 2008. Apple (Samsung S5L8900) applications processor with eDRAM, http://www.semiconductor.com/resources/reports_database/view_device.asp?sinumber=18016, accessed 2008.

[79] Sen S., Rexford J., and Towsley D., March 1999. Proxy prefix caching for multimedia streams. *IEEE INFOCOM*, 1310–1319.

[80] Shahabi C., Zarkesh A. M., Adibi J., and Shah V., 1997. Knowledge discovery from users web-page navigation. In *Proceedings of the Workshop on Research Issues in Data Engineering*, Birmingham, England, pp. 20–29.

[81] Shahabi C., Kashani F. B., Chen Y. S., and McLeod D., 2001. Yoda: An accurate and scalable web-based recommendation system. In *Proceedings of the 9th International Conference on Cooperative Information Systems*, pp. 418–432.

[82] Smeulders A. W., Worring M., Santini S., Gupta A., and Jain R., December 2000. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis Machine Intelligence*, **22**(12): 1349–1380.

[83] Somers M., and Paul J., January 2008. Webpage-based benchmarks for mobile device design. In *Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 795–800.

[84] SPEC—Standard Performance Evaluation Corporation, 2008. SPEC's benchmarks and published results, http://www.spec.org/benchmarks.html#web, accessed 2008.

[85] Spiliopoulou M., 1999. Data mining for the web. In *Proceedings of the Principles of Data Mining and Knowledge Discovery (PKDD)*, pp. 588–589.

[86] Spiliopoulou M., and Faulstich L. C., 1999. WUM: A web Utilization Miner. In *Proceedings of the EDBT Workshop WebDB98*, Valencia, Spain, LNCS 1590, Springer-Verlag, New York, pp. 184–203.

[87] Stefan P., and Laszlo B., 2003. A survey of web cache replacement strategies. *ACM Computing Surveys*, **35**(4): 374–398.

[88] SWF Macromedia Flash File Format, n.d. http://www.half-serious.com/swf/format/, accessed 2007.

[89] Wang J., 1999. A survey of web caching schemes for the Internet. *ACM Computer Communications Review*, **29**(5): 36–46.

[90] Website Optimization, 2008. Webpage analyzer, http://www.websiteoptimization.com/services/analyze/, accessed 2007.

[91] Woo S., Ohara M., Torrie E., Sing J., and Gupta A., June 1995. The SPLASH-2 Programs: Characterization and methodological considerations. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pp. 24–36.

[92] Yan T., Jacobsen M., Garcia-Molina H., and Dayal U., 1996. From user access patterns to dynamic hypertext linking. In *Proceedings of the 5th International World Wide Web Conference*, Paris, France.

[93] Yuan W., and Nahrstedt K., 2003. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *Proceedings of the 19th Symposium on Operating System Principles*, pp. 149–163.

Advances in Web Testing

CYNTRICA EATON

Department of Computer Science, Norfolk State University, 700 Park Avenue, Norfolk, Virginia 23504

ATIF M. MEMON

Department of Computer Science, University of Maryland, 4115 A. V. Williams Building, College Park, Maryland 20742

Abstract

Demand for high-quality Web applications continues to escalate as reliance on Web-based software increases and Web systems become increasingly complex. Given the importance of quality and its impact on the user experience, a significant research effort has been invested in developing tools and methodologies that facilitate effective quality assurance for Web applications. Testing, in particular, provides a critical inroad toward meeting the quality demand by enabling developers to discover failures and anomalies in applications before they are released. In this survey, we discuss advances in Web testing and begin by exploring the peculiarities of Web applications that makes evaluating their correctness a challenge and the direct translation of conventional software engineering principles impractical in some cases. We then provide an overview of research contributions in three critical aspects of Web testing: deriving adequate Web application models, defining appropriate Web testing strategies, and conducting Web portability analysis. In short, models are used to capture Web application components, their attributes, and interconnections; testing strategies use the models to generate test cases; and portability analysis enables Web developers to ensure that their applications remain correct as they are launched in highly diverse configurations.

1. Introduction

With a significant role in modern communication and commerce, Web applications have become critical to the global information infrastructure and, subsequently, one of the largest and most important sectors of the software industry [24, 39]. As a natural corollary, ensuring the quality of Web applications prior to release is highly important. Yet, given extreme time-to-market pressures, increasingly complex Web applications, constant shifts in user requirements, and rapidly evolving development technologies, achieving this quality is extremely difficult and presents novel challenges to software development [4, 56]. As a result, implementing high-quality Web applications using a cost-effective development process is currently one of the most challenging pursuits in software engineering; a significant research effort has been invested in developing systematic, quantifiable approaches that support Web quality assessment [4, 6, 38, 43].

Given the broad use of the word *quality* thus far, it is important to note that Web application quality is a complex, multifaceted attribute that has many dimensions including *usability* [9, 14, 35, 54], *performance* [12, 37, 51, 60], *accessibility*[1] [2, 11, 33, 46, 50, 52, 55], and *security* [25, 30, 31, 49]. While failure to assess each dimension prior to release can negatively impact the user experience, this chapter focuses on contributions to Web testing research. In particular, we explore work that applies structural and functional testing solutions to the discovery of Web failures. As a result, research devoted to usability, performance, accessibility, and security do not fall within the scope of this survey. For further clarity, because we are more interested in high-level, functional correctness, Web testing tools that verify links, validate Hypertext Markup Language (HTML) or Extensible Markup Language (XML) syntax, and perform stress testing fall outside the bounds of this discussion as well [10, 34].

While isolating faults and ensuring correct functionality is a vital process associated with any software development effort, it is widely considered tedious and time consuming even for more traditional software types with stable, well-defined, monolithic runtime environments [47, 53]; since Web applications are much more complex, the testing process can be even more involved. One overarching idea in Web testing research is to identify well-established software engineering methods that can address specific problems in Web development, adapt or modify them to account for peculiarities and complexities of Web applications, and define novel approaches when necessary [18, 45]. In this chapter, we provide more insight into how researchers are using this practice to advance the field of Web application testing by deriving solutions to three issues: the extraction of suitable test models, development of effective testing strategies, and assessment of configuration-independent quality through portability analysis. Our focus on these three particular issues aligns with the idea that effective testing of Web-based applications must include extracting models capable of representing components of the application and their interconnections, deriving and executing test cases based on those models, and ensuring that quality is preserved as Web applications are launched in diverse configurations.

We structure our discussion of Web testing research contributions in the following way: In Section 2, we take a look at how Web applications have evolved and overview characteristics that make testing them unique and challenging. In Section 3,

[1] The most widely used goal of accessibility is to ensure that Web applications accommodate the needs of physically and mentally handicapped users.

we discuss Web application models designed to capture characteristics useful for testing. In Section 4, we overview Web testing methodologies and processes. In Section 5, we discuss research in Web portability analysis where the goal is to ensure that quality does not diminish as Web applications are ported. In Section 6, we conclude.

2. Challenges of Web Testing

As use of the Web grew at a tremendous rate and the benefits of implementing high-quality Web-based systems became more apparent, the pursuit for expanded capabilities of Web applications simultaneously increased their complexity and drove the rapid evolution of Web technology. Over the years, Web infrastructure has evolved from primarily being a communication medium to a platform for elaborate Web applications that are interactive, highly functional software systems [56]. This evolution has had a notable impact on the pursuit and effective implementation of quality assurance strategies; with room for more complex interaction and increased computation, it is widely acknowledged that rigorous verification and validation approaches are necessary [41]. In the rest of this section, we discuss several factors inherent to Web development that contribute to the quality problem; in doing so, we also highlight the challenges and considerations that influence the practicality and usefulness of conventional software testing methodologies and tools.

