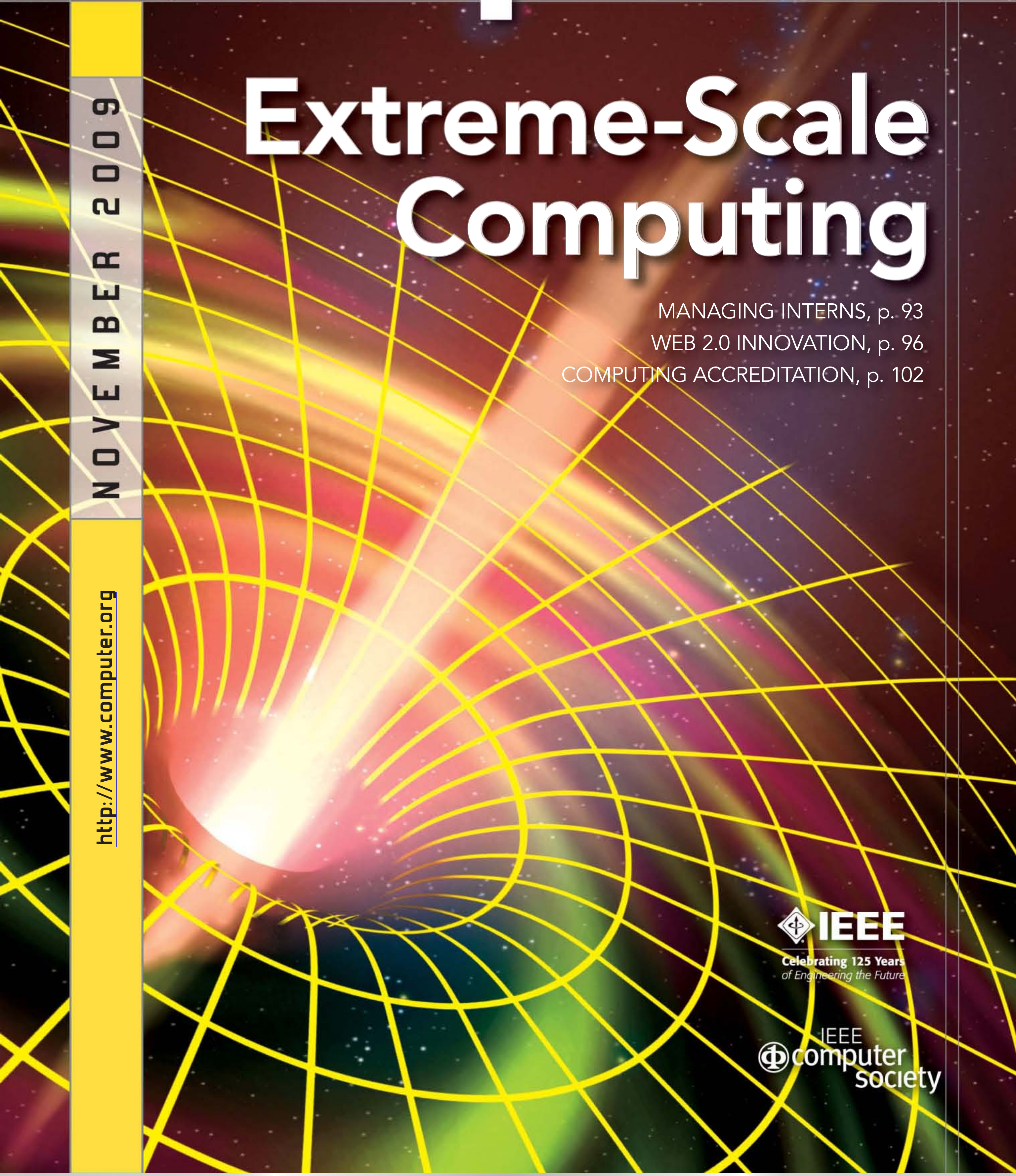
Innovative Technology for Computer Professionals

Computer

NOVEMBER 2009

http://www.computer.org



Extreme-Scale Computing

IEEE
Celebrating 125 Years
of Engineering the Future

IEEE computer society

IEEE
IEEE computer society

Computer
Innovative Technology for Computer Professionals

Editor in Chief
Carl K. Chang
Iowa State University
chang@cs.iastate.edu

Associate Editor in Chief
Sumi Helal
University of Florida
helal@cise.ufl.edu

Associate Editor in Chief, Research Features
Kathleen Swigger
University of North Texas
kathy@cs.unt.edu

Associate Editor in Chief, Special Issues
Bill N. Schilit
Google
schilit@computer.org

Computing Practices
Rohit Kapur
rohit.kapur@synopsys.com

Perspectives
Bob Colwell
bob.colwell@comcast.net

Web Editor
Ron Vetter
vetterr@uncw.edu

2009 IEEE Computer Society President
Susan K. (Kathy) Land, CSDP
president@computer.org

Area Editors

Computer Architectures
Steven K. Reinhardt
AMD
Databases and Information Retrieval
Erich Neuhold
University of Vienna
Distributed Systems
Jean Bacon
University of Cambridge
Graphics and Multimedia
Oliver Bimber
Bauhaus University Weimar Johannes Kepler University Linz
High-Performance Computing
Vladimir Getov
University of Westminster
Information and Data Management
Naren Ramakrishnan
Virginia Tech
Multimedia
Savitha Srinivasan
IBM Almaden Research Center
Networking
Sumi Helal
University of Florida
Software
Dan Cooke
Texas Tech University
Robert B. France
Colorado State University

Column Editors

Education
Ann E.K. Sobel
Miami University
Embedded Computing
Tom Conte
Georgia Tech
Green IT
Kirk W. Cameron
Virginia Tech
IT Systems Perspectives
Richard G. Mathieu
James Madison University
Invisible Computing
Bill N. Schilit
Google
The Known World
David A. Grier
George Washington University
The Profession
Neville Holmes
University of Tasmania
Security
Jeffrey M. Voas
SAIC
Software Technologies
Mike Hinchey
Lero—the Irish Software Engineering Research Centre
Web Technologies
Simon S.Y. Shim
San Jose State University

Advisory Panel
Thomas Cain
University of Pittsburgh
Doris L. Carver
Louisiana State University
Ralph Cavin
Semiconductor Research Corp.
Dan Cooke
Texas Tech University
Ron Hoelzeman
University of Pittsburgh
Naren Ramakrishnan
Virginia Tech
Ron Vetter
University of North Carolina at Wilmington
Alf Weaver
University of Virginia

CS Publications Board
Sorel Reisman (chair), Alain April, Angela R. Burgess, Frank E. Ferrante, David A. Grier, Audrey Kremer, Phillip A. Laplante, Paolo Montuschi, Jon Rokne, R. Sampath, Steve Seidman, Linda I. Shafer, Roy Sterritt, Steven L. Tanimoto

CS Magazine Operations Committee
David A. Grier (chair), David Albonesi, Isabel Beichl, Arnold (Jay) Bragg, Carl Chang, Kwang-Ting (Tim) Cheng, Fred Douglis, Hakan Erdogmus, Carl Landwehr, Dejan Milojicic, Sethuraman (Panch) Panchanathan, Crystal R. Shif, Maureen Stone, Fei-Yue Wang, Roy Want, Jeff Yost

Editorial Staff
Scott Hamilton
Senior Acquisitions Editor
shamilton@computer.org
Judith Prow
Managing Editor
jprow@computer.org
Chris Nelson
Senior Editor
James Sanders
Senior Editor

Contributing Editors
Lee Garber
Bob Ward

Design and Production
Larry Bauer
Design
Olga D'Astoli
Cover Design
Jennie Zhu

Administrative Staff
Products and Services Director
Evan Butterfield
Senior Editorial Services Manager
Crystal R. Shif
Magazine Editorial Manager
Jennifer Stout

Senior Business Development Manager
Sandy Brown
Senior Advertising Coordinator
Marian Anderson

Circulation: *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society. **IEEE Headquarters**, Three Park Avenue, 17th Floor, New York, NY 10016-5997; **IEEE Computer Society Publications Office**, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; voice +1 714 821 8380; fax +1 714 821 4010; **IEEE Computer Society Headquarters**, 2001 L Street NW, Suite 700, Washington, DC 20036. IEEE Computer Society membership includes $19 for a subscription to *Computer* magazine. Nonmember subscription rate available upon request. Single-copy prices: members $20.00; nonmembers $99.00.

Postmaster: Send undelivered copies and address changes to *Computer*, IEEE Membership Processing Dept., 445 Hoes Lane, Piscataway, NJ 08855. Periodicals Postage Paid at New York, New York, and at additional mailing offices. Canadian GST #125634188. Canada Post Corporation (Canadian distribution) publications mail agreement number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8 Canada. Printed in USA.

Innovative Technology for Computer Professionals

Computer

http://computer.org/computer

CONTENTS

ABOUT THIS ISSUE

Extreme-scale computing relates directly to the hardware, software, and applications enabling simulations in the petascale performance range and beyond. As the articles selected for inclusion in this special issue demonstrate, in addition to enabling science through simulations at unprecedented size and fidelity, extreme-scale computing serves as an incubator of scientific and technological ideas for the computing area. An additional article explores the advantages of a commitment-based service-oriented architecture.

For more information on computing topics, visit the Computer Society Digital Library at www.computer.org/csdl.

IEEE Computer Society: http://computer.org
Computer: http://computer.org/computer
computer@computer.org
IEEE Computer Society Publications Office: +1 714 821 8380
Cover image © Andreus | Dreamstime.com

Flagship Publication of the IEEE Computer Society

November 2009, Volume 42, Number 11

IEEE
Celebrating 125 Years
of Engineering the Future

IEEE computer society

PRINTED WITH SOY INK

BPA WORLDWIDE

Computer Highlights Society Magazines

The IEEE Computer Society offers a lineup of 13 peer-reviewed technical magazines that cover cutting-edge topics in computing including scientific applications, design and test, security, Internet computing, machine intelligence, digital graphics, and computer history. Select articles from recent issues of Computer Society magazines are highlighted below.

IEEE Software

More than 40 years ago, the term "software engineering" was coined as a challenge to establish software design and development on a firm engineering footing. Twenty years ago, Mary Shaw's classic article "Prospects for an Engineering Discipline of Software" assessed progress toward the establishment of software design and development on a firm engineering footing. Shaw's latest update, in the most recent *Software*, shows that the profession has made progress but still has much left to do.

IT Professional
TECHNOLOGY SOLUTIONS FOR THE ENTERPRISE

Ontologies represent items of knowledge—ideas, facts, things—in a way that defines the relationships and classifications of concepts within a specified domain of knowledge. It's this ability to define various useful relationships among items of knowledge, and to implement these relationships in software, that make an ontology such a powerful gadget in the knowledge manager's tool box. A new tutorial in *IT Pro*, "Just What Is an Ontology, Anyway?," by Thomas C. Jepsen, addresses several definitions of "ontology" as they relate to computer applications. Jepsen also gives an overview of common ontology-based applications.

IEEE Computer Graphics and Applications

Topics covered in *CG&A*'s special issue on recent developments in 3D user interface research include reality- and imagination-based interaction, pointing techniques, analysis of rapid aimed movements, temporal-data visualizations, and navigation of augmented CAD models.

IEEE Security & Privacy
Building Dependability, Reliability, and Trust

Virtually every Internet application relies on the Domain Name System, but security wasn't a major goal of its original design. The result is several critical vulnerabilities, reviewed in the September/October 2009 *S&P* special issue on DNS security. To address the security challenges, the security community developed DNS Security Extensions, which are undergoing deployment. Articles summarize key aspects of how to deploy DNSSEC at authoritative servers, resolvers, and public key learning.

IEEE Intelligent Systems

E-government and e-participation research aims to provide technologies and tools for more efficient public-administration systems and more participatory decision processes. To this end, interest is growing in how this challenging domain can benefit from emerging "intelligent" technologies, tools, and applications such as the Semantic Web, service-oriented architectures, Web 2.0, and social computing.

In "Transforming E-government and E-participation through IT," *IS* contributors Vassilios Peristeras, Gregoris Mentzas, Konstantinos A. Tarabanis, and Andreas Abecker note that governments invest heavily in information and communication technologies but are still far from satisfying their constituents.

IEEE Internet Computing

Cloud computing is location agnostic and provides dynamically scalable and virtualized resources as services over the Internet. In a recent special issue of *IC*, the guest editors provide broad introductory definitions of cloud computing concepts and introduce other articles that investigate some of the most fundamental issues concerning cloud services' development and deployment.

IEEE micro

Photonic networks-on-chip have distinct advantages over electronic NoCs, including communication bandwidth approaching multiple terabits per second with limited power dissipation. In the July/August issue of *Micro*, Columbia University researchers explore the design of photonic NoCs for delivering a scalable solution to future multicore processors performance requirements in "Photonics NoCs: System-Level Design Exploration."

IEEE MultiMedia

Steganography is the art and science of writing messages in a way that hides the existence of communication. It can be combined with cryptography to achieve a high level of security. Steganographic schemes abound for hiding messages in images with low dynamic ranges. However, these schemes operate in a fixed luminance range that doesn't work with images of high dynamic ranges, where each image has a different luminance range. An article in the July-September issue of *MultiMedia*, "A Novel Approach to Steganography in High-Dynamic-Range Images," presents a message-hiding approach for HDR images. The approach supports authentication and a large embedding capacity with low image distortion.

IEEE Design&Test of Computers

The September/October *D&T* special issue on 3D IC integration is guest-edited by David Kung of IBM T.J. Watson Research Center and Yuan Xie of Pennsylvania State University. Four articles address different challenges to giving chip architects the flexibility and design options of 3D IC technology as they pursue solutions to the complexities and cost of scaling to 22 nm and beyond.

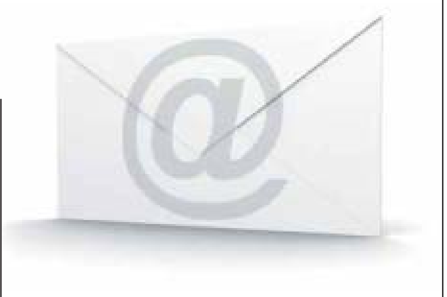
Nur Touba of the University of Texas at Austin, technical program chair for the 2008 International Test Conference, invited the authors of three outstanding conference papers to update their work for *D&T*.

IEEE Annals of the History of Computing

In *Annals*' July–September Anecdotes department, Stanley Mazor recollects his work as liaison to Magnavox in developing the Intel 8244 custom chip for Magnavox's Odyssey2 videogame console. Mazor teamed with Intel chip designer Peter Salmon to deliver the 8244 on time to meet Magnavox's announced plan to release the console by the 1977 holiday season. Intel met its schedule, although the Odyssey2 system didn't appear until 1979.

Mazor joined Intel in 1969. He worked with Ted Hoff and Federico Faggin to deliver the first working CPU, the Intel 4004, in 1971.

Editor: Bob Ward, *Computer*; bnward@computer.org



FORMAL VERSUS AGILE: AN OXYMORON

The article titled "Formal Versus Agile: Survival of the Fittest?" (S. Black et al., Sept. 2009, pp. 37-45) is at best an oxymoron.

Agile methods and formal methods have little in common. Or to put it simply: One size fits nobody, and agile fits very few.

Agile is good for very small projects and for throwaway code supporting research/experimentation. Formal methods should be used for any project that is nontrivial. The potential for useful overlap is miniscule.

The only place that agile fits, with a formal method, is at the front end of the *systems architecture* definition, where alternatives need to be explored. No amount of interchange will make people with *real* problems use agile methods except to support research about alternatives. And these days there is little that cannot be designed from real requirements without research that agile could actually be a help with. The exception may be to do drivers and test cases to evaluate hardware capabilities.

Nobody, at least nobody who is sane and rational, is going to give software folks a bunch of money just to see "what they can come up with." That is not to say that there are not PHBs who can get snookered by technobabble, but with the current economy, their numbers are being thinned down.

In the real world, real (functional) requirements are *required*, not optional. They are fixed; they do not change rapidly, if at all. They describe the essence of a solution to a problem. The best alternative solution will be further specified by derived requirements, which are subject to occasional change and constrained by nonfunctional requirements such as size, weight, power, security, reliability, and so on.

It is unclear what problems change so fast that agile would be useful. "Requirements" that change are a sign that the real problem has not been defined and that a systems architecture has been bypassed to start design and development without knowing the framework that the "solution" should fit in.

Management has to work to a budget. They need products that are guaranteed to work and solve their problem. And they need them on time. Agile just does not do that in any but the simplest cases.

William Adams
williamadams@ieee.org

The authors respond:

We thank William Adams for his comments. Unfortunately, his viewpoint seems to reflect the thinking of decades ago and is no longer appropriate. These days, agile is used very successfully on many large-scale projects and certainly not only for "very small projects and for throwaway code." For example, at the recent Agile 2008 Conference, Marcus Evans reported on the use of agile within the British Broadcasting Corporation (BBC) to develop the BBC iPlayer. The BBC iPlayer project has been described as a "project of the same importance as moving from B&W to color TV."

Another UK giant, British Telecommunications (BT), has used agile with great success. Indeed, according to Roger Leaton, agile advocate at BT, "Large-scale agile enablement, when done properly, really does work and transforms IT project delivery and business performance" ("How BT Learnt to Be Agile," 17 May 2008; www.computerworlduk.com/toolbox/software-quality-testing/quality-assurance/opinion/index.cfm?articleid=1416).

Regarding the comment that "Formal methods should be used for any project that is nontrivial," we are proponents of formal methods, but would never agree with this statement. Formal methods have their place, as do agile and other development paradigms in the modern world of software engineering.

In our article, we set out to get colleagues thinking about the big picture regarding practical software engineering in the 21st century. We wanted to challenge stereotypical, old-fashioned thinking. We are happy that we have achieved that aim.

SAAS LIMITATIONS

In "The Web as the Ubiquitous Computer" (Web Technologies, V.S. Pendyala and S.S.Y. Shim, Sept. 2009, pp. 90-92), the authors provide information about software as a service (SaaS). Technological trends are clearly going in the direction of having applications online. Zero maintenance results in a lower total cost of ownership; hosted offerings are designed and finely tuned to scale seamlessly for large numbers of simultaneous users; and upgrades are made frequently and, for the customer, effortlessly.

However, there are several significant limitations that users must bear in mind before they select a hosted application. The sad truth is that computers can fail, resulting in service disruption, which has the potential to be catastrophic. Users are not only subject to network outages, they also must always be tethered to the Internet.

Sensitive data also may be vulnerable when hosted on someone else's server. In this case, users can't modify the software and they can't upgrade it. They are forced to accept changes that they might not even be aware of.

They lose control over what version of the software they are using. Although keeping up to date is one of the selling points for SaaS, in fact, too many new software versions may turn out to be problematic.

With SaaS, users keep paying. They can't own a license and use it freely. Whenever the SaaS provider changes its rates, users must pay the increased fee or risk losing access to their information.

The cost savings with SaaS is a huge benefit. However, it is important to realize that many of these cost savings could just be upfront costs associated with SaaS implementation. In the long run, it could be more expensive to maintain SaaS due to subscription costs.

If software systems are absolutely critical to a company's operation, a much better choice is to invest in a more reliable and comprehensive solution to support making guarantees to customers.

Hong-Lok Li
lihl@ams.ubc.ca

The authors respond:

Points of failure cannot be avoided in any setup, including local environments. As described in the column, cloud environments tend to have better reliability and resilience intrinsic to large corporate setups. As we also stated, the Internet is now a household commodity and abundantly available. This is in fact a fundamental premise for the paradigm shift.

Vulnerability and losing control of data were covered in the column. It is not clear how that point applies to versioning, however. Irrespective of the environment, developers should always be in control of versioning their software. If the comment refers to versioning of the software on the leased servers, that is the case even with local servers. The users are bound by what the server vendor provides, whether deployed on a cloud or in the local environment. Vendors can discontinue support to older versions or include features in the new versions that the user may not like. In view of the economies of scale that the cloud provides, there is a greater likelihood of accessing better features.

This point about potential problems with the fee structure is covered in the "Challenges" section of the column. As mentioned there, users "could feel trapped and helpless when providers change their terms of service or operational methods after some time." Gains from economies of scale inherent in the cloud environment will help keep the costs down.

The new model is more likely to provide a reliable and comprehensive solution and make guarantees to customers than a homegrown environment. The investment and interest in this new model of computation itself is ample proof of this.

DEFINING COMPUTER ENGINEERING

Regarding "Defining Engineering" (Letters, Sept. 2009, p. 6-7): First, I contend that computer software development and programming has very little to do with mathematics. Computer software is a language solution to an automated world. A handyman with good communication skills will likely be as good at designing and developing software as an engineer who has memorized all the rules inside the box.

Many successful software developers have only modest ability in math—myself included—and so couldn't take a computer science degree in the average university because of the odious math prerequisites. So one of the most creative fields available is blocking many creative people because of the misperception that it requires math.

Second, when programmers are conceived of as technicians the software architecture and design become brain-numbing bureaucratic functions; just produce the ambiguous diagrams and documents, argue about the document structure, get it all signed, and you're done. Oh, and ship it offshore for programming—it's already late.

Maybe the reason engineering can't come to grips with programming is that programming requires craft and abstraction as well as science, and that knowing all the minutiae of the theory may have little to do with effective practice. Too often, an "engineered" software application has many impressive features, but is maddeningly "unintuitive" to use. Doesn't engineering do "intuitive"? Even an airliner's complex instrument panel is designed for optimal usability by the user (the pilot). This doesn't just relate to screens and buttons—most levels of computer systems need to be understandable. Humans need to easily work on them, upgrade them, and interface with them without making errors of misunderstanding that cost time, money, and safety.

My education was in music and language, I have a BA, and I don't do math—but I have been a successful programmer (and now software architect) since 1972. My own programming experience includes data communications, finance, compiler development, psychomotor testing, animation, graphics, the Web, databases, diagnostic systems, user interfaces, and much else. I have software patents granted and pending, and I authored a successful college textbook on computer programming.

Harry Gilbert
harrymgilbert@yahoo.com

We welcome your letters. Send them to letters@computer.org. Letters are subject to editing for style, clarity, and length.

Bad Alignment

David Alan Grier, *George Washington University*

Potential customers will be interested in new technology only if it somehow makes their lives better—if it moves them toward a goal they hold for themselves, their family, their company.

It wasn't working. Nothing was in alignment. The coffee was cold, the muffins were stale, and the software presentation was a forced march through a PowerPoint landscape denuded of life. Three hours had passed, and no civilization could be seen on the horizon. In command of the podium was a perky salesperson named Alena who led us through slide after slide with an infection to her voice that suggested ideas of great magnitude where no new ideas could be heard. The eight of us on the committee sat in stone cold silence, praying that an all-merciful divine being would send a band of pirates to attack our meeting, seize this woman, and hold her hostage in a distant country.

We held no personal grudge against Alena. We could find no fault in her person or character. However, we could all see that the poor dear wasn't succeeding in the task assigned to her. As the representative of her employer, she was supposed to educate our committee and harmonize our aspirations with her company's goals, to show us that they would meet our needs when we met theirs. Such a task isn't easy, as it demands analysis, imagination, and more than a little empathy. These qualities aren't usually associated with pirates, but we were willing to take our chances. An open and unilateral statement of criminal assessment would have been far preferable to the presentation we were attending.

The problem of communicating the beneflts of technology has bedeviled the industrial age from its inception. So often, new technology is conceived in terms of engineering specifications: processor speeds, data formats, power consumption, manufacturing costs. Rarely do potential customers see technology with the same eyes. They will be interested in new devices only if it somehow makes their lives better—if it moves them toward a goal they hold for themselves, their family, their company.

PATTERSON'S SALES CYCLE

As I watched the sales presentation, I could see that Alena had clearly been schooled in John Henry Patterson's sales cycle, though she probably knew little about the man.

Patterson was the president of the National Cash Register Company at the end of the 19th century. He was one of the flrst thinkers to understand the problem of explaining new technologies to nontechnical audiences. His novel technology, the cash register, was a marvel of mechanical engineering and attracted as much attention as the personal computer of a century later. Yet initially, that clever engineering—the high-precision gears, the multiple linkages, the novel bearing designs—wasn't sufflcient to attract more than a few paying customers. When new customers were flrst introduced to the machine, with its engraved case and polished brass, they would usually react with the question, "What's in it for me?"

Patterson developed a four-step cycle for connecting the technology of cash registers with his customers' goals and drilled this cycle into his stales staff. The steps were easy to remember: identify and propose, demonstrate and close. They were much harder to follow.

The first step in the process required the salespeople to listen to their customers and discover their human needs. Did they feel that they were in control of their business? Were they earning enough to support their family? Were they spending too much time at their work? After identifying such issues, Patterson's salespeople were to propose ways that cash registers could be used to solve problems and then demonstrate those solutions. For example, cash registers could reduce employee theft, simplify accounting, and track expenditures, which in turn would reduce anxiety, simplify labor, strengthen confidence.

The last step of the process, closure of the sale, brought the customer's goals in line with what National Cash Register had to offer. That was the most important part of the process for Patterson. "ABC," he told the National Cash Register representatives: "Always Be Closing."

MISSING THE POINT

Alena clearly understood the need to close a sale, but she didn't know how to work the other steps of the Patterson sales cycle. She started her presentation by saying that she wanted to listen to us and identify our needs, but she quickly proved that she was much more interested in identifying herself. In the course of 15 minutes, we learned about her academic credentials, the number of years she had worked for the company, the types of schools that had bought software from her, and, if memory serves correctly, the name of the Thai dressmaker who had created the suit she was wearing.

In telling her story, Alena was little different from the other salespeople who had appeared before us. As we listened to them, we had been regaled with corporate histories, stories of programmers who had developed their software, the revenue goals that had been met or exceeded by their regional office. Such stories were impressive at times but showed little analysis and less empathy.

Without identifying our needs, Alena and all the other salespeople had little to propose and less to demonstrate. They wandered through the remaining steps of the Patterson cycle like merchant ships lost on the stormy sea. They would tack toward one idea for a time before shifting to another in the hope that something might catch our attention. Eventually, they became lost and panicked. The first sign of trouble came when they started repeating the phrase, "The software will do whatever you want it to do."

The real evidence was seen in the acceleration of the PowerPoint slides. At the start of the presentation, they lingered for 30 or 40 seconds per slide. After the first hour, this dwell time dropped to 15 seconds or 20 at most. As we approached the third hour, slides were flashing on the screen for no more than 5 or 6 seconds.

The Shewhart cycle is the familiar four-step process that shaped large parts of engineering practice: Plan. Do. Check. Analyze. Repeat.

These slides were detailed, technical graphics. Nested matrices. Symantec networks. Predicate calculus. Horn clauses. Each of these ideas flew past our eyes accompanied by the plaintive cry, "whatever you want."

THE SHEWHART CYCLE

If Alena had asked me what I wanted to do, my response would have been unequivocal. I wanted to finish my service on the committee as quickly as possible. While I could see an obvious strategy that would have aligned that goal with Alena's hopes, I had to acknowledge that the result wouldn't have been entirely satisfactory to all concerned. Our committee actually had a more noble goal: It had been assembled for the purpose of improving the university's approach to assessment.

In the world of academe, "assessment" is the term for the more prosaic concept of "quality control." Although it's based on ideas that are considerably older, it has been part of higher education for a little more than a decade. It uses educational analyses that were developed in the 1950s and employs a basic tool of quality control, the Shewhart cycle of continuous improvement, that dates to 1931.

The Shewhart cycle is the familiar four-step process that shaped large parts of engineering practice: Plan. Do. Check. Analyze. Repeat. "We like to believe that there is law and order in the world," Shewhart wrote. "We seek causal explanations of phenomena so that we may predict the nature of these same phenomena at any future time." He knew that sometimes natural phenomena could hide its operations, but he also had great faith in the analytic powers of the human mind. To support this faith, he liked to quote the poet of the English Enlightenment, Alexander Pope:

> All Nature is but Art, unknown to thee;
> All chance, direction, which thou canst not see;
> All discord, harmony not understood;
> All partial evil, universal good.

Yet, Shewhart didn't accept the mechanistic view of the universal good that was shared by Pope's 18th-century contemporaries. He looked for universal good and hoped to find misapprehended harmonies, but knew that analytic reason couldn't always find them. We "are limited in doing what we want to do," Shewhart explained, because understanding every aspect of even a simple manufacturing task in its entirety "requires almost infinite knowledge." Therefore, quality control required the good engineer to "accept as axiomatic that we cannot do what we want to do and cannot hope to understand why we cannot."

AT&T STRATEGY

Shewhart developed his cycle in conjunction with a statistical methodology that allowed engineers to search for problems in production and find ways of bringing those problems under control, at least for a short time. This method was well suited for Shewhart's employer, the American Telephone and Telegraph Company (AT&T), because it was a highly complicated business that followed a simple, well-articulated goal: universal service.

When AT&T began expanding in the early 20th century, its leadership concluded that both its customers and investors would be best served by a company that offered a standard telephone service across the US and could dominate every market in which it operated. This goal required a uniformity beyond the scope of the accomplishments of any company operating at that time. Yet, management argued that its goals required "standardized operating methods, plant facilities and equipment," explained an early company president, and also "complete harmony and cooperation of operating forces through centralized and common control."

AT&T's common goal simplified the human dynamics of quality control, making it easier for a team of engineers to empathize with the needs of their neighbor. As would any team of engineers, marching through the steps of plan, do, check, and analyze, they would need to understand how other units might view their proposal, which in turn required them to imagine how these units approached the goal of universal service. Of course, such a goal didn't eliminate differences within the corporation, as all employees would interpret universal service in light of their own opinions and aspirations. However, it gave a much stronger foundation for resolving disagreements than the goals found at most universities.

EVALUATING ENGINEERING EDUCATION

In words that have often been expanded and embellished, a university president once described the modern institution of higher education as a collection of rival, warring tribal factions united by a common heating system. Lacking a common heating system, our school was united by complaints about the parking lots. As most of the members of the assessment committee took public transportation, we were barely united at all.

In spite of our discontent, some of our committee members were comfortable with the academic ideas of quality control and even with the presentations by the software sales teams. In particular, the engineers saw how they might employ these pieces of software as they had the longest history with educational assessment. Engineering faculty began developing the concepts of educational quality control at roughly the same time that Shewhart developed his methods for AT&T.

The committee established the idea that education could be studied with the tools of engineering and that education needed to be treated as a process that requires continuous improvement.

In 1929, the Carnegie Foundation for Teaching established a committee to investigate the state of engineering education. To chair the committee, they recruited William Wickenden, who had been a senior manager at AT&T and was then president of the Case School of Applied Science in Cleveland. Wickenden created an extraordinarily detailed plan for his committee. A preliminary plan shows subdivisions of authority, lines of communication, data flows, and responsibility for outcomes. He had grand designs for the committee, "a comprehensive survey of the whole situation—students and graduates, faculty and facilities, curricula and methods, professional engineers and industry, and the economic and social significance of engineering."

In the end, Wickenden's committee produced a sympathetic report. Well before they completed their work, the members concluded that "there were no glaring defects in the contemporaneous policies and methods of engineering education." At the same time, the committee established the idea that education could be studied with the tools of engineering and that education needed to be treated as a process that requires continuous improvement. "There were many readjustments which were needed," the committee concluded. "The situation called not for revolution but evolution." It was a call to apply the ideas of Walter Shewhart: Plan. Do. Check. Analyze. Repeat.

DIFFERING VIEWPOINTS

I don't know if Alena and her team did a formal assessment of their presentation to our committee. If they did, the feedback for their Shewhart cycle would have been swift, brutal, and obvious. We didn't buy their software. If they looked more deeply at our response, they might have discovered that the members of the committee responded in different ways to their presentation. The variations in these responses were determined not only by the different goals of each school, but also by the social structure in which each school operates.

The engineering school, by far the most sympathetic to Alena and her colleagues, is part of a complicated but unified social structure. This structure contains accrediting bodies, engineering societies, and professional exam boards. These groups are used to debating the nature of education and finding common ground among themselves.

By contrast, the College of Policy and Current Events abides in an anarchistic landscape and has little experience in working with other institutions to define the goals of higher education. The combative nature of the group is suggested by the titles of its courses: Civil Wars, Terrorism, Military Strategy. It even teaches classes in pirate theory, though we tend to call such courses "Transnational Security Threats" and limit the examples to the modern pirates of Sudan and the Strait of Malacca.

According to the best scholars of the fleld, pirates are most effective when they can control their operational goals, when they can build a strong bond of empathy among their band. To do this, they usually need the tacit approval of a nation-state, freedom from absent owners (usually achieved by stealing a boat or buying one with stolen funds), and assembling a team that accepts the twin goals of expanding plunder and avoiding capture. The situation doesn't quite parallel the AT&T of the 1930s with its goal of universal service, but it works tolerably well. In this circumstance, they have a simple debate over Shewhart's cycle. Will a new strategy increase the chances of gaining treasure? Will it make them more vulnerable to naval attack? They usually demand a full discussion of all hopes and doubts in such debates.

No pirate wants to worry about the concerns of others in the midst of a raid. Perhaps because of this, they are never available when you need them to disrupt a sales presentation that has gone badly out of alignment. C

David Alan Grier*, the author of* Too Soon to Tell *(IEEE CS Press, 2009), is an associate professor at the George Washington University, where he teaches science and technology policy and leaves the theory of piracy to others. Contact him at grier@gwu.edu.*

32 & 16 YEARS AGO

NOVEMBER 1977

COMPUTER NETWORKS (p. 11) "Networks of computers are making it possible to achieve computer-to-computer and terminal-to-computer communications that only a few years ago would have been impossible. Many of these networks are operational already, with more powerful ones seemingly in store for the future. This remarkable growth has opened new opportunities for designers, users, and managers—but it has posed some difficult problems for them as well. Knowledge of such network issues as topological design alternatives, common carrier communications services, value-added networks, hardware and software networking technology, cost factors, regulatory issues, measurement techniques, and network administration are of paramount importance."

NETWORK COMPLEXITY (p. 12) "The complexity of computer networks has taken a dramatic upswing, following significant developments in electronic technology such as medium- and large-scale integrated circuits and microprocessors. Along with this upswing in complexity, several sophisticated network classification schemes have evolved. Abstruse terminology—such as centralized, circuit switching, deterministic, distributed, packet switching, and stochastic—frequently appears in discussions and papers, to an extent that often confuses communication subnet users. For example, how many readers would immediately recognize that centralized networks inherently have a deterministic routing policy? or know whether or not distributed networks are necessarily packet switching networks, or may instead be message switching?"

SERVICE MEASUREMENT (p. 32) "Suitable measures for the comparison of computer services have been discussed in detail by Abrams and Treu, who identified more than 50 possible measures of the computer service delivered through a remote terminal, on the basis of time, lengths, rates, and ratios. Further, sophisticated hardware and software tools in the form of mechanized measurement drivers have been developed to collect performance measurements from the systems under study. The comparison methodology, however, is currently lacking an appropriate experimental design which will provide, with a specified level of confidence, the answer to the question of real interest: Which system is the best?"

NETWORK AVAILABILITY (p. 43) "The principal opportunity for increasing the availability of a network of computers results from the presence of multiple, reasonably autonomous processors. First, when a software failure does occur which disables a node of the network, and the malfunction can be confined to that node, the network may continue to provide at least some level of service to some of its users.

The network may even continue operation so as to make the failure transparent to nearly all users. On the other hand, if all service were being provided by a single central computer, an unrecoverable failure in the operating system running on the one CPU would interrupt service to all users until the system could be restarted."

ROUTING (p. 60) "There are basically two different types of system control procedures. An *open-loop* control system is one in which system control actions are taken in a prespecified manner that is independent of both current system state and past response. Open-loop control laws are usually simple in form but are practically useful only in those systems where the evolution of system response can be accurately predicted from given data. A *closed-loop* control system is one in which control action depends on current system state and possibly also on dynamic response history. Closed-loop systems are usually characterized by a relative insensitivity to parameter variations, and they are flexible enough to control system response successfully even in the presence of unpredictable system inputs. The design of closed-loop systems is frequently difficult because of the related problems of stability, accuracy, and speed of response."

PRODUCTIVITY AND COMPUTERS (p. 66) "The computer is a vital factor in productivity improvement. Economic research by Edward Dennison and others indicates that almost half of the US increase in productivity is attributable to technological innovation. Since computers represent one of the most important technological advances of the 20th century, as well as being a factor in the other major sources of productivity gains, we can conservatively estimate that computer usage has provided at least 15% of the 2% growth in productivity for the last decade, or 0.3 percentage point."

JOBS AND PEOPLE (p. 87) "On-the-spot matching of jobs to people as well as people to jobs, using real-time computer techniques, is now an everyday routine at seven job centers run by the Employment Service Agency in London. The computer system, which the ESA claims leads the rest of the world and is arousing interest in the U.S. as well as Europe, began live operation at the Romford job center in March."

NOVEMBER 1993

PARALLELISM (p. 20) "Today's workstations have redefined the way the computing community distributes processing resources, and tomorrow's machines will continue this trend with higher bandwidth networks and higher computational performance. One way to obtain higher computational performance is to use special parallel coprocessors to perform functions such as motion and color support of high-definition screens. Future computationally intensive applications suited for desktop computing machines include real-time text, speech, and image processing. These applications require massive parallelism."

SYSTEMS ENGINEERING (p. 54) "Developing large computer-based systems with complex dynamics and component interdependencies requires analysis of critical end-to-end processing flows to determine feasibility and proper allocation. Currently, no engineering discipline provides the knowledge base for the necessary trade-off studies concerning software, hardware, and communication components; a new discipline is needed at the systems engineering level.

"Industry has recognized the size and scope of problems with systems engineering processes, and several organizations have been formed to advance the systems engineering discipline. However, recognition of the need for a special discipline addressing the system engineering of computer-based systems (ECBS) is just emerging, as evidenced by the recently formed IEEE Computer Society Task Force on ECBS and recently published textbooks."

JOHN BACKUS (p. 78) "John Backus has been named recipient of the 1993 Charles Stark Draper Prize, the highest honor of the National Academy of Engineering (NAE). The award, which carries a $375,000 stipend and a gold medal, is the largest prize in engineering, according to the NAE.

"Backus was cited for the development of Fortran—the first general-purpose, high-level computer language—which ushered in the computer software revolution and a $23 billion industry. 'Before John Backus, only a handful of specialists could use the computer,' said Robert M. White, NAE president. 'Today, everyone from preschoolers to postgraduates can use the computer.'"

SPEECH TRANSLATION (p. 78) "A computer scientist specializing in artificial intelligence is bypassing traditional AI methods in an effort to improve the performance of speech-to-speech translation. So far he has achieved a 75 percent accuracy rate when testing his program on 1,600 sentences."

"[Hiroaki] Kitano, once a professional simultaneous interpreter, envisions a world in which hand-held computers will enable native and foreign speakers to converse anywhere."

DIGITAL LIBRARIES (p. 79) "The hypertext industry predicted by [Vannevar] Bush in 1945 emerged slowly, however. It was only in the late 1980s, years after Ted Nelson's vision of Xanadu, Brown University's IRIS (Institute for Research in Information and Scholarship) Project, and many other development efforts (Apple's HyperCard, for example) that this field came into international prominence. Yet hypertext still awaits the broader commercialization that will result from further R&D and technology transfer."

THE POSIX STANDARD (p. 81) "Through its efforts to develop standard interfaces for portable operating systems and publication of the Posix standard, the IEEE Computer Society has provided a dynamic environment for observing the changing role of testing in an open-systems environment. Clearly, the ability to achieve goals such as portability and interoperability is directly related to the ability to enforce adherence to a standard.

"Less obvious is the impact of developing standards and test methods concurrently. The generation of test methods during standard development promotes higher quality standards by providing immediate feedback to the standards development process. As the development process matures, lessons learned through experiences such as Posix will likely demonstrate that both these roles of enforcing adherence to the standard and improving its quality are crucial to its success."