2.1 Heterogeneity and Distribution of Components

Given current technology, Web developers are able to create software systems by integrating diverse components that are written in various programming languages and distributed across multiple server platforms; because of the ubiquitous presence of the Web, data can be transferred among completely different types of software components that reside and execute on different computers quite easily [18, 29, 39]. These factors have contributed to a significant growth in the Web services arena and sparked a keen research interest in applying semantic nets to help manage heterogeneity.[2] Since modern Web applications typically have complex, multitiered, heterogeneous architectures including Web servers, application servers, database servers, and clients acting as interpreters, testing approaches must be able to handle highly

[2] Please refer to the chapter titled ''Semantic web applied to service oriented computing'' by Fensel and Vitvar for further information.

complex architectures and account for the flow of data through the various architectural components [24, 39].

2.2 Dynamic Nature of Web Applications

There are several aspects that make Web applications highly dynamic. For one, unlike earlier Web pages that had static structure and hard-coded components, modern Web applications can react to user input, generate software components at runtime, assemble components from varied sources, and create Web pages on the fly [18, 41]. Moreover, interaction between clients and servers can change dynamically over a session depending on how users interface with a system. Finally, in Web development, application requirements routinely change because of advances in technology or in response to user demand. All combined, dynamically generated components, dynamic interaction among clients and servers, constant changes in application requirements, and continually evolving technologies make techniques that were effectively applied to simple Web applications with traditional client–server systems inadequate for testing dynamic functionality.

2.3 Unpredictable Control Flow

Variance in control flow was generally not a factor for traditional systems because flow was exclusively managed by program controllers. Since Web applications can have several entry points and users can arbitrarily navigate to previously visited Web pages by interacting with their Web browser interface, control flow in Web applications is largely unpredictable [1]. In terms of entry points, users can directly access Web pages when given the appropriate Uniform Resource Locator (URL). In cases when Web applications consist of several Web pages that are expected to be accessed in a particular order, users could find themselves at an improper starting point if they type in the URL to an intermediate page directly or they discover an intermediate page in a batch of search engine returns. To ensure proper functionality, this factor must be carefully accounted for during development to ensure that users can, in effect, find their way to the intended start page if they happen to land somewhere in the middle. Since users interact with Web applications through browsers, loose coupling of browser controls and the Web application can translate into unexpected failures and anomalies. For instance, a user can break normal control flow by refreshing a Web page or navigating to an earlier/later point in their navigation history with the help of the back and forward buttons. In either case, the execution context will have changed without notifying the program controller, possibly triggering unexpected results. To ensure that interaction with the browser does not have a negative effect, browser controls and their effects must be factored in during Web application testing [19].

2.4 Significant Variation in Web Access Tools

An important challenge to Web quality assurance stems from increased diversity of client platforms. Although users traditionally explored the Web with versions of either Internet Explorer or Netscape on desktop PCs, recent trends including the emergence of Mozilla, for instance, as a popular browser alternative and shifts toward Web-enabled appliances such as televisions and personal digital assistants (PDAs) suggest that the contemporary face of Web browsing environments is continuing to evolve. While the wide variety of tools used to navigate and interact with the Web provide users with expanded flexibility in choice of access platform, it complicates Web quality assurance. In essence, wide variation translates into a wide space of potential Web client configurations and complicates the testing effort by requiring that Web developers not only ensure that the systems they have developed are correct, but that correctness persists as software is ported. Failure to evaluate Web application portability across the configuration space can result in instances when Web application components render/execute correctly in some client configurations and incorrectly in others.

2.5 Development Factors, Adjusted Quality Requirements, and Novel Constructs

The process used to develop Web applications presents a significant challenge to Web testing. Web software is often developed without a formalized process; developers generally delve directly into the implementation phase, rarely engage in requirements acquisition, and go through a very informal design phase [6, 41]. This direct, incremental development is more than likely the result of two factors: time-to-market pressure and the ability for relatively untrained developers to create and modify Web sites using tools like KompoZer,[3] Amaya,[4] and Dreamweaver[5] that support What You See Is What You Get (WYSIWYG) implementation. To accommodate these factors, testing approaches would, ideally, be automatable and incorporate easily adaptable test suites [24].

[3] http://www.kompozer.net/.
[4] http://www.w3.org/Amaya/Amaya.html.
[5] http://www.adobe.com/products/dreamweaver/.

Shifts in quality requirements for Web applications, in comparison to more traditional software, also impact the Web testing process. According to Wu and Offutt [56], much of the software industry has been able to succeed with relatively low-quality requirements; a combination of timely releases and marketing strategies have almost always determined whether traditional software products succeed competitively. In contrast, Web traffic is heavily influenced by software quality; since users have *point-and-click* access to competitors, they can very easily take their business elsewhere. As a result, the goals of the development process must be reprioritized since producers only see a return on their investment if their Web sites meet consumer demand.

Finally, Web applications incorporate a host of novel constructs; integrating practices for adequate assessment during testing is key. For one, modern Web applications often have interface components that are completely hidden in that they do not correspond to any visible input elements on a Web page [26]. As a result, it is a important to support analysis for Web applications that take hidden elements into account; incomplete information about interfaces can limit the effectiveness of testing and preclude testers from exercising parts of an application that are accessible only through unidentified interfaces [1, 26].

2.6 Summary

In summary, Web applications can be described as heterogeneous, distributed systems that are highly dynamic with unpredictable control flow. Since Web applications have high-quality demands, are expected to run on a wide variety of client configurations, and incorporate novel constructs, it is important that testing approaches adequately address these factors as they apply. In the sections that follow, we take a look at how researchers are meeting these challenges in defining Web application models, Web testing strategies, and Web portability analysis approaches.

3. Web Application Models

In the field of software engineering, models are often used to aid developers in analysis. In general, models help to capture software features relevant to testing by abstracting components, their attributes, and interconnections; models can represent varying degrees of granularity depending on the features salient to the testing approach. This section provides an overview of various Web application modeling techniques and establishes a context for the testing strategies discussed in Section 4.

In particular, we discuss various approaches including Markov models and statecharts, object-oriented models, and regular expressions. In the next section, we look at how these models are used to derive test cases and evaluate the functional and structural correctness of Web applications.

It is important to note that, in their work, Alalfi, Cordy, and Dean [1] surveyed close to two dozen Web application models used to support Web testing and provided a comprehensive discussion of the various techniques used to model navigation, content, or behavior of Web applications. There is only a slight overlap in the models discussed in Ref. [1] and in this chapter.