APPLE'S NEWTON (p. 104) "With [John] Sculley's ouster—Apple didn't call it that mind you, but Silicon Valley insiders insist that he was squeezed out—Apple not only appears to be taking the conservative route vis-a-vis their once premiere visionary, but also the man who was responsible for the next generation of data management: the Newton. This palm-sized computer/fax machine/cellular telephone has the potential to virtually upend not only computing, but also the entire industry of personal communications. Unfortunately, without Sculley, the Newton loses its only advocate possessing the power and influence to ensure adequate allocation of further research and development funding. Without that funding, the Newton stands a poor chance of ever reaching its true potential."

PDFs of the articles and departments from the November 1977 and 1993 issues of Computer *are available through the IEEE Computer Society's website: www.computer.org/computer.*

Editor: Neville Holmes; neville.holmes@utas.edu.au

There's no such thing as science fiction.
Just stuff we haven't made yet.

From 20,000 feet under the sea to the dark side of the moon, and everywhere in between, you'll find the work of IEEE members. It's in a thousand things our founders could only dream of in 1884.

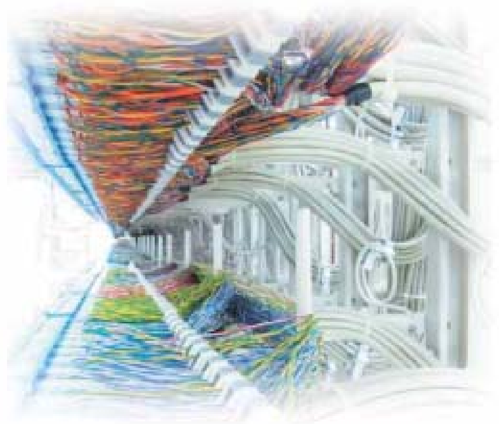
In fact, IEEE members have taken the stuff of science fiction and made it part of every day life. That's why, whether you need to draw on the knowledge of technology's past pioneers or today's innovators, one fact is clear—you'll see that IEEE members aren't just waiting for the future, they're engineering it—and the best chapters are yet to come.

Celebrating 125 Years of Engineering the Future

www.ieee125.org

IEEE
Celebrating 125 Years
of Engineering the Future

Anonymization Technology Takes a High Profile



Neal Leavitt

Increasingly, governments and various types of organizations are trying to either block or track Internet access and online communications by dissidents, employees, or others. To sidestep these activities, users are turning to anonymization technology.

Censorship and the increased tracking of users online have become important topics. Numerous governments censor computer- and network-based communications to keep their citizens from freely getting news from or transmitting information to the outside world.

Dissidents and everyday Internet users—as well as criminals and others who want their online identities to be secret—have turned to anonymization technology to keep from being identified. This has made the technology more important and widely used in recent years, sparking the start up of anonymization companies and the development of new techniques.

"The evolving threats, the introduction of new technologies and applications, and the emergence of Internet censorship are really driving [the approach] right now," said Lance Cottrell, the founder and chief scientist of anonymization vendor Anonymizer. "And events like the recent elections in Iran have really drawn attention to it." In Iran, protesters against the results of the recent presidential elections fought government censorship to communicate with the outside world.

Anonymization technology faces numerous challenges to increased adoption and commercial success. Moreover, the technology has generated controversy among those concerned that terrorists, pedophiles, criminals, and others could take advantage of it.

THE BASICS

Anonymity systems prevent observers from discovering the source of online communications. Typically, the system keeps a recipient or observer of a transmission from seeing the IP address of a source or tracking a message back to its originator.

"The name of the game is to keep the servers you visit from knowing your IP address, which means not connecting to them directly. This means going through one or more other computers [called *proxies*] to arrive at the desired destination," said James Marshall, an independent consultant and software developer who created CGIProxy, a free Web proxy.

Some anonymity systems also encrypt data.

History

The first popular anonymization tool was the Penet remailer developed by Johan Helsingius of Finland in the early 1990s. Penet was not totally safe for users because it kept a potentially accessible record of their names.

Some members of the Cypherpunk privacy and cryptography developers' group released their eponymous remailer in 1992.

Cottrell wrote the Mixmaster remailer in 1993. In 1995, he launched Anonymizer—the first Web-based anonymity system, initially a free service but now a commercial product.

Driving forces

Concerns about communications privacy are driving anonymization technology's increased adoption.

And as Internet use has grown, criminals have increasingly gone online to break the law, noted Rob Enderle, principal analyst with the

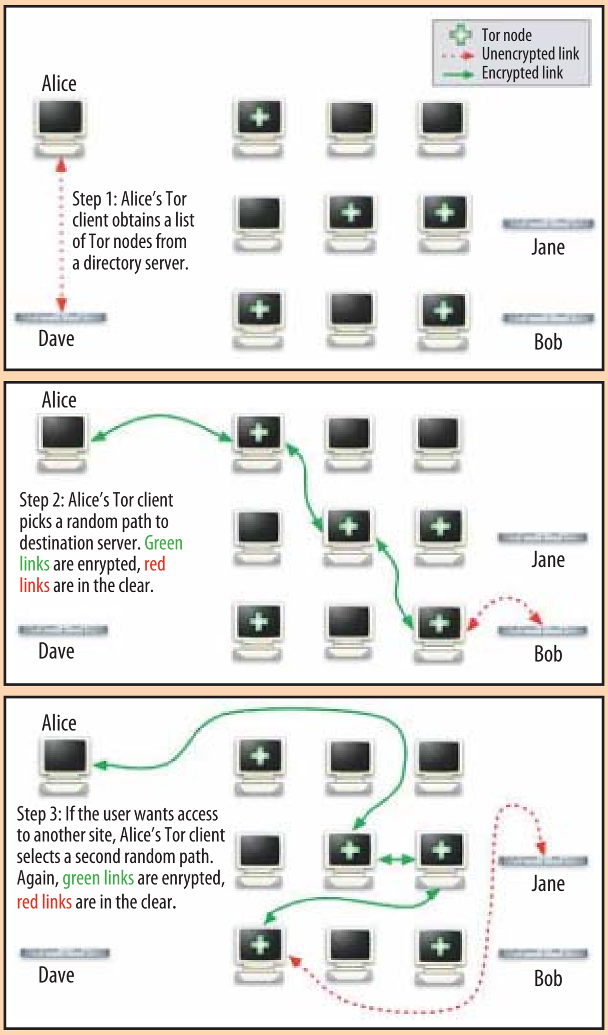

Tor node
Unencrypted link
Encrypted link
Alice
Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Jane
Dave
Bob
Step 2: Alice's Tor client picks a random path to destination server. Green links are enrypted, red links are in the clear.
Step 3: If the user wants access to another site, Alice's Tor client selects a second random path. Again, green links are enrypted, red links are in the clear.

Figure 1. The Tor anonymization system hides a user's identity by sending traffic through a series of participating nodes.

Enderle Group, a market research firm. In the process, they have looked for ways to keep law-enforcement agencies from identifying them.

In some countries, libraries and employers block content, and some ISPs and websites record people's Web habits for marketing purposes.

Dissidents and other protesters also want ways to communicate without governments being able to identify or trace transmissions back to them.

The 2006 OpenNet Initiative (http://opennet.net)—a research project by Harvard University and the Universities of Cambridge, Oxford, and Toronto—studied Internet censorship and surveillance in 46 countries. The study found that 25 of the nations filtered various types of communications—including political content, religious sites, and pornography—by blocking transmission to and, in some cases, from specific IP addresses.

Officials identify sources they want to block by tracking back communications and identifying their senders via tools such as traceroute and services such as whois. They also find sites to block by using search engines and monitoring discussion groups and chat rooms.

Some individuals and organizations identify various sources' IP addresses and sell the information to governments or other interested parties.

UNDER THE HOOD

Over time, demand for anonymization has grown and the types of uses, applications, and business models have evolved.

Commercial anonymization systems, such as SwissVPN (www.swissvpn.net), charge subscription fees for their services.

Noncommercial anonymization systems don't charge fees but instead generate revenue by selling advertising that appears on their webpages.

Home-brewed anonymization systems are based on anonymizing proxy packages, such as CGIProxy and Freenet, available online for free. They are popular with college students who use them to circumvent school networks' URL filtering systems.

Types of anonymizers

Users can install software to implement *simple virtual private network (VPN) systems*, such as Anonymizer Total Net Shield and Perfect Privacy. VPNs create encrypted tunnels through which traffic passes. Recipients or observers cannot read the encrypted traffic and thus cannot track it back to the sender.

Users can also install software for *simple proxies*, also known as *open proxies* and *anonymous proxies*. Users enter the proxy's IP address or hostname in their browser's network settings, and when they point their browser at a website, the browser tells the proxy which site to visit. The proxy visits the site on the users' behalf and sends the content back to them.

The systems remove the users' IP information from packets and replace it with their own IP information, said Rolf Wendolsky, a director of anonymization vendor JonDos.

The typical proxy provider sets up a server on the Internet through which users can relay traffic, which some anonymization applications encrypt. This *single-hop architecture* is easy to implement and maintain. However, users all enter and leave through the same server, thereby creating a single point of failure.

Daisy-chaining anonymization, which uses a *multihop* approach, sends a user's traffic through a series of participating nodes, as Figure 1 shows. The traffic travels a path which either the user or the anonymization software selects, depending on the application. The goal is to route traffic through nodes owned by different individuals or organizations. That way, no one organization can see enough packet information to identify the user.

With *form-based proxies*—such as Anonymizer Anonymous Surfing and Anonymouse—users enter the URL of websites they want to visit into a form field on an anonymization provider's page. The provider then takes the user to the desired site. The anonymization software rewrites links on the delivered page so that they connect to the provider, preventing anyone from tracing the transaction back to the original user.

Form-based proxies are written via either common-gateway-interface scripts, designed to transfer information from forms and other online sources between a Web server and a browser; or PHP Hypertext Preprocessor scripts, which run on a Web server and enable dynamic Web content such as forms.

Form-based proxies are popular because users don't have to configure or install any software.

However, they are the most insecure of all anonymization systems, said Wendolsky. For example, attacks could use form-based proxies to replace links on websites with URLs that send users to malicious sites.

"The disadvantage of such systems is [slower] performance, both because of the multiple hops and because of the poor performance of many nodes," explained Cottrell. Also, he added, users can't always judge the trustworthiness of the participating nodes' owners.

Protocol support

Protocol-specific systems like Anonymouse and PHProxy anonymize online activities—for example, e-mail or Web access—based on only one or several application-layer protocols, such as HTTP or the Simple Mail Transfer Protocol, thus they are not versatile.

Anonymization technology faces numerous challenges to increased adoption and commercial success.

But because they are designed to work in detail with only certain types of applications, they can effectively recognize and strip out all user-specific data from the traffic they send.

Protocol-independent systems such as JonDonym use approaches such as SOCKS—designed to send TCP traffic via a proxy server—which supports many communications protocols. They can also take advantage of VPNs, which also work with many protocols.

Although these systems obscure the path that traffic takes, they don't generally "understand" traffic well enough to actually change data in packets, which could reduce their effectiveness.

APPLICATIONS

The leading anonymizing applications include the following:

- Anonymizer (www.anonymizer.com): VPN- and form-based systems, supported by user payments, open to anyone on the Internet, protocol-independent, encrypts communications
- Anonymouse (http://anonymouse.org): form-based system, supported by on-site advertisements and user payments, open, protocol-specific
- I2P (Invisible Internet Project, www.i2p2.de): VPN system, free, closed to all but those on subscribing networks, open source, protocol-independent, encrypts communications
- JonDonym (https://www.jondos.de/en): multihop proxy system; free and commercial versions; open source; open; encrypts communications; originally developed by the Technical University of Dresden, the University of Regensburg, and JonDos
- Megaproxy (www.megaproxy.com): VPN system, supported by user payments, open, protocol-independent, encrypts communications
- Proxify (http://proxify.com): form-based proxy system, supported by advertisements or user payments, protocol-independent, encrypts communications
- Tor (www.torproject.org): multihop proxy system; free; open; open source; protocol-independent; encrypts communications; started in 2003 with 30 proxies on two continents, now has 2,000 on five continents and up to 500,000 users at any one time
- XeroBank (https://xerobank.com): multihop-proxy and VPN systems, supported by user payments, partially open source, open, protocol-independent, encrypts communications

CHALLENGES AND CONTROVERSY

Criminals could take advantage of improved anonymization technology to hide their identities, said analyst Enderle.

Also, anonymizers aren't foolproof. For example, if the first and last proxies in a system are malicious or compromised, the first proxy would know the client's identity and the last proxy would know the server's identity, explained Indiana University assistant professor Apu Kapadia. If the same person owns both proxies or if their separate owners communicate, this could break anonymity, he said.

Most open source projects publish enough information about their workings, including node addresses, to let governments or other organizations block traffic from at least some of those nodes, noted Cottrell.

According to Seth Schoen, staff technologist for the Electronic Frontier Foundation, a privacy and Internet-user-rights organization, there is a risk that some single-proxy anonymizer services may log users' IP addresses. If governments order them to turn over information or hackers break into their servers, users could lose their anonymity, he explained.

However, he noted, providing greater security would hurt performance because additional proxies and encryption increase overhead.

In fact, performance overhead sometimes causes anonymization to slow users' Internet access.

Expanding the number of nodes in anonymization systems could be difficult because users serving as nodes will experience a lot of traffic flowing through their computers.

Some ISPs block nodes to control spam. If, in the process, they block those used by anonymizers, Marshall said, this would hurt anonymization.

Browser complexity and the need to maintain browsing functionality could help proficient hackers sidestep anonymization, noted Wendolsky. Hackers could accomplish this in some cases, he explained, by exploiting browser plug-ins, JavaScript, cookies, caches, or HTML parsing engines.

Analyst Enderle stated, "Anonymizers are wrong-headed." The technology conceals identities, he said, which makes it attractive to criminals.

The technology's two biggest marketplace challenges are cultural and legal, according to Cottrell. "The legal challenge is that some countries are outlawing or could decide to prohibit the use of privacy tools and require all Internet providers to keep detailed access records. The cultural issue is the trend toward [openness on the Internet]."

However, proponents say that privacy and the desire to communicate online without fear of identification or government retribution are among the good reasons to use anonymization and that this will drive the technology's continued development and adoption.

Anonymizer, for example, has reported a 20 percent annual growth in its business over the past few years.

Marshall predicted that anonymization will have a bright future, with more organizations developing systems as people become aware of its importance. He said, "The demand is there."

Neal Leavitt is president of Leavitt Communications (www.leavcom.com), a Fallbrook, California-based international marketing communications company with affiliate offices in Brazil, France, Germany, Hong Kong, India, and the UK. He writes frequently on technology topics and can be reached at neal@leavcom.com.

Editor: Lee Garber, *Computer*, l.garber@computer.org

IEEE Symposium on Low-Power and High-Speed Chips

COOL Chips XIII

Yokohama Joho Bunka Center, Yokohama, Japan
(Yokohama Media & Communications Center, Yokohama, Japan)
April 14 - 16, 2010

CALL FOR CONTRIBUTIONS

COOL Chips is an International Symposium initiated in 1998 to present advancement of low-power and high-speed chips. The symposium covers leading-edge technologies in all areas of microprocessors and their applications. The COOL Chips XIII is to be held in Yokohama on April 14-16, 2010, and is targeted at the architecture, design and implementation of chips with special emphasis on the areas listed below. The COOL Chips Organizing Committee will ask the MICRO to publish selected papers in a special issue on COOL Chips XIII.

Contributions are solicited in the following areas:

- **Low Power-High Performance Processors, *"Eco-Processors"*, for - Multimedia, Digital Consumer Electronics, Mobile, Graphics, Encryption, Robotics, Automotive, Networking, Medical, Healthcare, and Biometrics.**
- **Novel Architectures and Schemes for - Single Core, Multi-Core, Embedded Systems, Reconfigurable Computing, Grid, Ubiquitous, Dependable Computing, GALS and Wireless.**
- **Cool Software including - Parallel Schedulers, Embedded Real-time Operating System, Binary Translations, Compiler Issues and Low Power Techniques.**

Proposals should consist of a title, an extended abstract (up to 3 pages) describing the product or topic to be presented and the name, job title, address, phone number, FAX number, and e-mail address of the presenter. The status of the product or topic should precisely be described. If this is a not-yet-announced product, and you would like to keep the submission confidential, please indicate it. We will do our best to maintain confidentiality. Proposals will be selected by the program committee's evaluation of interest to the audience. Submission should be made by e-mail, (Author's kit can be obtained from **http://www.coolchips.org/**)
to: Makoto Ikeda, Program Chair **e-mail: submit_xiii@coolchips.org**

Author Schedule:	**January 15, 2010**	**Extended Abstract Submission (by e-mail)**
	March 1, 2010	**Acceptance Notified (by e-mail)**
	March 15, 2010	**Final Manuscript Submission**

You are also invited to submit proposals for poster sessions by e-mail,
to : M. Muroyama , Poster Chair **e-mail: poster_xiii@coolchips.org**

Author Schedule:	**March 8, 2010**	**Poster Abstract Submission (by e-mail)**
	March 15, 2010	**Poster Acceptance Notified (by e-mail)**

For more information, please visit <**http://www.coolchips.org/**>.
For any questions, please contact the Secretariat <**cool_xiii@coolchips.org**>.

Sponsored by the Technical Committees on Microprocessors and Microcomputers and Computer Architecture of the IEEE Computer Society. In cooperation with the IEICE Electronics Society, ACM SIGARCH and IPSJ.

IEEE Celebrating 125 Years of Engineering the Future
IEEE computer society

System Takes New Approach to Speech Search

A new system promises to make it easier to search audio clips for specific phrases or names than using traditional speech-recognition software.

The AudioMining software that the Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS) developed would let, for example, a TV reporter quickly and accurately search a video clip for a specific part in which an interview subject said something important.

AudioMining software would let users search for terms within audio files without depending on a database of words that must be updated regularly, as is the case with standard speech-recognition systems.

Typically, automatic speech recognition (ASR) systems use a word dictionary and a statistical model for the typical usage of word sequences to produce a transcription, which then can be searched, said IAIS scientist Daniel Schneider.

The word dictionary contains only a limited number of words and names, so there are many that the ASR system won't recognize. Regularly, specialists must update the systems' name, word, and phrase database, a time-consuming and expensive process.

Schneider said his research team built AudioMining with a dictionary based on syllables, rather than words. The system breaks down word queries into syllable sequences and searches for matches in the syllable dictionary.

With about 10,000 stored syllables, the system can recognize any word, Schneider explained.

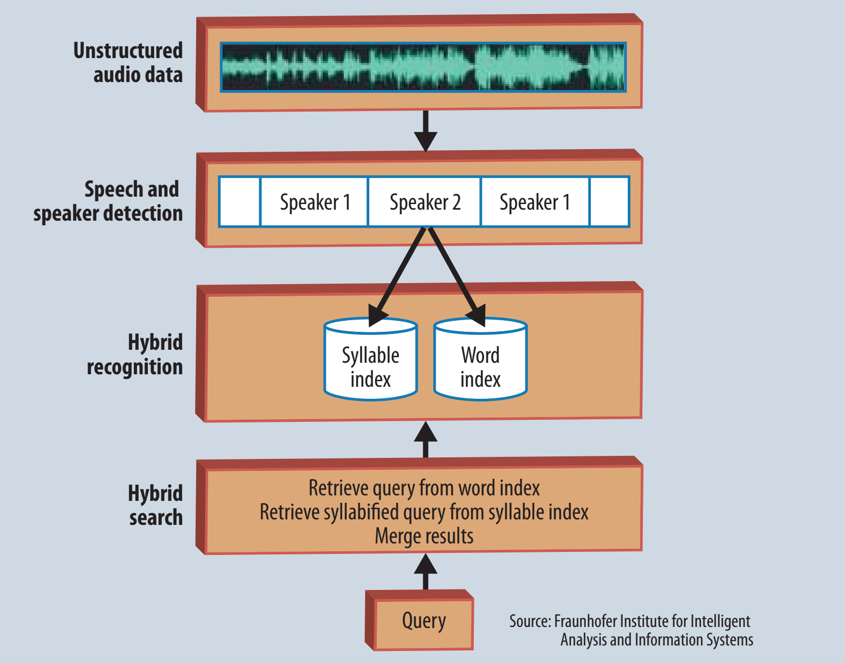
The approach analyzes an audio stream and uses efficient probabilistic algorithms, as well as "knowledge" about parts of speech, to segment the file. This lets the system identify different speakers and break down words into syllables, explained Schneider.

AudioMining uses *n*-gram language modeling, a probabilistic model for predicting language sequences, to more accurately identify syllables.

Users can add traditional word-recognition technology to improve accuracy.

AudioMining could be used to either analyze already-recorded speech—such as recordings of lectures and conferences—or monitor ongoing audio, such as a TV broadcast.

In testing, Schneider said, the system took only milliseconds to find 85 out of 100 searched-for utterances in audio files, with 99 percent


Unstructured audio data
Speech and speaker detection
Speaker 1
Speaker 2
Speaker 1
Hybrid recognition
Syllable index
Word index
Hybrid search
Retrieve query from word index
Retrieve syllabified query from syllable index
Merge results
Query
Source: Fraunhofer Institute for Intelligent Analysis and Information Systems

The Fraunhofer Institute for Intelligent Analysis and Information Systems has developed a system for searching audio clips for specific phrases or words. When a user submits a search query, the system breaks down the query's words into syllables. It then segments the audio data based on different speakers, words, and syllables. It uses a syllable dictionary to find the syllables in the speech that match those in the query. Traditional speech recognition uses a word dictionary, which is more difficult to work with.

of results being correct.

The Fraunhofer system doesn't perform speech recognition, as it doesn't identify what words mean, explained University of Calgary professor emeritus David Hill, a speech-recognition expert.

The system thus could be useful for searching for phrases in a library of spoken material but not for transcribing streaming audio, he said.

According to Schneider, his team is currently commercializing its technology. For a project the researchers are developing with Germany's ARD broadcast network, they built a system—with a public interface—for searching and comparing spoken quotes from German politicians during the 2009 election campaign. C

Program Uses Mobile Technology to Help with Crises

A nongovernmental organization has released open source software tools for collaboration and communication that let government agencies or humanitarian organizations quickly report, share, aggregate, and analyze important data via a cell phone.

This is part of the InSTEDD (Innovative Support to Emergencies, Diseases, and Disasters) NGO's project to use mobile technology to improve governments' and humanitarian agencies' ability to respond to health problems, disasters, and other regional crises.

InSTEDD's tools provide information on disease-outbreak epidemiology and natural disasters, and help humanitarian organizations collaborate to improve the services they offer.

Many health workers are isolated by distance or terrain, potentially making communications time-consuming and difficult. Thus, the ability to use fast, simple communications technology that works with basic cell phones is valuable, according to InSTEDD president and CEO Eric Rasmussen.

The GeoChat short-message-service tool lets NGO field workers or first responders to disasters use cell phones to send information via an SMS message to the groups for which they work, government agencies, or even international bodies such as the United Nations' World Health Organization.

For example, in GeoChat's first testbed, in Cambodia, eight district health officers utilize the tool to share observations, diagnoses, and disease-outbreak epidemiology reports with a provincial hospital, which could use it to report on a rapid response team's estimated arrival times to areas with problems.

Field workers include their locations with each message, which pops up as a conversation thread on an interactive map. The messages either go directly to an NGO or to the InSTEDD website, where users with the correct passwords could pick them up.

GeoChat works via SMS because it's convenient and because, in many areas, Internet communications aren't available, according to Rasmussen.

SMS requires payment by users. InSTEDD works with the PayPal online payment system and makes it convenient for organizations by, for example, letting the groups prepay for messages so that field workers don't have to worry about such matters.

When combined with InSTEDD's Mesh4X synchronous-communication tool, users can transmit data between established applications—such as Access, Excel, GoogleEarth, MySQL, and the Oracle Database—and between devices—like laptops, smart phones, PDAs, and servers—reliably, selectively, and securely in a distributed data mesh.

InSTEDD's Riff analytics tool can help governmental agencies and humanitarian organizations examine incoming data and make decisions, Rasmussen said.

InSTEDD is helping humanitarian organizations and government agencies deploy its free tools worldwide. Beta versions are already deployed in countries such as Bangladesh, Denmark, Ghana, Tanzania, and the US.

Rasmussen said the technologies could also be used for specialized commercial software applications. InSTEDD is currently negotiating agreements with several companies.

Google's philanthropic arm, Google.org, contributed $5 million to InSTEDD through 2008 and has just started funding a three-year, $6.67 million grant.

There is a huge need to improve collaboration and communications for humanitarian relief organizations responding to natural disasters and health crises, explained Google.org director Frank Rijsberman. He said InSTEDD is perhaps the only nonprofit organization able to accomplish this by combining the necessary software-engineering skills with a deep understanding of humanitarian relief organizations. C

News Briefs written by ***Linda Dailey Paulson****, a freelance technology writer based in Portland, Oregon. Contact her at ldpaulson@yahoo.com.*

FOOTBALL HELMET USES SENSORS TO PROVIDE HEATSTROKE WARNINGS

A company has designed sensor-based equipment that can be installed in a football helmet to warn that players may be unknowingly about to suffer heatstroke.

Hothead Technologies' Heat Observation Technology (HOT) system consists of one helmet-installed heat sensor that collects information about the wearer's skin temperature, explained Rick Lane, the company's general manager and director of operations.

The equipment relays this information to an accompanying PDA, whose user can monitor players' health in real time and download full reports after an event to spot trends.

Before a game, a coach or trainer can set a temperature threshold that, if exceeded, will prompt the system to send an alert, along with the player's name, jersey number, emergency contact information, and existing medical conditions. At that point, the system will also more frequently transmit information on the affiected player and monitor temperatures.

William Roberts, MD, a sports medicine expert and University of Minnesota professor, said although heat exhaustion occurs in various sports, heatstroke deaths are most frequent in high school and college football because helmets can keep the relatively small number of coaches and trainers from seeing signs of trouble. An average of five US high school and college football players have died of heat-related causes annually during the past few years, he noted.

Hothead Technologies has developed sensor equipment that can be installed in a football helmet to warn that a player may be about to suffer heatstroke.

Having technologies available to make monitoring heat levels easier is good because athletes are often so involved in their activities, they don't pay attention when they aren't feeling well, he said.

Scott Pyne, MD, a sports medicine specialist, said the prime contributor to heatstroke is the body's inability to cool itself via sweating. Humidity exacerbates the problem because saturated air won't enable the cooling provided by sweat evaporation.

HOT would be helpful if it accurately provides an athlete's core temperature, according to Roberts. However, he said, helmet-based sensoring isn't necessarily the best way to do this because the skin temperatures that such devices measure don't represent core-body temperatures.

There haven't been independent studies of the HOT helmet's effiectiveness, particularly in terms of accurately measuring core-body temperatures, noted Pyne.

Medical devices such as GI transducers—pill-shaped sensors that users swallow—can take an athlete's core temperature. However, they must be used in close proximity to devices that display information, which makes the systems expensive and difficult to use with a large sports team, Roberts said.

According to Lane, Hothead designed the HOT helmet to be an early-warning device, not medical equipment. C

Researcher Develops System for Distributed Debugging

Austrian researchers have developed an approach for the challenging task of debugging distributed systems.

Debugging is a critical process in the creation of any computer system. This has become increasingly important as systems have become more complex. For example, debugging is crucial for real-time, distributed systems, in which disparate machines must operate quickly and correctly in unison.

Debugging is significantly different for such systems than for conventional ones. Finding bugs or reproducing problematic scenarios in a distributed system frequently requires complex coordination, said professor Roland Höller with the University of Applied Sciences Technikum Wien.

Typical approaches to distributed debugging have linked systems' disparate elements with cables to an embedded control unit that connects via Ethernet or USB to one or more PCs running debugging software.

These approaches are often impractical because the elements of many distributed systems that must be debugged are embedded inside equipment such as industrial machinery or an automobile and are not easily accessible.

Höller's approach—which works with conventional debugging software—puts the debugging circuitry on a system element's chip, rather than an external device. The user issues the command to begin the debugging via a wireless or wired network, enabling the process to work even with difficult-to-access system elements.

According to Höller, his approach addresses distributed debugging by synchronizing the system clocks within each of the elements.

The synchronization enables the technique to coordinate even complex debugging activities across the distributed system by having each step run at the same time. This lets users know how the system being debugged will operate when it is actually running, with all elements functioning simultaneously.

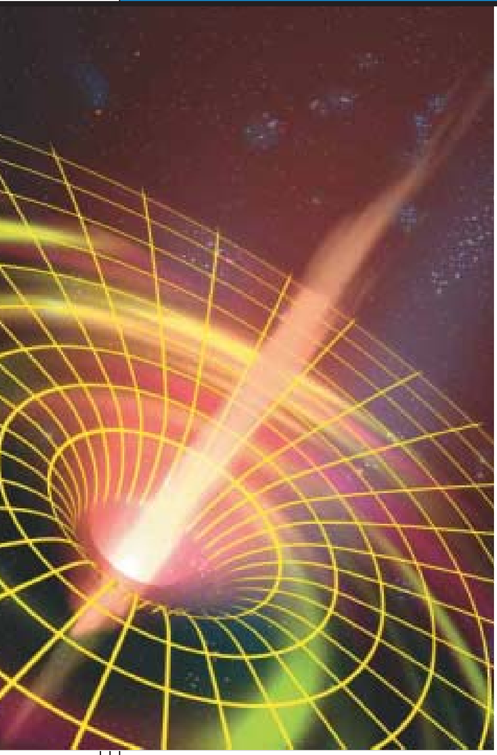
Höller said the challenge in developing his system was tightly integrating the clock synchronization with the debugging system.

The research team is running a prototype of its approach using both the Eclipse Foundation's Eclipse debugging software and the GNU Debugger.

His team plans to begin working to make its approach a debugging standard with several organizations, including the Nexus 5001 Forum, which develops embedded-processor debug-interface standards; and the Open Core Protocol International Partnership.

Höller said his team has patented and wants to commercialize its technique but will have to convince chip makers to provide valuable space on their processors for the debugging circuitry. C

Editor: Lee Garber, *Computer*, l.garber@computer.org



EXTREME-SCALE COMPUTING—WHERE 'JUST MORE OF THE SAME' DOES NOT WORK

Adolfy Hoisie, *Los Alamos National Laboratory*
Vladimir Getov, *University of Westminster*

In addition to enabling science through simulations at unprecedented size and fidelity, extreme-scale computing serves as an incubator of scientific and technological ideas for the computing area in general.

The leading edge of high-performance computing (HPC), an area of considerable growth and pace of progress, extreme-scale computing relates directly to the hardware, software, and applications enabling simulations in the petascale performance range and beyond. Moreover, extreme-scale computing acts as a scientific and technological driver for computing in general. In addition to enabling science through simulations at unprecedented size and fidelity, extreme-scale computing serves as an incubator of scientific and technological ideas for the computing area. As such, its rapid development significantly impacts several neighboring areas such as loosely coupled distributed systems, grid infrastructures, cloud computing, and sensor networks.

The complexity of computing at extreme scales is increasing rapidly, now matching the complexity of the simulations running on them. Therefore, the quest for higher processing speed has become only one of many challenges when designing novel high-end computer systems. This complexity arises from the interplay of various factors such as level of parallelism (systems in this range currently use hundreds of thousands of processing elements and are envisioned to reach millions of threads of parallelism), availability of parallelism in algorithms, design and implementation of system software, deep memory hierarchies, heterogeneity, reliability and resilience, and power consumption, just to name a few.

IT'S ALL ABOUT SCALABILITY

Achieving high levels of sustained performance in applications is a dauntingly challenging task. To respond to this never-ending demand for higher and higher performance, extreme-scale computing incorporates in a single topic area several research and development challenges related to scalability. The questions that have been attracting attention from the professional community at large include the following:

- Are there limits to manageable levels of parallelism? Are millions of threads tractable? What are the programming models that support application de-

velopment within reasonable levels of effort, while allowing high performance and efficiency?
- Is there a limit to the number of cores that can be used for building a single computer? What is the significance of heterogeneity and hybrid designs in this respect?
- Are there fundamental limits to an increasing footprint of the interconnect? What are the performance/reliability tradeoffs?
- What are the factors that hinder high levels of sustained performance? What are the best ways to assess, model, and predict performance in extreme-scale regimes?
- What are the system software challenges, limitations, and opportunities? Can we develop system software that harnesses heterogeneity and asynchronous designs?
- What are design considerations for the I/O and storage subsystems given the vast amounts of data generated by such simulations?
- What are the main characteristics and challenges in providing high-level quality of service by current and future extreme-scale systems? Given the size and complexity of the systems enabling extreme-scale computing, can we overcome the intrinsic limitations in reliability and resilience?
- Is it inevitable that extreme-scale supercomputers will be delivered together with an associated power plant? Can we reduce as much as possible the power consumption to save energy for a greener planet but also enable the design of even faster computers?

IN THIS ISSUE

In this special issue, we explore some of the salient aspects of extreme-scale computing. The selected articles cover a significant cross-section of the questions listed above.

In "Architectures for Extreme-Scale Computing," Josep Torrellas outlines the main architectural challenges of extreme-scale computing and describes potential paths forward to ensure the same fast pace of progress that this area sustained in the past decade. Key technologies such as near-threshold voltage operation, nonsilicon memories, photonics, 3D die stacking, and per-core efficient voltage and frequency management will be key to energy and power efficiency. Efficient, scalable synchronization and communication primitives, together with support for the creation, commit, and migration of lightweight tasks will enable fine-grained concurrency. A hierarchical machine organization, coupled with processing-in-memory will enhance locality. Resiliency will be addressed with a combination of techniques at different levels of the computing stack. Finally, programming the machine with a high-level data-parallel model and using an intelligent compiler to map the code to the hardware will ensure programmability and performance. Finally, the author outlines Thrifty, a novel extreme-scale architecture.

In "Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers," Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu describe their recently developed high-speed interconnect architecture for next-generation supercomputers that operate beyond 25 petaflops. The first such system, which will be one of the world's largest supercomputers, is scheduled to begin operation in 2011. The network topology of Tofu is a fault-tolerant 6D mesh/torus, and each link has 10 Gbytes of bidirectional bandwidth. Each of the computation nodes employs four communication engines with an integrated collective function. The Tofu interconnect is designed to run a 3D torus application even if there are some faulty nodes inside the system's submesh. A user can specify a 3D Cartesian space for a job, and the system allocates nodes to parallel processes of the job and ensures that a neighboring node of the application's Cartesian space is also a neighbor in the physical 6D space. Since there are several combinations of physical coordinates for folding application coordinates, the system can provide a suitable submesh shape from the available free nodes, which greatly improves system utilization. Additionally, system availability has been further improved by using a newly developed graceful degradation technique that allows a 3D Cartesian space to become available within a faulty 6D submesh.

Extreme-scale computing incorporates in a single topic area several research and development challenges related to scalability.

As supercomputing applications and architectures grow more complex, researchers need methodologies and tools to understand and reason about system performance and design. "Using Performance Modeling to Design Large-Scale Systems" by a team of authors from the Los Alamos National Laboratory, New Mexico, is dedicated to this important topic area. Existing petascale systems contain sufficient hardware complexity to make it impossible for application developers, hardware designers, and system buyers to have an intuitive "feeling" for those factors that have a bearing on performance; as we march toward exascale systems this problem will only get worse. In this article, the authors present a proven, highly accurate quasi-analytical performance modeling methodology that puts performance analysis tools in the hands of applications and systems researchers. As a case in point, the article demonstrates how performance modeling can accurately predict application performance on IBM's Blue Gene/P system, one of today's largest parallel machines, for three large-scale applications in application domains including shock hydrodynamics, deterministic particle transport, and plasma fusion modeling. Using this system as a baseline, a performance look-ahead is shown for the near-term future, theorizing how these applications will perform on potential future systems incorporating improved compute and interconnection network performance.

In "Parallel Scripting for Applications at the Petascale and Beyond," Michael Wilde and colleagues characterize the applications that can benefit from extreme-scale scripting, discuss the technical obstacles that such applications raise for the system and application architect, and present results achieved with parallel script execution on the extreme-scale computers available today. They show examples of the science that can be achieved with this approach, the scale that extreme machines make possible, the performance of applications at these scales, the systems and architectural challenges that were overcome to make this feasible, and the challenges and opportunities that remain. The article concludes by exploring the relationships—and promising connections—between parallel scripting and traditional memory.

In "Energy-Efficient Computing for Extreme-Scale Science," David Donofrio and colleagues describe the Green Flash project, which aims to deliver an order-of-magnitude increase in efficiency, both computationally and in cost-effectiveness. The main idea is based on offering a many-core processor design with novel alternatives to cache coherence that enable far more efficient interprocessor communication than a conventional symmetric multiprocessing approach coupled with autotuning technologies to improve kernels' computational efficiency. Application-driven HPC design represents the next transformational change for the industry and will be enabled by leveraging existing embedded ASIC design methods, autotuning for code optimization, and emerging hardware emulation environments for performance evaluation. Looking beyond climate models, the Green Flash approach could allow future exaflops-class systems to be defined by science rather than have the science artificially constrained by generic machine characteristics.

In June 2008, the world entered the petaflops era with the Roadrunner supercomputer installation at Los Alamos. It is widely anticipated that systems with millions of threads, capable of achieving tens of petaflops, will be in existence in just a couple of years. Exascale computing is now within reach.

Development in this area attracts support from funding agencies all around the globe, including the US, Asia (Japan, China, and India, most notably), Europe, and Australia. The main reasons for this are the strategically important application domains and the incubator role that this field has for computing in general. Extreme-scale computing, and HPC in general, is an exciting and fast-developing area with sizable contributions coming from different professional categories, including research and development, industry, education, and end users.

We hope you will enjoy reading the articles in this special issue. ■

Adolfy Hoisie *is the leader of the Center for Advanced Architectures and Usable Supercomputing (CAAUS) and of the Computer Science for High-Performance Computing group at the Los Alamos National Laboratory. His research interests are performance analysis and modeling of large-scale systems and applications, system architecture, and extreme-scale computing in general. He is a past recipient of the Gordon Bell award and of other awards for research excellence. Contact him at hoisie@lanl.gov.*

Vladimir Getov *is a professor of distributed and high-performance computing at the University of Westminster, London. His research interests include parallel architectures and performance, autonomous distributed computing, and high-performance programming environments. He received a PhD and DSc in computer science from the Bulgarian Academy of Sciences. He is a member of the IEEE and the ACM and is Computer's area editor for high-performance computing. Contact him at v.s.getov@westminster.ac.uk.*

CCGrid 2010

The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

May 17-20, 2010, Melbourne, Australia

http://www.manjrasoft.com/ccgrid2010

CALL FOR PAPERS

Tremendous advances in processing, communication and systems/middleware technologies are leading to new paradigms and platforms for computing, ranging from computing ***Clusters*** to widely distributed ***Grid*** and emerging ***Clouds***. CCGrid is a series of very successful conferences, sponsored by the IEEE Computer Society Technical Committee on Scalable Computing (TCSC) and ACM, with the overarching goal of bringing together international researchers, developers, and users and to provide an international forum to present leading research activities and results on a broad range of topics related to these platforms and paradigms and their applications. The conference features keynotes, technical presentations, posters, workshops, tutorials, as well as the SCALE challenges featuring live demonstrations.

In 2010, CCGrid will return to Australia, to Melbourne, to celebrate its 10th anniversary. CCGrid 2010 will have a special focus on three important and immediate issues that are significantly influencing all aspects of Cluster, Cloud and Grid computing: ***Economics, Environment and Autonomics***. Topics of interest include, but are not limited to:

- **Paradigms and Technologies:** System architectures, Design and deployment; Programming models, language, systems and tools/environments; Virtualization; Middleware technologies; Volunteer Computing.
- **Service-Orientation:** Service oriented architectures; Utility computing models; *aaS paradigm; Service composition and orchestration
- **Greening:** Environment friendly computing ecosystems; Hardware/software/application energy efficiency; Power and cooling; Thermal/power awareness
- **Autonomic Management:** Self-* behaviors, models and technologies; Autonomic paradigms and approaches (control-based, bio-inspired, emergent, etc.); Bio-inspired approaches to management; SLA definition and enforcement;
- **Economic Aspects:** Utility models and computing economies; Economic-based models and approaches.
- **Monitoring and Evaluation:** Performance models; Monitoring and evaluation tools, Analysis of system/application performance; Benchmarks and testbeds.
- **Security and Trust:** Cloud/Grid security and trust; Access control; Data privacy and integrity; Regulation.
- **Applications and Experiences:** Applications to real and complex problems in science, engineering, business and society; User studies; Experiences with large-scale deployments systems or applications.

CALL FOR WORKSHOP PROPOSALS

CGrid 2010 is inviting proposals for organising of co-located and peer-reviewed workshops on emerging topics.
For more details:
http://wiki.cs.cf.ac.uk/bin/view/Sandbox/WorkshopsCCGrid10

PAPER SUBMISSION

Authors are invited to submit papers electronically. Submitted manuscripts should be structured as technical papers and may not exceed 10 letter size (8.5 x 11) pages including figures, tables and references using the IEEE format for conference proceedings (print area of 6-1/2 inches (16.51 cm) wide by 8-7/8 inches (22.51 cm) high, two-column format with columns 3-1/16 inches (7.85 cm) wide with a 3/8 inch (0.81 cm) space between them, single-spaced 10-point Times fully justified text). Submissions not conforming to these guidelines may be returned without review. Authors should submit the manuscript in PDF format and make sure that the file will print on a printer that uses letter size (8.5 x 11) paper. The official language of the meeting is English. All manuscripts will be reviewed and will be judged on correctness, originality, technical strength, significance, quality of presentation, and interest and relevance to the conference attendees.

Submitted papers must represent original unpublished research that is not currently under review for any other conference or journal. Papers not following these guidelines will be rejected without review and further action may be taken, including (but not limited to) notifications sent to the heads of the institutions of the authors and sponsors of the conference. Submissions received after the due date, exceeding length limit, or not appropriately structured may also not be considered. Authors may contact the conference chairs for more information. The proceedings will be published through the IEEE Computer Society Press, USA and will be made online through the IEEE Digital Library.

CHAIRS & COMMITTEES

General Chair
Rajkumar Buyya, University of Melbourne and Manjrasoft Pty Ltd, Australia

Program Committee Chair
Manish Parashar, Rutgers University, USA

Workshop Chair
Omar Rana, Cardiff University, UK

Local Organising Chair
James Broberg, University of Melbourne, Australia

Publicity Chair
Cho-Li Wang, University of Hong Kong, China

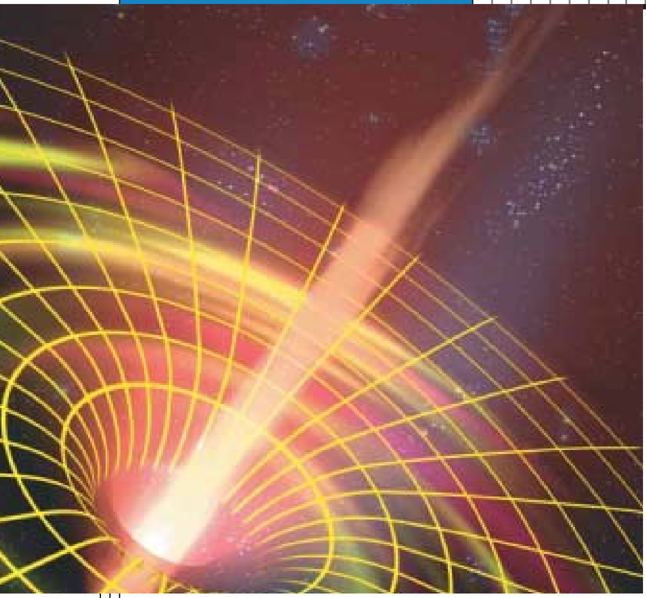
IMPORTANT DATES (TENTATIVE)

Papers Due:	2 November 2009
Notification of Acceptance:	18 December 2009
Camera Ready Papers Due:	25 January 2010

SPONSORSHIPS (PENDING)

IEEE Computer Society TCSC, ACM SIGARCH

COVER FEATURE



ARCHITECTURES FOR EXTREME-SCALE COMPUTING

Josep Torrellas, *University of Illinois at Urbana-Champaign*

Extreme-scale computers promise orders-of-magnitude improvement in performance over current high-end machines for the same machine power consumption and physical footprint. They also bring some important architectural challenges.

After many years of research and development, high-end computing has finally reached peta-op performance—that is, individual machines that can execute about 10^{15} operations per second (peta-ops). These machines attain such performance using extraordinary, highly concurrent architectures. They include Los Alamos National Laboratory's Roadrunner, which augments conventional processors with high-performance IBM Cell accelerators, providing a hybrid platform; or installations of IBM's Blue Gene/P, which use a massive number of simpler, conventional processors and rely on high chip and system integration. They will soon include the University of Illinois' Blue Waters system described in the "Blue Waters: Application-Driven System Design for Sustained Petascale Performance" sidebar, which uses high-performance processors and novel system components and packaging. With all of these architectures available, the next few years will bring breakthroughs in science and engineering.

However, these architectures are hardly scalable. They are simply too large and consume too much power. Indeed, petascale machines have a footprint of about 1/10th of a football field and consume several megawatts (MW). One megawatt costs about one million dollars per year. For these reasons, high-performance computer architects are focusing on extreme-scale computing.

Broadly speaking, an extreme-scale architecture is one thousand times more capable than a current architecture with the same power consumption and physical footprint. This means that a machine with the power consumption and physical footprint of a current petascale machine must be able to deliver exascale performance—namely, 10^{18} operations per second (exa-ops). It also means that, intuitively, the power consumption and physical footprint of a current departmental server should be enough to deliver petascale performance. Finally, a single commodity chip should deliver terascale performance—namely, 10^{12} operations per second (tera-ops).

Clearly, attaining such a general-purpose tera-op chip, peta-op departmental server, and exa-op data center would revolutionize computing. Conceiving and building such systems, however, poses technical challenges at all levels of the computing stack, including circuits, architecture, software systems, and applications. The sheer size of the challenges and opportunities of attaining such systems by the end of the next decade should act as a strong motivator for researchers.

ARCHITECTURAL CHALLENGES IN EXTREMEﬁSCALE COMPUTING

To attain extreme-scale computing, researchers must address architectural challenges in energy and power efficiency, concurrency and locality, resiliency, and programmability.

BLUE WATERS: APPLICATION-DRIVEN SYSTEM DESIGN FOR SUSTAINED PETASCALE PERFORMANCE

Robert Fiedler, Robert Wilhelmson, William Kramer, and Brett Bode
National Center for Supercomputing Applications

Blue Waters (www.ncsa.illinois.edu/BlueWaters), a system designed to provide sustained petaflops performance on a wide range of applications, is being developed by IBM in collaboration with the National Center for Supercomputing Applications (NCSA) and the University of Illinois, and is funded by the National Science Foundation (NSF) and the state of Illinois.

The heart of this system is the Power7 chip, which features eight cores, each with 32-Kbyte L1 instruction and data caches and a 256-Kbyte L2 unified cache. The entire 32-Mbyte on-chip L3 cache can be accessed by any core with latency approximately three times lower than local memory. For each core, up to 4 Mbytes of data in the L3 cache is automatically migrated to a private region with latency approximately 15 times lower than local memory. Each core includes four floating-point units, a VSX (vector) unit that supports single- and double-precision operands, and two load/store units. Simultaneous multithreading (SMT) is supported, allowing one, two, or four SMT threads per core. Dual double data rate 3 (DDR3) memory controllers provide a sustained memory bandwidth of 100 Gbytes per chip.

The 200,000+ Power7 cores in the system communicate via a unique, integrated, high-speed, low-latency interconnection fabric. Remote direct memory access technology enables effective overlap of communication and I/O operations with computational work. The system also includes well over 10 Pbytes of user disk space in a general parallel file system (GPFS), and an archival high-performance storage system (HPSS) that will expand to 500 Pbytes. GHI, a software interface between GPFS and HPSS, will enable automatic migration of disk files to archival storage while presenting users with a simple, unified view of their files.

The Blue Waters packaging extends the use of water-cooled designs, leading to greater energy efficiency. In cold weather, an outdoor cooling tower will chill the water. In addition, energy losses due to AC/DC conversion associated with an uninterruptable power supply are eliminated by exploiting the university's highly reliable electricity supply.

High levels of reliability and automated system health monitoring will undoubtedly be critical for practical use of extreme-scale systems. Blue Waters includes numerous reliability, availability, and serviceability (RAS) features designed to ensure that the mean time between failures is more than a few days, and includes automated checkpoint/restart capabilities. Moreover, an integrated system console will assist operators in monitoring the system's health by automatically filtering status data collected by very large numbers of system components, enabling them to quickly pinpoint any problematic hardware.

The programming environment for application developers and users is an important aspect of the Blue Waters system, and one that is under active development. It not only accommodates established tools and parallel programming models such as the message passing interface (MPI) and OpenMP, but also encourages the use of new tools and models such as Unified Parallel C (UPC), Co-Array Fortran, and global shared memory by ensuring interoperability for all of these programming languages and models.

The Blue Waters programming environment also includes advanced software development tools, which are a key factor in programmer productivity. Developers will have the opportunity to move beyond the traditional command-line environment to an expanded Eclipse-based integrated development environment. The IDE will provide intuitive access via a common communication interface to many powerful capabilities on Blue Waters, including tools for basic code development and building, for remote debugging, and for automated/expert-system-guided performance tuning. Further, interfaces provided to the Blue Waters resource manager will facilitate science and engineering discovery, including input dataset creation/data staging, batch job submission and monitoring, in-line data analysis using coscheduled processors, data reduction/postprocessing, remote scientific visualization, and archival data storage.

The Blue Waters system design was chosen to deliver sustained petascale performance for a broad range of science and engineering applications. A wide range of applications selected by the NSF through the Petascale Computing Resource Allocations (PRAC) program are currently being prepared for Blue Waters with the help of a team of application specialists at NCSA. The PRAC teams span many areas of investigation, including quantum chromodynamics (fundamental properties of matter), astrophysics (cosmology, galaxy formation, gamma ray bursts, turbulent stellar dynamics), chemistry (biomolecular dynamics, materials science, superconductors), turbulent fluid flows, geosciences (earth quakes), weather and climate modeling (tornadoes, global warming), social sciences (contagion), and evolutionary biology.

***Robert Fiedler** is a technical program manager at the National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign. Contact him at rfiedler@ncsa.uiuc.edu.*

***Robert Wilhelmson** is a chief science officer at NCSA. Contact him at bw@ncsa.uiuc.edu.*

***William Kramer** is a deputy project director at NCSA. Contact him at wkramer@ncsa.uiuc.edu.*

***Brett Bode** is a technical program manager at NCSA. Contact him at bbode@ncsa.uiuc.edu.*

Increasing energy and power efficiency

As Peter Kogge and his colleagues indicate in their DARPA-sponsored study on exascale computing, improving energy and power efficiency is the most formidable challenge facing designers of high-end systems.[1] Because extreme-scale machines must be three orders of magnitude more energy efficient than current machines, a possible target for extreme-scale computing is an exa-op data center that consumes 20 MW, a peta-op departmental server that consumes 20 kilowatts (KW), and a tera-op chip multiprocessor that consumes 20 watts (W). These numbers imply that the machine must deliver 50 giga operations (or 50×10^9 operations) per watt. Because these operations must be performed in a second, each operation can only consume, on average, an energy of 20 pico-Joules (pJ). For reference, consider Intel's Core Duo mobile processor circa 2006, which consumed, on average, more than 10,000 pJ per instruction.[2] Our target is even harder to attain than these numbers suggest. This is because large machines spend most of the energy transferring data from or to remote caches, memories, and disks. Minimizing data transport energy, rather than arithmetic logic unit (ALU) energy, is the real challenge.

Several evolutionary approaches to attaining more energy-efficient architectures exist. At the circuit level, these approaches emphasize designing circuits for energy and power efficiency, rather than for speed, as in most current approaches. Such designs include on-chip interconnection network circuits for low swing, or new memory layouts and bank organizations that minimize the amount of capacitance switched per access. The latter is important because current memory designs focus on providing cost-effective bandwidth, and are wasteful when activating portions of the memory. Future designs must minimize the energy spent charging and discharging lines, possibly through memory designs that include hierarchical bit-line organizations.

Phase change memory's main attraction is its scalability with process technology.

At the microarchitecture level, evolutionary approaches involve simplifying the cores, making their pipelines shallower and their execution engines less speculative. Finally, at the machine architecture level, a popular approach is to augment the processing nodes with accelerators that are energy-efficient for some operations.

Unfortunately, attaining three orders of magnitude higher efficiency in energy and power requires all of this and much more. In particular, it calls for the maturity of several technologies that are now being developed or investigated.

Near-threshold voltage operation. One of the most effective approaches for energy-efficient operation is to reduce the supply voltage (V_{dd}) to a value only slightly higher than the transistor threshold voltage (V_{th}). This is called near-threshold voltage (NTV) operation. It corresponds to a V_{dd} value of around 0.4 V, compared to a V_{dd} of around 1 V for current designs.

Broadly speaking, operation under NTV can reduce the gates' power consumption by about 100× while increasing their delay by 10×. The result is a total energy savings of one order of magnitude.[3] In addition to the 10× increase in circuit delay, the close proximity of V_{dd} and V_{th} induces a 5× increase in gate delay variation due to process variation, and a several orders-of-magnitude increase in logic failures—especially in memory structures, which are less variation tolerant.

Effective use of NTV in extreme-scale architectures will require solving these challenges. For example, increasing the number of cores in the machine can help make up for the cores' lower speed. Moreover, the optimal NTV for memory structures is slightly higher than the optimal NTV for logic. Thus, caches can cycle at a few times higher frequency than cores, suggesting novel architectural designs in which a group of cores might share a cache.[3]

Aggressive use of circuit or architectural techniques that minimize or tolerate process variation can address the higher-variation shortcoming. This includes techniques such as body biasing and variation-aware job scheduling. Finally, novel designs of memory cells and other logic can solve the problem of higher probability of logic failure. Overall, NTV operation is a promising direction that several research groups are pursuing.

Nonsilicon memory. Nonsilicon memory is another relevant technology. Phase change memory (PCM), which is currently receiving much attention, is one type of nonsilicon memory. PCM uses a storage element composed of two electrodes separated by a resistor and phase-change material such as $Ge_2Sb_2Te_5$.[4] A current through the resistor heats the phase-change material, which, depending on the temperature conditions, changes between a crystalline (low-resistivity) state and an amorphous (high-resistivity) one—hence recording one of the two values of a bit.

PCM's main attraction is its scalability with process technology. Indeed, both the heating contact areas and the required heating current shrink with each technology generation. Therefore, PCM will enable denser, larger, and very energy-efficient main memories. DRAM, on the other hand, is largely a nonscalable technology, which needs sizable capacitors to store charge and, therefore, requires sizable transistors.

Currently, PCM has longer access latencies than DRAM, higher energy per access (especially for writes), and limited lifetime in the number of writes. However, advances in circuits and memory architectures will hopefully deliver advances in all these axes while retaining PCM scalability.

Finally, because PCM is nonvolatile, it can potentially support novel, inexpensive checkpointing schemes for extreme-scale architectures. Researchers can also use it to design interesting, hybrid main memory organizations by combining it with plain DRAM modules.

Photonic interconnects. Optics have several key properties that can be used for interconnects. They include low-loss communication, very large message bandwidths enabled by wavelength parallelism, and low transport latencies, as given by the speed of light.[5] Consequently, they are especially good substrates for long-range communication. Indeed, when used for long-haul data communication, they deliver substantial end-to-end reductions in energy per bit and time per access. We therefore expect extreme-scale machines to use photonic interconnects extensively, especially to support communication between far-away nodes in larger machines. Some researchers are also proposing the use of photonics for on-chip interconnects,[6] targeting the technology to large, high-bandwidth message transfers on chip. An area of intense current research is efficient interfaces between electronic and photonic signaling.

Other system technologies. Several other technologies will likely significantly impact energy and power efficiency. An obvious one is 3D die stacking, which will reduce memory access power. A 3D stack might contain a processor die and memory dies, or it might contain only memory dies. The resulting compact design eliminates energy-expensive data transfers, but introduces manufacturing challenges, such as the interconnection between stacked dies through vias. Interestingly, such designs, by enabling high-bandwidth connections between memories and processors, might also induce a reorganization of the processor's memory hierarchy. Very high bandwidth caches near the cores are possible.

Efficient on-chip voltage conversion is another enabling system technology. The goal here is for the machine to be able to change the voltage of small groups of cores in tens of nanoseconds, so they can adapt their power to the threads running on them or to environmental conditions. A voltage controller in each group of cores can regulate the group's voltage. Hopefully, the next few years will see advances in this area.

Enabling concurrency and locality

In general, the performance of future energy-efficient extreme-scale machines will not be attained through high frequency—dynamic power consumption is roughly proportional to the cube of the frequency. Instead, circuits will likely be designed for low voltages and modest frequencies. Consequently, we will have to rely on more threads running concurrently.

Assume, for example, 1-GHz cores, each completing one operation per cycle. In this case, a chip will need 1,024 cores to attain one tera-op, a server will need about 1 million cores to attain one peta-op, and a data center will need about 1 billion cores to attain one exa-op. In reality, a thread will often stall waiting for data, rather than committing one operation per cycle. Consequently, to hide the stall time and attain the desired performance level, the system will have to support several times more threads. Extreme-scale architectures will need memory hierarchy organizations, synchronization primitives, and network links that support these concurrency levels. Moreover, designers cannot optimize such structures in a way that ignores, let alone penalizes, locality. Exploiting high degrees of spatial and temporal data locality is the only way to attain the desired performance at the target power budget.

We suggest two architectural supports to enable the fine-grained parallelism that extreme-scale machines require. The first support is efficient, scalable synchronization and communication primitives—especially for dynamic and irregular parallelism. These primitives must provide efficient point-to-point synchronization between two cores and collective operations.[7] Examples include low-overhead dynamic hierarchical barriers, producer-consumer synchronization through full-empty bits, and broadcast updates.

The second architectural support for fine-grained parallelism is low-overhead primitives for the creation, commit, and migration of lightweight tasks. Such lightweight tasks are likely to be created by the compiler, spawned with a single instruction, and managed with scalable queuing structures that minimize overheads and stall times.

In addition, several hardware architecture structures can help minimize data movement and exploit locality.

A processing-in-memory unit typically performs memory-intensive operations on arrays or sets of data.

The most obvious one is a many-core chip organization based on clusters. A cluster is a set of cores that have physical proximity and share some cache or other storage structure. The compiler can break a task into smaller subtasks and assign them to the cores in a cluster.

A second structure for locality is simple compute engines in the memory controllers or in the L3 cache controllers that perform certain memory-intensive computations. Such an approach—often known as processing-in-memory (PIM)—seeks to avoid transferring large amounts of data between the memory and the main cores and then back to perform a simple computation. PIM can be embodied in a variety of hardware—from simple functional units to specialized compute engines. A PIM unit typically performs memory-intensive operations on arrays or sets of data, such as element-by-element operations, reductions of various sorts, and recurrences.

Finally, although conventional cache hierarchies seek to minimize data movement, it is important to note that they sometimes end up moving sizable amounts of data unnecessarily. They do this through their use of cache lines and automatic mapping of lines in the cache. Mechanisms to prevent such data movements are needed.

Bolstering resiliency

Resiliency will be another key challenge in extreme-scale machines due to a combination of several effects:

- Spatial variations in process, voltage, and temperature, as well as logic wearout (aging), will likely become relatively more acute as semiconductor feature sizes decrease.
- Smaller feature sizes imply less charge in storage elements, making these elements more vulnerable to soft

Table 1. Characteristics of several popular techniques for resiliency.

Level	Detection and isolation	Recovery
Hardware	Error-correcting codes (ECC) for on- and ofi-chip memory structures; parity for processor data path and network; testing circuitry, sensors, detectors, and watchdog timers; hardware replication; checker engines or cores	Low-overhead checkpointing and rollback; hardware reconflguration, including disabling or salvaging the faulty component
Operating system/ runtime	Resiliency module to diagnose and isolate	Job scheduling around faulty components; virtualization
Compiler	Augmenting the code with checks	Compiler support for checkpointing
Application/ programming system	Applications that check themselves	Applications that checkpoint themselves; transactional model

errors induced by particle impacts.

- The use of V_{dd} values that are close to V_{th} will increase process variation.

All these effects will increase the chances of transient and permanent faults. At the same time, the largest extreme-scale machines will have many components, which will also increase the chance of faults. For example, an exa-op supercomputer might have 10 to 100 petabytes of memory, requiring tens or hundreds of millions of memory chips. The machine might also have hundreds of exabytes of secondary storage, requiring millions of disk drives.

No single solution can fully address the resiliency challenge. Instead, acceptable resilience in extreme-scale architectures might only be attainable through a combination of techniques at different levels of the computing stack. Each of these techniques has advantages and shortcomings, making the most cost-effective combination a complex function depending on, among other things, the workload executed.

Table 1 lists some of the most popular techniques for resiliency. We classify them based on whether they detect and isolate (and sometimes also correct) errors or recover from errors. Within each group, we classify the techniques based on the level at which they operate—namely, hardware, operating system and runtime, compiler, or application and programming system.

Several popular techniques support error detection and isolation in hardware. They include error-correcting codes (ECCs)—which also correct errors—for on- and off-chip memory structures and parity for processor data path and network links. Extreme-scale architectures might need to augment single-bit parity in the data path and network with more expensive detection and perhaps even correction codes. Testing circuitry (exercised right after manufacture and periodically in the field), various sensors and detectors (wearout, power, and temperature), and watchdog timers to detect timeouts will likely also be needed. Widespread hardware replication or using engines or cores to check the work of other cores are probably too expensive for these machines.

Software can also help detect and isolate errors. A resiliency module can diagnose and isolate (and even correct) the error at runtime, while a compiler can augment the code with checks on code control flow or data accesses. Finally, some applications can check themselves.

Error recovery is typically based on checkpointing and rollback. Checkpointing can be implemented in different layers. For example, as Table 1 shows, it can be supported in hardware. One hardware-based implementation with especially low overhead is ReVive,[8] which uses in-memory, incremental checkpointing—possibly coupled with nonvolatile memory.

Checkpointing can also be compiler driven. In this case, the compiler, fully aware of the program state at any point, chooses to checkpoint when the program has the least state. Finally, the application itself can decide when to checkpoint, again based on the size of the program state. In general, checkpointing should induce only minor overhead during error-free operation and, in a rollback, result in little work loss and in low error-recovery latency.

Unfortunately, conventional checkpointing consumes substantial time and power, and requires high disk bandwidth and capacity. As the processor core count increases to more than one billion for exa-op machines, conventional checkpointing can consume most of the execution time and power, hence becoming infeasible. Thus, effective checkpointing support in extreme-scale architectures is a major challenge. These machines will require novel, highly energy-efficient checkpointing and rollback mechanisms that combine techniques from multiple layers of the stack.

Other recovery mechanisms rely on reconfigurable or redundant hardware, such as spare cores. On an error, the hardware can be reconfigured and the faulty component can be disabled or salvaged. The operating system or runtime system can schedule jobs around faulty components, and even rely on virtualization to transparently mask away the faulty component. Finally, at the application and programming system level, the application can be written using intrinsically resilient programming models. For example, in the transactional model, the transaction is rolled back if an error occurs.

Designing for programmability

Programming highly concurrent machines has traditionally required heroic efforts. Extreme-scale architectures, with their emphasis on power efficiency, can

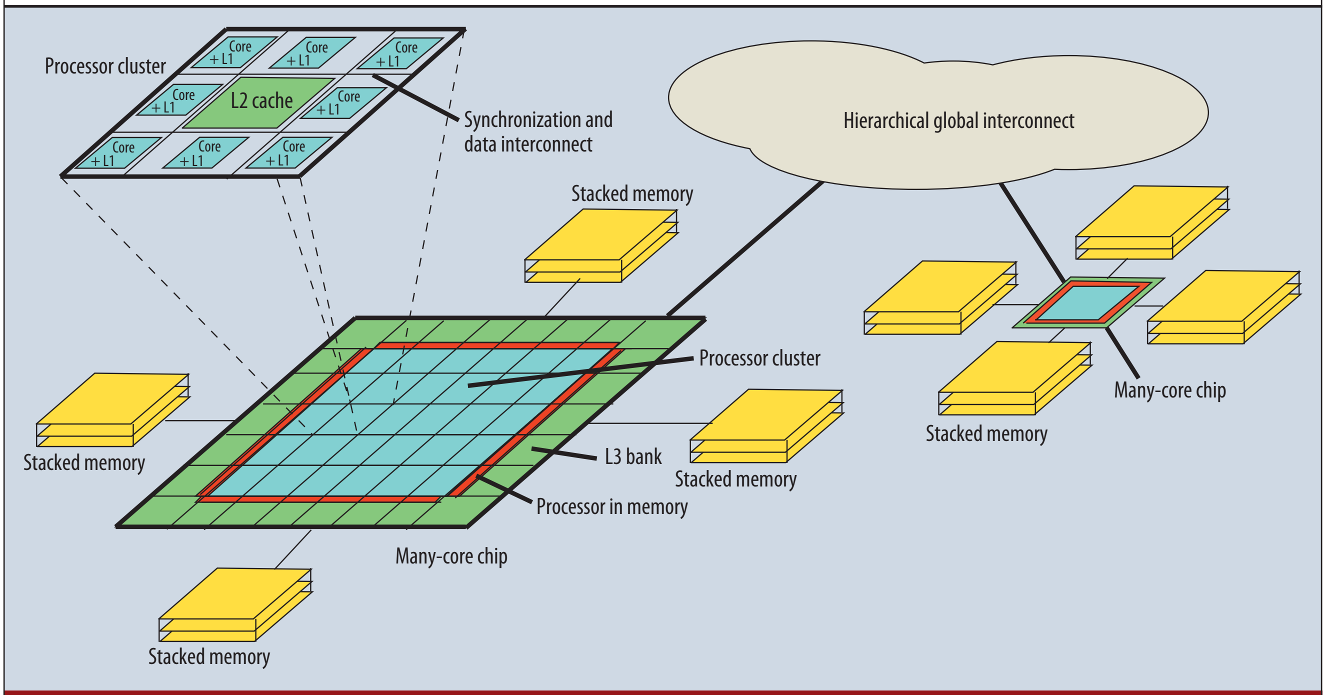


Figure 1. Overview of the Thrifty architecture. The architecture comprises 1,024-core many-core chips.

make the task even more difficult. Indeed, by keeping V_{dd} low, they need several times more concurrency to attain the same performance level. Moreover, power efficiency mandates carefully managing locality and minimizing communication. All of these requirements can easily increase programming complexity.

Programmers of extreme-scale machines must be able to express a high degree of parallelism in a way that does not preclude careful locality management and communication minimization. An appealing approach is to program the machine using a high-level programming model, and then rely on intelligent static and dynamic compilation layers to efficiently map the code to the hardware. The most appropriate high-level model will likely be a data-parallel one, where programmers apply high-level operations to data aggregates.[9] This model can compactly express high degrees of parallelism. With this model, execution appears as computation segments, in which the cores compute largely independently, logically separated by barriers, with some data shuffling between the segments.

A compiler takes the computation assigned to individual elements of the data aggregate and, possibly driven by programmer annotations, breaks it into fine-grain tasks. Then, the compiler maps these tasks to cores, trying to leverage the different locality levels provided by the hardware, such as core cluster, chip, and node. Because the architecture is complex, this technique requires a level of code autotuning or adaptation. Autotuning can be provided through libraries that generate multiple versions of code with different parameters and choose the best one, or through continuous optimization, an approach that relies on a slow adaptation of the program as it runs.

Several architectural features can help support the execution environment described. For example, the machine can provide a single address space to the software. With such a capability, programmers developing irregular codes have a substantially simpler task. Hardware mechanisms that support compiler optimizations and high-level languages are also helpful. For example, such mechanisms could detect dependences within and across threads inexpensively[10] or manage the caches in software. The architecture can also provide features to detect data transfer patterns and eliminate, minimize, or hide data movement. For example, this includes efficient primitives for prefetching, multicast-update of the copies of a datum, and movement of computation to the data's location. It also includes efficient implementations of synchronization primitives—in particular, various forms of barriers.

Finally, autotuning libraries and continuous optimization software will benefit from tight coupling with performance or energy-monitoring hardware structures, such as counters, signatures, and trace buffers. Ideally, these structures should be programmable by the user software. They should also enable a low-latency feedback loop to the application, so program adaptation can be effective.

OUTLINE OF AN EXTREMEffSCALE ARCHITECTURE

Figure 1 gives an overview of the Thrifty extreme-scale architecture concept, which we are developing. The

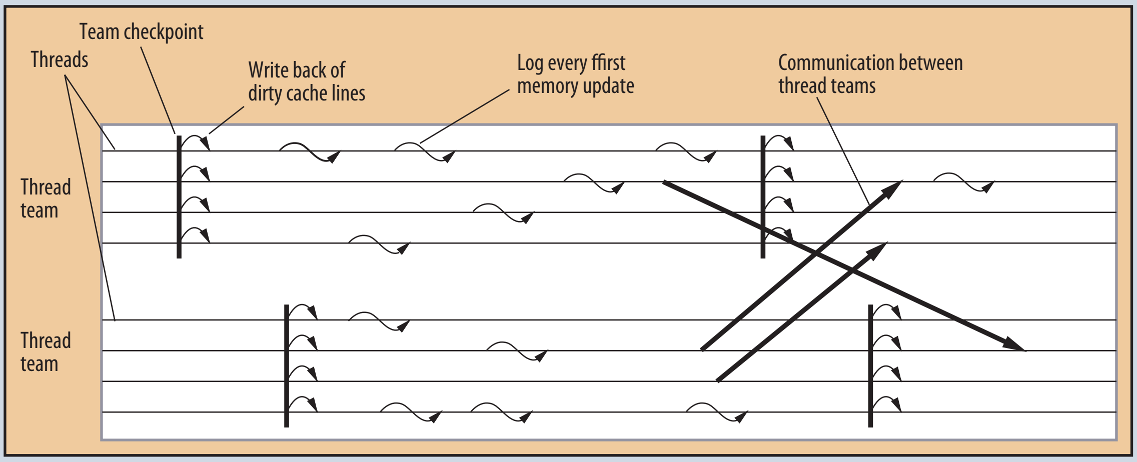

Team checkpoint
Threads
Write back of dirty cache lines
Log every ffirst memory update
Communication between thread teams
Thread team
Thread team

Figure 2. Hierarchical operation of ReVive. Threads are organized in teams, and each team independently follows the ReVive operation. This involves performing regular team checkpoints, in which processors write their registers to memory and caches write their dirty lines to memory. Between checkpoints, a controller logs the value in memory before every first memory update. Communication between teams is explicitly recorded and causes additional checkpoints.

architecture comprises 1,024-core many-core chips, and supports a shared address space. Because Thrifty has no hardware cache coherence, the software must maintain data coherence.

The chips' cores are homogeneous, two-issue, in-order, and single-threaded. They have floating-point support and a reconfigurable vector unit. To enable locality optimizations, groups of eight cores are organized in a cluster, sharing an L2 cache. Each group of four clusters is an independent voltage and frequency domain, because Thrifty relies heavily on voltage and frequency changes to control its power-performance operation. All the clusters share an on-chip L3 cache, which is banked and distributed across the chip. Each L3 bank has a simple PIM engine that performs reductions and other simple memory-intensive operations. The chip has efficient hardware primitives for synchronization and communication (including hierarchical barrier and multicast update). A wide hierarchical network connects the cores.

The machine can connect thousands of these chips hierarchically into boards and cabinets, using a network organized in a fat tree. Memories are organized in 3D stacks and associated with PIM engines. A low-overhead, in-memory incremental checkpointing system based on a hierarchical form of ReVive provides recovery for applications that are explicitly willing to pay for it.[8] The scheme is outlined in Figure 2.

The architecture is programmed with high-level, data-parallel operators on data aggregates. Tasks are mapped to processor clusters. A back-end compiler further divides these tasks into small subtasks, which it then maps to the cores in a cluster for locality.

Substantial advances in architecture and hardware technologies should appear in the next few years. For extreme-scale computing to become a reality, we need to revamp most of the subsystems of current multiprocessors. Many aspects remain wide open, including effective NTV many-core design and operation; highly energy-efficient checkpointing; rearchitecting the memory and disk subsystems for low energy and fewer parts; incorporating high-impact technologies such as nonvolatile memory, optics, and 3D die stacking; and developing cost-effective cooling technologies. A new generation of computer designers will deliver these advances. C

Acknowledgments

I thank Peter Kogge and the writers of the ExaScale report. I also thank Rishi Agarwal, Pranav Garg, David Padua, Maria Garzaran, Brian Greskamp, Ulya Karpuzcu, Shekhar Borkar, David Dunning, Allan Snavely, and William Harrod.

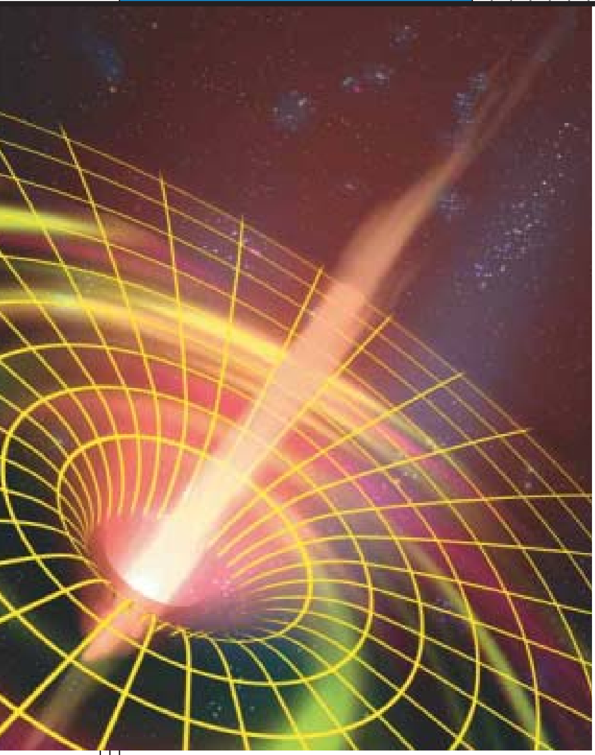
References

1. P. Kogge et al., *ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems*, DARPA Information Processing Techniques Office (IPTO) sponsored study, 2008; www.cse.nd.edu/Reports/2008/TR-2008-13.pdf.
2. E. Grochowski and M. Annavaram, "Energy per Instruction Trends in Intel Microprocessors," *Technology@Intel Magazine*, Mar. 2006, pp. 1-8.
3. R. Dreslinski et al., "Near Threshold Computing: Overcoming Performance Degradation from Aggressive Voltage Scaling," *Proc. Workshop Energy-Efficient Design*, 2009, pp. 44-49.

4. B. Lee et al., "Architecting Phase Change Memory as a Scalable DRAM Alternative," *Proc. Int'l Symp. Computer Architecture* (ISCA 09), ACM Press, 2009, pp. 2-13.
5. R. Ramaswami and K. Sivarajan, *Optical Networks: A Practical Perspective*, Morgan Kaufmann, 2nd ed., 2002.
6. D. Vantrease et al., "Corona: System Implications of Emerging Nanophotonic Technology," *Proc. Int'l Symp. Computer Architecture* (ISCA 08), IEEE Press, 2008, pp. 153-164.
7. J. Shirako et al., "Phasers: a Unified Deadlock-Free Construct for Collective and Point-to-Point Synchronization," *Proc. Int'l Conf. Supercomputing* (SC 08), ACM Press, 2008, pp. 277-288.
8. M. Prvulovic, Z. Zhang, and J. Torrellas, "ReVive: Cost-Effective Architectural Support for Rollback Recovery in Shared-Memory Multiprocessors," *Proc. Int'l Symp. Computer Architecture* (ISCA 02), ACM Press, 2002, pp. 111-122.
9. J. Brodman et al., "New Abstractions for Data Parallel Programming," *Proc. First Usenix Workshop on Hot Topics in Parallelism* (HotPar), Usenix Assoc., 2009; www.usenix.org/event/hotpar09/tech/full_papers/brodman/brodman.pdf.
10. J. Tuck et al., "SoftSig: Software-Exposed Hardware Signatures for Code Analysis and Optimization," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 08), ACM Press, 2008, pp. 145-156.

***Josep Torrellas** is a professor and Willett faculty scholar in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His research interests include computer architectures, parallel computing, hardware and software reliability, and low-power design. Torrellas received a PhD in electrical engineering from Stanford University. He is an IEEE Fellow and a member of the ACM. Contact him at torrellas@cs.uiuc.edu.*

COVER FEATURE



TOFU: A 6D MESH/TORUS INTERCONNECT FOR EXASCALE COMPUTERS

Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu, *Fujitsu*

A new architecture with a six-dimensional mesh/torus topology achieves highly scalable and fault-tolerant interconnection networks for large-scale supercomputers that can exceed 10 petaflops.

Researchers continue to improve high-performance computing systems by increasing the number of processor cores per node and nodes per system. To interconnect tens of thousands of nodes, many HPC systems employ mesh/torus topologies because of their high scalability and low cost/performance ratio.

On a mesh-connected system, topology-aware tuning is important for many applications. The system should be able to allocate a job contiguously and provide a torus topology for an individual job. To achieve high system utilization, a flexible-sized submesh is also important. To meet these needs, we have developed Tofu, an interconnect architecture that features a higher-radix mesh/torus topology (Tofu stands for "torus fusion" or "torus-connected full connection"). A Tofu system can be divided into an arbitrary size of rectangular submeshes just like a block of tofu, and provides a torus topology for each submesh.

TOFU INTERCONNECT ARCHITECTURE

Topology-aware tunability and system utilization are the primary trade-offs for mesh/torus-connected systems. Contiguous job allocation is intended to fulfill users' needs for topology-aware tuning, while noncontiguous job allocation maximizes system utilization. QCDOC (Quantum Chromodynamics on a Chip),[1] Blue Gene,[2] and QPACE (QCD Parallel Computing on the Cell Broadband Engine)[3] systems employ contiguous job allocation, while the Cray XT series[4] employs noncontiguous job allocation.

A contiguous job allocation scheme potentially requires additional hardware like partition switches to provide a consistent view of network topology wherever an application runs on a full system or subsystem. Assuming particular submesh shapes, architectural designers can reduce the number of necessary partition switches. However, a more flexible mechanism is required when assuming various job sizes. Because a multidimensional mesh topology can embed a ring topology, a higher-radix mesh/torus offers one solution. QCDOC employs a 6D torus topology and 12 network links per node.

We designed Tofu to be a 6D mesh similar to QCDOC but with a reduced number of network links, as a link's peak throughput is roughly inversely proportional to the number of network links per node. Tofu reduces links by restricting the length of some additional dimensions to two. A dimension of length two requires only one additional network link and is sufficient to combine with another longer dimension to embed a ring topology.

Tofu has six coordinate axes: *x*, *y*, *z*, *a*, *b*, and *c*. The lengths of the *ac* axes are restricted to two, so each node has a total of 10 links. The length of the *b*-axis is restricted to three instead of two for fault tolerance. Twelve nodes having the same *xyz* coordinates constitute a *node group* and are interconnected by the *abc*-axes. A node group can be considered a unit of job allocation. Figure 1 shows examples of three node groups whose *xyz* coordinates are (0,0,0), (1,0,0), and (2,0,0). The spherical vertices represent nodes and the cylindrical edges *abc*-axes.