3.1 Markov Models and Statecharts

Kallepalli and Tian [32] proposed unified Markov models (UMMs) for testing Web applications. In essence, UMMs are variants of Markov models that are defined as a set of hierarchical Markov chains where states are operational units (Web files), edges between states correspond with hypertext links and indicate a possible transition (navigation), and usage probabilities indicate the likelihood of a transition. The UMM shown in Fig. 1 represents a simple Web application that is comprised of a homepage (the intended start page) with links to frequently asked questions (faq) and a registration page. From the faq page, the user can either navigate back to the homepage or leave the Web application altogether; from the registration page, users can either return to the homepage, go to a confirmation page, or quit the Web application. The probability of making transitions between Web pages is listed

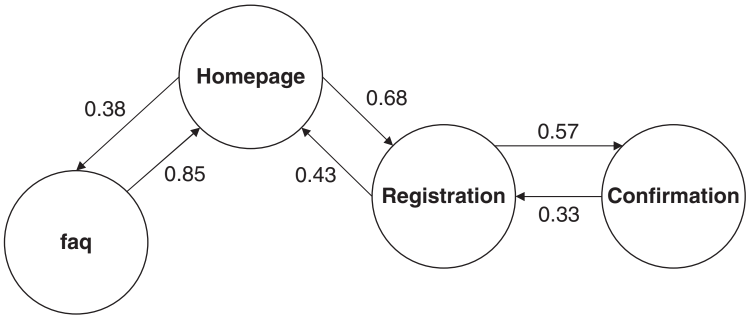


FIG. 1. Unified Markov model (UMM) example for a simple Web application. States represent individual Web pages, edges correspond to hyperlinks, and the probabilities shown represent the likelihood that a user will select the corresponding hyperlink from a certain page.

alongside the edges; as an example, users navigate from the homepage to the faq 38% of the time. The underlying idea is to have the UMM represent execution flow, information flow, and probabilistic usage information. States, edges, and usage probabilities are each recovered from Web logs that maintain a record of user interaction with the system including usage frequency and failure reports; since Web logs are quite common and routinely maintained on the server side, this approach incurs low overhead. The models extracted are eventually used to support statistical testing.

There is quite a bit of similarity between the work of Kallepalli and Tian [32] and that of Sant *et al.* [48]; both use Markov models (or some variation thereof) to represent Web applications, generate the model based on logged user data, and use the model as a basis for testing. The major difference between the two is that Sant *et al.* experiment with using varying degrees of history to estimate whether users will visit a given Web page during a session. In particular, they look at *unigram* models where page visitation is considered independent of previous actions, *bigram* models where the previous Web page visited has an impact, and *trigram* models where the previous two pages help to define the probability that users will visit a given page.

Statecharts generally model reactive systems as a series of states, transitions, events, conditions, and their interrelations. Di Lucca and Penta [19] proposed a Web application model based on statecharts that can be used to analyze how interaction with the Web browser interface affects Web application correctness. As mentioned in Section 2, users can disrupt normal control flow and cause anomalous behavior of a Web-based system by refreshing a Web page or navigating to recently visited Web pages using either the forward or back buttons. As a result, it is important to detect problems that may unintentionally arise from user interaction with the Web browser interface. Di Lucca and Penta [19] presented the following model: the browser is characterized by the Web page displayed, by the state of its buttons (*enabled* or *disabled*), and the history of Web pages visited using the browser buttons. Each of these features is captured in a statechart, where each state is defined by the page displayed and by the state of the buttons while the user actions on page links or browser buttons determine the state transactions. Consider the statechart shown in Fig. 2; in this example, the user starts at a search engine, gets a list for search results from a query, and follows a link of interest. Each of the states is labeled with a brief description of the page loaded in the browser (i.e., search engine) and the state of the back and forward buttons. As an example, if the back button is disabled and the forward button is enabled, the corresponding label would be *BDFE* where *B* corresponds with back, *D* indicates that the button is disabled, *F* corresponds with forward, and *E* indicates that the button is disabled.

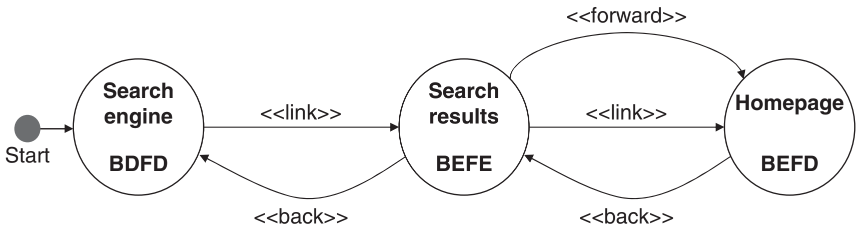


FIG. 2. Statechart example for a simple user session. In this example, the user starts at a search engine, submits a query, and activates a link to retrieve search results. From there, a link is activated to reach the homepage of a particular search return. Note, once the user reaches the homepage, they can only return to the previous page, search results, since there are no links to another page; as a result the forward button is disabled and the back button is enabled.

3.2 Object-Oriented Models

Several researchers have explored the use of object-oriented models for Web applications. This is largely because components and attributes of Web applications can be easily and accurately represented using object-oriented models and using such models facilities application of pre-existing object-oriented software testing techniques [57]. In this section, we explore object-oriented support for Web application modeling. In general, with object-oriented approaches, the central entity in a Web site is the Web page; a Web page contains the information to be displayed to the user and the navigation links toward other pages. It also includes components that facilitate organization (i.e., frames) and interaction (i.e., forms). Web pages can be static or dynamic; while the content of a static page is fixed, the content of a dynamic page is computed at runtime by a server and may depend on input provided by the user through input fields. Subclasses of the Web page model are generally defined to capture differences between the two.

One of the earlier papers defining an object-oriented approach to Web application modeling was written by Coda *et al.* [16] and provided an overview of WOOM (Web object-oriented model). Defined as a modeling framework that could be used to support Web site implementation, WOOM instances were designed to interface between the underlying concept for a Web site and its actual implementation. WOOM uses resources, elements, sites, server, links, and transformers to define Web sites. Liu *et al.* [36] introduced an object-oriented model, called the Web Application Test Model (WATM), that was designed to support data flow testing for Web applications. Liu *et al.* use static analysis of source files to create a model that represents Web applications as a set of interactive software components. Components include client pages, server pages, or program modules; attributes can be program variables or widgets and operators are defined as functions written in scripting or programming languages. Xu and Xu [57] defined an object-oriented model with three levels: the object model, the interactive relation model, and the architecture model. The object model captures attributes of and possible actions on objects (Web page components); the interactive model captures relationships between objects (how Web components influence and connect with each other); and the architecture model provides an overview of the Web application as a whole. These three levels were designed to, respectively, support unit, integration, and system testing.

A significant research effort in the area of object-oriented Web application models has been invested in extending and applying the Unified Modeling Language (UML), a family of languages primarily used in modeling and specification of more traditional object-oriented systems [1]. Very early on, Conallen [17] introduced extensions to UML, namely a new set of class and association UML stereotypes, that could support the capture of Web application-specific elements; the idea behind [17] was to provide a common way for application designers to express the entirety of their applications design with UML. Note both WOOM and the work by Conallen were motivated by design-based goals as opposed to testing; they are primarily mentioned here because of their novelty in this area when they were introduced.

Ricca and Tonella [42–45] developed a UML metamodel for high-level representation of Web applications which supports evaluation of static site structure and can be used to semiautomatically generate test cases. The analysis model primarily captures navigation and interaction capabilities and it is derived from artifacts used by a Web server such as Common Gateway Interface (CGI) scripts as well as information manually provided by developers [42]. When performing testing, Ricca and Tonella reinterpret the UML model into a graph by associating objects with nodes and associations with edges. This enables traditional analyses that use graphs as a basis, such as traversal algorithms, to be applied; simple analysis can detect unreachable pages and support flow analysis to detect data dependences.

Di Lucca *et al.* also base their Web application model on UML. In Ref. [20], the authors present a tool that supports construction of UML diagrams for Web applications that lack design documents; Di Lucca *et al.* use UML to depict several aspects of a Web application including its structure and static/dynamic behavior at different abstraction levels. The UML diagram is generated by a tool that analyzes the source code of the application, extracts and abstracts relevant information form it, and populates a repository with the recovered information.