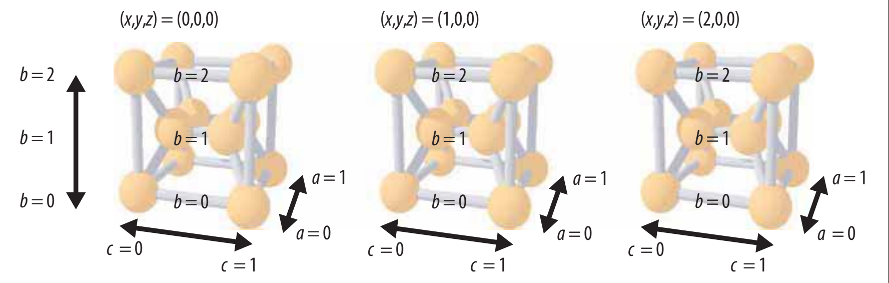


Figure 1. Example Tofu node groups. Twelve nodes (spherical vertices) having the same *xyz* coordinates constitute a node group and are interconnected by the *abc* axes (cylindrical edges).

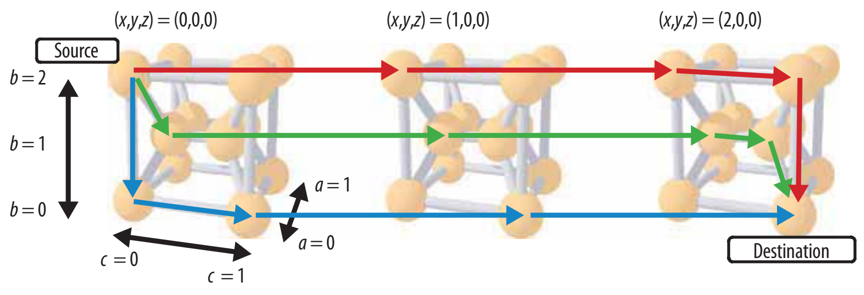


Figure 2. Multipath routing in Tofu. Example of 3 out of 12 paths from the node (0,0,0,0,2,0) to the node (2,0,0,0,0,1).

Multipath routing function

A routing algorithm that detours a unit under maintenance is necessary to enable hot-swap maintenance. We therefore developed an algorithm for Tofu that divides packet routing into three phases. First, a packet traverses the *abc*-axes to select a path at a source node group. It then moves from the source node group to the destination node group along the *xyz*-axes. Finally, the packet travels along the *abc*-axes again to a destination node at the destination node group. Although the routing path in each phase is minimal, the total routing path is not because the Tofu routing algorithm can take *abc*-axes twice. Tofu routing can relay through an arbitrary node in a source node group, so there are a total of 12 paths for an arbitrary destination.

Figure 2 shows examples of multiple paths. The arrows represent three routes from node (0,0,0,0,2,0) to node (2,0,0,0,0,1).

Torus mapping and fault tolerance

For many applications, topology-aware tuning is recommended to achieve the scalability of tens of thousands of nodes. Therefore, a Tofu system offers every job a network topology view of a 3D torus. It embeds a 3D torus view into a 6D mesh space by corresponding two axes of the 6D mesh to one application-view axis. The system sequentially allocates the coordinates of each application-view axis to form a loop in the plane composed of the 6D mesh's two axes.

Unfortunately, faults inevitably occur in extremely large systems, despite efforts to reduce the failure rate. Therefore, it is important to build a fault-tolerant system that isolates and replaces faulty components without stopping system operation. To achieve higher system utilization while tolerating faults and failures, Tofu has the ability to offer a 3D torus view on a submesh with a faulty node.

Figure 3a shows an example of mapping coordinates of an application-view axis with a length of 9 on a nonfaulty submesh. In this case, the system chooses the *x*-axis and *b*-axis of the 6D mesh to be an allocation plane for the application-view axis. It maps coordinates 0 on node (0,0,0,0,0,0) and 5 on node (2,0,0,0,0,0).

Figure 3b shows an example of fault-tolerant mapping. Node (1,0,0,0,1,1) is down, and the length of the application-view axis is reduced to 8. The system removes the four nodes (1,0,0,0,1,0), (1,0,0,0,1,1), (1,0,0,1,1,0), and (1,0,0,1,1,1) from the coordinate allocation. This mechanism is also effective for the hot-swap maintenance scheme.

High system utilization

High system utilization is required for cost-effectiveness. Job allocation scheduling is especially effective and requires few additional hardware costs. For example, the RIKEN Super Combined Cluster (RSCC) system achieved 78-79 percent utilization during 2005-2006, but there was still room for improvement. Consequently, RIKEN and Fujitsu Laboratories collaboratively introduced the

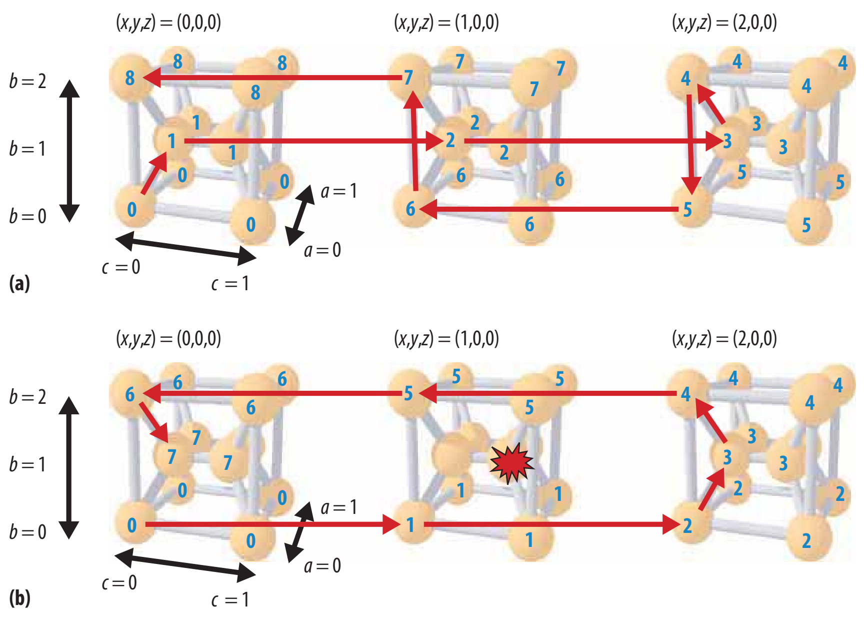


Figure 3. Fault-tolerant mapping in Tofu: (a) example of mapping on a nonfaulty submesh and (b) example of fault-tolerant mapping.

Meta Job Scheduler, job allocation scheduling software that performs backfill scheduling, which improved RSCC's system utilization to 92-93 percent during the following two years.[5] This was roughly equivalent to a 17 percent hardware enhancement.

For mesh-connected supercomputers, fragmentation is the main obstacle to high system utilization. Working from the premise that flexible submesh shapes would help fill small fragments of free nodes, we utilized dimensional combinations to map the same 3D torus view. In Tofu, the *b*-axis has a different length than the *ac*-axes; this asymmetricity increases a 6D mesh shape's variation to achieve the specified 3D torus view.

Consider, for example, all the possible mesh/torus shapes that can map a 12 × 12 × 6 torus view for a 3D torus-connected system and a Tofu system:

3D torus-connected system: 12 × 12 × 6, 12 × 6 × 12, 6 × 12 × 12
Tofu system: 6 × 6 × 2 × 2 × 3 × 2, 6 × 2 × 6 × 2 × 3 × 2, 2 × 6 × 6 × 2 × 3 × 2, 6 × 4 × 3 × 2 × 3 × 2, 6 × 3 × 4 × 2 × 3 × 2, 4 × 6 × 3 × 2 × 3 × 2, 4 × 3 × 6 × 2 × 3 × 2, 3 × 6 × 4 × 2 × 3 × 2, 3 × 4 × 6 × 2 × 3 × 2

There are three possible shapes for the 3D torus-connected system and nine for the Tofu system. A larger number of allocation candidate mesh shapes would be expected to improve system utilization.

We simulated system utilization using a real workload log from the Parallel Workloads Archive (www.cs.huji.ac.il/labs/parallel/workload). We scaled a job's number of nodes, with properties of candidate mesh shapes corresponding to the number of nodes, and assumed a large job had redundancy nodes for fault avoidance. The simulation results with backfill job scheduling showed that system utilization of a single candidate job shape was about 70 percent and that of multiple candidates improved to about 80 percent. Based on these results, we are developing a new job allocation scheduler which supports job allocation from multiple job shape candidates and utilizes enhanced scheduling algorithms of the Meta Job Scheduler.

ADDITIONAL TOFU FEATURES

The Tofu interconnect architecture has several other features.

Throughput and packet transfer

Tofu has high-throughput links with 10 gigabytes per second of fully bidirectional bandwidth for each. The link throughput is roughly derived from the off-chip bandwidth and network degree 10. We implemented 100 GBps of the off-chip bandwidth for each node to feed enough data to a massive array of 128-Gflops processors.[6]

Packet length is variable to minimize packet header overhead and improve effective bandwidth. The minimum packet length is 32 bytes and the maximum is 2,048 bytes, including the header and cyclic redundancy check (CRC). Packets are transferred via virtual cut-through,[7] which achieves low latency by buffering packets only when the destination link is blocked.

Each link has four 8-Kbyte receive buffers and an 8-Kbyte retransmission buffer. The receive buffers correspond to four virtual channels and are used while the destination link is blocked. The retransmission buffer is used for link-level retransmission that recovers CRC errors.

I/O communication routing

Preparing a dedicated I/O interconnect ensures I/O bandwidth. To achieve a beneficial cost/performance tradeoff, computation and I/O communication should share bandwidth—this is one of the reasons why computation and I/O phases are often separated in a single application. However, even if computation and I/O communications share bandwidth, I/O communication should not pass the area in which other jobs communicate for computation. To minimize the path on which I/O communication crosses other jobs, we arranged the coordinates of the I/O node and routing orders.

Figure 4 shows a desirable I/O communication path. A partial network of z coordinates equal to 0 forms the I/O network, and the remaining network of z coordinates larger than 0 defines the computation network. Rather than have I/O communication pass the x-axis and y-axis on the computation network, it is preferable that it only pass the z-axis. We designed two groups of virtual channels with different routing orders, ensuring that a desirable virtual channel could transmit a packet dedicated for I/O communication. Computation nodes transmit packets by virtual channels that have the z-axis as the first order, and I/O nodes transmit packets by virtual channels that have the z-axis as the last order.

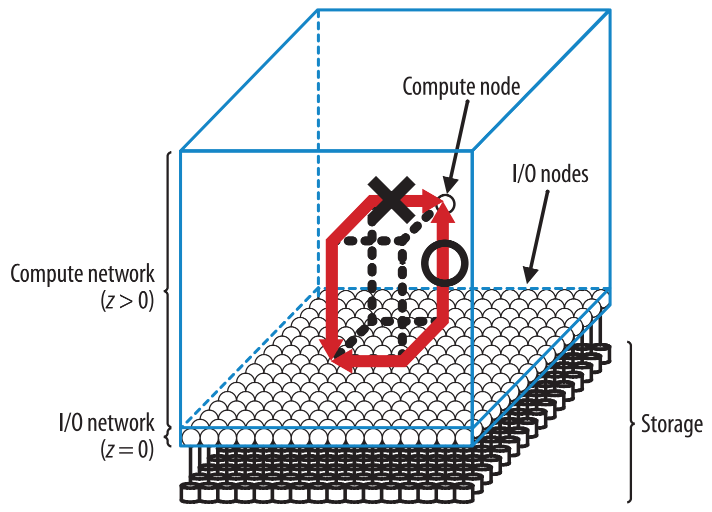


Figure 4. Desirable I/O communication path. A partial network of *z* coordinates equal to 0 forms the I/O network, and the remaining network of *z* coordinates larger than 0 deffines the computation network. Computation nodes transmit packets by virtual channels that have the *z*-axis as the ffirst order, and I/O nodes transmit packets by virtual channels that have the *z*-axis as the last order.

Multiple communications engines

Some applications can overlap communications to multiple nodes simultaneously. Parallel execution of communication primitives by multiple communication engines may improve performance, especially of communication-intensive and nearest-neighbor communicating applications like Lattice QCD. We therefore implemented four communication engines in each node in Tofu. As Figure 5 shows, each engine can transmit and receive packets simultaneously, transmit packets to any direction, and receive packets from any direction. Four communication engines can also improve the throughput of point-to-point communication by simultaneously using multiple paths between nodes.

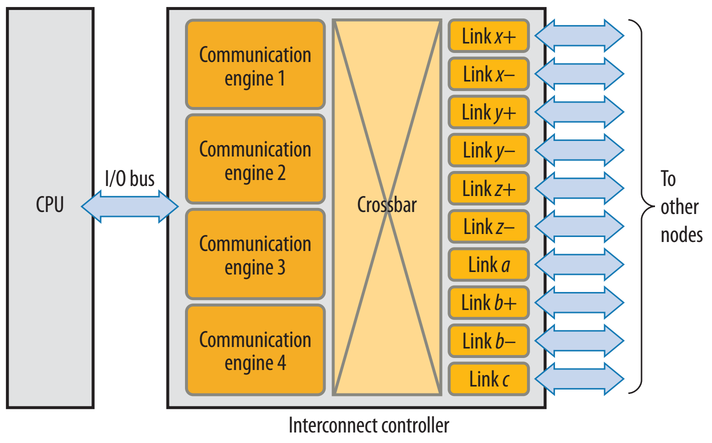


Figure 5. Block diagram of a node with four communication engines. Each engine can transmit and receive packets simultaneously, transmit packets to any direction, and receive packets from any direction.

Integrated collective function

The collective communication function's communication time often suppresses a highly parallel application's performance scalability. Collective communications are generally implemented by combining point-to-point communications, whose frequency increases with the number of nodes engaging in the collective communication function. Each point-to-point communication involves CPU processing time that often gets unexpectedly longer, and this variability greatly affects the completion time of a highly parallel collective communication function.

We addressed this problem by designing in the hardwired logic an integrated collective function, which processes frequently used collective communication functions (barrier and reduce) without any CPU intervention. Collective communication functions are operated by chained barrier gates. A barrier gate waits for and transmits one barrier packet. Figure 6 shows an example of a chain of four barrier gates. This chain of barrier gates repeats transmission and reception of a barrier packet four times to synchronize 16 nodes.

A chain of barrier gates can also perform the all-reduce operations of a 64-bit integer or a double-precision floating-point number. It converts a double-precision floating-point number into two 160-bit floating-point data to obtain the same result at all nodes with a single barrier

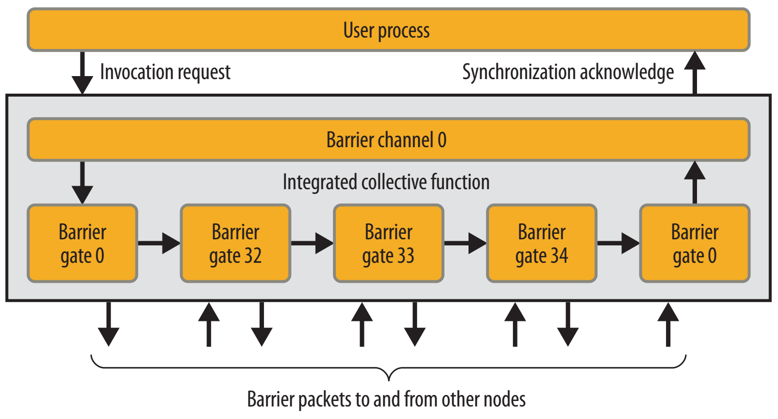


Figure 6. Example of four barrier gates operating and connected to a barrier channel. This chain of barrier gates repeats transmission and reception of a barrier packet four times to synchronize 16 nodes.

synchronization sequence. This floating-point calculation method was developed by the Petascale System Interconnect project sponsored by Japan's Ministry of Education, Culture, Sports, Science and Technology, and was previously implemented in the Highly Functional Switch of Fujitsu's FX1 supercomputer.[8]

High-performance computing systems are becoming denser, leaner, and more integrated, making it increasingly challenging to manage and repair failures. By effectively isolating components that require maintenance and repair, the Tofu interconnect architecture promises to be a fundamental technology for future exascale systems. C

Acknowledgments

The authors thank Yuji Oinaga and the many Tofu project members for their valuable suggestions and efforts.

References

1. P. Boyle et al., "The QCDOC Project," *Nuclear Physics B—Proc. Supplements*, Mar. 2005, pp. 169-175.
2. The BlueGene/L Team, "An Overview of the BlueGene/L Supercomputer," *Proc. 2002 ACM/IEEE Conf. Supercomputing* (SC 02), IEEE CS Press, 2002; https://asc.llnl.gov/computing_resources/bluegenel/pdf/sc2002-pap207.pdf.
3. G. Goldrian et al., "QPACE: Quantum Chromodynamics Parallel Computing on the Cell Broadband Engine," *IEEE Computing in Science and Eng.*, Nov./Dec. 2008, pp. 46-54.
4. W.J. Camp and J.L. Tomkins, "Thor's Hammer: The First Version of the Red Storm MPP Architecture," *Proc. 2002 ACM/IEEE Conf. Supercomputing* (SC 02), IEEE CS Press, 2002.
5. T. Shigetani, "RIKEN Super Combined Cluster Operations Report" (in Japanese), Mar. 2009; http://accc.riken.jp/HPC/Symposium/2008/shige.pdf.
6. T. Maruyama, "SPARC64 VIIIfx: Fujitsu's New Generation Octo Core Processor for PETA Scale Computing," presentation, Hot Chips 21, 2009.
7. P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, Jan. 1979, pp. 267-286.
8. T. Abe, T. Inari, and K. Seki, "JAXA Supercomputer Systems with Fujitsu FX1 as Core Computer," *Fujitsu Scientific & Technical J.*, Oct. 2008, pp. 426-434.

***Yuichiro Ajima** is a system architect in the Next-Generation Technical Computing unit at Fujitsu Limited. His research interests are in technical computing system architectures. He received a PhD in information engineering from the University of Tokyo. Ajima is a member of the Information Processing Society of Japan (IPSJ). Contact him at aji@jp.fujitsu.com.*

***Shinji Sumimoto** is a senior architect in the Next-Generation Technical Computing unit at Fujitsu Limited, and a research fellow at Fujitsu Laboratories Limited. His research interests are in high-performance cluster system architectures, especially high-performance communication, and large-scale computing architectures. Sumimoto received a PhD in electrical engineering from Keio University. He is a member of IPSJ. Contact him at s-sumi@labs.fujitsu.com.*

***Toshiyuki Shimizu** is a director in the Next-Generation Technical Computing unit at Fujitsu Limited. His research interests include high-performance computer system architectures, particularly interconnect architectures for highly scalable systems. Shimizu is a member of IPSJ and the Institute of Electronics, Information, and Communication Engineers. Contact him at t.shimizu@jp.fujitsu.com.*

CALL FOR PARTICIPATION

CTS 2010

Chicago, Illinois, USA

The 2010 International Symposium on Collaborative Technologies and Systems

May 17 – 21, 2010
The Westin Lombard Yorktown Center Hotel
Chicago, Illinois, USA

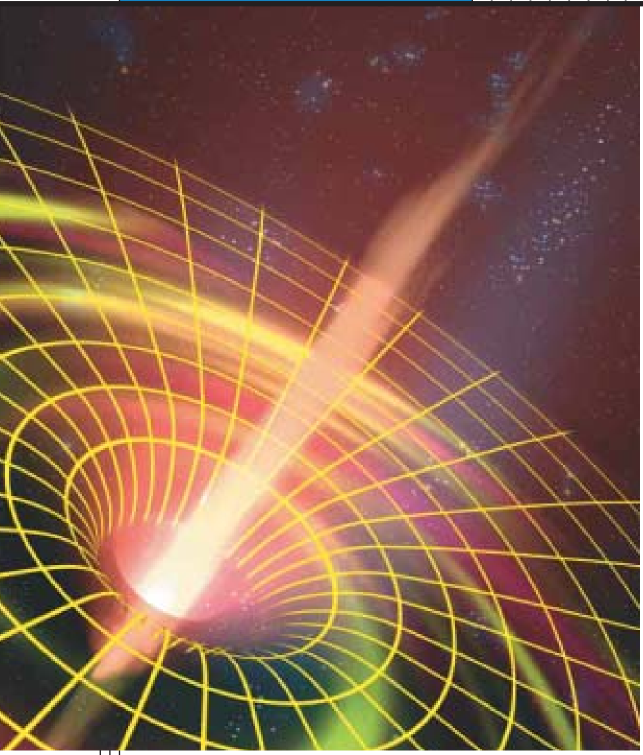
Important Dates:

Paper Submission Deadline	**December 20, 2009**
Workshop/Special Session Proposal Deadline	**November 15, 2009**
Tutorial/Demo/Panel Proposal Deadline	**January 8, 2010**
Notification of Acceptance	**February 1, 2010**
Final Papers Due	**March 1, 2010**

For more information, visit the CTS 2010 web site at:
http://cisedu.us/cis/cts/10/main/callForPapers.jsp

In cooperation with the ACM, IEEE, IFIP

COVER FEATURE



USING PERFORMANCE MODELING TO DESIGN LARGE-SCALE SYSTEMS

Kevin J. Barker, Kei Davis, Adolfy Hoisie, Darren J. Kerbyson, Michael Lang, Scott Pakin, and José Carlos Sancho, *Los Alamos National Laboratory, New Mexico*

A methodology for accurately modeling large applications explores the performance of ultrascale systems at different stages in their life cycle, from early design through production use.

Increasingly, scientific discovery has relied on computer simulation, fueling an insatiable demand for ever-faster supercomputers. Such computers can deliver results sooner, often with higher fidelity. However, as supercomputers get faster, they also get more complex. While a high-end desktop computer might contain 16 processor cores, a single memory address space, and a simple, internal, all-to-all network connecting the cores, today's fastest supercomputers—the Roadrunner system at Los Alamos National Laboratory,[1] the Jaguar system at Oak Ridge National Laboratory, and Jugene at Forschungszentrum Juelich—contain from tens to hundreds of thousands of cores, as many separate address spaces, and multiple interconnection networks with different features and performance characteristics.

All of these components interact in complex ways because of their hierarchical organization and contention for limited system resources. Thus, adjacent processor cores might coordinate their activities faster than distant cores, but distant cores need not compete with each other for access to memory. Roadrunner, Jaguar, and Jugene are all petascale systems, which can process 1×10^{15} floating-point operations per second (1 petaflop/s). Already, the high-performance computing community is investigating the challenges of exascale systems, which, while possibly only six years away, will have 1,000 times the peak performance of today's systems (1 exaflop per second) and a corresponding increase in complexity.

PERFORMANCE MODELING

Given the complexity of supercomputing architectures, it is hard to predict how fast an application running on today's petascale (or smaller) supercomputers will run on an exascale supercomputer. A thousandfold increase in peak performance rarely translates to an identical increase in application speed. Managing complexity necessarily exacts a performance toll. Accurately extrapolating performance from one system to another merely by gut feeling and guesswork is virtually impossible. However, being able to predict performance is an important capability because it helps system architects determine how to address the various design tradeoffs they face. Furthermore, it helps computational scientists decide which supercomputer to port their applications to, as porting to a state-of-the-art supercomputer can be a time-consuming process.

Published by the IEEE Computer Society 0018-9162/09/$26.00 © 2009 IEEE

A common approach to predicting performance is to use a machine simulator, a program that runs on an existing system but mimics a target system. While simulation works well for quick-running applications on modest numbers of processors, this approach struggles with predicting performance of long-running applications on extreme-scale systems. In this case a simulator needs either to run on a system of a size comparable to the target system, run for an unacceptable length of time, or forgo simulation accuracy.

We present an alternative to system simulation: *performance modeling*. Just as a model of a physical process provides a set of analytical formulas that describe the salient features of a complex natural phenomenon, a performance model offers a set of analytical formulas that describe the salient performance characteristics of a complex artificial phenomenon: a scientific application running on a current or future supercomputer. Furthermore, like its natural-world counterpart, a performance model supports not only prediction but also insight and interpretation.

At Los Alamos we have been using performance modeling almost daily over the past decade to explore the performance of numerous systems at different stages in their life cycle, from early design through production use. Figure 1 shows how widely applicable performance modeling is and how performance models are invaluable tools for performance analysis.

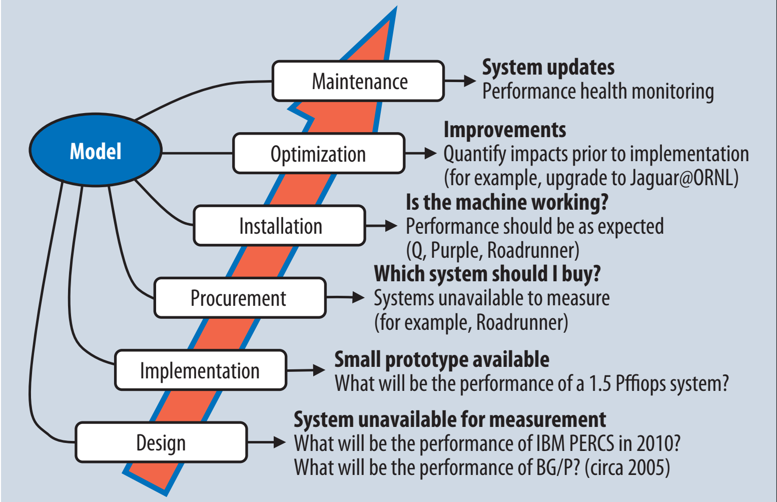

Model
Maintenance
System updates
Performance health monitoring
Optimization
Improvements
Quantify impacts prior to implementation (for example, upgrade to Jaguar@ORNL)
Installation
Is the machine working?
Performance should be as expected (Q, Purple, Roadrunner)
Procurement
Which system should I buy?
Systems unavailable to measure (for example, Roadrunner)
Implementation
Small prototype available
What will be the performance of a 1.5 Pflops system?
Design
System unavailable for measurement
What will be the performance of IBM PERCS in 2010?
What will be the performance of BG/P? (circa 2005)

Figure 1. Widely applicable throughout a system's life cycle, performance modeling offiers invaluable tools for analyzing performance.

Early system design. Many models have been used to explore the performance of machines still on the drawing board. For example, we utilized our performance models to guide the design of the innovative rich-interconnection network that will be used in the forthcoming IBM Blue Waters system at the National Center for Supercomputing Applications. In addition, we also used performance models to examine hybrid accelerated systems prior to Roadrunner's procurement.[1]

Implementation. When a small, perhaps prototype, system becomes available, we can use modeling to predict expected performance for larger-scale systems. Often, only a single node is available during early access, as was the case with modeling the full-scale Roadrunner and 270 compute-node Tri-Lab Computing Clusters now deployed at Los Alamos, Sandia, and Lawrence Livermore National Laboratories.

Procurement. We also have used performance modeling to compare competing vendor systems during procurement. This was first done in 2003 for the procurement of the ASC workhorse system Purple, deployed at Lawrence Livermore. Modeling was subsequently used in the procurement of Roadrunner and several other laboratory HPC systems. Measuring performance is typically not possible, either because the system scale is larger than anything currently available, or the system uses yet-to-become-available next-generation components.

Installation. Performance modeling helps to predict an expected level of performance and thereby verify that a system is correctly installed and configured. We first applied this process to the Accelerated Strategic Computing Initiative (ASCI) "Q" system, installed at Los Alamos in 2002. We used performance models to correctly identify a performance issue with the initial installation[2]: A factor of two performance loss attributed to significant operating system noise (or jitter) that, after optimization, resulted in achieved application performance being very close to the initial model predictions. We have subsequently applied this process to all recent systems at Los Alamos, and to ASC Purple at Lawrence Livermore.

Optimization. We also have used performance modeling to quantify systems prior to performing optimizations or upgrades. For example, the Jaguar machine at Oak Ridge National Laboratory has been through several upgrade cycles, including migrating the processors from dual-core to quad-core and also from single-processor to dual-processor nodes. Modeling provided an expected level of application performance in advance, which was used as a verification mechanism for a successful system upgrade. In addition, we have used models to guide routing optimizations for InfiniBand networks.

Maintenance. Performance modeling can be incorporated into a system's everyday operation and maintenance. We are currently exploring a multitude of modeling applications to assist with identifying potential performance problems during normal production use. By using a per-

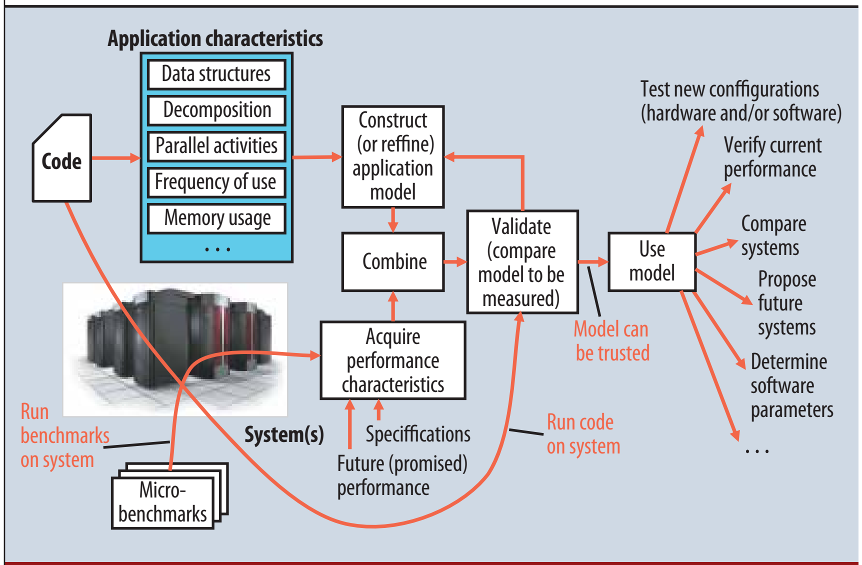

Application characteristics
Data structures
Decomposition
Parallel activities
Frequency of use
Memory usage
. . .
Code
Construct (or reffine) application model
Combine
Validate (compare model to be measured)
Use model
Acquire performance characteristics
Model can be trusted
Test new confiigurations (hardware and/or software)
Verify current performance
Compare systems
Propose future systems
Determine software parameters
. . .
Run benchmarks on system
System(s)
Speciffications
Future (promised) performance
Micro-benchmarks
Run code on system

Figure 2. Development of a performance model. An application-centric approach starts with the application code plus a representative input deck. Unlike other forms of performance prediction, this approach treats the application as a white box to be examined and understood.

formance model to determine expected performance, we can report in real time the system's "performance health" as its deviation from that expectation.

We have also used performance modeling to compare the performance of many systems, some in existence and some not, without the need for lengthy benchmarking activities. In the past, this has included a comparison of the Japanese Earth Simulator with other large-scale systems,[3] as well as comparing the use of different accelerator hardware in Roadrunner prior to procurement.

In addition, we have used performance modeling to compare code design alternatives—in terms of how to optimize a code's underlying data decomposition and mapping to a particular system to improve performance. Our design study uses performance modeling to explore the design space of future potential systems. Using validated performance models, we can quantify the impact on application performance given various improvements to system components.

IMPORTANT FACTORS IN PERFORMANCE MODELING

Figure 2 shows our approach to developing a performance model. Because our models are application-centric we begin with the application code plus a representative input deck. Unlike other forms of performance prediction, performance modeling treats the application as a white box to be examined and understood rather than a black box that simply spits out a runtime.

The first step identifies the application characteristics that contribute most to overall runtime. These typically include information about how the global data is distributed across processes, how much memory is accessed per data unit, how much communication is performed per data unit, and which processes communicate with which others. We are concerned only with first-order effects on performance; it does not matter what happens during initialization, for example.

A variety of techniques can be used to identify application characteristics. We start by profiling the code to see where the bulk of the processing time is spent and to understand the communication patterns, such as neighbor relationships, message counts, and message sizes. By comparing profiles across different problem sizes and process counts, we can form hypotheses about how communication and computation vary with program inputs and machine sizes.

Next, we manually examine the code to test our hypotheses and refine them accordingly. The goal is to produce a performance model that expresses runtime in terms of both application characteristics (for example, message sizes and counts as a function of the number of particles in a mesh) and system characteristics (such as the time needed to send a message as a function of message size).

Then, we fill in values for the performance model's system parameters. We acquire primitive performance characteristics from the target system in one of two ways. If access to the system is possible, we run a suite of microbenchmarks to gather primitive performance information. If not, as in the case of a system that has not yet been built, we rely upon system specifications or reasonable extrapolations based on existing hardware. In either case, an exciting prospect is that a range of values can be used to explore the application's sensitivity to the various system parameters.

We now have enough information to map the performance model onto the target system. For example, if we know the size of an application's main data structure, how that data structure is distributed among parallel processes, and how pieces of the data structure are communicated among processes, we can calculate the amount of data transferred in each message and the total number of messages that must be transmitted. If we measure the time needed to transmit a single message on a target system, computing the total time the application will spend communicating on that system becomes straightforward.

Not unexpectedly, the next step validates the predicted times by running the application on the target system using as many problem sizes and process counts as feasible.

Inaccuracies can occur from oversimplifications in the model: Perhaps an application characteristic that seemed unlikely to impact performance and was therefore excluded from the model in fact must be included.

As the model is validated on increasing numbers of input parameters, system architectures, and process counts, greater levels of trust can be placed in the model's accuracy. A trusted performance model can be used for many purposes:

- testing hardware or software upgrades to ensure that they deliver the expected increase in performance,
- examining a new system's performance to verify that it runs as fast as possible,
- comparing proposed systems to determine which will run the application faster,
- exploring the architectural design space to specify an optimal system under a set of constraints, and
- determining application input parameters, data structures, or data-decomposition techniques that can be expected to improve performance.

In short, a validated performance model is useful for application developers, users, those in charge of procuring new systems, and system architects.

SIMPLE PERFORMANCE MODEL EXAMPLE

Performance modeling normally studies the workings of large, complex, scientific applications. We choose as our example a generic five-point stencil, as might be used for solving a partial differential equation. Although a five-point stencil is not an "application" in its own right, it could form a piece of a larger application. For example, it may be used in the heat-transfer step of a climate-modeling application that also includes routines for simulating salinity, evaporation, radiation, and cloud cover. A complete performance model would need to include several of these components. In a parallel implementation of a five-point stencil, each process manages a rectangular piece of a large array and exchanges its subarray's boundary rows and columns, called ghost cells, with its neighbor process. Then, for each element in the array, it performs some computation, generally expressed in the following form:

$$A(i, j) \leftarrow (A(i, j) + A(i-1, j) + A(i+1, j) + A(i, j-1) + A(i, j+1)) / 5$$

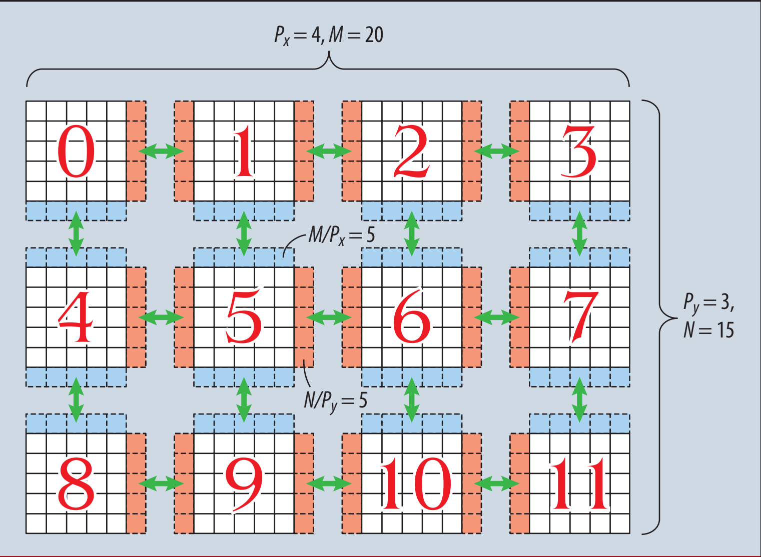


Figure 3. Generic five-point stencil application. Processes are numbered 0–11. White boxes represent the cells in a process's local fragment of the global array. Shaded boxes indicate ghost cells, and arrows indicate interprocess communication from one process's local cells to another process's ghost cells.

Assume the stencil runs on an $M \times N$ array of doublewords (that is, 8-byte elements). Initially, we assume that blocking operations are used for all communication, with only one message sent or received at a time. Figure 3 shows a 20×15 array and the stencil communication and ghost zones used when this array is decomposed onto a 4×3 array of processes.

Given that problem specification, what do we need to know to predict how long our stencil application will run on a given system? Clearly, we need to know the values of M and N. We also need to know the number of parallel processes, P, and how the $M \times N$ array is distributed across those P processes. That is, given that the P processes will be laid out into a $P_x \times P_y$ arrangement, we need to know the values of P_x and P_y. On the system side, we need to know how long it takes to transmit a message of a given size ($T_{\text{msg}}(k)$) and how long it takes for the CPU to compute a value for a single array element, T_{elt}.

To combine those individual values into a performance model, we take a top-down approach, starting with the "fundamental equation of modeling":

$$T_{\text{total}} = T_{\text{comp}} + T_{\text{comm}} - T_{\text{overlap}}$$

That is, we separately model computation time, communication time, and the overlap between them, then combine these into an expression of total execution time. We model the time for a single iteration because the stencil application behaves identically with each iteration.

In our stencil example, T_{comp} is the time per array element multiplied by the number of array elements. The number of elements is determined by the size of a process's subarray, namely MN/P. The total time to process a subarray is therefore

$$T_{comp} = T_{elt}\, MN/P$$

per iteration. Because all processes compute in parallel, T_{comp} does not need to include an additional factor of P; the time for one process to compute is the time for all processes to compute. To model T_{comm} we need to know the maximum number of neighbors a process has. Because all processes communicate concurrently, the process performing the most communication determines T_{comm}.

The methodology for quasi-analytically modeling application performance applies just as well to actual multimillion-line applications.

Let's reason through some examples. In a sequential run, no process has any neighbors. If the processes are laid out in a horizontal line, the middle processes have the maximum number of neighbors, two: one to the left and one to the right. In an array of processes with at least three processes in each dimension, at least one process will have four neighbors, one each to the north, south, east, and west. By defining $N_{max}(p) = min\,(2, p - 1)$ we can more formally express the maximum number of neighbors in a 2D process array as $N_{max}(P_x)+N_{max}(P_y)$. Messages sent east or west are $8N/P_y$ bytes long (assuming 8-byte double words), while messages sent north or south are $8M/P_x$ bytes long. Because we assume blocking communication, all message times contribute to T_{comm}, implying that

$$T_{comm} = T_{msg}\,(8N/P_y)N_{max}\,(P_x) + T_{msg}\,(8M/P_x)N_{max}(P_y)$$

per iteration. Because there is no overlap of communication and computation in our stencil application,

$$T_{overlap} = 0.$$

We now have a complete performance model. Using only a couple of empirically or otherwise determined primitive-operation costs ($T_{msg}(k)$ and T_{elt}), and knowing a few application parameters (M, N, P_x, and P_y), we should in theory be able to predict the performance of our stencil code (T_{total}) all the way out to ultrascale systems. In practice, the model must be enhanced to take into account the additional complexity of a large-scale system. For example, T_{elt} might need to be made cognizant of vector or streaming operations (or even fused multiply-add instructions) and deep memory hierarchies; T_{msg} might need to take into account hierarchical communication costs—communication time within a processor socket, compute node, or network switch, and across network switches—as well as performance penalties due to network contention at each level of the communication hierarchy.

Nevertheless, we now have a straightforward methodology for quasi-analytically modeling application performance. This same methodology applies just as well to actual multimillion-line applications.

MODELING A LARGE-SCALE BLUE GENE SYSTEM

Blue Gene systems are some of the highest-performing machines available today, ranked in terms of peak performance on the Top 500 list of the world's fastest supercomputers (www.top500.org). Relative to other supercomputers, they provide as their salient characteristic enormous numbers of modest-speed processors. It is Blue Gene's massive parallelism that provides the potential for very high performance. We first consider the IBM Blue Gene architecture's latest incarnation: the Blue Gene/P.[4]

This system, used to validate our performance models, consists of 36,864 compute nodes. Sited at Lawrence Livermore National Laboratory, it arranges the compute nodes in a 72 × 32 × 16 3D torus. Each compute node contains four PowerPC 450 processor cores running at a clock rate of 850 MHz. Each core can issue four flops per cycle, for a peak performance of 3.4 gigaflops per second. The system's peak performance, totaled across all 147,456 processor cores, is a little more than 500 teraflops per second.