Bellettini *et al.* [5] discuss WebUML, a tool that generates UML models using static analysis to extract the navigational structure and dynamic analysis to recover behavior-related information about the application. In particular, WebUML constructs class and state diagrams through static source code analysis and dynamic

Web server interaction. Class diagrams represent components of a Web application including forms, frames, applets, and input fields; state diagram models are used to model entities such as active documents, which couple HTML with scripting code, and capture function call flow and navigation to other entities. It is important to note that dynamic analysis is performed by generating a set of server-side script mutants and using them in a navigation simulation; the Web pages that result from the previous step are then analyzed using static source code techniques [5].

3.3 Regular Expressions

Two lines of research incorporate regular expression notation in modeling Web applications; the idea in each is to capture structural information (i.e., arrangement of text, widgets, etc.) and possible arrangement of dynamic content (i.e., two widgets as opposed to one when the user provides a given input). Wu and Offutt [56] model individual Web pages as regular expressions to represent the static arrangement of Java servlet-based software and the dynamic sections that can vary from instance to instance. In their approach, the overall Web page P is comprised on various elements p_n. Dynamic sections are modeled as the basic elements that can be generated and standard regular expression notation is used to concisely model conditions on the appearance of each in the final HTML file. As an example, consider that $P \rightarrow p_1 \cdot (p_2 \mid p_3)^* \cdot p_4$ indicates that p_1 and p_4 will always be at the beginning and end of the corresponding HTML file; this captures the static arrangement of Web page elements. Meanwhile, either p_2 or p_3 can occur 0 or more times in the resulting page; this of course represents the dynamic nature of the page. This is a basic example of their overall approach but it captures the spirit of their work quite nicely.

In the second line of research, Stone and Dhiensa [54] present a generalized output expression that represents every possible output. While the spirit and basic motivation behind Refs. [54, 56] are the same, the former includes more advanced notation that allows other interaction factors (i.e., the affect of browser interaction on the state of dynamic elements) to be represented and the latter uses metatags instead of standard regular expression notation.

4. Web Test Case Generation

One of the basic goals of Web testing is fault discovery; characterizing the faults that affect Web applications, developing methodologies for deriving test cases, and establishing effective testing strategies have been active research areas.

For instance, in their work, Ricca and Tonella [44] derived a Web application fault classification model by analyzing publicly available fault reports for Web applications. While some faults included in the model occur in conventional software, others are more specific to Web applications and arise from their peculiarities. The faults in the model include authentication issues, hyperlink problems, crossbrowser compatibility (which we call portability; see Section 5), Web page structure errors, cookie/value setting issues, and incorrect protocols. In this section, we look at how the Web application models introduced in Section 3 are used to generate test cases and provide a basis for testing techniques that support fault discovery in Web applications.

4.1 Markov Models and Statecharts

Kallepalli and Tian [32] use UMMs to model usage patterns in Web applications; in particular, their model captures the likelihood that users transition from one page to the next and can be used to determine the probability of a given path from an arbitrary source state (Web page) to a sink state. To support statistical testing, Kallepalli and Tian suggest setting a probability threshold and exercising each navigation path with a higher likelihood; this approach focuses testing efforts on the most likely usage scenarios. Hao and Mendes [27] replicated this work and show that UMMs are effective in statistical testing. They extended the work of Kallepalli and Tian to account for the existence of various entry nodes, or start Web pages, in the course of a usage session. As noted in Section 2, users can start at various places in Web applications. To account for this factor Hao and Mendes use various UMMs to model a Web site, each with a different entry node; for clarity, Kallepalli and Tian only use one UMM. Similarly, Sant *et al.* [48] discuss generating test cases from random walks through Markov models.

Di Lucca and Penta [19] designed a Web application model to help developers evaluate how interaction with Web browser buttons could adversely affect system functionality. Recall, the model they developed is a statechart in which each state is defined by the Web page displayed in the browser and the status of the forward, backward, and refresh buttons; transitions between states occur when a new Web page is loaded (e.g., through a link) or either of the three buttons is activated. Di Lucca and Penta expect this approach to be integrated with other testing strategies; once a set of source-to-sink test case paths have been generated using some other approach, the idea is to create a statechart that corresponds to that sequence and include the effect of activating the forward, backward, and refresh buttons as states. The next step would be to define the coverage criteria that must be satisfied and generate a test suite that meets the given criteria. In this work, Di Lucca and Penta primarily outline a model to complement existing techniques; the testing approach ultimately applied is left open.

4.2 Object-Oriented Models

Liu *et al.* [36] extend data flow testing techniques to HTML and XML to ensure that Web application data is stored, computed, and used properly. In data flow testing, program execution paths are selected based on definition–use[6] chains of variables. Since script variables and document widgets store the variables in Web applications, they are a primary target for this approach. In particular, script variables, widgets, and the Web pages that contain them are each considered objects with attributes; relationships among objects are used to generate test cases that monitor the flow of data. To accommodate the various data dependencies, Liu *et al.* define intraobject testing where test paths are selected for the variables that have definition–use chains within the object, interobject testing, where test paths are selected for variables that have def-use chains across objects, and interclient testing, where tests are derived to evaluate data interactions among clients. To test Web applications, definition–use chains need to be extended to HTML and XML documents and to cross HTTP client/server boundaries.

Ricca and Tonella [42–45] essentially use a graph-based version of their UML model to generate test cases and perform Web application testing. The tool they developed for testing is called TestWeb and the tool they use for model extraction is called ReWeb. In their approach, Web application test cases are represented as a sequence of URLs (to correspond with the Web pages in a navigation path) and values for form inputs when necessary. To derive test cases, TestWeb uses the Web application model extracted by ReWeb to generate a set of navigation paths based on some coverage criteria; testers are then responsible for manually providing values for form inputs. Once the paths are defined and values have been provided, TestWeb then automates test case execution; testers must then evaluate the results to distinguish passing test cases from failing ones. One limiting factor in this approach is the need for testers to provide form input values. In response to this issue, Elbaum *et al.* [24] apply the same basic technique as Ricca and Tonella but they further automate this approach by using data captured in actual user sessions to supply form inputs; the goal of this work is to minimize the need for tester intervention.

Finally, Bellettini *et al.* [6] introduce a semiautomatic technique for test case definition that uses a UML model at its base. Much like Kallepalli and Tian [32],

[6] Note, definition–use chains correspond to the definition of a variable v in a given program and all reachable uses of v that occur prior to any redefinition.

Bellettini *et al.* use knowledge of previous user interactions to determine the most exercised navigation paths and focus the testing effort on them. The tool they have developed to implement this technique, TestUML, is a testing suite that uses generated models to define test cases, coverage testing criteria, and also reliability analysis.

4.3 Regular Expressions

Regular expressions have been used to model Web applications because, using the notation, a compact representation of structure, variation, and iteration of HTML files can be expressed in a generalized way. Wu and Offutt [56] use regular expressions to model Web applications and characterize test case execution as a sequence of interactions between client and servers that begin at a source Web page and uses composition and transition rules to reach the sink. Variation in transition rules can lead to different navigation paths and each of those paths can be used as a test case.

Stone and Dhiensa [54] also use regular expressions to model Web applications and, like Wu and Offutt, are motivated by a need to evaluate dynamic Web pages to find errors. The goal of this work, in particular, is to minimize the possibility for code support errors in all possible output from a script. The basic idea of their approach is to compare the expected structure of script statements with the output actually produced. The authors suggest that only a slight extension to currently existing tools is needed to ensure they can accept and validate the more generalized model.

5. Portability Analysis

Though the process of detecting and correcting faults in an implemented software system is inherently difficult, software quality assurance becomes increasingly complex when faults only surface in precise configurations [28]. In such cases, the number, nature, and interconnection of constituent parts that define the configuration[7] can significantly impact software quality. To adequately reduce the number of faults in the delivered product, developers must evaluate the overall correctness of the implementation in addition to how that correctness is affected by variation in configurations. We refer to the process of detecting and diagnosing faults that are only triggered in precise configurations as *portability analysis*. Without an efficient, thorough technique for assessing software portability, quality could degrade as software is ported and configuration faults, or faults that are only activated in

[7] http://www.chambers.com.au/glossary/configur.htm.

specific configurations, have the potential to remain latent until they are encountered by users in the field.

While configuration faults affect portability for a wide range of software types, they are a particular challenge in Web application development. Given that there are several different browsers,[8] each with different versions,[9] a number of operating systems on which to run them,[10] and dozens of settings,[11] users have expanded flexibility in Web access options and the client configurations used to explore the Web are highly varied. Though this expanded variation and flexibility allows for more customized Web user experiences, subsequent differences across configurations present a serious challenge for Web developers to ensure universal quality.

Ideally, Web applications would behave and execute uniformly across heterogeneous client configurations; in such a situation, quality assurance could effectively be carried out on one client configuration and the results extrapolated for the entire set. Yet, in practice, the makeup of the client configuration has a significant impact on Web application execution (Fig. 3). Since Web applications are expected to enable crossplatform access to resources for the large, diverse user community, it is important to evaluate how well a given system meets that demand [32].

In the previous sections, most of the discussion centered on testing Web applications to ensure functional and structural correctness. In this section, we discuss various Web application portability analyses which include launching Web applications in varied configurations, looking for unsupported HTML in source code, and attempting to transform code into a form that is supported. In particular, we outline existing approaches along with their limitations and briefly discuss tools that implement them.

5.1 Manual and Automated Execution-Based Approach

Execution-based approaches to Web portability analysis primarily involve launching Web applications in target configurations and qualitatively comparing expected and observed results to verify correctness. In practical terms, this means that Web applications must be physically loaded to perform execution-based quality assurance. In the brute-force application of this approach, Web application

[8] For example, Microsoft Internet Explorer (IE), Netscape, AOL Browser, Opera, Mozilla, Safari for Mac OS X, Konqueror for Linux, Amaya, Lynx, Camino, Java-based browsers, WebTV.

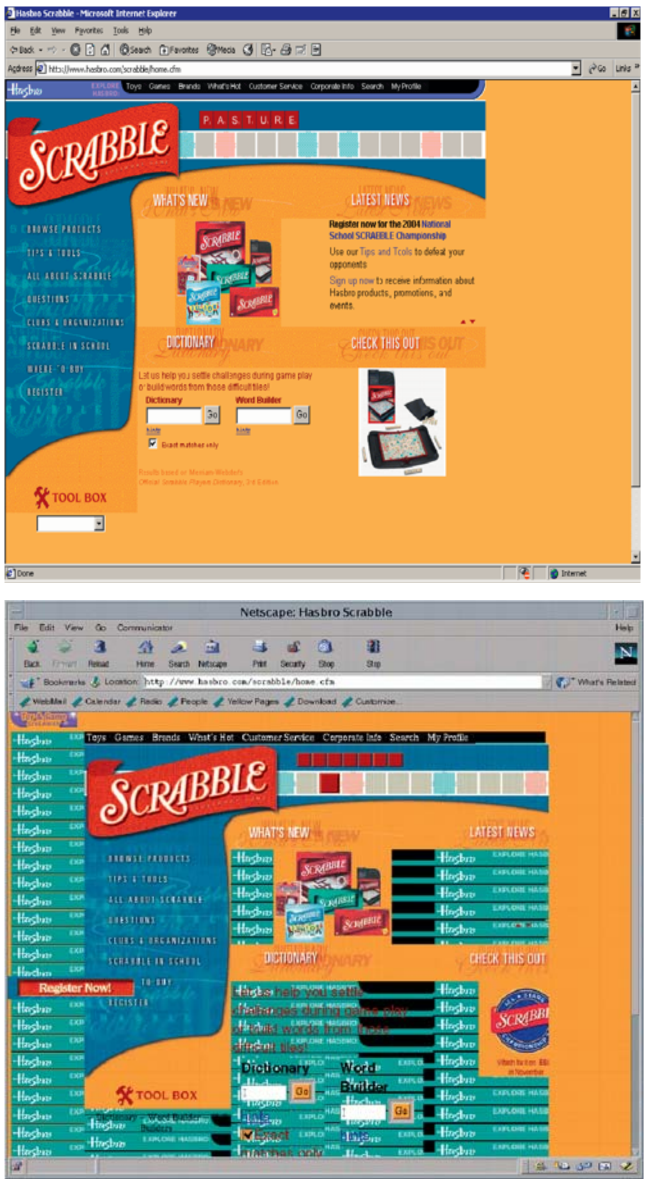
[9] For example, IE 4.0, IE 5.0, IE 6.0.

[10] For example, Windows, Power Macintosh.

[11] For example, browser view, security options, script enabling/disabling.

A

B



Register now for the 2004 National School SCRABBLE Championship

Use our Tips and Tools to defeat your opponents

Sign up now to receive information about Hasbro products, promotions, and events.

Let us help you settle challenges during game play or build words from those difficult tiles!

FIG. 3. When rendered in (A) Internet Explorer 6.0 and (B) Netscape 4.8, both on Windows XP, the Scrabble Homepage is significantly different.

deployment and analysis are both carried out manually. Though exhaustive coverage of the configuration space would allow thorough portability analysis, physical access to each possible browsing environment is extremely difficult and nearly impossible; as a result, there is a notable conflict between the need to test each potential client configuration and the constraints imposed by limited development resources. Testing on all necessary combinations of hardware, operating systems, browsers, connection settings, etc., normally requires labor-intensive setups on many dedicated machines making it extremely difficult for a test team working with limited equipment to replicate certain configuration faults. Even with access to each possible configuration, the time and effort required to effectively assess Web pages using this strategy can also impede the depth of the Web application evaluated. Because this approach can be weakened by client configuration availability and limited time, this technique is highly ineffective and impractical for Web developers interested in establishing portability across a vast, richly defined configuration space.

While the brute-force strategy evaluates Web application portability postimplementation, Berghel [7, 8] presented a manual execution-based approach designed for preimplementation use. The basic idea outlined in Refs. [7, 8] is to launch a suite of test Web pages, called *Web Test Patterns*,[12] and use the results to gauge the level of HTML support in varied configurations. This approach allows Web developers to derive a cognitive model of HTML support criteria across various configurations; they could then use this model to drive decisions regarding which HTML tags to include in an implementation during code synthesis. Much like the brute-force strategy, the effectiveness of this approach is mainly restricted by resource limitations. In addition, this strain of Berghel's approach only allows users to develop a mental model of tag support criteria; effective application of this model can be severely flawed in practice given the expansive set of HTML tags that can be included in source code and the intricacy of support criteria. Retaining this information and attempting to use this strategy effectively is clearly time-, cognition-, and resource-intensive.

To minimize the effort and, ultimately, the cost of analysis using execution-based approaches, researchers have explored collapsing the space of test configurations through combinatorial testing approaches. In particular, Xu *et al.* [58, 59] propose applying single-factor and pairwise coverage criteria to systematically reduce the space of distinct configurations evaluated during quality assurance. This process applies sampling heuristics to define the minimal set of client configurations that must be assessed to establish confidence in the entire configuration space. While this approach can make subsequent analysis more cost-effective in terms of resources

[12] Each Web test pattern in the suite incorporates several HTML tags and descriptions of the impact they would have if processed correctly.

and effort, it can also create false confidence in analysis results when the set of test configurations does not accurately represent the entire space.