Several interconnection networks connect the nodes of a Blue Gene system. The main network is a 3D torus used for routine data communication among processors. This network has a peak bandwidth of 450 Mbytes in each of six directions, although software overheads limit the achievable performance to only 375 Mbytes per second. Communication latency between compute nodes depends on the number of hops a message must travel through the torus. We measured a minimum of 3 μs between adjacent nodes and a maximum of 6 μs between distant nodes. Additional networks support global synchronizations, global collective operations, and control functionality.[4]

Model validation

Our modeling approach is exemplified using three applications of interest to Los Alamos. These represent large-scale production applications used on many of the largest production supercomputers today.

SAGE. SAIC's Adaptive Grid Eulerian (SAGE) code is a multidimensional, multimaterial, Eulerian hydrodynamics application that utilizes adaptive mesh refinement.[5] SAGE is used with a wide variety of scientific and engineering problems, including water shock, energy coupling, cratering and ground shock, explosively generated air blast, and hydrodynamic instability problems. Processing consists of three steps repeated many times in each iteration: one or more data gather operations to obtain a copy of remote neighbor boundary data, computation on each of the local cells, and a scatter operation to update boundary conditions on remote processors. The interprocessor communication volume can be large in SAGE and also requires a large number of collective communications.[5]

Sweep3D. This time-independent deterministic particle transport kernel represents the core of a widely used method for solving the Boltzmann transport equation. Each cell in the grid has a data dependency on three upstream cells—computation cannot proceed until the results of computation on these upstream cells completes. Results from processing the cell are passed to three downstream cells. Computation forms a set of wavefronts that originate at each corner of the three-dimensional grid and preserve the data dependencies. The wavefront operation is synonymous with a pipeline—which must be filled and emptied—and can lead to low parallel utilization at scale. Interprocessor communications are fine-grained with small payloads.[6]

VPIC. A relativistic three-dimensional particle-in-cell code self-consistently evolves a kinetic plasma. VPIC (Vector Particle-In-Cell) simulations of laser-plasma interactions have been conducted at extreme fidelity, modeling up to 10^{12} particles on 136×106 voxels.[7] While particle-in-cell codes typically require more data movement per computation unit than other methods—such as dense matrix calculations, N-body computations, and Monte Carlo simulations—VPIC implements a novel method to reduce the volume of interprocessor communication required during simulation. Because of this, performance for the VPIC code is typically compute-bound, meaning VPIC is rather insensitive to message-passing performance.

Each application is most often run in a weak-scaling mode: The problem per processor remains constant as the system size scales, hence larger systems achieve higher fidelity simulations. This is typical of memory capacity bound codes, in which all available memory is used in the simulations. Note also that the problem size per processor core is inversely proportional to the number of cores in a processor.

Figure 4 shows the measured and modeled performance of SAGE, Sweep3D, and VPIC on Blue Gene/P when using between 1 and 32,768 compute nodes. In all cases, the time per iteration for each application appears as a function of node count. The subgrid size per node is fixed, which implies that perfect scaling would appear as a horizontal line at the single-node iteration time. Such performance is seldom achievable in practice due to both application and system characteristics.

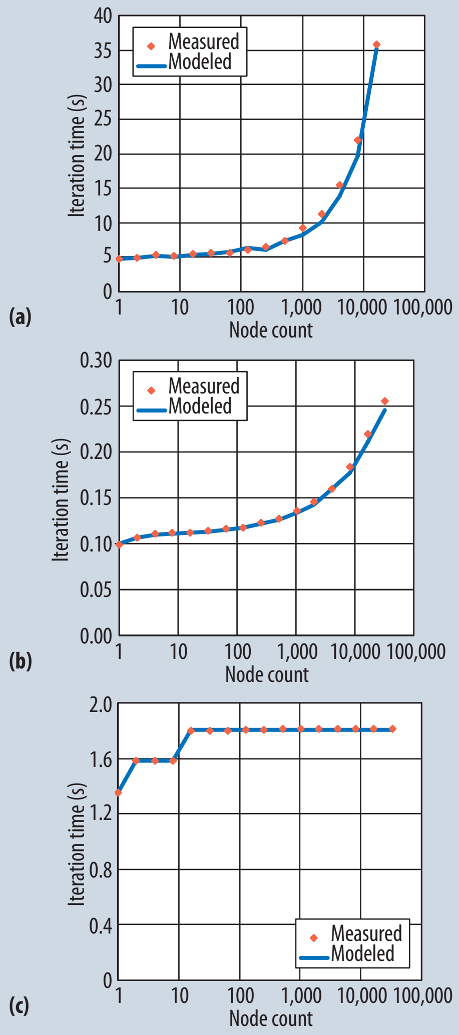


Figure 4. Validation of application performance models on a Blue Gene/P system. Graphs depict the measured and modeled performance of (a) SAGE, (b) Sweep3D, and (c) VPIC on Blue Gene/P when using between 1 and 32,768 compute nodes.

These graphs raise several points. First, our performance models predicted application performance extremely accurately out to all 32,768 nodes, even though the input to the models was based on measurements taken on only a few nodes. For SAGE, Sweep3D, and VPIC, the maximum prediction errors are 10 percent, 4 percent, and less than 1 percent, respectively, and the average prediction errors are 5 percent, 2 percent, and less than 1 percent.

Second, each application exhibits a qualitatively different scaling curve because of the manner in which it utilizes the system resources. Each process in SAGE sends many messages to a large number and varying set of recipients, depending on node count. Therefore, SAGE's communication performance depends heavily on network bandwidth and the dimensions of the subset for the torus being used.

Table 1. Main characteristics of Blue Gene/P and possible future con-gurations

Characteristics	Current Blue Gene/P	Possible system		
		A	B	C
Computation				
Cores per processor	4			×4
Communication				
Near-neighbor latency (fis)	3.0		÷2	
Per-hop latency (ns)	50		÷2	
Near-neighbor bandwidth (Gbyte/s)	375	×4		

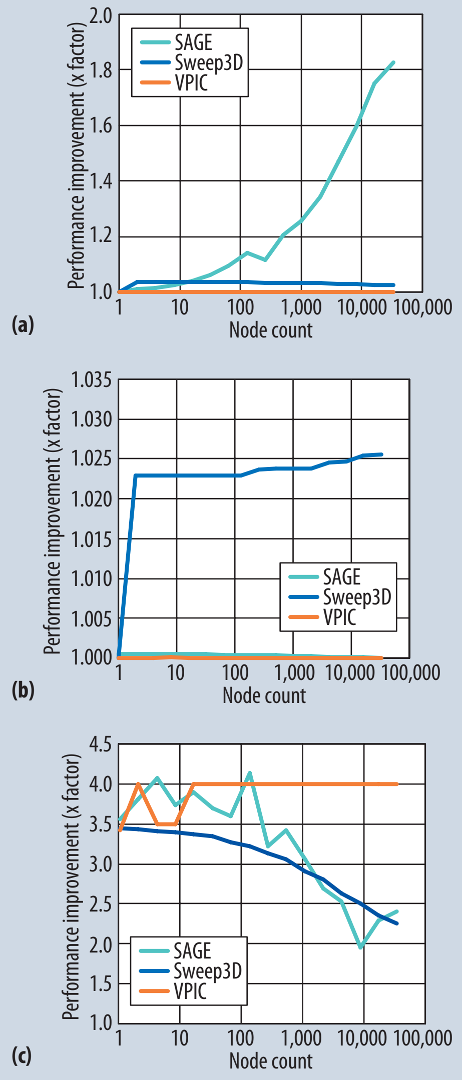


Figure 5. Expected improvements in application performance on a possible future Blue Gene/P type system.

Sweep3D's performance is compute-bound at small scale, but largely determined by the effect of the algorithmic pipeline at large scale. VPIC performance is determined almost exclusively by the speed at which particles are processed, varying only slightly when using 16 or more compute nodes. These differences reinforce the point that performance prediction based on simple extrapolation does not work: Consider measuring these applications on 100 nodes and predicting their performance on 10,000 nodes *without* the use of a performance model.

Exploring possible future system performance

In general, a model can be used to explore a parameter space that cannot be empirically measured. A performance model is no exception. In the context of exploring ultrascale system performance, investigating the performance of possible future systems offers an important use for modeling. At Los Alamos, we constantly focus on machines that do not yet exist but that could be deployed in 5 to 10 years. We illustrate this approach by modifying the baseline Blue Gene/P architecture's performance characteristics to reflect potential future hardware performance. In particular, we consider three separate possibilities: improving internode communication bandwidth, reducing internode communication latency, and increasing the number of cores per processor—all summarized in Table 1.

Figure 5 uses performance modeling to plot the performance improvement over the current Blue Gene/P system, given each of those possible system improvements. As in our validation experiments, qualitative differences in the performance behavior can be seen across the three applications. SAGE's performance is improved at large node counts when the communication bandwidth is quadrupled, as in Figure 5a, while Sweep3D's performance is barely affected, and VPIC's not at all.

Sweep3D's performance is, however, slightly affected by the reduction in communication latency shown in Figure 5b. As Figure 5c shows, the increase in core count per processor socket significantly improves the performance of all three applications at modest node counts, but this improvement diminishes with scale for SAGE and Sweep3D. In SAGE's case, the way its communication pattern increases in complexity with process count causes the diminishing improvement, which makes SAGE more sensitive to network performance on the 4× system than the baseline system. Sweep3D's diminishing performance improvement stems from the algorithmic pipeline effects, which are an inherent aspect of the code. VPIC is compute-bound even at 131,072 processes (32,768 nodes in the 4× system). Hence, its performance improvement at scale exactly matches the increase in core count.

Our methodology for modeling the performance of large applications running on ultrascale systems comprises an analysis of the key contributors to application performance as an analytical expression that maps primitive system performance characteristics to a predicted execution time. Using three production applications—SAGE, Sweep3D, and VPIC—we demonstrated that our performance models can predict execution times out to 32,768 nodes/131,072 processor cores with a worst-case prediction error across all data points of only 10 percent. This is a significant accomplishment because these applications all exhibit nonlinear performance characteristics as system size increases. Hence, extrapolations based on intuition are unlikely to produce accurate predictions.

In our analysis example of a possible future Blue Gene system, we expect that some applications will run significantly faster given an improvement in communication bandwidth. Others will see no performance increase from bandwidth improvements, but will from an increase in the number of cores per processor socket.

We apply performance modeling almost daily to answer performance-related questions. The ability to examine different system configurations on both existing and possible future architectures is proving a valuable capability both at Los Alamos National Laboratory and elsewhere. Although our performance-modeling work has always been focused on large applications running on the world's fastest supercomputers, our methodology is equally applicable to smaller applications on more modest-sized systems. Anyone who is concerned about understanding, predicting, and possibly improving application performance should seriously consider utilizing the analytical performance-modeling methodology we have presented in this article. C

Acknowledgments

We thank Steve Louis at Lawrence Livermore National Laboratory for dedicated access to the Dawn Blue Gene/P system. This work was funded in part by the Advanced Simulation and Computing (ASC) program and the Office of Science of the Department of Energy. Los Alamos National Laboratory is operated by Los Alamos National Security LLC for the US Department of Energy under contract DE-AC52-06NA25396.

References

1. K.J. Barker et al., "Entering the Petaflop Era: The Architecture and Performance of Roadrunner," *Proc. IEEE/ACM Supercomputing* (SC08), IEEE CS Press, 2008.
2. F. Petrini, D.J. Kerbyson, and S. Pakin. "The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q," *Proc. IEEE/ACM Supercomputing* (SC 03), IEEE CS Press, 2003.
3. D.J. Kerbyson, A. Hoisie, and H.J. Wasserman. "A Performance Comparison between the Earth Simulator and other Top Terascale Systems on a Characteristic ASCI Workload," *Concurrency and Computation Practise and Experience*, vol. 17, no. 10, 2005, pp. 1219-1238.
4. G. Almasi et al., "Overview of the IBM Blue Gene/P Project," *IBM J. Research & Development*, vol. 52, no. 1/2, 2008, pp. 199-220.
5. D.J. Kerbyson et al., "Predictive Performance and Scalability Modeling of a Large-Scale Application," *Proc. IEEE/ACM Supercomputing* (SC 01), IEEE CS Press, 2001.
6. A. Hoisie, O. Lubeck, and H. Wasserman, "Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications," *Int'l J. High-Performance Computing Applications*, vol. 14, no. 4, 2000, pp. 330-346.
7. K.J. Bowers et al., "0.374 Pflop/s Trillion-Particle Particle-in-Cell Modeling of Laser Plasma Interactions on Roadrunner," Gordon Bell Finalist, *Proc. IEEE/ACM Supercomputing* (SC 08), IEEE CS Press, 2008.

***Kevin J. Barker** is with the Performance and Architecture Lab (PAL) at Los Alamos National Laboratory. Barker received a PhD in computer science from the College of William and Mary. Contact him at kjbarker@lanl.gov.*

***Kei Davis** is with PAL in the Computer Science for High Performance Computing Group (CCS-1) at Los Alamos National Laboratory. Davis received a PhD in computing science from the University of Glasgow and an MSc in computation from the University of Oxford. Contact him at kei@lanl.gov.*

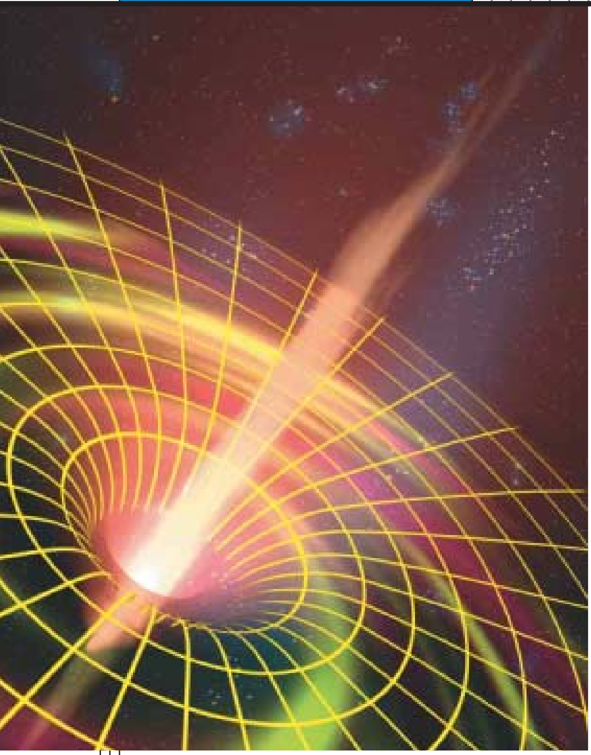
***Adolfy Hoisie** is leader of computer sciences for the HPC Group and director of the Center of Advanced Architectures and Usable Supercomputing at Los Alamos National Laboratory. Hoisie won the Gordon Bell Award in 1996 and is a coauthor of the recently published SIAM monograph on performance optimization. Contact him at hoisie@lanl.gov.*

***Darren J. Kerbyson** is the team lead of the Performance and Architecture Lab (PAL) in the Computer Science for High-Performance Computing Group (CCS-1) at Los Alamos National Laboratory. He received a PhD in computer science from the University of Warwick, UK. Kerbyson is a member of the IEEE Computer Society. Contact him at djk@lanl.gov.*

***Michael Lang** is with PAL in the Computer and Computational Sciences Division at Los Alamos National Laboratory. Lang received an MS in electrical engineering from the University of New Mexico. Contact him at mlang@lanl.gov.*

***Scott Pakin** is with PAL at Los Alamos National Laboratory. Pakin received a PhD in computer science from the University of Illinois at Urbana-Champaign. Contact him at pakin@lanl.gov.*

***José Carlos Sancho** is with PAL at Los Alamos National Laboratory. He received a PhD in computer science from the Technical University of Valencia, Spain. Contact him at jcsancho@lanl.gov.*



PARALLEL SCRIPTING FOR APPLICATIONS AT THE PETASCALE AND BEYOND

Michael Wilde, Ian Foster, Kamil Iskra, and Pete Beckman,
University of Chicago and Argonne National Laboratory
Zhao Zhang, Allan Espinosa, Mihael Hategan, and Ben Clifford, *University of Chicago*
Ioan Raicu, *Northwestern University*

Scripting accelerates and simplifies the composition of existing codes to form more powerful applications. Parallel scripting extends this technique to allow for the rapid development of highly parallel applications that can run efficiently on platforms ranging from multicore workstations to petascale supercomputers.

John Ousterhout aptly characterized scripting as "higher-level programming for the 21st century."[1] Scripting has revolutionized application development on the desktop and server, accelerating and simplifying programming by allowing programmers to focus on the composition of programs to form more powerful applications.

Might scripting provide the same benefits for parallel computers—including extreme-scale computers—as it does for workstations and servers? We believe that the answer is yes. Scripting languages let users assemble sophisticated application logic quickly by composing existing codes. In *parallel scripting*, users apply parallel composition constructs to existing sequential or parallel programs. With such methods, programmers can quickly specify highly parallel applications that may, depending on problem scale, require for their execution a 16-core workstation, a 16,000-core cluster, or a 160,000-core petascale system.

Understanding how to scale scripting to 21st-century computers should thus be a priority for researchers of next-generation parallel programming models. In addressing this priority, we have focused on parallel scripting for systems such as the IBM Blue Gene/P (BG/P) and Sun Constellation.

MOTIVATION FOR PARALLEL SCRIPTING

Most research and development on programming models for exascale machines is concerned with tightly coupled single-program, multiple-data (SPMD) applications—for example, computational fluid dynamic codes applied to weather modeling and structural mechanics codes applied to automobile design. Such applications certainly require large amounts of computing power and a high-performance messaging infrastructure.

Published by the IEEE Computer Society 0018-9162/09/$26.00 © 2009 IEEE

However, it would be shortsighted to assume that such exascale applications are the only ones that require high-end supercomputers. Our experience suggests a substantial and unmet need to run existing programs at large scale, via the simple expedient of running many copies of programs at once. Each such application may itself be a parallel message-passing, multithreaded, or serial code. Developers of such applications, like developers of SPMD applications, require methods and tools to reduce complexity, enhance reuse, and optimize performance on different platforms. Parallel scripting can provide a basis for such methods and tools.

Example

A simple example illustrates parallel scripting in practice.

It is increasingly common for a weather modeler to run many instances of a model, each with different initial conditions, to quantify forecast uncertainty. In pseudocode, the modeler wants to do something like the following:

```
initial_conditions[ ] = initialize( )
forecast[ ] = null
foreach condition, index in initial_conditions:
   forecast[index] = weather_model(condition)
uncertainty = analyze(forecast)
```

This program first creates an array of files, each comprising a different set of initial conditions for the weather model. Then, it invokes the multiple instances of the weather model proper, using an operator (foreach) that performs parallel execution based on available resources. (The weather model runs on many processors; thus, on a small parallel computer, the multiple model invocations may be run one after the other. However, on a large parallel computer, many or all can be run in parallel.) The output from these multiple invocations is stored in a second array of files. The final step analyzes the computed forecasts.

A researcher may wish to explore the sensitivity of the same model to an input parameter, again for a range of initial conditions. This new strategy can be defined via a script that calls the same program in a different manner, this time sweeping over a range of parameters:

```
parameters[ ] = getParameterSets( )
initial_conditions[ ] = initialize()
foreach condition, cindex in initial_conditions:
   foreach parameterSet, pindex in parameters:
      forecast[cindex, pindex] = weather_model
         (parameterSet, condition)
```

Other variants of these simple scripts could select just those runs that generate excessive rainfall, pass their output to a flood model, and/or generate specialized images to highlight unusual conditions. Indeed, parallel scripts can become quite complex. Whether simple or complex, they have in common that they express large amounts of parallelism concisely, via the composition of existing programs that read and write files.

Advantages

As this example shows, parallel scripting is ideal for parameter sweeps and ensemble studies, methods that are increasingly used to explore sensitivity to parametric, structural, and initial condition uncertainty.

Parallel scripting enables developers to build on the codes of today to create the applications of tomorrow on the full spectrum of available parallel systems.

Another important problem class for parallel scripting is data analysis. A parallel script can be a natural tool for both specifying and accelerating the analysis of a large collection of discrete files or database records, particularly in the case of application programs designed to analyze a single file or database record. Biomedical researchers apply this form of parallel scripting, for example, to process images for training computer-aided medical diagnosis algorithms and for research in surgical planning. Starting with programs designed for analyzing single images, they use parallel constructs to create concise scripts capable of rapidly analyzing thousands of such images.

The compelling conclusion from such experiences is that parallel scripting enables developers to build on the codes of today to create the applications of tomorrow on the full spectrum of available parallel systems.

SWIFT: A LANGUAGE FOR PARALLEL SCRIPTING

The framework within which we investigate parallel scripting is the Swift language and system[2] (www.ci.uchicago.edu/swift). Linguistically, Swift blends a C-like syntax with functional programming characteristics. The language is designed to expose opportunities for parallel execution, avoid the unnecessary introduction of nondeterminism, simplify the development of programs that operate on file systems, and permit efficient implementation on distributed-memory parallel computers.

Swift integrates external persistent data—typically contained in files and directories—into the language model, improving the development process for programs that read and/or write large datasets. This integration is achieved via a mapping system that allows files and directories to be represented within programs as typed language variables. Thus, a nested directory structure may be represented in Swift as a nested data structure, permitting a program that operates over all files in those directories to be written as a nested set of `foreach` statements. Similar constructs allow for the definition of typed interfaces to external executables.

Swift reveals opportunities for parallel execution via a combination of explicitly parallel constructs (such as `foreach`) and a dataflow programming model. This model is based on single-assignment variables, a construct that also avoids unnecessary nondeterminism: If one program produces a file that a second program consumes, then Swift ensures that the shared variable representing that file is not assigned a value until the first program has completed execution. As a result of that assignment, the second program then becomes executable. Studies indicate that the amount of code needed to express applications in this form is substantially lower than by ad hoc scripting in shell scripts or less expressive notations such as directed acyclic graphs.[3] The Swift runtime system handles the dispatch of executable tasks to computers and the movement of the data that these programs consume and produce.

Swift integrates external persistent data—typically contained in files and directories—into the language model, improving the development process for programs that read and/or write large datasets.

PARALLEL SCRIPTING CASE STUDY

University of Chicago researchers have developed the Open Protein Simulator,[4] an application that predicts tertiary (3D) protein structure, an important computational problem in biochemistry due to the difficulty of experimental structure determination. Their approach to this problem involves running many instances of a structure prediction simulation, each with different random initial conditions. The simulation uses an "iterative fixing" algorithm[5] (ItFix) that performs multiple "rounds," each involving many parallel Monte Carlo simulated annealing models of molecular moves with energy minimization. After each round, ItFix analyzes the results and picks the best (usually lowest-energy) candidate structure as the basis for the next round, continuing until a convergence criterion is satisfied or a maximum number of rounds have been completed.

This application is a natural candidate for parallel scripting with Swift. Given an external executable `PSim` that computes a single model structure, we want to specify the higher-level structure of the ItFix application. A traditional implementation might involve multiple Bash or Perl scripts to allocate resources, structure on-disk data, and manage the thousands of concurrent tasks. In contrast, the following simplified Swift example of a single ItFix round emphasizes how concise a parallel script can be when using appropriate concepts and constructs:

```
app (ProtGeo pg) predict (Protein pseq)
{
     PSim @pseq.fasta @pg;
}

(ProtGeo pg[]) doRound (Protein p, int n) {
    foreach sim in [0:n-1] {
       pg[sim] = predict(p);
    }
}

Protein p <ext; exec="Pmap", id="1af7">;
ProtGeo structure[];
int nsim = 10000;
structure = doRound(p, nsim);
```

The `app` declaration defines an interface to the `PSim` (Open Protein Simulator) executable. This interface specifies how to map from the typed Swift variables `pg` (protein geometry file) and `pseq` (protein sequence structure) in the header of procedure `predict()` to the command-line program syntax expected by `PSim`. The expressions `@pseq.fasta` and `@pg` insert the filenames mapped to those arguments into the command line. The `predict` procedure expects a protein structure containing a FASTA-format file as its argument and returns a structure prediction in the form of a PDB (Protein Data Bank) file that describes the geometric locations of the protein's atoms in its predicted 3D structure. The `doRound()` procedure performs one "round" of parallel simulations by invoking the `predict()` procedure `n` times in parallel, with each `PSim` invocation executed by `predict()` performing a Monte-Carlo-based structure prediction and returning an array of predictions. The last four statements invoke `doRound` for one protein sequence, running the `PSim` application program 10,000 times in parallel.

Swift's dataflow model enables the multiple invocations of `predict()` to run concurrently, as none depend on data produced by another. Swift's runtime system handles the dispatch of each `predict()` call to an available node and the movement of the associated data to and from that node.

Having thus defined the form of a single round, we can then specify the iterative fixing algorithm proper. We do this as follows, with declarations and parameter lists elided:

```
ItFix( Protein p, int nsim, int maxr,
        float temp, float dt)
{
    ProtSim prediction[][];
    boolean converged[];
    PSimCf config;
    ...
    iterate r {
       prediction[r] =
          doRoundCf(p, nsim, config);
       converged[r] =
          analyze(prediction[r], r, maxr);
    } until ( converged[r] );
}
```

This code fragment uses the Swift `iterate` statement to perform prediction rounds until a convergence criterion has been satisfied or a maximum number of rounds have been performed. The procedure `doRoundCf()` enables science configuration parameters to be passed to the `PSim` application.

Given these Swift procedures, researchers can then use flexible scripts to leverage many processors with relative ease, as in the following parameter sweep script:

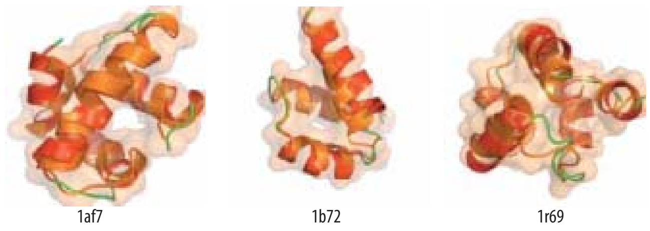
```
int nSim = 1000;
int maxRounds = 3;
Protein pSet[] <ext; exec="Protein.map">;
float startTemp[]=[100.0, 200.0];
float delT[]=[1.0, 1.5, 2.0, 5.0, 8.0];
foreach p, pn in pSet {
    foreach t in startTemp {
       foreach d in delT {
          ItFix(p, nSim, maxRounds, t, d);
       }
    }
}
```

Given 10 protein sequences from the external mapper script "Protein.map", nsim = 1,000, two starting temperatures, and five temperature increments (to control the simulated annealing algorithm), this script would execute 10 × 1,000 × 2 × 5 = 100,000 simulations in each of up to three prediction rounds. On highly parallel systems such as the Argonne BG/P Intrepid, this script can use a substantial portion of the machine's 160,000 processor cores. (ItFix has run on up to 64,000 cores on Intrepid.) Similar code with a generalized parameterization of ItFix can sweep across any combination of settable parameters that govern the structure prediction algorithm.

Figure 1 shows results of running ItFix with Swift for eight protein structure predictions that were executed with a Swift script on 8,000 CPUs. The images show the predicted structure of three proteins from the run; the table shows their lowest root mean square deviation from the experimentally known structure, and their improvement over older runs done on clusters with ad hoc scripts ("DeBartolo"). The scatter plot indicates the correlation between statistical energy potential and protein structure accuracy for 985 simulations of protein 1af7. A parallel Swift script performs the predictions and then generates the plots, images, and a statistics summary table, which are made available to researchers via a Web interface.[4]

In the first two weeks of April 2009, shortly after development of the ItFix Swift script, the system saw impressive use: 67,178 structure predictions, totaling 208,763 CPU-hours, on Intrepid; and 17,488 jobs, totaling 1,425 CPU-hours, on Ranger, the TeraGrid Constellation at the University of Texas at Austin. The same scripts were used in that period to perform 22,495 predictions totaling 2,397 CPU-hours on other TeraGrid sites with between 4,000 and 9,000 cores each. The Intrepid runs alone produced more than 100 gigabytes of compressed protein structure trajectory data.

Protein	Length	ST	TUI	Lowest RMSD (Å, BG/P)	Lowest RMSD (Å, DeBartalo)
T1af7	69	25	100	2.07	2.5
T1b72	50	25	100	1.41	1.6
T1r69	61	25	100	2.11	2.4



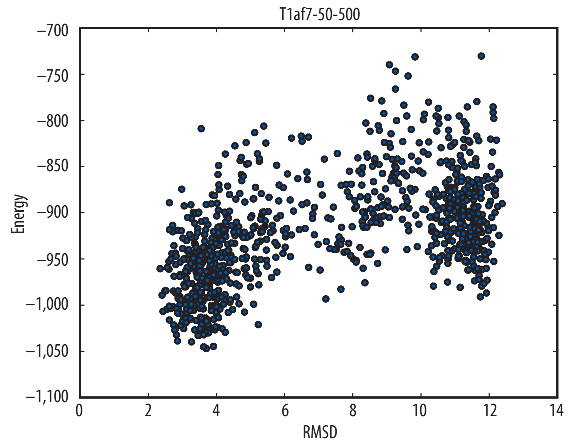
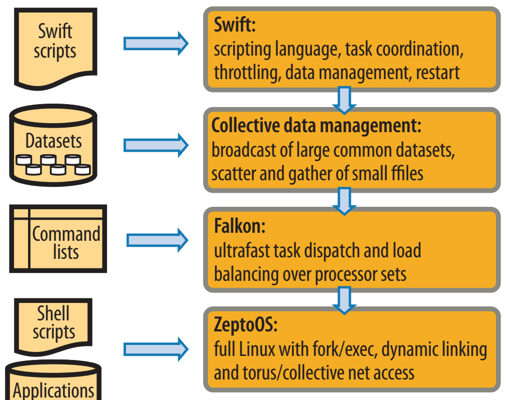
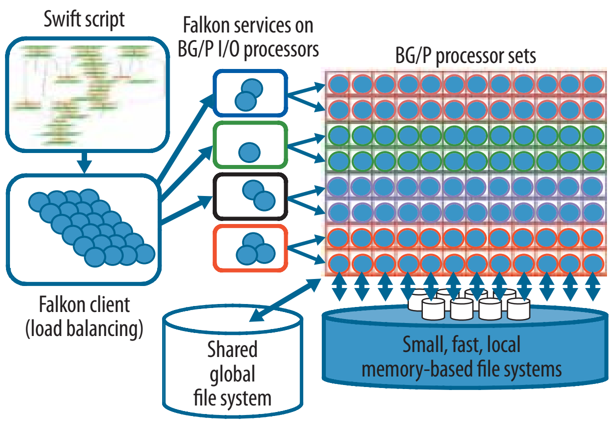


Figure 1. Results of script for predicting eight protein structures on 8,192 CPUs of the Intrepid BG/P, with details for three proteins and the Monte Carlo results for 1af7.



Swift scripts
Datasets
Command lists
Shell scripts
Applications
Swift: scripting language, task coordination, throttling, data management, restart
Collective data management: broadcast of large common datasets, scatter and gather of small ffiles
Falkon: ultrafast task dispatch and load balancing over processor sets
ZeptoOS: full Linux with fork/exec, dynamic linking and torus/collective net access

Figure 2. Architecture for petascale scripting.



Swift script
Falkon services on BG/P I/O processors
BG/P processor sets
Falkon client (load balancing)
Shared global file system
Small, fast, local memory-based file systems

Figure 3. Swift scripts execute using the Falkon distributed resource manager on the BG/P architecture.

AN ARCHITECTURE FOR PETASCALE PARALLEL SCRIPTING

Petascale computing raises challenging problems for implementers of parallel scripting systems. Even a simple parallel script can define large numbers of concurrent tasks that may operate on even larger numbers of files. Task dispatch, data management and movement, mixed-mode parallelism, resource management, failure detection and recovery—these and other programming model functions can lead to difficulties when millions of tasks must execute efficiently and reliably on hundreds of thousands of cores.

Figure 2 shows the four-layer software architecture that we have developed in our investigations of parallel scripting systems. The four layers address, from the top down, the parallel scripting language and its engine and runtime system (Swift); a layer to support the data management demands that parallel scripting—and many-task computing in general—places on cluster file systems; runtime system support for high-performance resource provisioning and task dispatch (for example, the Falkon multilevel resource manager); and operating system support for basic script execution and access to high-performance communications networks (for example, the ZeptoOS Linux-based compute-node kernel). We have used these components to run parallel scripts on up to (so far) 160,000 cores.

Data management for petascale scripting

In a straightforward implementation of parallel scripting, large numbers of programs operate concurrently and independently on a shared parallel file system such as IBM's General Parallel File System (GPFS) on the BG/P. Such I/O patterns place a high burden on a persistent storage infrastructure and tend to be inefficient due to the consistency mechanisms enforced by traditional file system semantics. Our solution to this problem, which we refer to as *collective data management* (CDM),[6] is loosely inspired by collective parallel programming operations such as broadcast, gather, and two-phase I/O.

CDM, as currently conceived, comprises a set of communication strategies that leverage fast local file systems as a high-speed local file cache, use broadcast operations to handle distribution of common input data, employ efficient scatter/gather and caching techniques for input and output, and aggregate compute node storage into larger file systems that leverage a high-performance interconnect to deliver data to applications. In this way, CDM enables efficient and easy distribution of data files to and from computing nodes and can greatly reduce load on the underlying persistent storage system.

Our work to date with CDM has been performed largely on the BG/P, and leverages features such as the BG/P interconnect architecture with its separate collective network, ZeptoOS compute-node kernels with I/O forwarding, and GPFS with full multiprocessor data consistency guarantees. Most of these considerations apply to other deployed petascale systems, all of which run some form of parallel file system, such as GPFS, Lustre, or the Parallel Virtual File System (PVFS). Moreover, all have some form of high-performance, often hierarchical or heterogeneous, network interconnects—for example, a mix of torus, tree, or Clos networks.

We are currently experimenting with CDM concepts through the explicit insertion of CDM primitives and heuristics into applications. Our goal is that CDM operations will ultimately be invoked automatically and transparently by the Swift implementation, making them fully transparent to the programming model and user.

Falkon

To maximize the range of parallel scripts that we can run efficiently, we require rapid task dispatch and execution. For example, keeping 160,000 cores efficiently utilized running 60-second single-thread tasks requires that tasks be dispatched at more than 160,000/60 = 2,700

Table 1. Example parallel scripting applications.

Field	Description	Characteristics	Status
Astronomy	Creation of montages from many digital images	Many 1-core tasks, much communication, complex dependencies	Experimental
Astronomy	Stacking of cutouts from digital sky surveys	Many 1-core tasks, much communication	Experimental
Biochemistry*	Analysis of mass-spectrometer data for post-translational protein modifications	10,000-100 million jobs for proteomic searches using custom serial codes	In development
Biochemistry*	Protein structure prediction using iterative fixing algorithm; exploring other biomolecular interactions	Hundreds to thousands of 1- to 1,000-core simulations and data analysis	Operational
Biochemistry*	Identification of drug targets via computational docking/screening	Up to 1 million 1-core docking operations	Operational
Bioinformatics*	Metagenome modeling	Thousands of 1-core integer programming problems	In development
Business economics	Mining of large text corpora to study media bias	Analysis and comparison of over 70 million text files of news articles	In development
Climate science	Ensemble climate model runs and analysis of output data	Tens to hundreds of 100- to 1,000-core simulations	Experimental
Economics*	Generation of response surfaces for various economic models	1,000 to 1 million 1-core runs (10,000 typical), then data analysis	Operational
Neuroscience*	Analysis of functional MRI datasets	Comparison of images; connectivity analysis with structural equation modeling, 100,000+ tasks	Operational
Radiology	Training of computer-aided diagnosis algorithms	Comparison of images; many tasks, much communication	In development
Radiology	Image processing and brain mapping for neurosurgical planning research	Execution of MPI application in parallel	In development

Note: Asterisks indicate applications being run on Argonne National Laboratory's Blue Gene/P (Intrepid) and/or the TeraGrid Sun Constellation at the University of Texas at Austin (Ranger).

tasks per second. Given that the batch schedulers typically run on parallel computers can take 60 seconds to dispatch a single task, there is a clear need for alternative technologies.

In our work to date we have used the Falkon distributed resource manager[7] to address this need, as shown in Figure 3. Falkon uses a combination of multilevel scheduling and a hierarchical task dispatch architecture to enable rapid task dispatch. Its multilevel scheduling architecture—similar to that used in systems such as Condor and MyCluster—separates two activities that are normally combined on a supercomputer, namely allocating a node to a user and dispatching tasks to that node. In the first provisioning phase, Falkon requests nodes in large quantities, using a system's native batch scheduler, and starts a persistent task execution agent on each core capable of rapidly executing arbitrary and independent Posix processes. Once nodes are thus allocated, Falkon uses a hierarchical network of dispatchers to pass tasks to nodes that are, or soon will be, ready to execute them. These methods have allowed Falkon to dispatch more than 3,000 tasks per second on the BG/P and to run on up to 160,000 cores.[7]

ZeptoOS

The lowest layer in our parallel scripting architecture is a Posix-compliant operating system that provides system services such as the fork() and exec() used to launch a new application program and the I/O functions used to gain high-performance access to specialized communication networks. On the BG/P, we provide these features through the ZeptoOS Linux compute node kernel,[6] which implements Posix-compliant system services, full dynamic loading of executables, access to the BG/P collective ("tree") network through higher-level broadcast operations, IP connectivity over the torus network, and facilities to stripe the RAM-disk file systems of compute nodes and mount them as high-performance intermediate file systems. On other computers, such as the Constellation, we currently use the native compute node OS that provides a complete Posix interface, but we envision a role for ZeptoOS as a vehicle for kernel experimentation even on the Constellation and Cray XT5.

PARALLEL SCRIPTING APPLICATIONS

We have applied large-scale parallel scripting to numerous applications.[3-5,7-9] Each scripted application can consume a large fraction, or even all, of a petascale computer. All involve executing many tasks at once, often with substantial amounts of communication both within each task and among tasks. Table 1 lists some representative examples.