Commercial tools, like Browser Photo[13] and BrowserShots,[14] that are designed to make execution-based approaches more cost effective mainly automate the launch of Web applications in varied configurations to mitigate the necessity for in-house access to configurations during quality assurance. Such tools work on the behalf of Web developers by launching applications in a set of target configurations and capturing a screenshot of the rendered result; developers assess Web application correctness by manually examining the returned screenshots and relying on visual cues (i.e., misrendered pages) to discover errors. Once errors are detected, the developer must employ additional methods, such as manually examining source code, to identify fault causes. The main flaws of this approach stem from the fact that fault detection efficacy is generally constrained by the dimensions of the screen capture and the extent to which the set of client configurations used during analysis accurately represent the entire configuration space. In other words, since the result of this approach only yields visual evidence of an error, faults triggered by user action or those that fall out of the range of the screenshot will remain undetected since a single snapshot cannot capture such defects. Also, since there is no indication as to why the error occurred, it is *nondiagnostic*; identifying factors that contribute to the anomaly requires more work and effort. In addition, if the space of configurations is not adequately inclusive, critical faults could remain undetected.

In general, execution-based approaches are deficient because of limited configuration coverage, lack of diagnostic ability, limited applicability of results or some combination of these factors. As a result, practical implementation of execution-based strategies generally involves configuration sampling. Such issues give rise to an incomplete, resource-intensive analysis of the Web application that does not provide an adequate basis for establishing confidence in Web application portability.

5.2 Lookup-Based Approach

Lookup-based approaches, like Doctor HTML[15] and Bobby,[16] detect configuration faults by maintaining an account of unsupported HTML tags in a predefined subset of Web configurations and essentially scanning source code for them.

[13] http://www.netmechanic.com/browser-index.htm.
[14] ttp://browsershots.org/.
[15] http://www2.imagiware.com/RxHTML/.
[16] http://www.watchfire.com/products/webxm/bobby.aspx.

Results of analysis are returned as a list of the unsupported tags found in the given Web application source code and the configurations with support violations.

One problem of this approach is captured nicely by Fig. 3. In this example, quality is clearly diminished for Netscape users of the Scrabble Web site, however, Doctor HTML did not include this particular support violation, namely lack of support for the HTML tag <div style=background-image:url(. . .)>, in the analysis report. This factor drives home the point that this type of analysis will only be as thorough as the knowledge of configuration support criteria utilized. In instances when incomplete or inaccurate support criteria is used, configuration faults will continue to remain latent after analysis. Since the tool approach is proprietary, it is unclear whether this oversight occurred because the given HTML tag was missing from the checklist.

In our own work [21–23], we define an advanced framework that incorporates aspects of both the lookup-based and execution-based approaches. In particular, we have defined an automated, model-based framework that uses static analysis to detect and diagnose Web configuration faults. Our approach overcomes the limitations of current techniques by enabling efficient portability analysis across the vast array of client environments. The basic idea behind this approach is that source code fragments [i.e., HTML tags and Cascading Style Sheet (CSS) rules] embedded in Web application source code adversely impact portability of Web applications when they are unsupported in target client configurations. Without proper support, the source code is either processed incorrectly or ignored and the aesthetic or functional properties associated with the code may be lost resulting in configuration faults. Our approach is to model source code support in various configurations and perform portability analysis by checking for support violations in source code inclusion. In the effort to fully exploit this approach, improve practicality, and maximize fault detection efficiency, manual and automated approaches to client support knowledge acquisition have been implemented, variations of Web application and support criteria models have been investigated, and visualization of configuration fault detection results has been explored. To optimize the automated acquisition of client support knowledge, alternate machine learning strategies have been empirically investigated and provisions for capturing tag/rule interaction have been integrated into the process.

Figure 4 provides a high-level overview of four processes implemented in our framework. In particular, updateKB() is used to acquire knowledge of code support; processURL() and query() are key in portability analysis; and generateReport() is mainly responsible for presenting analysis results. To initiate analysis, Web developers submit the URL associated with the homepage of a Web application to the *Oracle* and processURL() activates a Web crawler that retrieves Web application source code and forwards it to query(). Next, query() analyzes the source code to detect support violations. Any violations discovered are presented to the Web developer by way of generateReport(). It is important to note that the integrity of the report generated is largely a factor of how comprehensive the knowledge base is;

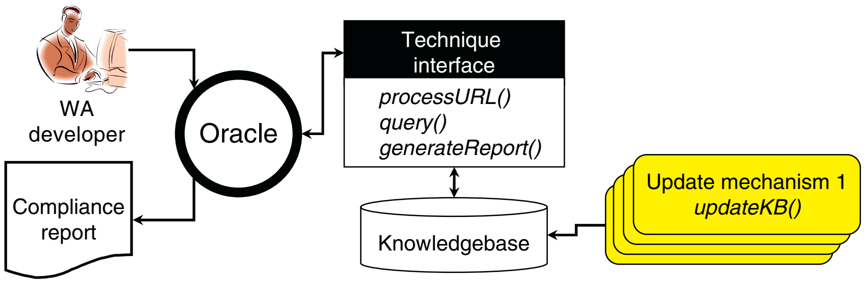


FIG. 4. High-level overview of a framework for detecting configuration-specific faults in Web applications.

subsequently, updateKB() allows both automated updates using machine learning methods and manual updates in which Web developers import support rules they, more than likely, know from experience. The underlying goal of automated, or machine learning-based, knowledge updates is to compare the source code of Web application components, namely Web pages, that render/execute properly in a given configuration with those that do not to discover possible support violations. If, for instance, a given tag consistently appears in Web pages that do not execute properly yet never in a correct Web page, it is expected to be unsupported.

In comparison to execution-based techniques, our approach bypasses the need to launch Web applications and applies a static, model-based analysis. This enables more efficient fault detection and diagnosis by reducing the need for configuration access and simultaneously reducing the threat of inaccurate equivalence assumptions. In terms of lookup-based approaches, our work uses a more inclusive model of both HTML and CSS crossconfiguration support during analysis; integrates diverse knowledge acquisition strategies to build an accurate, thorough model of support knowledge; and incorporates an extensible knowledge base model that allows support criteria to continually evolve.

5.3 Source Code Transformation Approach

Though Chen and Shen [13] do not specifically focus on Web configuration fault detection, correcting Web portability threats is a key aspect of their work and is highly applicable to the domain of Web portability analysis. In their research, Chen and Shen base their approach on the assumption that Web source code standards, as defined by The World Wide Web Consortium (W3C),[17] provide the most effective

[17] http://www.w3.org/.

basis for developing Web applications that are portable. The crux of their technique is to transform the source code of a Web application into a standardized form in which all nonstandard code fragments have been eliminated from the source yet the appearance of the original implementation is preserved. One problem with this approach stems from the fact that, as noted by Phillips [40], even if browsers fully comply with published standards, source code may still be processed differently since standards do not address every detail of implementation; in addition, there are instances in which browsers claim to be standards-compliant yet some HTML tags deemed standard by the W3C are unsupported or supported improperly [15]. In some instances, Web developers only get acquainted with the parts of the standards that work in most browsers through experience [40]; subsequently, developers may still have to employ a variant of the execution-based approach to assess source code support in client configurations.