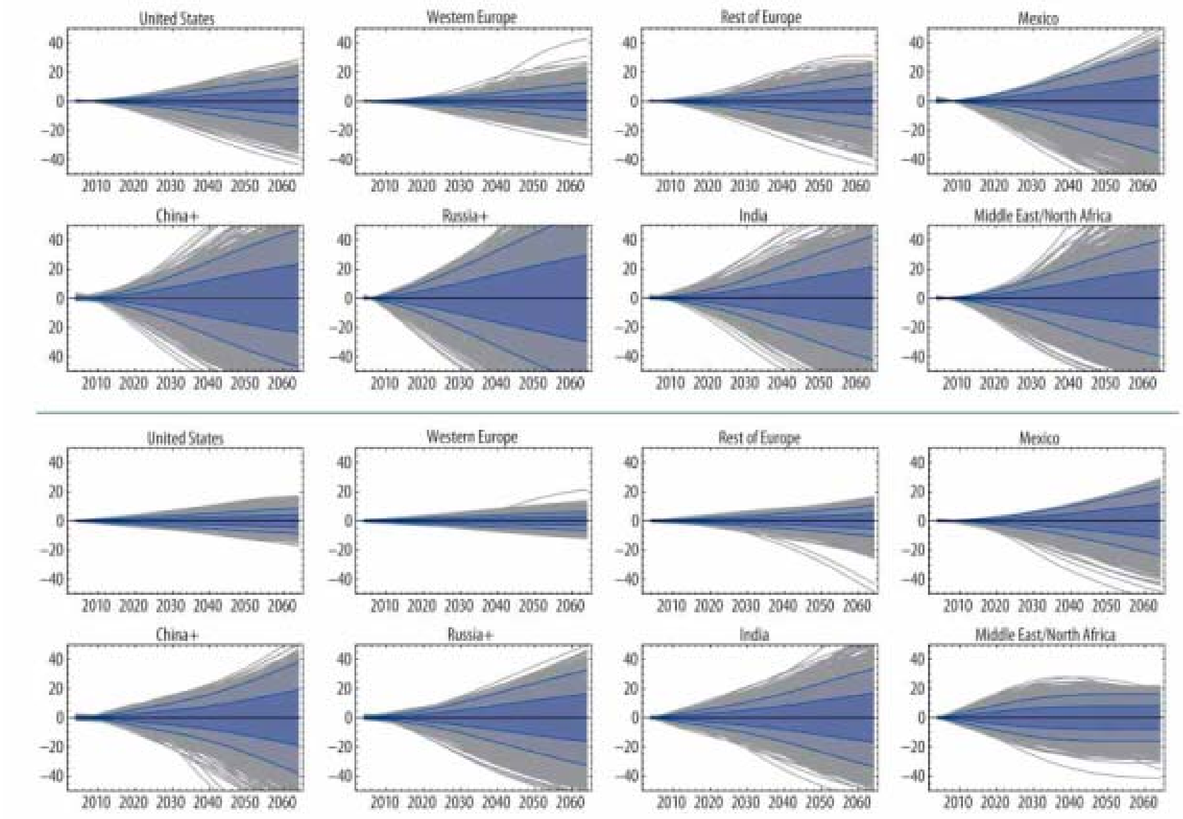


Figure 4. CIM-EARTH energy-economics parameter sweeps of 5,000 models exploring uncertainty in consumer (top) and industrial (bottom) electricity usage projections by region for the next five decades.

Molecular docking

The DOCK molecular dynamics application is run regularly on Intrepid to simulate the docking of small ligand molecules to large macromolecules (receptors). A compound that interacts strongly with a receptor associated with a disease may inhibit its function and thus prove useful in a beneficial drug.

This application is challenging because it involves many tasks, each with a wide range of execution times, and each computation involves significant I/O. Protein description files for docking range from tens to hundreds of megabytes and must be read for each computation.

Argonne biochemists use Falkon for molecular docking and surface screening, running at scales of up to 64,000 cores in a single scripted workload.

Uncertainty in economic models

The University of Chicago-Argonne CIM-EARTH project for integrated social, economic, and environmental modeling (www.cim-earth.org) uses Swift on petascale systems to execute parameter sweeps of economic models that forecast energy use and other commodity demands to examine the effects of uncertainty. CIM-EARTH researchers use the parallel scripting paradigm to refine several models for exploring uncertainty through large-scale parallelism.

Figure 4 shows the results of a parallel script exploring the implications of uncertainty—in this case, parametric uncertainty in substitution elasticities. Researchers analyzed 5,000 samples from a perturbed input dataset in parallel on Ranger and other parallel systems of 5,000+ cores each. The model evaluates relative sensitivity to uncertainty (percent from the mean) for consumer and industrial demand for electricity in eight geographical regions. The dark-blue and light-blue envelopes are one and two standard deviations from the mean.

Structural equation modeling

The University of Chicago's Human Neuroscience Laboratory has developed a computational framework for a data-driven approach to structural equation modeling[8] (SEM) and has implemented several parallel scripts for modeling functional MRI data within this framework. The Computational Neuroscience Applications Research Infrastructure[8] (CNARI, www.cnari.org) uses Swift to execute hundreds of thousands of simultaneous processes running the R data analysis language, consisting of self-contained structural equation models, on Ranger. These self-contained

R processing jobs are data objects generated by OpenMx (http://openmx.psyc.virginia.edu), a structural equation modeling package for R that can generate a single model object containing the matrices and algebraic information necessary to estimate the model's parameters. With the CNARI framework, neuroscientists run OpenMx from Swift scripts to conduct exhaustive searches of the model space.

Posttranslational protein modiffcation

The University of Chicago's Ben May Department for Cancer Research is applying petascale parallel scripting to the analysis of posttranslational protein modifications (PTMs), complex changes to proteins that play essential roles in protein function and cellular physiology. The PTMap application takes in raw data files from mass-spectrometry analysis of biological samples, along with the entire set of sequences of the organism's proteome, and searches them for statistically significant evidence of unidentified PTMs. The tool reads in a mass-spectrometry file—typically 200 megabytes of data in mzXML format—and protein sequences in FASTA format.

The analysis of a mass-spectrometry run for a single proteome has abundant opportunities for parallelization at the extreme scale. Researchers want to apply the latest version of PTMap to identify unknown PTMs across a wide range of organisms including *E. coli*, yeast, cows, mice, and humans.

PARALLEL SCRIPTING MODEL PERFORMANCE

Performance measurements indicate that on Intrepid, Falkon can execute more than 3,000 tasks per second, and launch, execute, and terminate 160,000 tasks on 160,000 cores in under one minute.[7]

Running DOCK under Falkon with a workload of 934,803 molecules (performing a DOCK execution for each one) on 116,000 CPU cores of the Intrepid BG/P took two hours,[7] as shown in Figure 5a, delivering 21.4 CPU-years. Per-task execution time varied considerably, from a minimum of 1 second to a maximum of 5,030 seconds, and a mean of 713±560 seconds. The two-hour run achieved a sustained utilization of 99.6 percent for the first 5,700 seconds and an overall utilization of 78 percent due to the workload tapering off at the end of the run. Despite the loosely coupled nature of this application, our results show that DOCK performs and scales well on a significant fraction of Intrepid, with 99.7 percent efficiency when compared to the same workload at 64,000 CPUs.

Figure 5b shows the progress and active processes of an SEM workflow with over 418,000 jobs, executing as a single Swift script invocation on Ranger to model neural pathway connectivity from experimental fMRI data.[8]

We performed preliminary measurements of the new PTMap application at modest scales, running the stage 1 processing of the *E. coli* K12 genome (4,127 sequences) on

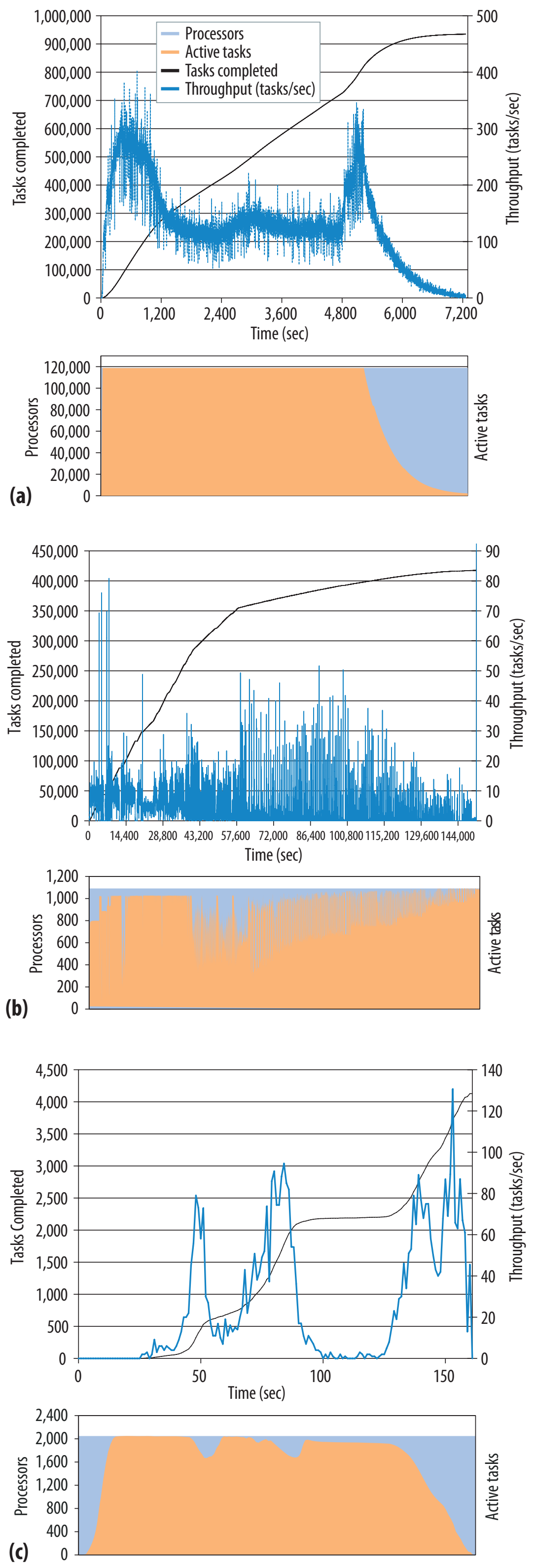


Figure 5. **Performance of three parallel application scripts: (a) DOCK on BG/P—Falkon, 934,803 tasks, 2 hours; (b) SEM on Constellation—Swift, 418,000 tasks, 41 hours; (c) PTMap on BG/P—Swift, 4,127 tasks, 3 minutes.**

2,048 Intrepid cores. Figure 5c summarizes this run. Overall, the average per-task execution time was 64 seconds, with a standard deviation of 14 seconds. These 4,127 tasks consumed a total of 73 CPU-hours, in a span of 161 seconds on 2,048 processor cores, achieving 80 percent utilization from a high-level Swift script.

We view these measurements—all on challenging short-task-length applications—as a promising milestone in meeting and in some cases exceeding the performance needed for petascale scripting and beyond.

Tools that make it easy to couple existing programs and apply programs to different data—in other words, scripting tools—align well with how people approach problem solving.

RELATED WORK

MapReduce,[10] Sphere,[11] and Dryad[12] implement library-based approaches to parallel processing of large datasets. For example, in the MapReduce paradigm, data is distributed over many nodes. SPMD applications can then call both local functions that execute on local data and reduction operations to combine distributed data. This model can require both substantial rewriting of programs and reorganization of data. In contrast, Swift programs require no modifications to application programs. Instead, Swift allows the programmer to focus on composing those programs into larger applications. We view the ability to leverage the vast value embedded in modern sequential and parallel application codes as an important property of parallel scripting. Swift's `foreach` construct performs a simple map operation, and the act of passing a multimember dataset to a procedure provides a simple and natural way to implement reduction operations.

The Nimrod system[13] is an example of a more specialized form of parallel programming system. Nimrod supports parallel computations involving many invocations of an external executable, driven by a high-level specification of a parameter study or, in more recent versions, a numerical optimization strategy.

SPMD message-passing systems such as MPI can be used to express some task-parallel computations. However, MPI is less well suited for the dynamic environments and applications at which Swift excels. In addition, any SPMD programming model, including MPI, faces issues of reliability when scaling to millions of processors and beyond, because of shorter mean time to failure as machines grow in size. The parallel scripting model is more flexible in this regard because failures are typically localized within a compute node task that can be re-executed and thus need not cause an entire application to fail. We view Swift and MPI as complementary in that Swift can be used to coordinate the execution of MPI applications.

Numerous dynamic load balancing libraries have been implemented over the years, varying in details but not general approach. Condor's manager-worker library is one example. Another, implemented within the MPI paradigm, is the Asynchronous Dynamic Load Balancing library.[14] ADLB moves MPI programming closer to the loosely coupled Swift model, in that tasks are freed from the restrictions of two-sided communication and execute in a manner similar to the traditional master-worker model. It is still, however, a model for executing in-memory tasks, unlike the Swift model of executing independent programs linked by file exchange.

The design of Falkon was inspired by the Condor Glide-in facility,[15] which established the utility of multilevel scheduling. Falkon is based on similar principles but implements a simpler facility that contains only the essential semantics needed for first-in, first-out task scheduling and thereby delivers orders of magnitude better scalability and throughput on petascale systems.[7]

High-performance languages for tightly coupled programming, such as Chapel,[16] also offer features similar to those found in Swift. Swift and Chapel share the same goal of programming productivity. However, Chapel is oriented toward in-memory computing, while Swift focuses on loosely coupled application program coordination. Like Chapel, Swift is a "global view" rather than a "fragmented model" programming language, in which the compiler and runtime system determine a program's mapping to the available runtime parallel resources. Like Chapel's `forall` statement, Swift's `foreach` determines a parallel execution strategy for the programmer, without the explicit task assignment of MPI-style fragmented models. Swift is also strongly typed like Chapel, but offers the programmer fewer ways to circumvent the typing model and lacks Chapel's semantics for type inference.

Ousterhout's observation concerning the power of scripting reflects a profound truth about programming. As in other fields of human endeavor, complex artifacts are often created by coupling existing components. Thus, tools that make it easy to couple existing programs and apply programs to different data—in other words, scripting tools—align well with how people approach problem solving.

Historically, people used scripting to prototype programs on workstations, but for more serious programming tasks, such as for parallel computers, they used different methods and tools. It is time to reconsider that position on parallel computers, just as people are doing in other environments. Not only can a scripting approach facilitate the rapid construction of large computations via the composition of existing components, but a scripting language's composition operators often reveal opportunities for parallel execution. Swift shows how a language that supports simple dataflow concepts and file system mapping constructs can allow for the concise specification of highly parallel computations within a scripting framework.

There might be some skepticism about whether scripting methods can be implemented efficiently on large-scale parallel computers, given the need to schedule, dispatch, and manage many tasks on many processors, all the while supporting large numbers of fine-grained I/O operations within both shared and local file system namespaces. Yet our experience shows that these issues need not stand in the way of performance. Data dependency and task management activities can be scaled relatively easily with the use of hierarchical scheduling methods. File system operations can also be scaled, within the constraints that single-assignment semantics place on how parallel scripts access the file system: A file may have many readers, but only one writer. The resulting computations may sometimes stress a parallel computer's communication network, but they usually perform sufficiently well to accomplish a vast array of important scientific tasks with unprecedented speed.

We continue to explore new applications that benefit from parallel scripting and to extend the power and performance of the Swift scripting system. Based on what we have learned to date, we believe that parallel scripting has proven its value on petascale systems and will play an indispensable role in the exascale programming tool chest. C

Acknowledgments

We thank many people for collaborating with us: Sarah Kenny, Steve Small, Michael Andric, and the OpenMx project team on SEM and neuroscience applications; Glen Hocky, Joe DeBartolo, Karl Freed, and Tobin Sosnick on the Open Protein Simulator; Yingming Zhao and Yue Chen on PTMap; Andrew Binkowski and Mike Kubal on molecular docking; Joshua Elliott, Meredith Franklin, and Todd Munson on CIM-EARTH; Andriy Fedorov on image processing for surgical planning; Yong Zhao for early work on Swift; and Justin Wozniak and Gail Pieper for comments on this article. This research is supported in part by NSF grants OCI-721939 and OCI-0944332, NIH grants DC08638 and DA024304, the U.S. Department of Energy under contract DE-AC02-06CH11357, NASA Ames Research Center GSRP grant NNA06CB89H, and the University of Chicago/Argonne National Laboratory Computation Institute. Computing resources were provided by the Argonne Leadership Computing Facility, TeraGrid, the Open Science Grid, and Amazon Web Services.

References

1. J. Ousterhout, "Scripting: Higher-Level Programming for the 21st Century," *Computer*, Mar. 1998, pp. 23-30.
2. Y. Zhao et al., "Swift: Fast, Reliable, Loosely Coupled Parallel Computation," *Proc. 2007 IEEE Congress on Services*, IEEE Press, 2007, pp. 199-206.
3. Y. Zhao et al., "A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data," *ACM SIGMOD Record*, Sept. 2005, pp. 37-43.
4. G. Hocky et al., *Toward Petascale ab initio Protein Folding through Parallel Scripting*, tech. report ANL/MCS-P1645-0609, Argonne National Laboratory, 2009.
5. J. DeBartolo et al., "Mimicking the Folding Pathway to Improve Homology-Free Protein Structure Prediction," *Proc. National Academy of Sciences*, 10 Mar. 2009, pp. 3734-3739.
6. Z. Zhang et al., "Design and Evaluation of a Collective I/O Model for Loosely-Coupled Petascale Programming," *Proc. 2008 IEEE Workshop Many-Task Computing on Grids and Supercomputers (*MTAGS 08), IEEE Press, 2008, pp. 1-10.
7. I. Raicu et al., "Toward Loosely Coupled Programming on Petascale Systems," article no. 22, *Proc. 2008 IEEE/ACM Conf. Supercomputing* (SC 08), IEEE Press, 2008.
8. S. Kenny et al., "Parallel Workflows for Data-Driven Structural Equation Modeling in Functional Neuroimaging," *Frontiers in Neuroinformatics*, Nov. 2009.
9. A. Fedorov et al., *Non-Rigid Registration for Image-Guided Neurosurgery on the TeraGrid: A Case Study*, tech. report WM-CS-2009-05, Dept. of Computer Science, College of William and Mary, 2009.
10. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Comm. ACM*, Jan. 2008, pp. 107-113.
11. Y. Gu and R.L. Grossman, "Sector and Sphere: The Design and Implementation of a High Performance Data Cloud," *Philosophical Trans. Royal Society A*, 28 June 2009, pp. 2429-2445.
12. M. Isard et al., "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," *ACM SIGOPS Operating System Rev.*, June 2007, pp. 59-72.
13. D. Abramson et al., "Parameter Space Exploration Using Scientific Workflows," *Computational Science—ICCS 2009*, LNCS 5544, Springer, 2009, pp. 104-113.
14. P. Balaji et al., "MPI on a Million Processors," *Proc. 2009 European PVM/MPI Users' Group Conf.* (EuroPVM/MPI 09), CSC-IT Center for Science, 2009.
15. D. Thain and M. Livny, "Building Reliable Clients and Services," *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, eds., Morgan Kaufmann, 2005, pp. 285-318.
16. B. Chamberlain, D. Callahan, and H. Zima, "Parallel Programmability and the Chapel Language," *Int'l J. High-Performance Computing Applications*, Aug. 2007, pp. 291-312.

***Michael Wilde** is a software architect in the Mathematics and Computer Science Division, Argonne National Laboratory, and a Fellow at the University of Chicago/Argonne National Laboratory Computation Institute. Contact him at wilde@mcs.anl.gov.*

Ian Foster *is a Distinguished Fellow at Argonne National Laboratory, director of the University of Chicago/Argonne National Laboratory Computation Institute, Chan Soon-Shiong Scholar, and Arthur Holly Compton Distinguished Service Professor of Computer Science at the University of Chicago. Contact him at foster@anl.gov.*

Kamil Iskra *is an assistant computer scientist in the Mathematics and Computer Science Division, Argonne National Laboratory, and a Fellow at the University of Chicago/Argonne National Laboratory Computation Institute. Contact him at iskra@mcs.anl.gov.*

Pete Beckman *is division director at the Argonne Leadership Computing Facility, a computer scientist in the Mathematics and Computer Science Division, Argonne National Laboratory, and a Senior Fellow at the University of Chicago/Argonne National Laboratory Computation Institute. Contact him at beckman@mcs.anl.gov.*

Zhao Zhang *is a PhD student in the Department of Computer Science at the University of Chicago. Contact him at zhaozhang@wuchicago.edu.*

Allan Espinosa *is a PhD student in the Department of Computer Science at the University of Chicago. Contact him at aespinosa@cs.uchicago.edu.*

Mihael Hategan *is a systems software developer at the University of Chicago/Argonne National Laboratory Computation Institute. Contact him at hategan@mcs.anl.gov.*

Ben Clifford *was formerly a systems software developer at the University of Chicago/Argonne National Laboratory Computation Institute. Contact him at benc@hawaga.org.uk.*

Ioan Raicu *is an NSF/CRA Computing Innovation Fellow at the Center for Ultra-scale Computing and Information Security, Department of Electrical Engineering and Computer Science, Northwestern University. Contact him at iraicu@eecs.northwestern.edu.*

IEEE computer society

PURPOSE: The IEEE Computer Society is the world's largest association of computing professionals and is the leading provider of technical information in the field.
MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.
COMPUTER SOCIETY WEB SITE: www.computer.org
OMBUDSMAN: To check membership status or report a change of address, call the IEEE Member Services toll-free number, +1 800 678 4333 (US) or +1 732 981 0060 (international). Direct all other Computer Society-related questions—magazine delivery or unresolved complaints—to help@computer.org.
CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.
AVAILABLE INFORMATION: To obtain more information on any of the following, contact Customer Service at +1 714 821 8380 or +1 800 272 6657:

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES

***Computer*:** The flagship publication of the IEEE Computer Society, *Computer*, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.
Periodicals: The society publishes 14 magazines, 12 transactions, and one letters. Refer to membership application or request information as noted above.
Conference Proceedings & Books: Conference Publishing Services publishes more than 175 titles every year. CS Press publishes books in partnership with John Wiley & Sons.
Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.
Technical Committees: TCs provide professional interaction in over 45 technical areas and directly influence computer engineering conferences and publications.
Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.
Certifications: The society offers two software developer credentials.
For more information, visit www.computer.org/certification.

IEEE
Celebrating 125 Years
of Engineering the Future

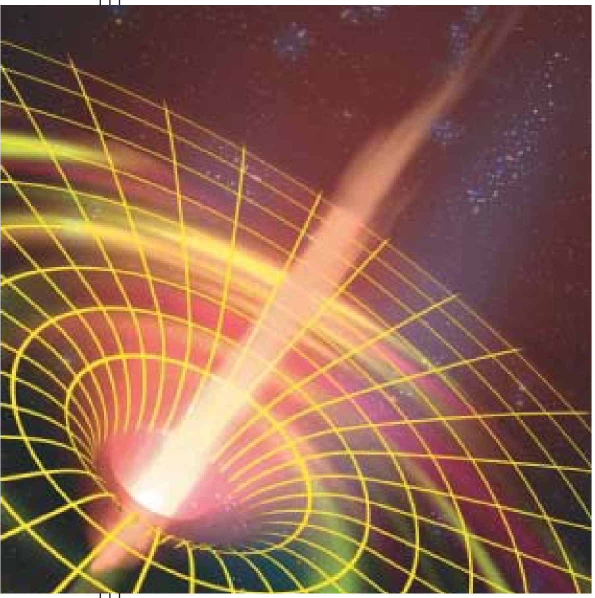
revised 1 May 2009

COMPUTER SOCIETY OFFICES

Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036
Phone: +1 202 371 0101 • **Fax:** +1 202 728 9614
Email: hq.ofc@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314
Phone: +1 714 821 8380
Email: help@computer.org
Membership & Publication Orders:
Phone: +1 800 272 6657 • **Fax:** +1 714 821 4641
Email: help@computer.org
Asia/Pacific: Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • **Fax:** +81 3 3408 3553
Email: tokyo.ofc@computer.org

Next Board Meeting:
17 Nov. 2009, New Brunswick, NJ, USA

COVER FEATURE



ENERGY-EFFICIENT COMPUTING FOR EXTREME-SCALE SCIENCE

David Donofrio, Leonid Oliker, John Shalf, and Michael F. Wehner, *Lawrence Berkeley National Laboratory*
Chris Rowen, *Tensilica*
Jens Krueger, *Fraunhofer Institute, Germany*
Shoaib Kamil and Marghoob Mohiyuddin, *University of California, Berkeley*

A many-core processor design for high-performance systems draws from embedded computing's low-power architectures and design processes, providing a radical alternative to cluster solutions.

The computational power required to accurately model extreme problem spaces, such as climate change, requires more than a business-as-usual approach. Building ever-larger clusters of commercial off-the-shelf (COTS) hardware will be increasingly constrained by power and cooling—with power consumption projected to be hundreds of megawatts for exascale-class problems according to recent DARPA and DOE reports. It makes more sense therefore to leverage the considerable innovation of the low-power architectures developed for embedded computing markets and design a machine capable of the exaflops performance (1 billion-billion floating-point operations per second) required for this and similarly demanding scientific applications.

To that end, we have developed Green Flash, an application-driven design that combines a many-core processor with novel alternatives to cache coherence and autotuning to improve the kernels' computational efficiency. This approach can achieve two-orders-of magnitude improvement in computational efficiency for climate simulation relative to a conventional symmetric multiprocessor (SMP) approach.

The challenge of moving high-performance computing architecture toward exaflops has staggering economic and political ramifications. The computational power required for extreme-scale modeling accurate enough to inform critical policy decisions requires a new breed of extreme-scale computers. The "A Page from Embedded Computing" sidebar describes the architectural philosophy behind Green Flash.

To test our design philosophy, we chose a truly exascale problem: kilometer-scale models of the global atmosphere system requiring simulations 1,000 times faster than real time. The kilometer-scale model decomposes Earth's atmosphere into 20 billion individual cells, demanding a machine with unprecedented performance.

Applying energy-efficient, embedded processors, although a crucial first step, is not in and of itself sufficient to meet this challenge. The computing industry has arrived at a rare inflection point: Fundamental principles of computer architecture are open to question, and new ideas are being explored. Green Flash not only offers a glimpse of how design processes that have been successful in the embedded space can be applied to scientific

A PAGE FROM EMBEDDED COMPUTING

Peter Ungaro, CEO of Cray Computing, recently remarked that "Our current technologies can get us to the 10-20 petaflops range. But then to start to think about 100 [petaflops], we really need a major shift in technology."[1]

The high-performance computing community has long subscribed to architectural specialization as the best way to boost efficiency, but design and verification costs and lead times have made the cost of creating full-custom designs impractical. Thus, to think about exaflops other than in the context of science fiction means abandoning the idea of building ever-larger clusters and turning to another set of proven design strategies that don't come at a prohibitive cost.

In our search for a radical alternative, we turned to the embedded-processor market, which successfully addresses the custom and cost issues. The industry relies on sophisticated tool chains that enable the rapid and cost-effective turnaround of power-efficient semicustom design implementations appropriate to each application.

Our design, Green Flash, leverages the same tool chains to design power-efficient exascale systems, tailoring embedded chips to target scientific applications. Rather than ask, What kind of scientific applications can run on our high-performance computing cluster? after it arrives, we have turned the question around to ask, What kind of system should be built to meet the needs of the most important science problems? This approach lets us realize the most substantial gains in energy efficiency because we essentially peel back the complexity of a high-frequency microprocessor design point to reduce waste—wasted opcodes, wasted bandwidth, waste caused by orienting architectures toward serial performance. We also change the notion of commodity from that of component-level integration of clusters to integration of commodity circuit designs within a chip for a system-on-chip.

By using hardware-software cotuning, our design enables rapid hardware design and establishes a feedback path from application programmer to hardware designer. By combining an autotuning environment for software optimization with an emulation platform based on an FPGA, we can simultaneously develop software optimizations and a semispecialized processor design. Essentially, we have not only built on proven ideas, but we have taken them in a new direction.

Reference

1. G. Huang, "Cray's Comeback: CEO Peter Ungaro on Clouds, Exaflops, and the Future of Supercomputing," 30 July 2009; www.xconomy.com/seattle/2009/07/30/crays-comeback-ceo-peter-ungaro-on-clouds-exaflops-and-the-future-of-supercomputing.

computing, but also addresses some of the most daunting problems of managing the exponential growth of on-chip-parallelism across the entire information technology (IT) industry.

MODELING EARTH'S CLIMATE SYSTEM

Current-generation climate models are comprehensive representations of the systems that determine Earth's climate. Models prepared for the *IPCC Fourth Assessment Report: Climate Change 2007*[1] coupled submodels of the atmosphere, ocean, and sea ice to provide simulations of the past, present, and future climate. Models already being prepared for the next report will represent the major remaining climate system components—the terrestrial and oceanic biosphere, the Greenland and Antarctic ice sheets, and certain aspects of atmospheric chemistry.

Each subsystem model has its own strengths and weaknesses and introduces a particular amount of uncertainty into climate projections. Current computational resources limit the resolution of these submodels, thereby contributing to the uncertainty. Resolution constraints on atmospheric process models, for example, do not allow clouds to be resolved, which means that model developers must rely on subgrid-scale parameterizations that are based on statistical methods. Simulations with these constraints produce cloud distributions that do not correlate well with observations.

Such disagreements can be traced to cumulus convection parameterization. Current global atmospheric models have resolutions of approximately 200 km—many times larger than individual clouds. A few groups are attempting to develop models at the limit of cumulus parameterization validity (approximately 25 km), but the necessary century-scale integrations are just barely feasible on the largest current computing platforms. Although this increase in horizontal fidelity should rectify many issues, the fundamental limitations of cumulus parameterization are likely to remain.

For this reason, it makes more sense to simulate cloud processes directly rather than model them statistically. At a horizontal grid spacing of approximately 1 km, a model could resolve cloud systems individually, providing a direct numerical simulation. However, because numerical stability requirements impose time-step limitations, the computational burden of fluid dynamics algorithms scales nonlinearly with the number of grid points. Consequently, the resources to carry out century-scale simulations of Earth's climate would overwhelm the capability of any traditional machine.

Resource requirements

In previous work,[2] we estimated the resources needed for a resolution of approximately 1 km. To fine-tune these estimates, we have since partnered with David Randall's group at Colorado State University (CSU) that is using a mesh representation of the globe with an icosahedron as the starting point. By successively bisecting the sides of the triangles making up this object, the group was able to generate a remarkably uniform mesh on the sphere. However, this is not the only way to discretize the globe at this resolution; a variety of independent resolving models are necessary to make credible projections about climate change. Consequently, we wanted to be sure that Green Flash could run a class of global climate models, not just a particular model.

We originally estimated 10 petaflops as the sustained computational rate necessary to simulate Earth's climate 1,000 times faster than it actually occurs. An updated estimate of the requirements for the CSU model raised that to as high as 70 petaflops—an example of the considerable uncertainty in making these estimates. As the CSU model matures, we expect to determine this rate even more accurately. An exaflops-scale machine would provide multiple realizations of individual simulations, a necessary tool in addressing the climate system's statistical complexities. The exact peak flops rate required would depend greatly on the machine's potential efficiency.

To help maintain backward and general-purpose compatibility, the processor's instruction set architecture is expandable but must be functional enough to allow general-purpose code execution.

A strawman decomposition

Without sufficient parallelism in the climate problem, these enormous sustained computational rates are not even imaginable. Fortunately, the CSU group has demonstrated that the icosahedral formulation of cloud-system resolving models at the kilometer scale can offer plenty of opportunity to decompose the physical domain. Their decomposition bisects the triangles composing the icosahedron 12 successive times, producing a global mesh with 167,772,162 vertices spaced 1 to 2 km apart. It is then possible to apply a logically rectangular two-dimensional domain-decomposition strategy horizontally to the icosahedral grid. Choosing square segments of the mesh containing 64 grid points each (8 × 8) results in 2,621,440 horizontal domains. The vertical dimension offers additional parallelism. Assuming that we could decompose 128 layers into eight separate vertical domains, the total number of physical subdomains could be 20,971,520.

Even given 20-million-way parallelism, we continued to pursue the strawman decomposition, keeping in mind the practical constraints on an SMP core's performance. With a single core assigned to each subdomain, individual cores must be capable of a computational rate of about 3.5 gigaflops for the icosahedral code to achieve a simulation 1,000 times faster than real time. These rates are based on the computational efficiency rates of current mainstream rates. The efficiency gained through autotuning can bring the peak flops rate requirement down considerably. About 25 Mbytes of memory would also be required per subdomain as would about 7,500 nearest-neighbor messages per second with a size of 8 to 10 Kbytes each. This last requirement translates to a bandwidth of about 78 Mbytes per second between nearest-neighbor processors.

Designing 128 cores onto a single chip would result in 163,840 individual sockets—numbers that were not implausible. We were thus encouraged to take our strawman decomposition to the design stage.

DESIGN PROCESS

Because power constraints have long directed the development of embedded architectures, we began with an embedded core and some of the sophisticated tool chains developed to minimize time from architectural specifications to the application-specific IC. We then looked at how we could maximize efficiency by tuning the hardware and software to optimize performance as well as how we could provide rapid design prototypes and cope with fault resilience.

Leveraging an embedded tool chain

The sophisticated tool chains for developing the system-on-chip application-specific ICs popular in the embedded computing market give designers the flexibility to combine verified functional units in myriad ways to rapidly produce semicustom designs. To test these ideas, we adopted the tool chain from Tensilica,[3] an embedded-design firm. The chain starts with a base architecture, to which a designer can add floating-point support to a processor or perhaps choose a larger cache or local store. Adding features to the processor core (or removing them) is as simple as clicking a checkbox or selecting from a dropdown menu.

The tool then selects the unit from its library and integrates it into the design—which substantially reduces the writing and rewriting of the full custom logic typically required when changing a processor's architecture. To help maintain backward and general-purpose compatibility, the processor's instruction set architecture is expandable but must be functional enough to allow general-purpose code execution. The tools also allow designers to flexibly define application-specific extensions to the base instruction set architecture. Of course, the tools have their limits (a designer can't have hundreds of read ports from a single memory, for example), but their flexibility vastly outweighs any inherent restrictions.

Much like current high-performance computing designs, our approach continues to use off-the-shelf components except at a finer grain. Rather than using entire off-the-shelf processors at a socket-level granularity, we can tailor individual functional units within a core and their interconnections to create a semicustom system-on-chip (SOC) design.

The ability to rapidly generate processor cores that are tailored to scientific applications makes these tools compelling, but the excessive overhead in verifying hardware and creating a usable software stack for each new processor negates any time saved in hardware development. To address this drawback, the tools generate optimizing compilers—test benches as well as a functional simulator—in parallel with the design's register transfer logic. Constructing the processor with verified building blocks and automatically generating test benches greatly reduce the risk and time spent in formal verification.

Rapid design prototyping

Traditionally, the complexity of coding in Verilog or VHDL versus C++ or Python and the inability to emulate large designs have outweighed the speed and accuracy advantages of using field-programmable gate arrays (FPGAs). However, FPGA use has become much more practical over the past decade because, unlike commercial microprocessors, FPGAs are not experiencing a clock-rate and power plateau. The lookup table count on FPGAs continues to increase, enabling the emulation of more complex designs. In addition, FPGA clock rates have been growing steadily, closing the gap between emulated and production clock rates. Recent advances in FPGA I/O features have made accessing large, dynamic memories much more palatable.

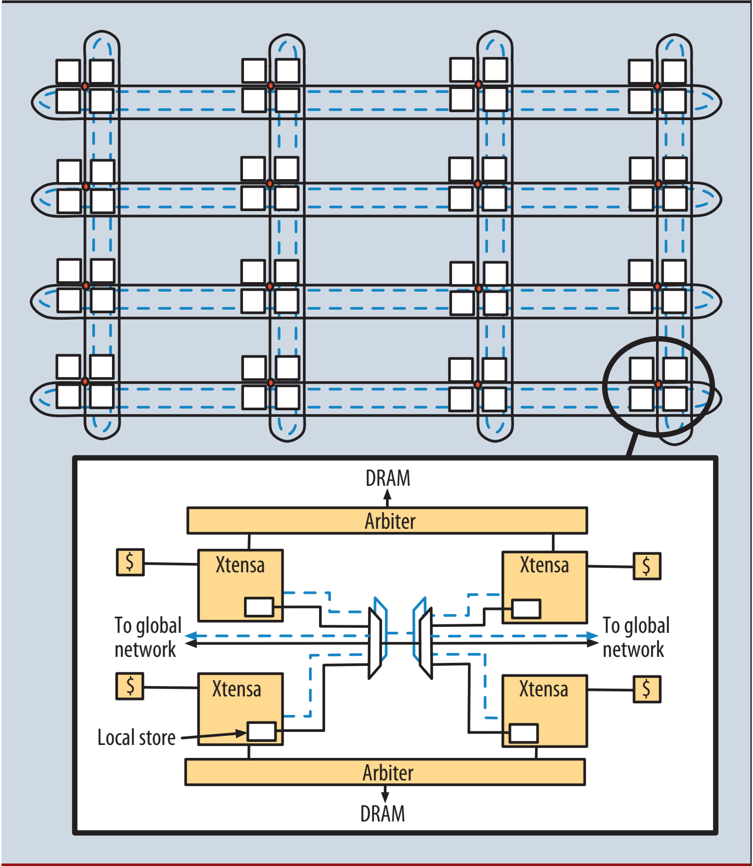


Figure 1. The on-chip network fabric for the Green Flash system-on-chip. A concentrated torus network fabric yields the highest performance and most power-efficient design for scientific codes.

To accelerate the creation of prototype system designs, we are using the Research Accelerator for Multiple Processors (RAMP),[6] an FPGA emulation platform that makes the hardware configuration available for evaluation while the actual hardware is still on the drawing board. RAMP is a cooperative effort among six universities to build a new standard emulation system for parallel processors.

Although the steady growth in FPGA lookup table count has enabled the emulation of more complex designs, a strawman architecture of 128 cores per socket requires emulating more than the two or four cores that will fit on a single FPGA. To address this limitation, we have employed version 3 of the Berkeley Emulation Engine (BEE3), a board populated with four Virtex-5 155 FPGAs, each with two dedicated channels of double data rate memory, connected in a ring with a crossover connection.

Using the BEE3, we effectively emulate eight networked cores, each running at 33 MHz. To scale beyond eight cores, the BEE3 includes 10-Gbit Ethernet connections, allowing the boards to be linked and enabling the emulation of an entire socket. There is significant precedent for emulating massively multithreaded architectures across multiple FPGAs. The Berkeley RAMP Blue project, for example, demonstrated the emulation of more than 1,000 cores using a stack of 16 BEE2 boards.[7]

Maximizing effciency

Opting to follow the design philosophy that the best way to reduce power consumption and increase efficiency is to reduce waste, we chose an architecture with a very simple in-order core and no branch prediction. Because the climate model's demands for memory and communication are high, both aspects drive Green Flash's core design. Reducing the computational burden through autotuning also contributes to efficiency. Finally, hardware-software cotuning tunes the hardware to the autotuned software for additional efficiency gains.

Network topology. Our experience evaluating the STI Cell processor[4] shows that, for memory-intensive applications, cores with a local store use a higher percentage of the available dynamic RAM (DRAM) bandwidth. On the basis of these results, we decided to include a local store in our processor architecture. As Figure 1 shows, the design uses a torus network fabric with two on-chip networks. Predictably, most of the communication among the climate model's subdomains is nearest neighbor. We did

PHOTONIC NETWORKS: A MORE EFFICIENT NETWORK INTERCONNECT

Power eciency requires reducing the power consumption of all system components. With these highly ecient tiny processing elements there is a danger that communication bottlenecks—both in energy and time—will result in a less ecient overall system. To mitigate this danger, long-term research requires exploring interconnect architectures that will both increase performance and reduce energy use.

One promising approach is to combine 3D CMOS integration with research into silicon photonics to build hybrid electronic-photonic interconnects on-chip.[1,2] Designers place photonic detectors and emitters along with specialized low-power photonic switching elements on a special interconnect layer and interface them with processing elements using conventional electronic routers. Figure A shows how the switching elements work. Large-scale communications occur over photonic links, which have several strong advantages over electronic networks. Energy consumption for photonics is less dependent on signaling rate and distance compared to electronics, and the photonic switches are much simpler as they do not require buers or repeaters.

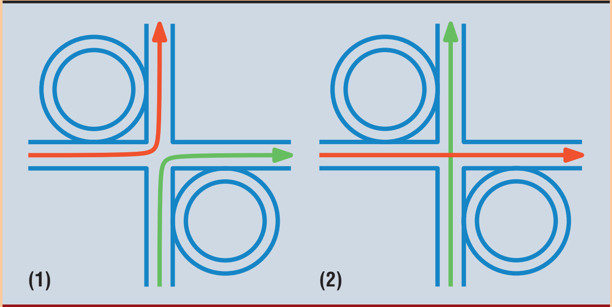


Figure A. Photonic switching elements. (1) Light is coupled onto a perpendicular path; (2) messages propagate straight through. The lack of distance and complex structures are strong advantages over a purely electrical interconnect.