Artail and Raydan [3] address the problem of enabling Web applications designed and tested on desktops to render properly on small-screen devices such as PDAs. At the root of the problem, mobile devices have constraints in resources and processing capabilities that make them unable to launch the vast majority of Web pages developed for desktop computers properly; Artail and Raydan describe a method for automatically re-authoring source code so that Web applications can render on a smaller screen and maintain the overall integrity of the original structure. The crux of the approach probes HTTP request headers to detect when clients are small-screen devices, to obtain dimensions of the screen size, and to use that information as a guide to transform the original source into a more compatible version while preserving the structural format of the Web page. Artail and Raydan use heuristics to reduce the size of page elements, resize images, hide text, and transform tables into text. While screen dimension constraints provide significant motivation for this work, it is important to note that there are also constraints on computing power and other resources as well; subsequently, Artail and Raydan retrieve the original source code all at once and incorporate Javascript code to display/hide parts of the Web page without having to revisit the server. This ultimately reduces user wait time considerably, saves battery power, and minimizes wireless network traffic.

6. Conclusion

Web development presents a myriad of unique challenges and requires adapted or newly developed techniques for various stages of the process. As one of the most widely used class of software to date with continually evolving design technologies, increased expectation of correctness from the user community, and short time-to-market pressures, a need for more rigorous testing approaches that can effectively reveal faults is key. Researchers are currently working to address testing problems for Web applications and to propose effective solutions.

This chapter explored research contributions toward cost efficient, effective testing of Web-based applications. In particular, we looked at the challenges involved in Web testing and discussed Web application models used, various Web testing strategies, and Web portability analysis approaches. In terms of models, we looked at variant uses of Markov models and statecharts, object-oriented models, and regular expressions. We used the discussion of those models as a basis for exploring various Web testing approaches. We then provided an overview of research efforts aimed at developing approaches for effective discovery of Web configuration faults; the goal of work in this area is to fulfill the challenge of the Web to provide configuration-independent quality to users.

As with testing research directed toward more conventional software systems, the quest to achieving quality is rather elusive and continually evolving. We expect future work in Web application testing to build upon the ideas expressed in this chapter and to become increasingly important as the Web continues to grow and evolve.

References

[1] Alalfi M. H., Cordy J. R., and Dean T. R., 2007. A survey of analysis models and methods in website verification and testing. In *ICWE, Volume 4607 of Lecture Notes in Computer Science*, L. Baresi, P. Fraternali, and G.-J. Houben, eds. Springer, Berlin. ISBN 978-3-540-73596-0.

[2] Amsler D. A., 2003. Establishing standards for usable and accessible user services web sites. In *SIGUCCS'03: Proceedings of the 31st Annual ACM SIGUCCS Conference on User Services*. ACM Press, New York, NY.

[3] Artail H. A., and Raydan M., 2005. Device-aware desktop web page transformation for rendering on handhelds. *Personal and Ubiquitous Computing*, **9:** 368–380. ISSN 1617-4909.

[4] Atkinson C., Bunse C., Grosz H.-G., and Kühne T., 2002. Towards a general component model for web-based applications. *Annals of Software Engineering*, **13:** 35–69. ISSN 1022-7091.

[5] Bellettini C., Marchetto A., and Trentini A., 2004. WebUML: Reverse engineering of web applications. In *SAC'04: Proceedings of the 2004 ACM Symposium on Applied Computing*. ACM Press, New York, NY. ISBN 1-58113-812-1.

[6] Bellettini C., Marchetto A., and Trentini A., 2005. TestUML: User-metrics driven web applications testing. In *SAC'05: Proceedings of the 2005 ACM Symposium on Applied Computing*. ACM Press, New York, NY. ISBN 1-58113-964-0.

[7] Berghel H., 1995. Using the www test pattern to check HTML compliance. *Computer*, **28:** 63–65.

[8] Berghel H., 1996. HTML compliance and the return of the test pattern. *Communications of the ACM*, **39:** 19–22.

[9] Bevan N., Barnum C., Cockton G., Nielsen J., Spool J., and Wixon D., 2003. The ''magic number 5'': Is it enough for web testing? In *CHI'03: CHI'03 Extended Abstracts on Human Factors in Computing Systems*. ACM Press, New York, NY. ISBN 1-58113-637-4.

[10] Braband C., Møller A., and Schwartzbach M., 2001. Static validation of dynamically generated HTML. In *PASTE'01: Proceedings of the 2001 ACM SIGPLAN–SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*. ACM Press, New York, NY. ISBN 1-58113-413-4.

[11] Brajnik G., 2004. Using automatic tools in accessibility and usability assurance processes. In *LNCS Proceedings of the 8th ERCIM Workshop on User Interfaces for All*. Springer, Berlin.

[12] Cai Y., Grundy J., and Hosking J., 2007. Synthesizing client load models for performance engineering via web crawling. In *ASE'07: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*. ACM Press, New York, NY. ISBN 978-1-59593-882-4.

[13] Chen B., and Shen V. Y., 2006. Transforming web pages to become standard-compliant through reverse engineering. In *W4A: Proceedings of the 2006 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*. ACM Press, New York, NY. ISBN 1-59593-281-X.

[14] Chi E. H., Rosien A., Supattanasiri G., Williams A., Royer C., Chow C., Robles E., Dalal B., Chen J., and Cousins S., 2003. The bloodhound project: Automating discovery of web usability issues using the Infoscent™ simulator. In *CHI'03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, New York, NY. ISBN 1-58113-630-7.

[15] Clark J., 2000. The glorious peoples myth of standards compliance, http://joeclark.org/glorious.html.

[16] Coda F., Ghezzi C., Vigna G., and Garzotto F., 1998. Towards a software engineering approach to web site development. In *IWSSD'98: Proceedings of the 9th International Workshop on Software Specification and Design*. IEEE Computer Society, Washington, DC.

[17] Conallen J., 1999. Modeling web application architectures with UML. *Communications of the ACM*, **42:** 63–70. ISSN 0001-0782.

[18] Di Lucca G. A., and Fasolino A. R., 2006. Testing web-based applications: The state of the art and future trends. *Information and Software Technology*, **48:** 1172–1186. ISSN 0950-5849.

[19] Di Lucca G. A., and Penta M. D., 2003. Considering browser interaction in web application testing. In *Proceedings of the 5th International Workshop on Web Site Evolution* IEEE Computer Society, Washington, DC. ISBN 0-7695-2016-2.

[20] Di Lucca G. A., Fasolino A. R., Pace F., Tramontana P., and de Carlini U., 2002. Ware: A tool for the reverse engineering of web applications. In *CSMR'02: Proceedings of the 6th European Conference on Software Maintenance and Reengineering*. IEEE Computer Society, Washington, DC.

[21] Eaton C., and Memon A. M., 2004. Evaluating web page reliability across varied browsing environments. In *Proceedings of the 15th IEEE International Symposium on Software Reliability Engineering (ISSRE'04)*, Saint-Malo, Bretagne, France.

[22] Eaton C., and Memon A. M., 2004. Improving browsing environment compliance evaluations for websites. In *Proceedings of the International Workshop on Web Quality (WQ 2004).*

[23] Eaton C., and Memon A. M., 2007. An empirical approach to testing web applications across diverse client platform configurations. *International Journal of Web Engineering and Technology (IJWET), Special Issue on Empirical Studies in Web Engineering*, **3:** 227–253.

[24] Elbaum S., Karre S., and Rothermel G., 2003. Improving web application testing with user session data. In *ICSE'03: Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society, Washington, DC. ISBN 0-7695-1877-X.

[25] Gaur N., 2000. Assessing the security of your web applications. *Linux Journal*, **3**. ISSN 1075-3583.

[26] Halfond W. G. J., and Orso A., 2007. Improving test case generation for web applications using automated interface discovery. In *ESEC-FSE'07: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The foundations of Software Engineering*. ACM Press, New York, NY. ISBN 978-1-59593-811-4.