Preliminary research with messaging patterns arising from scientic applications shows that such hybrid networks have the potential to bring major gains in eciency, due to their lower power consumption combined with fast propagation speed. Early research studies done in collaboration with the Lightwave Research Laboratory at Columbia University, for example, show that a hybrid electronic-photonic interconnect composed of ring resonators can deliver 27x better energy eciency than electrical interconnects alone.[3]

References

1. C. Batten et al., "Building Many-Core Processor-to-DRAM Networks with Monolithic CMOS Silicon Photonics," *IEEE Micro*, Special Issue: Micro's Top Picks from Hot Interconnects 16, vol. 29, no. 4, 2008, pp. 8-21.
2. A. Shacham, K. Bergman, and L.P. Carloni, "Photonic Networks-on-Chip for Future Generations of Chip Multiprocessors," *IEEE Trans. Computers*, vol. 57, no. 9, 2008, pp. 1246-1260.
3. G. Hendry et al., "Analysis of Photonic Networks for a Chip Multiprocessor Using Scientific Applications," *Proc. 2009 3rd ACM/IEEE Int'l Symp. Networks-on-Chip* (NOCs 09), IEEE CS Press, 2009, pp. 104-113.

additional experiments with cycle-accurate models of an on-chip packet-switched network to determine that a concentrated torus topology provides superior performance and energy efficiency for codes in which a nearest-neighbor communication pattern dominates.[5] We are currently targeting a core with a clock speed of 500 MHz, a 32-Kbyte conventional error correction code (ECC)-protected cache per core, and a 128-Kbyte local store. The availability of a conventional cache will allow code to be incrementally ported to use the local store. Each socket of 128 cores will have a 50-Gbyte-per-second interface to DRAM.

The traditional cache-coherent memory consistency schemes typical of most modern SMPs make fine-grained synchronization among cores very difficult, and greatly increase the amount of undesired interprocessor data movement. For example, to achieve our target execution rate on 20 million processors, we must compute on a local mesh size that is 8 × 8 × 10 cells. We have observed that the code would spend 90 percent of its time in communication if it were to run on a conventional cache-based hardware, due to the overhead penalty of exchanging extremely small messages between cores.

In Green Flash, we have added specialized hardware to each core to enable extremely low-overhead messaging between cores, bringing the communication overhead below 20 percent of the total execution time. We have used Tensilica's tools to create multiple designer-defined ports with a simple first-in, first-out interface, and each port can send and receive a word-sized data packet on each clock. This ultra-low-overhead streaming interface bypasses the cache to minimize latency and connects to one of the on-chip torus networks. The narrow network is for address exchange; the wider torus network is for bulk data exchange using asynchronous direct memory access (DMA) data transfers. The address space for each processor's local store is mapped into the global address space, and the data exchange is done as a DMA from local store to local store.

From a logical programming view, all processors are directly connected to each other, but physically are connected using a concentrated torus network to the chip's 2D planar geometry. To further simplify programming, a traditional cache hierarchy is also in place to allow the slow porting of codes to the more efficient interprocessor network. To minimize power, we are investigating the use of photonic interconnects for the intercore network, which could prove to be an efficient way of transferring long messages. The "Photonic Networks: A More Efficient

Network Interconnect" sidebar describes the advantages of this approach.

Autotuning. Communication was not our only challenge in the climate model computation. Meeting the requirement of simulating at 1000× real time per core in a power-efficient design is a daunting task, so to optimize the code and reduce the computational burden, we created an autotuning framework that automatically searches a range of optimizations to improve the application kernels' computational efficiency. The autotuner first systematically applies compiler optimizations and then uses domain-specific knowledge of the algorithm to take more aggressive steps, such as loop reordering, to produce optimal, but functionally equivalent, code. In this way, it maintains performance across a diverse set of architectures.

Figure 2 shows the results for the climate model. We ran the autotuning framework using the Tensilica architectural simulator, reducing the cache footprint and overall instruction count and increasing the kernel's computational density. We first generated the original requirement of 3.5 gigaflops per core using a machine that ran with approximately 5 percent efficiency. Autotuners, combined with hardware optimizations, will play a key role in dramatically increasing the efficiency of Green Flash. Through these combined optimizations, we expect Green Flash to realize a two-orders-of-magnitude increase in efficiency.

Hardware-software codesign. Conventional approaches to hardware design use benchmark codes to search for a power-efficient architecture. However, modern compilers fail to generate even close to optimal code for target machines, which strongly implies that a benchmark-based approach to hardware design does not exploit the full performance potential of the architecture design points and can lead to possibly suboptimal hardware solutions. The success of autotuners proves the feasibility of generating efficient code using domain knowledge. Therefore, we created cotuning as a technique to tailor the hardware to autotuned software to get better energy efficiency. Using our autotuning technology, we can automate the exploration for the optimal combination of tuned software *and* hardware in a coordinated design cycle.

As Figure 3 shows, our cotuning approach incorporates extensive software tuning into the hardware design process. The autotuned software tailors the application to the hardware design point under consideration by empirically searching software implementations to find the best mapping of software to microarchitecture.

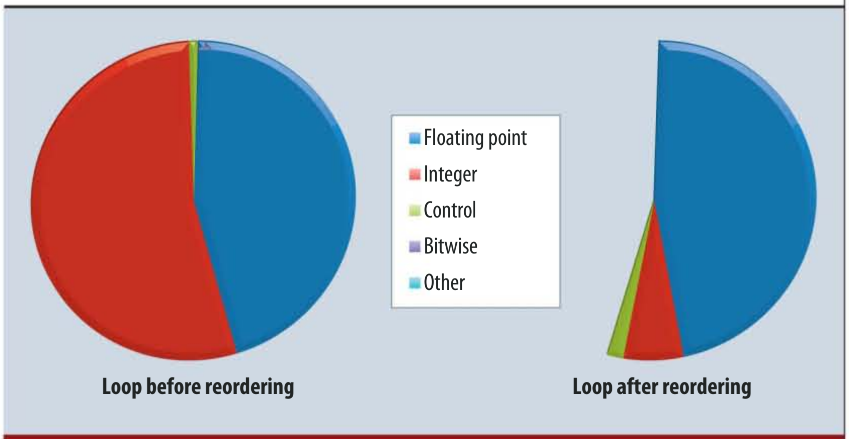


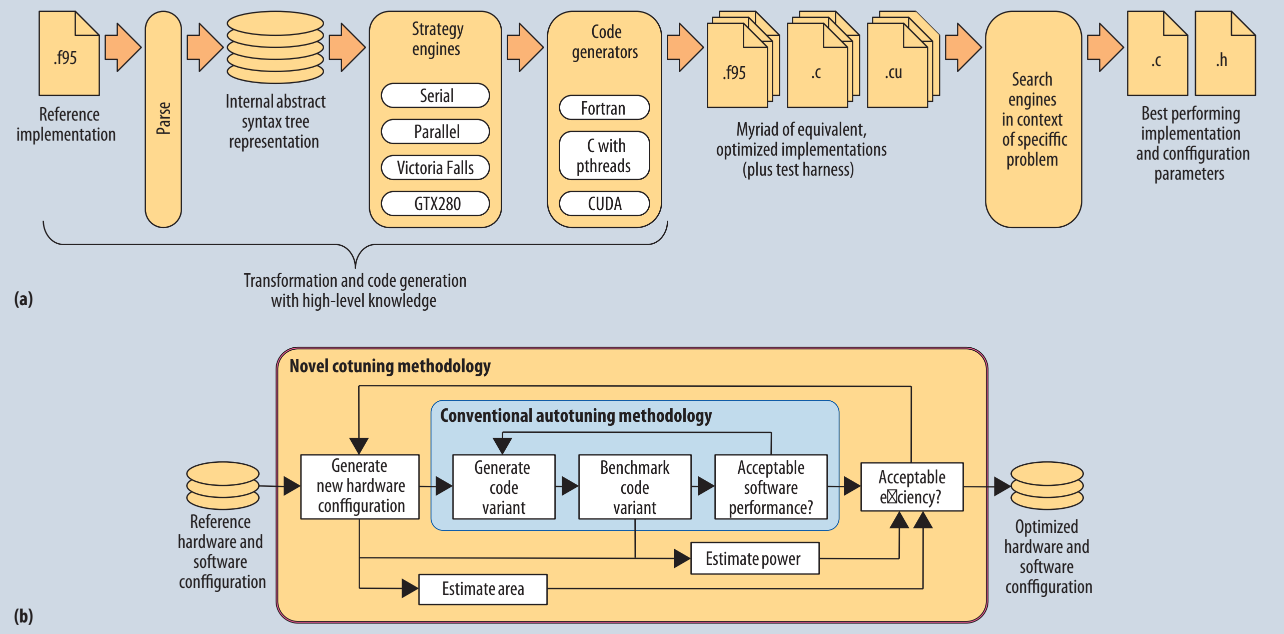
Figure 2. Effect of optimization on a single loop in the climate model. In addition to greatly reducing the instruction count, optimization reduced the cache footprint of this loop by more than 100 times. With software tuning, Green Flash can reduce a per-core computational requirement of 3.5 gigaflops to a more feasible 0.5 gigaflops.

As a demonstration of our proposed cotuning methodology, we used the Smart Memories multiprocessor (based on Tensilica cores) as the target architecture and three widely used kernels from scientific computing—dense matrix-matrix multiplication, stencil codes, and sparse matrix vector multiplication. As part of exploring the hardware design space, we varied four hardware parameters: number of cores, whether caches are hardware- or software-managed, cache size per core, and total memory bandwidth available.

We used tools to estimate the area and power of each hardware configuration that had the corresponding best software configuration, which we obtained through autotuning. As Figure 4 shows, power and area efficiencies improved dramatically for the three kernels.

One hindrance to practical cotuning is the large hardware-software design space to be explored to tailor the hardware design parameters to the target applications. Conventional hardware design approaches use a software simulation of the hardware to perform this exploration. However, cotuning in Green Flash must explore the software design space at each hardware design point, making it impractical to cotune using software simulation.

Instead, we took advantage of the Tensilica tool chain's ability to create synthesizable register-transfer logic for any processor and, by loading this design onto an FPGA, we were able to emulate a potential processor design running 500 times faster than a functional simulator. With this speedup, designers can benchmark true applications rather than having to rely on representative code snippets or statically defined benchmarks. More important, this speed advantage does not come at the expense of accuracy; FPGA emulation is arguably much more accurate than a software simulation environment because it truly represents the hardware design.



.f95
Reference implementation
Parse
Internal abstract syntax tree representation
Strategy engines
Serial
Parallel
Victoria Falls
GTX280
Code generators
Fortran
C with pthreads
CUDA
.f95
.c
.cu
Myriad of equivalent, optimized implementations (plus test harness)
Search engines in context of speciffic problem
.c
.h
Best performing implementation and confﬁguration parameters
Transformation and code generation with high-level knowledge
(a)
Novel cotuning methodology
Conventional autotuning methodology
Reference hardware and software confﬁguration
Generate new hardware confﬁguration
Generate code variant
Benchmark code variant
Acceptable software performance?
Acceptable efficiency?
Optimized hardware and software confﬁguration
Estimate power
Estimate area
(b)

Figure 3. Cotuning in the Green Flash design. (a) Conventional autotuning uses source code generators and search heuristics to empirically choose an efficient software implementation given a high-level representation of a kernel. (b) Hardware-software cotuning extends conventional hardware design space exploration by using autotuning to tailor software to each hardware design point.

The hardware-software codesign process enables scientific application developers to directly participate in the design process for future supercomputers in an unprecedented way. With this fast, accurate emulation environment, designers can run and benchmark the actual climate model as it is being developed and use cotuning to quickly search a large design space.

We believe that these experiments outline a path for bringing the concept of hardware-software codesign—already prevalent in embedded design practices—into the realm of supercomputing system design.

SCALING UP

This article focuses primarily on hardware and software design methodology. However, in considering any system of this scale, a myriad of system software issues come to the forefront, such as scalable operating systems, fault resilience infrastructure, and the development of entirely new programming models to make billion-way parallelism more tractable.

Fault resilience

An important question arises when proposing a 20-million-processor computing system: How do you deal with fault resilience? Although the problem is certainly not trivial, neither is it unusual. As long as the total number of discrete chips is not dramatically different, any large-scale design faces the challenge of aggregating conventional server chips into large-scale systems.

Across silicon design processes with the same design rules, hard failure rates are proportional to the number of system sockets and typically stem from a mechanical failure. Soft error rates are proportional to the chip surface area—not how many cores are on a chip—and bit error rates tend to increase with clock rate. The Green Flash architecture is unremarkable in all these respects and should not pose challenges beyond those that a conventional approach faces.

To deal with hard errors, designers often add redundant cores per chip to cover defects. An old trick in the memory business, the strategy is apparent in designs such as the 188-core Cisco Metro chip, and it is entirely feasible for our design as well. Moreover, Green Flash's low power dissipation per chip (7 to 15W) will reduce the mechanical and thermal stresses that often result in a hard error.

To address soft errors, we have included all the basics for reliability and error recovery in the memory subsystem, including full ECC protection for all hierarchical levels. Green Flash's low target clock frequency provides a lower signal-to-noise ratio for on-chip data transfers. Finally, to enable faster rollback if an error does occur, our design makes it possible to incorporate a nonvolatile RAM controller onto each SMP so that each node can perform a local rollback as needed. This strategy enables much faster rollback, relative to user-space checkpointing.

The Blue Gene system at Lawrence Livermore National Laboratory uses similar fault resilience strategies and contains a comparable number of sockets to Green Flash, yet its mean time between failures (MTBF) is 7 to 10 days[8]—much longer than systems with far fewer processor cores. Because we tailor our architecture to the application, Green Flash can deliver more performance than a machine with a comparable number of sockets, thus reducing its exposure to both hard and soft errors. It proves that carefully applying well-known fault-resilience techniques together with a few novel mechanisms that extend fault resilience, such as localized nonvolatile RAM checkpoints, can yield an acceptable MTBF for extreme-scale implementations.

Programming model

Future hardware constraints and growth in explicit on-chip parallelism will likely require a mass migration to new algorithms and software architecture that is as broad and disruptive as the migration from vector to parallel computing systems that occurred 15 years ago. Applications and algorithms will need to rely increasingly on fine-grained parallelism and strong scaling and support fault resilience.

History shows that the application-driven approach we are using for Green Flash offers the most productive strategy for evaluating and selecting among the myriad choices for refactoring algorithms for full scientific application codes as we move through this transitional phase. We are exploring novel programming models together with hardware support to express fine-grained parallelism to achieve performance, productivity, and correctness for leading-edge application codes in the face of massive parallelism and increasingly hierarchical hardware. The goal of this development thrust is to create a new software model that can provide a stable platform for software development for the next decade and beyond for all scales of scientific computing.

We have developed direct hardware support for both the message passing interface (MPI) and partitioned global address space (PGAS) programming models to enable scaling of these familiar single program, multiple data (SPMD) programming styles to much larger-scale systems. The modest hardware support enables relatively well-known programming paradigms to utilize massive on-chip concurrency and to use hierarchical parallelism to enable use of larger messages for interchip communication. The icosahedral formulation of the climate problem can expose a massive degree of parallelism through domain decomposition, which can use a 20-million processor computing system. The autotuning framework is rapidly evolving into a generalized code generator, which allows the programmer to express the solver kernels at a much higher level of abstraction—enabling a productive programming environment that supports portability, performance, and correctness without exposing scientists to the details of the computer architecture. We think this approach can be scaled out to support a broad range of codes that have such inherent explicit parallelism.

However, not all applications will be able to express parallelism through simple divide-and-conquer problem partitioning. We are only just beginning to explore new asymmetric and asynchronous approaches to achieving strong-scaling performance improvements from explicit parallelism. Techniques that resemble class static dataflow

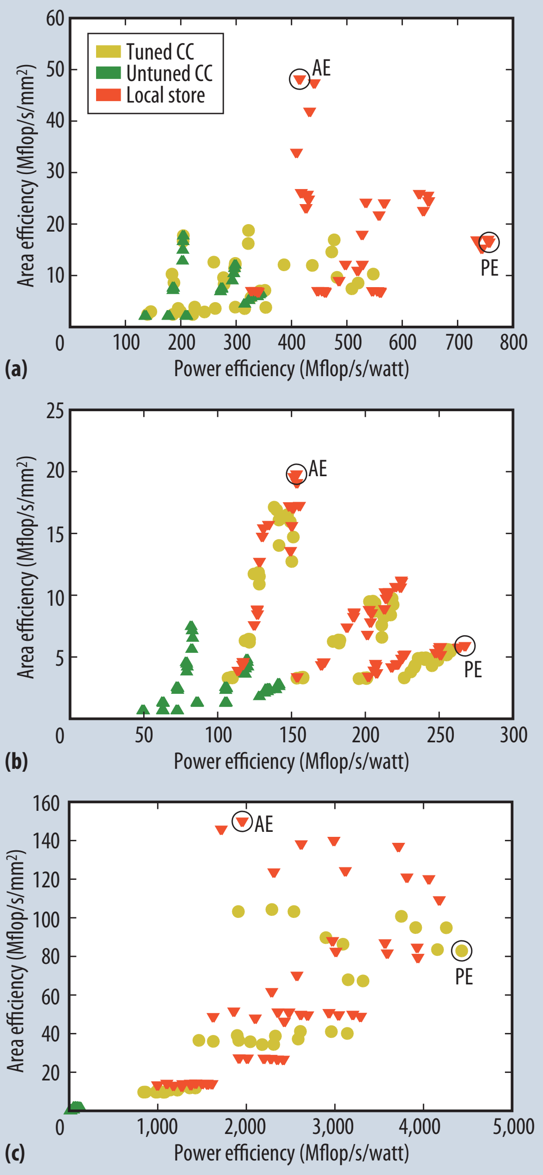


Figure 4. The advantages of cotuning for three kernel types common in scientific applications. AE and PE points denote configurations with highest area and power efficiencies. Improvements varied from 2x to 50x.

methods are garnering renewed interest because of their ability to flexibly schedule work and to accommodate state migration to correct load imbalances and failures.

In the case of the climate code, we can use dataflow techniques to concurrently schedule the physics computations with the dynamic core of the climate code, thereby doubling our concurrency without moving to a finer domain decomposition. This approach also benefits from the unique interprocessor communication interfaces developed for Green Flash. Successful demonstration of the new parallelization procedure for a range of leading extreme-scale applications can then be utilized by other similar codes, accelerating development efforts for the entire field.

What's next?

Designs that follow our approach have the potential to open a market demand for massively concurrent components that can also be the building blocks for mid- and extreme-scale computing systems. New programming models must be part of a new software development ecosystem that spans all system scales so that the industry has a viable migration path from development to large-scale production computing systems. We have demonstrated the value of FPGA-based hardware emulation platforms, such as RAMP, in prototyping and running hardware prototypes at near-real-time speeds before they are built. Such a capability will make it possible to test full-fledged application code and advanced software development many years ahead of the hardware platform construction.

Although machines such as Blue Gene or SciCortex have demonstrated the advantages of using simple, low-power embedded cores, our approach goes beyond these traditional designs by optimizing data movement through explicit message queues and software controlled memories. Relative to models such as CUDA[9] (Compute Unified Device Architecture) and Streaming,[10] our simple hardware support for lightweight on-chip interprocessor synchronization and communication provides a straightforward approach to programming a massive array of processors. Rather than limit implementation to off-the-shelf embedded ASIC tools, we also investigated more exotic technologies, such as silicon photonic interconnects.

Cost is and will continue to be a critical driver in evolving new technologies. The scientific computing community cannot sustain the end-to-end cost of developing and maintaining technologies that apply only to the narrow market of leading-edge high-performance computing systems. Broad-based market support is a prerequisite to make such an ecosystem both practical and sustainable. We believe that our decision to draw from the embedded computing industry will produce technology that reduces economic and manufacturing barriers to constructing computing systems useful to science. It will also ensure that selected technologies have broad market impact for everything from the smallest handheld to the largest supercomputer. The investment will thus be the center of a sustainable software-hardware universe supported by applications across the IT industry.

For the past decade, the current methodologies of message-passing interfaces and Fortran have adequately served the development of high-performance computing applications. But parallelism is no longer an exotic problem. It is an industry-wide challenge that affects everything from cell phones to data centers. Future hardware constraints and growth in explicit on-chip parallelism will require a mass migration to new algorithms and software architecture—a migration as broad and disruptive as that from vector to parallel computing systems.

Green Flash represents a radical approach that breaks through the slow pace of incremental change. It demonstrates that application-driven computing design can foster a sustainable hardware-software ecosystem with broad-based support across the IT industry. In evolving Green Flash, we explored practical advanced programming models together with lightweight hardware support mechanisms that allow programmers to use massive on-chip concurrency.

Green Flash has provided insights into how designers can evolve massively parallel chip architectures through a feedback path that closely couples application, algorithm, and hardware design. Application-driven design ensures that hardware design is not driven by reactions to hardware constraints—reactions that ignore programmability and delivered application performance. Our exploration of the climate model allowed us to investigate questions that cut across all application areas and have ramifications for the next generation of fully general-purpose architectures. Ultimately, we envision an architecture that can exploit reusable components from the mass embedded computing market while improving programmability for a many-core design. The future building blocks of a high-performance computing system will serve the performance and programmability needs of the smallest high-performance, energy-efficient embedded system all the way to extreme-scale machines. C

Acknowledgments

We thank Mark Horowitz and the rest of the Smart Memories Team of Stanford University for early support and advice. We thank the Berkeley Wireless Research Center for early and ongoing assistance with the RAMP platform. We thank Dave Randall's modeling group in the Department of Atmospheric Science at Colorado State University for early access to their icosahedral model. Finally, we would like to acknowledge the Berkeley ParLab and the View from Berkeley discussion that led us to take this direction for HPC architecture. All authors from LBNL were supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231.

References

1. Intergovernmental Panel on Climate Change (IPCC); www.ipcc.ch/publications_and_data/publications_and_data_reports.htm.
2. M. Wehner, L. Oliker, and J. Shalf, "Towards Ultra-High Resolution Models of Climate and Weather," *Int'l J. High Performance Computing Applications*, Apr. 2008, pp. 149-165.
3. Tensilica Inc., "Hardware and Software Development Tools;" www.tensilica.com/products/hw-sw-dev-tools.htm.
4. S. Williams et al., "Scientific Computing Kernels on the Cell Processor," *Int'l J. Parallel Programming*, vol. 35, no. 3, 2007, pp. 263-298.
5. G. Hendry et al., "Analysis of Photonic Networks for a Chip Multiprocessor Using Scientific Applications," *Proc. Int'l Symp. Networks-on-Chip* (NOCs 09), 2009; www.cs.columbia.edu/~luca/research/pnocs_NOCS09.pdf.
6. Research Accelerator for Multiple Processors (RAMP); http://ramp.eecs.berkeley.edu.
7. D. Burke et al., "RAMP Blue: Implementation of a Manycore 1008 Processor FPGA System," *Proc. Reconfigurable Systems Summer Institute 2008* (RSSI 08), July 2008; www.rssi2008.org/proceedings/papers/presentations/19_Burke.pdf.
8. "Into the Wide Blue Yonder with Blue Gene/L," *Science and Technology Rev.*, Lawrence Livermore National Laboratory, Apr. 2005; www.llnl.gov/str/April05/Seager.html.
9. Nvidia Inc.; www.nvidia.com/object/cuda_home.html.
10. AMD/ATI Inc., "ATI Stream Software Development Kit;" http://developer.amd.com/gpu/ATIStreamSDK.

David Donofrio *is a computer systems engineer at Lawrence Berkeley National Laboratory and a member of the Science Driven System Architecture Group at the National Energy Research Supercomputing Center. His research interests include computer architecture and multicore performance optimization. Donofrio received a BS in computer engineering from Virginia Tech. He is a member of the IEEE Computer Society. Contact him at ddonofrio@lbl.gov.*

Leonid Oliker *is a staff computer scientist in the Future Technologies Group at Lawrence Berkeley National Laboratory. His research interests include supercomputing evaluation, multicore autotuning, and power-efficient computing. Oliker received a PhD in computer science from the University of Colorado at Boulder. Contact him at loliker@lbl.gov.*

John Shalf *is a staff computer scientist at Lawrence Berkeley National Laboratory and the Group Leader of the Science Driven System Architecture Group of the National Energy Research Supercomputing Center and leads the Green Flash Project. His research interests include computer architecture, programming models, and frameworks for large-scale scientific application development. Shalf received a degree in electrical engineering from Virginia Tech. He is a member of the IEEE Computer Society. Contact him at jshalf@lbl.gov.*

Michael F. Wehner *is a member of the Scientific Computing Group at the Lawrence Berkeley National Laboratory. His research interests include the design of global climate models and the analysis of their output. Wehner received a PhD in nuclear engineering from the University of Wisconsin-Madison. Contact him at mfwehner@lbl.gov.*

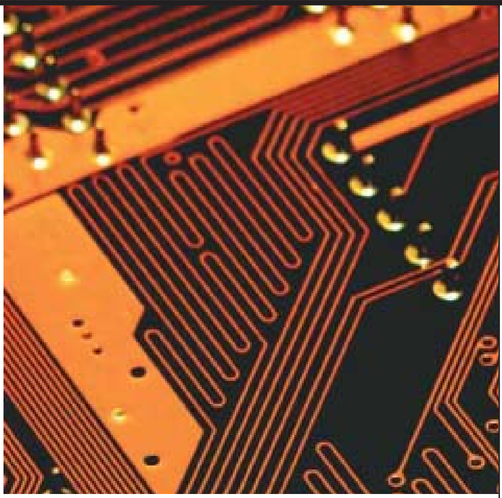
Chris Rowen *is the CTO and founder of Tensilica. His research interests include computer architecture, parallel programming models, and energy-efficient hardware architecture. Rowen received a PhD in electrical engineering from Stanford University. Contact him at rowen@tensilica.com.*

Jens Krueger *is a research assistant at the Fraunhofer Institute for Industrial Mathematics and a visiting researcher at the Lawrence Berkeley National Laboratory Future Technologies Group. His research interests include high-performance and scientific computing focused on future hardware architectures and code optimization. Contact him at jtkrueger@lbl.gov.*

Shoaib Kamil *is a PhD student in the Department of Computer Science at the University of California, Berkeley. His research, conducted with the Future Technologies Group at Lawrence Berkeley National Laboratory, is in autotuning, power efficiency, and interconnect optimization. Contact him at skamil@eecs.berkeley.edu.*

Marghoob Mohiyuddin *is a PhD student in the Department of Electrical Engineering and Computer Science at the University of California, Berkeley, and a member of the Future Technologies Group at Lawrence Berkeley National Laboratory. His research interests include computer architecture and performance tuning for scientific computing. Contact him at marghoob@eecs.berkeley.edu.*

RESEARCH FEATURE



Commitment-Based Service-Oriented Architecture

Munindar P. Singh, *North Carolina State University*
Amit K. Chopra, *Università degli Studi di Trento, Italy*
Nirmit Desai, *IBM India Research Labs*

Existing service-oriented architectures are formulated in terms of low-level abstractions far removed from business services. In a new SOA, the components are business services and the connectors are patterns, modeled as commitments, that support key elements of service engagements.

The vision of service-oriented computing (SOC) promises the creation of a dynamic Web of value. According to this vision, anyone desiring to offer something of value can create and deploy a corresponding service; anyone wishing to benefit from that value can simply select one or more services and compose them into a desired application—or another service.

Current service-oriented architectures (SOAs) purport to support the SOC vision, but what they realize is fundamentally more limited than the vision. The SOC vision implies that services are business services. However, current SOAs interpret services narrowly—as surrogates for computational objects. Whereas business services are *engaged* (often involving subtle business considerations), objects are *invoked* (with business considerations hidden within computational artifacts). More importantly, business services are usually autonomous entities that come together in a service engagement.

Consider the familiar purchase scenario as modeled in leading SOA approaches. Purchasing, say, books is a business service that combines individual services such as placing an order, paying, and shipping. Different organizations could provide these services.

Business Process Modeling Notation (BPMN; http://bpmn.org) and the Business Process Execution Language (BPEL; http://docs.oasis-open.org/wsbpel/2.0) represent composed services as processes specified via control and data flows over tasks (the differences between BPMN and BPEL are syntactic; http://bpmn.org/Documents/Mapping%20BPMN%20to%20BPEL%20Example.pdf). For example, BPMN would model a purchase as three tasks—ordering, paying, shipping—where control and data (book identifier and price) flow from ordering to both paying and shipping.

The Choreography Description Language (WS-CDL; www.w3.org/TR/ws-cdl-10), another leading SOA approach, specifies how services exchange messages. Unlike procedure calls, messaging decouples the parties involved and is thus better suited for distributed systems. WS-CDL would specify how the ordering service sends messages to the paying and shipping services, which perform their work upon receipt of such messages. Declarative approaches for constraining task or message order and occurrence improve modularity and inspectability[1,2] but continue to emphasize control and data flow.

Published by the IEEE Computer Society

COMMITMENTS AND CSOA BENEFITS

In contrast to existing approaches, commitment-based SOA (CSOA) gives primacy to service engagements' *business meanings*, which it captures through participants' *commitments* to one another. CSOA constrains tasks or messages only when doing so affects the business meaning. Computationally, it represents each participant as an agent; interacting agents carry out a service engagement by creating and manipulating commitments to one another.

Commitments

A commitment relates three parties: a *debtor* who is committed to a *creditor*, typically within the scope of an organizational *context*. The context may be an institution—for example, a marketplace such as eBay or a legal jurisdiction such as California—in which the interaction occurs. Institution members who fail to discharge their commitments risk sanction. The Uniform Commercial Code (UCC; www.law.cornell.edu/ucc), which applies in many US jurisdictions, dictates conditions such as when a customer need not pay for purchased goods—for instance, if the goods arrive damaged and the customer returns them immediately. In general, the context is crucial in handling exceptions, which are rife in business settings. For modeling purposes, CSOA treats the context as an agent in its own right.

Importantly, commitments can be manipulated, which supports flexibility. A debtor may *create* a commitment, thus activating it, or *discharge* it, thus satisfying it. Given a commitment, its creditor may *assign* it to a new creditor and its debtor may *delegate* it to a new debtor. A debtor may cancel a commitment, whereas a creditor may *release* the debtor from the commitment.

CSOA benefits

CSOA thus offers the following specific benefits.

Enactment and compliance. Service enactments can be judged correct as long as the parties don't violate their commitments. This notion of correctness enhances flexibility by expanding the operational choices for each party.[3] For example, if the customer substitutes a new way to make a payment or elects to pay first, no harm is done because the behavior is correct at the business level. The seller can employ a new shipper; the buyer can return damaged goods for credit; and so on. Conversely, a customer would be in violation if he keeps the goods but fails to pay. Thus, commitments support business-level compliance without dictating specific operationalizations:[4] Without business meaning, exercising such flexibility could cause noncompliance.

Specification and composition. Commitment-based specifications explicitly reflect business requirements, which are natural for stakeholders. For example, upon placing an order, the customer becomes conditionally committed to the merchant to pay for the goods if they are delivered. The delivery of the goods unconditionally commits the customer to paying for them. When the customer pays, this commitment to pay is discharged. Commitments provide clear conceptual boundaries at which to compose service engagements. For example, we can specify an alternative service engagement that employs independent delivery and payment services. Without business meaning, there would be no basis for establishing that this alternative engagement was valid.

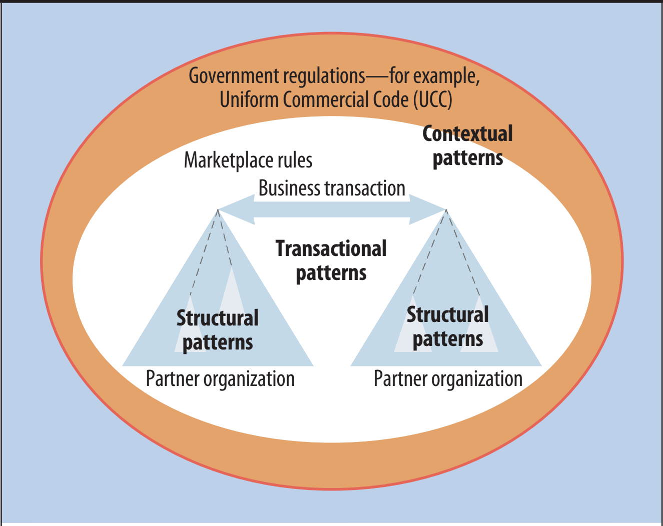


Figure 1. Commitment-based SOA patterns. Transactional patterns refer to the dealings among two or more participants, structural patterns refer to how a participant is organized, and contextual patterns refer to the organizational context in which the service engagement takes place.

CSOA PATTERNS

As Figure 1 shows, CSOA is characterized by a family of reusable *patterns* that form the elements of a service engagement: *Transactional* patterns refer to the dealings among two or more participants; *structural* patterns refer to how a participant, including subcontractors, is organized; and *contextual* patterns refer to the organizational context in which the engagement takes place.

Key CSOA patterns are induced from existing approaches, including UCC, RosettaNet (www.rosettanet.org), the Transaction Workflow Innovation Standards Team (TWIST; www.twiststandards.org), the MIT Process Handbook (MITPH; http://ccs.mit.edu/ph), and extended transaction models.[5] These approaches are not commitment based, but we analyze them via commitments and include the induced patterns within CSOA.

Commitment life cycle

CSOA pattern implementations are expressed as statecharts[6] as shown in Figure 2. Labeled rectangles denote *states*. A state that refines another state is contained within

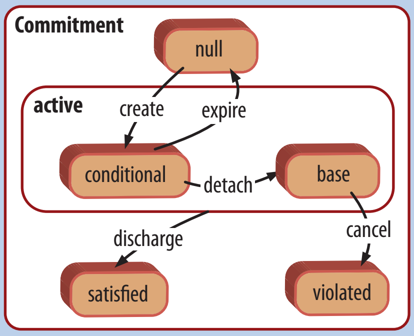


Figure 2. CSOA patterns are expressed in statecharts like this one, which captures a commitment's life cycle.

it. For example, **null** and **active** (containing **conditional** and **base**) are states. An arrow denotes a *transition* wherein the labeled event, if any, occurs. A transition takes the system from one state to the next. When the source state has substates, the transition occurs from each of them—for example, *discharge*.

Figure 2 captures a commitment's life cycle: **null** means it does not exist, **active** means it is fully in force, **satisfied** means it has been discharged, and **violated** means it cannot be discharged. A commitment in **base** may become **violated**; a commitment in **conditional** cannot directly become **violated** but transitions to **null** upon expiration. For example, a customer may offer to buy some goods by creating the commitment "If you ship I will pay." The commitment may expire or the customer may pay. If the merchant delivers, that would detach the commitment, unconditionally committing the customer to pay.

Each commitment has an *antecedent* and a *consequent*. The expression C (debtor, creditor, context, antecedent, consequent) means that the debtor commits to the creditor in the context that if the antecedent becomes true, the debtor would bring about the consequent. When the antecedent holds, the commitment undergoes a detach, meaning that the debtor becomes unconditionally committed to bringing about the consequent. When the consequent holds, the commitment undergoes a discharge. Figure 2 shows detach and discharge as transitions. An **active** commitment must be in either **conditional** or **base**, and this depends solely on whether its antecedent holds (**base**) or not (**conditional**).

Importantly, an agent explicitly performs *create* whereas *detach* and *discharge* occur automatically when antecedent and consequent, respectively, hold; *expire* occurs implicitly upon timeout, but an agent may perform *cancel* explicitly or it may occur via timeout.

Pattern language

Of the 13 attributes in the classical template for design patterns,[7] the following are relevant for CSOA: *classification* (according to Figure 1), *intent*, *motivation*, *applicability*, *consequences*, *implementation*, and *known uses*. A common consequence for CSOA patterns is that the parties involved be proactive and able to communicate flexibly—this is why they are modeled as agents.

The implementation, specified via a statechart, incorporates the *participants* and *structure*. To make the patterns modular, each statechart includes only the relevant states and transitions. (In this sense, our statecharts are not individually complete, and rely upon other patterns to have brought about the states from which they begin.) The commitment operations corresponding to a transition would be realized via business actions such as sending purchase orders, delivering goods, and so on, thereby enacting the corresponding business scenarios.

TRANSACTIONAL PATTERNS

The core of a service engagement is the business transaction that it seeks to accomplish. Transactional patterns describe the corresponding interactions in terms of how the associated commitments are created and manipulated. These patterns deal with common transactional primitives such as initiating a business transaction, formally creating suitable commitments, satisfying the commitments, and possibly updating, retrying, or compensating actions in light of the stated gating conditions. Each transactional pattern involves the same two participants.

We define the commitment life cycle in Figure 2 as the transactional pattern Commit, with the following attributes:

- *Intent:* Expressing an offer.
- *Applicability:* When an offer is made as part of setting up a service engagement.
- *Consequences:* For progress, the creditor should be ready to bring about the antecedent.
- *Known uses:* Purchase, MITPH's Purchase, RosettaNet's Purchase Order (PIP3A4).

Another important transactional pattern is Compensate, which has the following attributes:

- *Intent:* Some business action needs to be undone.
- *Motivation:* A customer sends payment, which commits the merchant to sending the goods; later, if the merchant fails to deliver the goods on time, thus violating its commitment, it must make amends by, for example, refunding the payment.
- *Applicability:* Supporting an extended form of transactional rollback to maintain an all-or-none effect despite exceptions.[5]
- *Consequences:* Typical usage is when the debtor is unable to discharge the original commitment.

- *Implementation:* Upon violation of the original commitment, the transaction requires creating a compensating commitment.
- *Known uses:* RosettaNet's Return Product (PIP3C1).

In the same vein, we can define transactional patterns for real-life cases such as Relieve based on RosettaNet's Purchase Order Cancel (PIP3A9) and the MITPH's Notify, Update based on MITPH's Update and RosettaNet's Purchase Order Change (PIP3A8), and Retry based on MITPH Rework to retry a failed task.

STRUCTURAL PATTERNS

Service engagements involve subtle relationships among the parties involved in a transaction. Structural patterns capture constraints on which party can play which role, or whether a party can delegate or assign certain commitments to another party. Each of these patterns involves two or more participants.

The simplest illustration of a structural pattern is a service engagement involving an organization with an internal structure. A participating organization may delegate its commitments under the engagement to appropriate members that could themselves be organizations. For example, auto insurance companies often delegate their customer service commitments to a regional branch, which might further delegate the commitments to a specific agency.

Composite states help describe patterns involving more than one commitment. For example, in Figure 3, a dotted vertical line separates **Original** and **Delegated**. Thus, if **Original** is in **pending** and **Delegated** is in **active**, the composite state is given by **Original** being in **pending** *and* **Delegated** being in **active**.

The structural pattern Delegate, Retaining Responsibility, shown in Figure 3a, has the following attributes:

- *Intent:* A debtor delegates its commitment but remains responsible for its satisfaction.
- *Motivation:* The merchant delegates its commitment to ship goods to a shipping service but remains committed to deliver the goods to the customer; discharging the delegated commitment discharges the original pending commitment.
- *Applicability:* When the delegatee and creditor don't have a strong business relationship.
- *Consequences:* The creditor is safe because the delegator remains responsible; this pattern enables and coheres with Escalate and Withdraw.
- *Implementation:* **Original** becomes **pending** and **Delegated** becomes **active**.
- *Known uses:* When an insurance company delegates a claimant's auto repair work to a mechanic, it remains responsible if the mechanic fails to make adequate repairs.

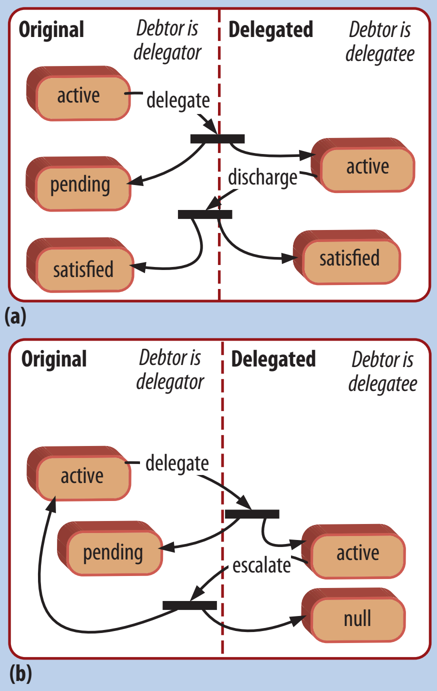


Figure 3. Structural patterns: (a) Delegate, Retaining Responsibility; (b) Escalate (Delegated Commitment). A solid bar with incoming and outgoing arrows is a synchronization primitive: When all events corresponding to the incoming arrows occur, the transitions corresponding to each outgoing arrow also execute.