[27] Hao J., and Mendes E., 2006. Usage-based statistical testing of web applications. In *ICWE'06: Proceedings of the 6th International Conference on Web Engineering*. ACM Press, New York, NY. ISBN 1-59593-352-2.

[28] Hao D., Zhang L., Mei H., and Sun J., 2006. Towards interactive fault localization using test information. In *APSEC'06: Proceedings of the XIII Asia Pacific Software Engineering Conference*. IEEE Computer Society, Washington, DC. ISBN 0-7695-2685-3.

[29] Hassan A. E., and Holt R. C., 2002. Architecture recovery of web applications. In *ICSE'02: Proceedings of the 24th International Conference on Software Engineering*. ACM Press, New York, NY.

[30] Huang Y.-W., Huang S.-K., Lin T.-P., and Tsai C.-H., 2003. Web application security assessment by fault injection and behavior monitoring. In *WWW'03: Proceedings of the 12th International Conference on World Wide Web*. ACM Press, New York, NY. ISBN 1-58113-680-3.

[31] Joshi J. B. D., Aref W. G., Ghafoor A., and Spafford E. H., 2001. Security models for web-based applications. *Communications of the ACM*, **44:** 38–44. ISSN 0001-0782.

[32] Kallepalli C., and Tian J., 2001. Measuring and modeling usage and reliability for statistical web testing. *IEEE Transactions on Software Engineering*, **27:** 1023–1036. ISSN 0098-5589.

[33] Kasday L. R., 2000. A tool to evaluate universal web accessibility. In *CUU'00: Proceedings on the 2000 Conference on Universal Usability*. ACM Press, New York, NY.

[34] Kostoulas M. G., Matsa M., Mendelsohn N., Perkins E., Heifets A., and Mercaldi M., 2006. XML screamer: An integrated approach to high performance XML parsing, validation and deserialization. In *WWW'06: Proceedings of the 15th International Conference on World Wide Web*. ACM Press, New York, NY. ISBN 1-59593-323-9.

[35] Levi M. D., and Conrad F. G., 1997. Usability testing of World Wide Web sites. In *CHI'97: CHI'97 Extended Abstracts on Human Factors in Computing Systems*. ACM Press, New York, NY. ISBN 0-89791-926-2.

[36] Liu C.-H., Kung D. C., Hsia P., and Hsu C.-T., 2000. Structural testing of web applications. In *Proceedings of ISSRE 2000*, p. 84. ISSN 1071-9458.

[37] Martin E., Basu S., and Xie T., 2007. WebSob: A tool for robustness testing of web services. In *ICSE COMPANION'07: Companion to the Proceedings of the 29th International Conference on Software Engineering*. IEEE Computer Society, Washington, DC. ISBN 0-7695-2892-9.

[38] Murugesan S., and Deshpande Y., 2002. Meeting the challenges of web application development: The web engineering approach. In *ICSE'02: Proceedings of the 24th International Conference on Software Engineering*. ACM Press, New York, NY. ISBN 1-58113-472-X.

[39] Offutt J., 2002. Quality attributes of web software applications. *IEEE Software*, **19:** 25–32.

[40] Phillips B., 1998. Designers: The browser war casualties. *Computer*, **31:** 14–16ISSN 0018-9162.

[41] Ricca F., 2004. Analysis, testing and re-structuring of web applications. In *ICSM'04: Proceedings of the 20th IEEE International Conference on Software Maintenance*. IEEE Computer Society, Washington, DC. ISBN 0-7695-2213-0.

[42] Ricca F., and Tonella P., 2001. Analysis and testing of web applications. In *ICSE'01: Proceedings of the 23rd International Conference on Software Engineering*. IEEE Computer Society, Washington, DC.

[43] Ricca F., and Tonella P., 2001. Building a tool for the analysis and testing of web applications: Problems and solutions. In *Proceedings of TACAS*, pp. 373–388.

[44] Ricca F., and Tonella P., 2005. Web testing: A roadmap for the empirical research. In *WSE*. IEEE Computer Society, Washington, DC. ISBN 0-7695-2470-2.

[45] Ricca F., and Tonella P., 2006. Detecting anomaly and failure in web applications. *IEEE Multimedia*, **13:** 44–51ISSN 1070-986X.

[46] Rosmaita B. J., 2006. Accessibility first!: A new approach to web design. In *SIGCSE'06: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education.* ACM Press, New York, NY. ISBN 1-59593-259-3.

[47] Sanchez A., Vega B., Gonzalez A., and Jackson G., 2006. Automatic support for testing web-based enterprise applications. In *ACM-SE 44: Proceedings of the 44th Annual Southeast Regional Conference.* ACM Press, New York, NY. ISBN 1-59593-315-8.

[48] Sant J., Souter A., and Greenwald L., 2005. An exploration of statistical models for automated test case generation. *ACM SIGSOFT Software Engineering Notes*, **30:** 1–7ISSN 0163-5948.

[49] Scott D., and Sharp R., 2002. Abstracting application-level web security. In *WWW'02: Proceedings of the 11th International Conference on World Wide Web.* ACM Press, New York, NY. ISBN 1-58113-449-5.

[50] Sevilla J., Herrera G., Martínez B., and Alcantud F., 2007. Web accessibility for individuals with cognitive deficits: A comparative study between an existing commercial web and its cognitively accessible equivalent. *ACM Transactions on Computer–Human Interaction*, **14:** 12ISSN 1073-0516.

[51] Shams M., Krishnamurthy D., and Far B., 2006. A model-based approach for testing the performance of web applications. In *SOQUA'06: Proceedings of the 3rd International Workshop on Software Quality Assurance.* ACM Press, New York, NY. ISBN 1-59593-584-3.

[52] Sierkowski B., 2002. Achieving web accessibility. In *Proceedings of the ACM SIGUCS Conference on User Services.*

[53] Sneed H. M., 2004. Testing a web application. In *Proceedings of the 6th International Workshop on Web Site Evolution.* IEEE Computer Society, Washington, DC. ISBN 0-7695-2224-6.

[54] Stone R. G., and Dhiensa J., 2004. Proving the validity and accessibility of dynamic web-pages. In *W4A'04: Proceedings of the 2004 International Cross-Disciplinary Workshop on Web Accessibility (W4A).* ACM Press, New York, NY. ISBN 1-58113-903-9.

[55] Velasco C. A., and Verelst T., 2001. Raising awareness among designers accessibility issues. *ACM SIGCAPH Computers and the Physically Handicapped*, 8–13ISSN 0163-5727.

[56] Wu Y., and Offutt J., 2002. In *Modeling and testing web-based applications.* George Mason UniversityTechnical Report ISE-TR-02-08.

[57] Xu L., and Xu B., 2004. A framework for web applications testing. In *International Conference on Cyberworlds.*

[58] Xu B., Xu L., Nie C., Chu W., and Chang C. H., 2003. Applying combinatorial method to test browser compatibility. In *Proceedings of International Symposium on Multimedia Software Engineering*, pp. 156–162.

[59] Xu L., Xu B., Nie C., Chen H., and Yang H., 2003. A browser compatibility testing method based on combinatorial testing. In *International Conference on Web Engineering.* Springer, Heidelberg.

[60] Zhu L., Gorton I., Liu Y., and Bui N. B., 2006. Model driven benchmark generation for web services. In *SOSE'06: Proceedings of the 2006 International Workshop on Service-Oriented Software Engineering.* ACM Press, New York, NY. ISBN 1-59593-398-0.

Author Index

Numbers in *italics* indicate the pages on which complete references are given.

Subject Index

Contents of Volumes in This Series