A related structural pattern is Escalate (Delegated Commitment), shown in Figure 3b, which has the following attributes:

- *Intent:* The failure of a delegatee reactivates the original commitment.
- *Motivation:* If a shipper fails to deliver the goods, the merchant is held responsible.
- *Applicability:* When the delegatee does not provide guaranteed service.
- *Consequences:* The creditor would be the instigator.
- *Implementation:* **Delegated** goes to **null** and **Original** goes to **active**, thus reactivating the original commitment.
- *Known uses:* A customer who pays with a check delegates to the bank his commitment to pay the merchant; if the bank fails to pay—say, because of insufficient funds—the escalation reactivates the customer's original commitment to pay.

Sometimes the delegation transfers responsibility. This corresponds to a variation of the delegation pattern wherein the original commitment simply ends instead of becoming **pending**. Its becoming **null** forecloses the possibility of escalation.

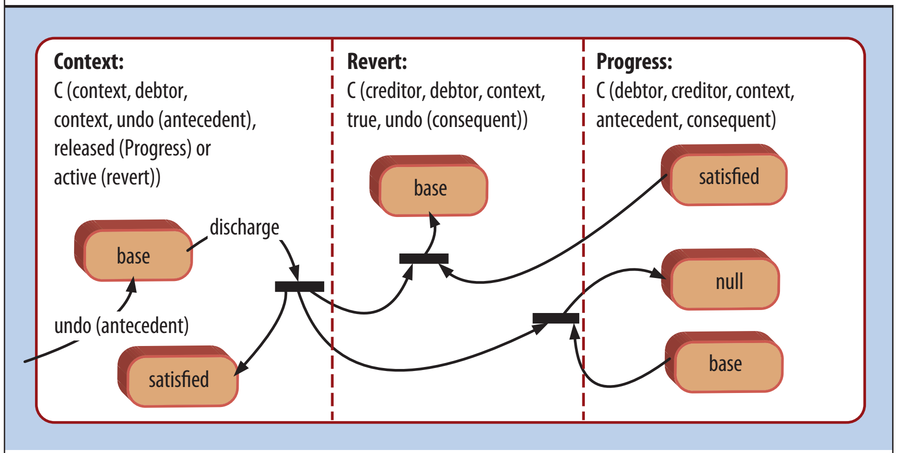

Context:
C (context, debtor, context, undo (antecedent), released (Progress) or active (revert))
Revert:
C (creditor, debtor, context, true, undo (consequent))
Progress:
C (debtor, creditor, context, antecedent, consequent)
base
discharge
undo (antecedent)
satisfied
base
satisfied
null
base

Figure 4. Contextual pattern Revert Ofier. To understand this pattern, imagine a commitment Progress whose debtor is a customer who has received some goods (the antecedent) from a merchant and is therefore committed to paying for them (the consequent). The context is an agency that regulates this service engagement; it commits to the debtor that if the debtor undoes the antecedent (returns the goods) and hasn't already discharged Progress, it is released from Progress (need not pay). Conversely, if the debtor has discharged Progress, then the context activates a commitment Revert that reverses the debtor and creditor roles of Progress: Its debtor is the original creditor who must now undo the original consequent (return the payment).

The structural pattern Transfer Responsibility has the following attributes:

- *Intent:* A debtor nullifies its original commitment by delegating it to another party and is no longer concerned with the delegated commitment's satisfaction or violation.
- *Motivation:* If the customer delegates to his credit card company the payment to the merchant, the subsequent interactions for the payment occur between the company and the merchant; the customer need no longer be involved.
- *Applicability:* When the delegatee and creditor have a strong business relationship.
- *Consequences:* The creditor must accept the delegation and perhaps seek proof that the delegatee accepts it; the delegation may be risky for the creditor.
- *Implementation:* **Original** becomes **null** and **Delegated** becomes **active**.
- *Known uses:* When an airline "endorses" a ticket over to another airline based on a passenger's request, the second airline becomes responsible for transporting the ticketed passenger.

In addition, the structural pattern Withdraw Delegation applies when a delegated commitment is not yet satisfied. It nullifies the delegated commitment and restores the original commitment to **active**. An example is when an airline with an overbooked flight delegates its commitment to transport a passenger to another airline. If the second airline's flight is excessively delayed due to weather, the first airline may reactivate its commitment to transport the passenger.

Yet another structural pattern is Division of Labor, where a service subcontracts a task to two or more other services. This pattern has numerous uses, including RosettaNet's Distribute Work (PIP7B1).

CONTEXTUAL PATTERNS

A service engagement's business context dictates the rules of encounter to which it is subject. For example, eBay users are subject to the online marketplace's terms and conditions, such as that they may not attempt to place false bids. More pertinently, the rules for dispute resolution are also contextual in nature. Each of these patterns involves the three participants—debtor, creditor, and context—with the context explicitly acting as a debtor of a *metacommitment* whose antecedent and consequent involve commitments. The context has the power to create and manipulate commitments among the agents in its scope. Metacommitments provide guarantees to the participants.

In contextual patterns, the context agent itself features as a debtor or creditor. Often in such patterns the context commits to another party such that if some conditions prevail it will cause a specified commitment to transition to a suitable state.

Figure 4 shows the contextual pattern Revert Offer, which has the following attributes:

- *Intent:* To enable a party to back out of a transaction.
- *Motivation:* A customer commits to paying for some goods, which the merchant delivers. If the customer returns the goods before paying, the merchant releases him from paying; if the customer has paid, the merchant refunds the payment.
- *Applicability:* When an agency regulates the service engagement.
- *Consequences:* The context has the means to determine that the requisite conditions hold; it has power over the debtor such as removing it from a marketplace or voiding its license to operate.
- *Implementation:* An *undo(antecedent)* undoes the offer's antecedent. If **Progress** is in **base**, the system releases the debtor—**Progress** becomes **null**—and no further action is needed; if **Progress** is **satisfied**, undo(antecedent) cause the creation of **Revert**.
- *Known uses:* UCC.

An alternative contextual pattern is Penalize, which seeks to punish a party that violates a commitment. For example, if the debtor fails to pay $10 by Monday, the new commitment could be to pay $11 by Tuesday. If the original means commitment delivering the goods, the penalty could mean refunding the deposit and an additional 10 percent—this can be implemented by making a penalty commitment **active**.

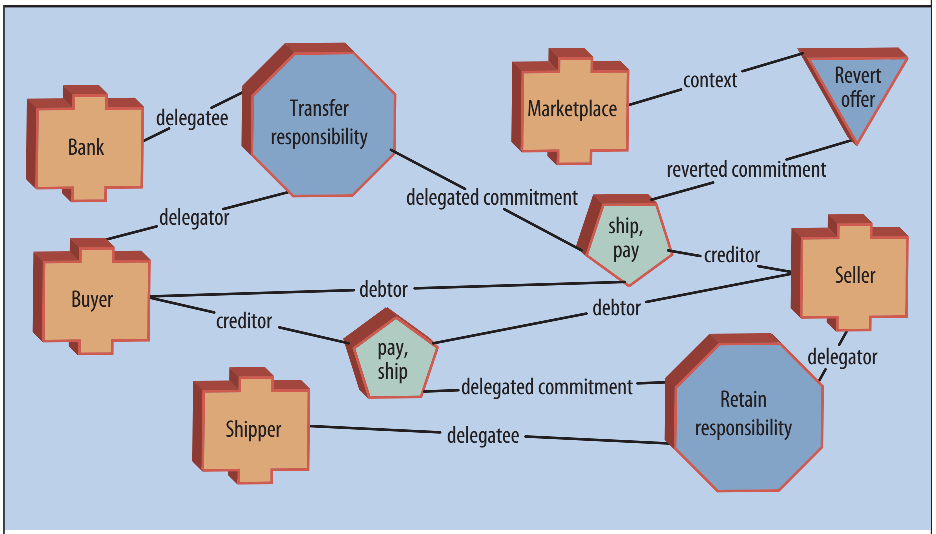


Figure 5. CSOA model of a purchase service engagement. Additional business requirements are accommodated simply by applying additional patterns, while the existing patterns remain as they are.

APPLYING THE PATTERNS

Designing a service engagement using the CSOA patterns requires three steps:

- identify the commitments regarding the services involved,
- apply selected patterns to appropriate commitments, and
- map the operations occurring in the patterns to the engagement's business actions.

Let's revisit our purchase example. We begin with the main partner roles, buyer and seller, and their commitments: The buyer offers to pay if the seller ships him the goods; the seller offers to ship the goods if the buyer pays. Next, we introduce a bank and a shipper: The buyer delegates the payment commitment to the bank, and the seller delegates the shipping commitment to the shipper; the two apply different structural patterns. Last, we apply a contextual pattern enabling refunds upon return.

Figure 5 shows the resulting model, which captures the essential business meaning of the service agreement. Note that additional business requirements are accommodated simply by applying additional patterns, while the existing patterns remain as they are. In some cases, a service engagement may require additional operational constraints, such as that payments should precede shipping.

In contrast, traditional approaches such as BPMN are based solely on operational constraints. The control and data flows to capture the meaning of Figure 5 could be quite complex. Not only do the flows hide the business meaning, they also complicate accommodating additional business requirements: Even a simple change can lead to many additional intricate changes in the existing flows. Further, traditional models lack a formal representation of business meaning, instead relegating meaning to documentation. Modelers need the operationalizations, of course, but should be concerned with business meanings, not low-level operationalizations.

To instantiate an engagement, business partners would adopt the specified roles and perform the services and other business actions specified. The patterns refer to several explicit actions, including *create*, *delegate*, *assign*, *release*, and *update*. Each such action is governed by the corresponding partner's policy; at enactment, such policies determine what computations occur. Our prototype tools map commitment patterns to computations[3] and produce role skeletons, which can be used to implement agents that can participate in an engagement.[4]

CSOA patterns describe abstract possibilities. However, applying the patterns involves matching them to the concrete business realities of a service engagement. For example, a transactional pattern allowing cancellation would make sense only if a commitment can be reasonably canceled. Further, it may not be possible to delegate a commitment if the intended delegatee would not accept the delegation. Finally, the context may not be able to ensure that an agent will discharge any commitments created by the context. In general, CSOA patterns work best when there is a suitable prior business or legal relationship among the parties involved. The patterns can guide the specification of the appropriate relationships or constraints to realize desired service engagements.

ARCHITECTURAL STYLES

An architectural style specifies a family of configurations of *components* and *connectors* subject to stated *constraints*.[8]

In these terms, existing SOAs are an architectural style in which the major components are service provider and consumer, and an invocation protocol serves as connec-

Table 1. Architectural styles of commitment-based SOA versus existing SOAs.

Elements	Existing SOAs	Commitment-based SOA
Components	Service provider and consumer	Business service provider and consumer agents
Connectors	Operations and message patterns (in, out, in-out, out-in)	Commitment patterns
Invariants	Match operation and message signatures	Debtor fl creditor; delegator fl delegatee
Model	Control and data flow	Operations on commitments

tor. (For simplicity, we ignore registries as well as service publication and discovery.) A practical SOA includes specialized components and connectors, such as for resource management and other enterprise functions (identity, billing, and such), and imposes additional constraints so that appropriate components interoperate with each other.

Boualem Benatallah and colleagues proposed patterns called *business-level interfaces and protocols*.[9] However, like WS-CDL and BPEL, their patterns ignore business meanings and thereby lead to rigid interoperation. For example, if a message interface specifies that a customer should make a payment subsequent to the receipt of goods, then a service realizing such an interface must behave accordingly. It ought not to take any liberties such as reversing the messages' order, interposing other messages, or introducing another party such as a payment agency. However, real-life service engagements typically presume such flexibility—thus traditional approaches subvert the SOC vision by creating avoidable friction in the web of value.

The motivation for considering business meaning is to improve the naturalness, maintainability, and reusability of service specifications and the flexibility of enactments. As Table 1, which contrasts commitment-based SOA with existing SOAs,[8] shows, CSOA is not a unique style but has many flavors depending on the patterns selected. Such flexibility is necessary to support the nuances of service engagements. The primary constraint on a sound implementation of CSOA is that at runtime all commitments eventually become **null** or **satisfied**.

The reader may reasonably wonder why, given these differences, CSOA is still a SOA. The answer is twofold. First, CSOA is centered on services and is, arguably, more true to the SOC vision than existing SOAs. Second, CSOA doesn't seek to replace existing SOAs and their implementations. Specifically, the service engagements modeled in CSOA could translate into business processes expressed in BPMN.

A model-driven architecture (MDA; www.omg.org/mda) provides a useful way to think of the relationship between CSOA and existing SOAs. In MDA terms, CSOA is a *computation-independent* model whereas existing SOAs are *platform-independent* models. In other words, the move to CSOA would represent the step—often repeated in computer science—of moving from lower to higher abstractions. Because commitments are computation independent, yet lend themselves to rigorous operationalization, CSOA can help bridge the well-recognized gap between business and IT.[10] Others have begun to recognize the importance of high-level abstractions, but their work still employs operational abstractions (www.ip-super.org).

Santhosh Kumaran[11] presented four abstraction layers for enterprise modeling: strategy (business considerations), operation (business functions conceptualized via tasks and artifacts), execution (analogous to existing SOAs), and implementation. CSOA would help extend Kumaran's operation layer to multienterprise service engagements, and commitment patterns would provide richer representations that facilitate modeling enterprise operations perspicuously and reusably.

Because of the subtleties of real-life service engagements, no small set of patterns would be provably complete. This is analogous to object-oriented design patterns, which are numerous and varied even though the underlying programming languages need only a few primitives.

However, despite their subtlety, service engagements for the most part exhibit regularities in how their transactions, structures, and contexts are applied. Consequently, a reasonably small set of patterns can help describe a large number of practical engagements. Thus, our main contributions are introducing a SOA that gives primacy to business interactions and showing how to formalize the concomitant patterns that provide an expressive vocabulary for modeling service engagements.

Typical service engagement models would include several CSOA patterns applied in routine ways. Thus, *aggregate service patterns*, which capture best practices in designing service engagement, can potentially be abstracted and applied in designing new engagements using CSOA.

References

1. M.P. Singh, "Distributed Enactment of Multiagent Workflows: Temporal Logic for Service Composition," *Proc. 2nd Int'l Joint Conf. Autonomous Agents and Multiagent Systems* (AAMAS 03), ACM Press, 2003, pp. 907-914.
2. W.M.P. van der Aalst and M. Pesic, "DecSerFlow: Towards a Truly Declarative Service Flow Language," *Proc. 3rd Int'l Workshop Web Services and Formal Methods* (WS-FM 06), LNCS 4184, Springer, 2006, pp. 1-23.

3. A.K. Chopra and M.P. Singh, "Contextualizing Commitment Protocols," *Proc. 5th Int'l Joint Conf. Autonomous Agents and Multiagent Systems* (AAMAS 06), ACM Press, 2006, pp. 1345-1352.
4. N. Desai et al., "Interaction Protocols as Design Abstractions for Business Processes," *IEEE Trans. Software Eng.*, Dec. 2005, pp. 1015-1027.
5. A.K. Elmagarmid, ed., *Database Transaction Models for Advanced Applications*, Morgan Kaufmann, 1992.
6. D. Harel, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, June 1987, pp. 231-274.
7. E. Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
8. M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.
9. B. Benatallah et al., "Service Mosaic: A Model-Driven Framework for Web Services Life-Cycle Management," *IEEE Internet Computing*, July 2006, pp. 55-63.
10. H. Smith and P. Fingar, *Business Process Management: The Third Wave*, Meghan-Kiffer Press, 2002.
11. S. Kumaran, "Model-Driven Enterprise," *Proc. Global Enterprise Application Integration Summit* (GIS 04), Integration Consortium, 2004, pp. 166-180.

***Munindar P. Singh** is a professor in the Department of Computer Science at North Carolina State University. His research interests include multiagent systems and service-oriented computing. Singh received a PhD in computer science from the University of Texas at Austin. He is a Fellow of the IEEE. Contact him at singh@ncsu.edu.*

***Amit K. Chopra** is a postdoctoral fellow at the Università degli Studi di Trento, Italy. His research interests include service-oriented architectures and multiagent systems. Chopra received a PhD in computer science from North Carolina State University. Contact him at akchopra.mail@gmail.com.*

***Nirmit Desai** is a research staff member at IBM India Research Labs, Bangalore. His research interests include cross-organizational business processes. Desai received a PhD in computer science from North Carolina State University. Contact him at nirmitv@gmail.com.*

cn Selected CS articles and columns are available for free at http://ComputingNow.computer.org.

CAREER OPPORTUNITIES

THE UNIVERSITY OF PENNSYLVANIA invites applicants for tenure-track appointments in computer science to start July 1, 2010. Tenured appointments will also be considered. The Department of Computer and Information Science seeks individuals with exceptional promise for, or a proven record of, research achievement who will excel in teaching undergraduate and graduate courses and take a position of international leadership in defining their field of study. While exceptional candidates in all areas of core computer science may apply, of particular interest this year are candidates in who are working on the foundations of Market and Social Systems Engineering - the formalization, analysis, optimization, and realization of systems that increasingly integrate engineering, computational, and economic systems and methods. Candidates should have a vision and interest in defining the research and educational frontiers of this rapidly growing field. The University of Pennsylvania is an Equal Opportunity/Affirmative Action Employer. The Penn CIS Faculty is sensitive to "two –body problems" and would be pleased to assist with opportunities in the Philadelphia region. For more detailed information regarding this position and application link please visit: http://www.cis.upenn.edu/departmental/facultyRecruiting.shtml

TEXAS STATE UNIVERSITY – SAN MARCOS, Department of Computer Science. Applications are invited for a tenure-track position at the rank of Assistant, Associate or Professor. Consult the department recruiting page at http://www.cs.txstate.edu/recruitment/ for job duties, required and preferred qualifications, application procedures, and information about the university and the department. Texas State University-San Marcos is an equal opportunity educational institution and as such does not discriminate on grounds of race, religion, sex, national origin, age, physical or mental disabilities, or status as a disabled or Vietnam era veteran. Texas State is committed to increasing the number of women and minorities in faculty and senior administrative positions. Texas State University-San Marcos is a member of the Texas State University System.

THE UNIVERSITY OF PENNSYLVANIA invites applicants for the position of Lecturer in Computer Science to start July 1, 2010.Applicants should hold a graduate degree (preferably a Ph.D.) in Computer Science or Computer Engineering, and have a strong interest in teaching with practical application. Lecturer duties include undergraduate and graduate level courses within the Master of Computer and Information Technology program,(www.cis.upenn.edu/grad/mcit/). Of particular interest are applicants with expertise and/or interest in teaching computer hardware and architecture. The position is for one year and is renewable annually up to three years. Successful applicants will find Penn to be a stimulating environment conducive to professional growth in both teaching and research. The University of Pennsylvania is an Equal Opportunity/Affirmative Action Employer. The Penn CIS Faculty is sensitive to "two –body problems" and would be pleased to assist with opportunities in the Philadelphia region. For more detailed information regarding this position and application link please visit: http://www.cis.upenn.edu/departmental/facultyRecruiting.shtml.

WASHINGTON UNIVERSITY IN SAINT LOUIS, Department of Computer Science and Engineering, Multiple Tenure-Track/Tenured Faculty Positions. The Department of Computer Science and Engineering (CSE) and the School of Medicine (WUSM) are jointly searching for multiple tenure-track faculty members with outstanding records of computing research and a serious interest in collaborative research on problems related to biology and/or medicine. Appointments may be made wholly within CSE or jointly with the Departments of Medicine or Pathology & Immunology. A key initiative in the CSE Department's strategic plan is Integrating Computing and Science. As part of that initiative, we expect to make synergistic hires with a combined research portfolio spanning the range from fundamental computer science/engineering to applied research focused on science or medicine. Specific areas of interest include, but are not limited to: • Analysis of complex genetic, genomic, proteomic, and metabolomic datasets; • Theory/Algorithms with the potential for biomedical applications; • Image analysis or visualization with the potential for biomedical applications; • Databases, medical informatics, clinical or public-health informatics; • Computer engineering with applications to medicine or the natural sciences; • All areas of computational biology and biomedical informatics These positions will continue a successful, ongoing strategy of collaborative research between CSE and the School of Medicine, which is consistently ranked among the top 3 medical schools in the United States. CSE currently consists of 24 tenured and tenure-track faculty members, 71 Ph.D. students, and a stellar group of undergraduates with a history of significant research contributions. The Department seeks to build on and complement its strengths in biological sequence analysis, biomedical image analysis, and biomedical applications of novel computing architectures. Exceptional candidates conducting research in other areas of Computer Science are also encouraged to apply. Washington University is a private university with roughly 6,000 full-time undergraduates and 6,000 graduate students. It has one of the most attractive university campuses anywhere, and is located in a lovely residential neighborhood, adjacent to one of the nation's largest urban parks, in the heart of a vibrant metropolitan area. St. Louis is a wonderful place to live, providing access to a wealth of cultural and entertainment opportunities without the everyday hassles of the largest cities. We anticipate appointments at the rank of Assistant Professor; however, in the case of exceptionally qualified candidates appointments at any rank may be considered. Applicants must have a Ph.D. in computer science, computer engineering, electrical engineering, biomedical engineering, or a closely related field and a record of excellence in teaching and research appropriate to the appointment level. The selected candidate is expected to build an externally-supported research program, teach and mentor students at the graduate and undergraduate levels, and foster interdisciplinary interactions with colleagues throughout the university. Candidates who would contribute to enhancing diversity at the depart-

SUBMISSION DETAILS: Rates are $299.00 per column inch ($320 minimum). Eight lines per column inch and average five typeset words per line. Free online listing on careers.computer.org with print ad. Send copy at least one month prior to publication date to: Marian Anderson, Classified Advertising, Computer Magazine, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; (714) 821-8380; fax (714) 821-4010. Email: manderson@computer.org.

mental and university levels are strongly encouraged to apply. Applications from academic couples are welcomed and encouraged. Qualified applicants should submit a complete application (cover letter, curriculum vita, research statement, teaching statement, and names of at least three references) electronically by following the directions provided at http://cse.wustl.edu/faculty-recruiting/. Other communications may be directed to Prof. Michael Brent, Department of Computer Science and Engineering, Campus Box 1045, Washington University, One Brookings Drive, St. Louis, MO 63130-4899. Applications submitted before January 31, 2010 will receive full consideration. Washington University is an equal opportunity/affirmative action institution and encourages applications from women and minority candidates.

UNIVERSITY AT BUFFALO, THE STATE UNIVERSITY OF NEW YORK, Faculty Position in Computer Science and Engineering. The CSE Department invites excellent candidates in all core areas of Computer science and Engineering, especially experimental and systems areas, to apply for an opening at the assistant professor level. The department is affiliated with successful centers devoted to biometrics, bioinformatics, biomedical computing, cognitive science, document analysis and recognition, high performance computing, and information assurance. Candidates are expected to have a Ph.D. in Computer Science/Engineering or related field by August 2010, with an excellent publication record and potential for developing a strong funded research program. Applications should be submitted by December 31, 2009 electronically via recruit.cse.buffalo.edu. The University at Buffalo is an Equal Opportunity Employer/Recruiter.

BUSINESS ANALYST (NY, NY) for systems work & Oracle IT consulting. Bach.'s or equivalent req'd, plus 1 yr. exper. May involve travel 100% of time and/or relocation. Resumes to Fadel Partners, Job KB, 38 E32nd St., 11th fl., NY, NY 10016.

ASSOCIATE, QUANTITATIVE DEVELOPER. Roc Capital Management LP seeks Associate, Quantitative Developer in New York, NY to design and develop application software for the reconciliation of an international financial and investment services firm utilizing Java, C++ and SQL. Requires Master's degree in Computer Science or related field or equivalent plus 2 years of experience formulating project strategies for application software outlining the steps required to develop system applications utilizing structured analysis and design, and preparing flow charts and diagrams to illustrate sequencing, including program codes, commands and testing applications. Qualified applicants please submit resumes to Roc Capital Management LP by email to recruiting@roccapital.com. Resumes/cover letters must indicate job code 2009IEEE.

HEWLETT-PACKARD COMPANY has an opportunity for the following position in Cupertino, CA. Business Planning Manager. Reqs. knwldge & undrstdg of CKM & boundary processes; exp. using a CRM tool, analyzing data making data-driven decisions, providing presentations, communication across CRM Teams, working with Visio; knwldge of customer data quality; proficiency in MS office. Reqs. incl. Bachelor's deg. or foreign deg. equiv. in Bus. Admin, Bus Mgt, Mgt, Eng, Civil

FACULTY POSITIONS
Computer Science
NYU ABU DHABI

New York University is establishing a new comprehensive liberal arts campus in Abu Dhabi, the capital of the United Arab Emirates. New York University Abu Dhabi (NYUAD) will consist of a highly selective liberal arts college (Arts, Humanities, Social Sciences, and Engineering), distinctive graduate programs, and a world-class Institute for advanced research, scholarship, and creative work. NYU in New York and NYUAD will be integrally connected, together forming the foundation of a unique global network university, actively linked as well to NYU's study and research sites on five continents.

As part of a multi-year hiring plan, NYUAD's Division of Science, Technology, Engineering and Mathematics invites applications for faculty positions in Computer Science, at all ranks, to begin September 2010. We are seeking individuals with a strong record of accomplishment in research and teaching, in any area of computer science, and who have leadership skills to establish a new program in Computer Science. Faculty may spend time at NYU in New York and at its other global campuses. The terms of employment are highly competitive compared to U.S. benchmarks and include housing and educational subsidies for children.

*The deadline for submission is **December 1, 2009***. Applicants must submit a cover letter, curriculum vitae, statement of research and teaching interests, and the names and addresses of three references in PDF format in order to be considered. Please do not submit preprints or publications at this time. Complete instructions for the application process and additional information can be found at **http://nyuad.nyu.edu/human.resources/open.positions.html**. If you have any questions, please e-mail **nyuad.science@nyu.edu**.

NEW YORK UNIVERSITY
ABU DHABI

NYU Abu Dhabi is an Equal Opportunity/Affirmative Action Employer.

جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar
School of Computer Science
مؤسسة قطر
Qatar Foundation

Computer Science Faculty Positions

Carnegie Mellon University in Qatar invites applications for teaching-track positions at all levels in the field of Computer Science. These career-oriented renewable appointments involve teaching international undergraduate students, and maintaining a significant research program. Candidates must have a Ph.D. in Computer Science or related field, substantial exposure to Western-style education, outstanding teaching record and excellent research accomplishments or potential.

Specifically, we are seeking candidates with expertise in databases, data mining, web technology and human-computer interaction. Truly exceptional candidates in other areas also will be considered.

The position offers a competitive salary, foreign service premium, research seed grant, excellent international health coverage and allowances for housing, transportation, dependent schooling and travel.

Carnegie Mellon is internationally recognized as a leader in research and higher education. In 2004, the university established itself in Education City, a state-of-the-art campus that is home to six top universities. Collaboration opportunities with internationally-known researchers and world-class businesses are abundant.

For further information or to apply, visit
http://www.qatar.cmu.edu/cs/positions/

Smartphone Programming in the Classroom

The **Centre for Mobile Education and Research (CMER) Academic Kit** provides the materials you need to integrate mobile devices into the CS/IT/Engineering curricula. The kit includes:

- A 12-week course on mobile application development with Java ME and BlackBerry API.
- A 6-week course on mobile Web-based application development.
- Five plug-in teaching modules to supplement courses on software engineering, information security, game development, web services, and operating systems.

All courses and modules can be taught across the CS/IT/Engineering curricula and most include: lesson slides, labs, tutorials, quizzes, and assignments.

CMER
Centre for Mobile Education and Research

These resources are for academic use only.

http://cmer.cis.uoguelph.ca

KU THE UNIVERSITY OF KANSAS

Assistant Professor

The University of Kansas Electrical Engineering and Computer Science Department seeks to hire one tenure-track assistant professor to support its computer systems design research area. We are seeking to fill research and teaching needs in digital system design, multi-core architectures, systems-on-chip, electronic system-level design, computer system (hardware and software) synthesis, FPGA synthesis, and high-assurance systems. The successful applicant will have an earned Ph.D. in computer engineering, computer science, or related area. Faculty members in our department are expected to develop nationally recognized research programs while supporting our undergraduate and graduate teaching missions. The appointment will be effective as negotiated. Apply at **https://jobs.ku.edu** and search position no. 00004009. A complete application includes a letter of application, curriculum vita, and the names and addresses of three references. Applications will be reviewed beginning January 15, 2010 and will be accepted until the position is filled.

EO/AA

Eng., Comp Sci or related field & 2 yrs. of related exp. Send resume & refer to job #CUPMKA2. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

UNIVERSITY OF HOUSTON, Department of Computer Science, Chair. The Department of Computer Science at the University of Houston (www.cs.uh.edu) is looking for a new Department Chair. The department is a highly dynamic place on an ascending trajectory with 256 undergraduates, 310 graduate students out of which 88 are PhD students. The University of Houston is located in one of the most vibrant metropolitan areas in the nation. Currently, the Department has 22 tenure-track faculty members, slated to expand to 30 faculty members within the next four years. In FY2009, the department received over $6 M in competitive research funding from federal, state, and corporate sources. All our recently recruited faculty members have federal support for their research, and three are recipients of the prestigious CAREER Award from the National Science Foundation. The Department has strong research programs in Computer Systems (high performance computing, networks, real-time systems, security), Data Analysis (information retrieval, data mining, machine learning) and Computational Life Sciences (biomedical image analysis, bioinformatics, biometrics, graphics). The Department's research is the epitome of innovation, mixing advances in core computer science areas with pace-setting multi-disciplinary programs in computational medicine, biology, and psychology. The combination of fundamental research and innovations has led to numerous local, national and international collaborations, the strongest of which are with the Texas Medical Center. The University of Houston, one of the largest in the nation with over 36,000 students, is located in one of the most vibrant metropolitan areas. Houston, the 4th largest U.S. city, is the epicenter of the energy industry, features the largest medical center in the world, and hosts the Johnson Space Center. The Department's research laboratories have joint programs with laboratories from the local medical schools and hospitals, NASA, and the high-tech industry. The ideal Department Chair candidate should be an established leader in his/her field and widely known in the computer science community and beyond. S/he should have proven managerial and marketing skills running another department or a major lab. S/he should also have excellent people skills and be privy to the academic, fund-raising, and publicity system's inner workings in the United States. The Chair's designated mission would be to further accelerate the department's ascendancy to top ranking positions. Qualified applicants need to apply on-line at http://www.cs.uh.edu/chair-search. A CV, at least six recommendation letters, and a vision statement are required. In the vision statement the applicants should clearly describe their vision for the growth of the department and how their track record will support the University mission. The deadline for submission of all documents (including recommendation letters) is January 31, 2010. However, screening of applications and interviews will be ongoing and applicants are encouraged to apply as soon as possible. Interested applicants may further inquire with the Chair of the Search Committee, Prof. Pavlidis at ipavlidis@uh.edu , 713-743-0101. The University of Houston is an Equal Opportunity/Affirmative Action institution. Minorities, women, veterans and persons with disabilities are encouraged to apply.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY, Faculty Positions. The Department of Electrical Engineering and Computer Science (EECS) seeks candidates for faculty positions starting in September 2010. Appointment would be at the assistant or untenured associate professor level. In special cases, a senior faculty appointment may be possible. Faculty duties include teaching at the graduate and undergraduate levels, research, and supervision of student research. We will consider candidates with backgrounds and interests in any area of electrical engineering and computer science. Faculty appointments will commence after completion of a doctoral degree. Candidates must register with the EECS search website at https://eecs-search.eecs.mit.edu, and must submit application materials electronically to this website. Candidate applications should include a description of professional interests and goals in both teaching and research. Each application should include a curriculum vita and the names and addresses of three or more individuals who will provide letters of recommendation. Letter writers should submit their letters directly to MIT, preferably on the website or by mailing to the address below. Please submit complete application by December 15, 2009. Send all materials not submitted on the website to: Professor W.

Eric L. Grimson, Department Head, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Room 38-401, Cambridge, MA 02139, M.I.T. is an equal opportunity/affirmative action employer.

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY, Department of Computer Science and Engineering, Faculty Positions. The Department of Computer Science and Engineering is one of the largest departments in the School of Engineering. The Department currently has 40 faculty members recruited from major universities and research institutions around the world, with about 1000 students (including 600 undergraduate and 170 postgraduate students). The medium of instruction is English. More information on the Department can be found at http://www.cse.ust.hk/. The Department will have at least two tenure-track faculty openings at Assistant Professor/Associate Professor/Professor levels for the 2010-2011 academic year. We are looking for faculty candidates with interests in multidisciplinary research areas related to computational science and engineering such as bioinformatics and financial engineering. Strong candidates in core computer science and engineering research areas will also be considered. Applicants at Assistant Professor level should have an earned PhD degree and demonstrated potential in teaching and research. Salary is highly competitive and will be commensurate with qualifications and experience. Fringe benefits include medical/dental benefits and annual leave. Housing will also be provided where applicable. For appointment at Assistant Professor/Associate Professor level, initial appointment will normally be on a three-year contract, renewable subject to mutual agreement. A gratuity will be payable upon satisfactory completion of contract. Applications should be sent through e-mail including a cover letter, curriculum vitae (including the names and contact information of at least three referees), a research statement and a teaching statement (all in PDF format) to csrecruit@cse.ust.hk. Priority will be given to applications received by 28 February 2010. Applicants will be promptly acknowledged through e-mail upon receiving the electronic application material. (Information provided by applicants will be used for recruitment and other employment-related purposes.)

ELECTRONIC DATA SYSTEMS, HP Enterprise Services is accepting resumes for the following positions: **INFORMATION SPECIALIST IN RANCHO CORDOVA, CA.** (Ref. # EDSRCRNA1). Conceptualize, design, construct, test, & implement portions of business & tech IT solutions through application of appropriate SW devlpmt life cycle methodology. Requires Master's or foreign degree equivalent in Engineering, Comp Sci, Maths, Info Sys, or related field + 3 yrs exp in job offered, or as a developer, project engineer/technical lead, SW engineer, or related occupation. Object Oriented Analysis; Unified Modeling Language design patterns; Java Server Pages; Struts Framework; JavaScript; and Java Database Connectivity. Please mail resumes with reference # to: Ref. EDSRCRNA1, Jim York, Applications Manager, EDS, HP Enterprise Services, 10888 White Rock Road, Rancho Cordova, CA 95670. No phone calls. Must be legally authorized to work in the U.S. without sponsorship. EOE. **TESTING SPECIALIST IN MOUNTAIN VIEW, CA** (Ref. # EDSMOUCAR1). Under minimal direction, utilize appropriate testing methodology & apply specialization to develop test plan for test level to be executed for project, as

Advertisement for Researchers and Post-Doctoral Fellows

The Advanced Digital Sciences Center (ADSC) invites applications for full-time research positions in Singapore. From its space in Singapore's newest science and engineering research complex at Fusionopolis, ADSC is led by outstanding faculty from the College of Engineering at the University of Illinois at Urbana-Champaign, under funding provided by Singapore's Agency for Science, Technology and Research (A*STAR). ADSC's signature project is the Human Sixth Sense Programme (HSSP), which addresses the seamless integration of man, machine and the environment in the digital age. Technology innovations from ADSC will provide many exciting opportunities for new corporate spin-offs and economic development.

Research areas of interest include communications, networking and control; computer systems; cyber-physical infrastructures; multimedia and human-machine interfaces; trusted information management; and related application areas. Candidates working in interdisciplinary areas related to these fields are strongly encouraged to apply.

Qualifications for researchers: PhD in Electrical Engineering, Computer Engineering, Computer Science or a closely related field; outstanding academic credentials and demonstrated excellence in research; and the ability to supervise graduate and undergraduate student while working with post-doctoral fellows and other researchers. Applications from senior researchers are especially encouraged. Post-doctoral fellows in the same areas are also sought. Salary is open, based on qualifications. To ensure full consideration, applications must be received by December 1, 2009; however, applications will be accepted until the positions are filled. Interviews may take place during the application period, but final decisions will not be made until after this date. Further information, including instructions for applying, can be found through the ADSC home page, www.adsc.illinois.edu. Questions may be directed to adscjobsearch@csl.illinois.edu. ADSC is an employer committed to diversity and principles of equal opportunity.

Chair of Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT AUSTIN

The University of Texas at Austin seeks an experienced and visionary academic leader with a distinguished record of research for the position of Chair of the Department of Electrical and Computer Engineering. The Department is currently ranked in the top ten of both computer and electrical engineering, and has grown to 67 faculty members whose research interests encompass all aspects of electrical and computer engineering.

The Chair is expected to advance the department's mission and ranking by attracting and mentoring the highest quality faculty, expanding and broadening the Department's interdisciplinary research activities, and initiating innovative educational programs both at the undergraduate and graduate levels, which have 1200 and 700 students, respectively.

The successful candidate for this position will have a Ph.D. with academic achievements commensurate with appointment as a full professor with tenure; a highly distinguished record of research in electrical engineering, computer engineering, or a closely allied field; international recognition in his or her field; and demonstrated leadership and administrative skills and experience.

An application consisting of a curriculum vitae, a concise statement regarding teaching, research, and leadership experience, and the names of at least five references (with titles, affiliations, and complete contact information), should be both e-mailed and mailed to the attention of Ms. Carole Bearden, Department of Electrical & Computer Engineering, Cockrell School of Engineering, The University of Texas at Austin, 1 University Station, Mail code C0803, Austin, Texas 78712; and e-mail *cjjp@mail.utexas.edu.* Applications submitted by December 15, 2009, are strongly preferred; however, applications will be accepted until the position is filled.

Women and minority candidates are encouraged to apply. The University of Texas at Austin is an EO/AA employer. M/F/D/V. This is a security sensitive position and thereby subject to Texas Education Code §51.215. A background check will be required for the final candidate. Discretion will be exercised in an attempt to maintain the privacy of applicants.

THE UNIVERSITY OF TEXAS AT AUSTIN
UT ECE
ELECTRICAL & COMPUTER ENGINEERING

specified in testing strategy for project. Coordinate & collaborate with others in analyzing collected requirements to ensure test plan & identify testing solutions meet customer needs & expectations. Requires Master's or foreign degree equivalent in Electrical Engg, Comp Sci, Comp Engg, Electronic Engg, or related field, plus 4 yrs exp in job offered, or as a Principal IT Application Analyst, Programmer Analyst or related occupation. Will accept Bachelor's or foreign degree equivalent in Electrical Engg, Comp Sci, Comp Engg, Electronic Engg, or related field, plus 6 years post-baccalaureate, progressive experience. SQL, Oracle database, LoadRunner, Quality Center, Quick Test Professional, MS Office (Excel, PowerPoint, Word). Please mail resumes with reference number to: Ref. #EDSMOUCAR1, Paul Schwartz, Technical Delivery Manager, EDS, HP Enterprise Services, 585 South Blvd East, MS 2C, Pontiac, MI 48341. No phone calls. Must be legally authorized to work in the U.S. without sponsorship. EOE.

UNIVERSITY OF SOUTH CAROLINA, Department Chair – Computer Science and Engineering. The Department of Computer Science and Engineering (www.cse.sc.edu) in the College of Engineering and Computing, University of South Carolina, seeks nominations and applications for the position of Department Chair. The Department offers bachelor's degrees in Computer Engineering, Computer Information Systems, and Computer Science, M.S., M.E. and Ph.D. degrees in Computer Science and Engineering, a Master of Software Engineering, and a Certificate of Graduate Studies in Information Assurance and Security. This is an active and engaged Department with 21 faculty members, including 20 with current research funding and 8 NSF CAREER award winners. Enrollment is over 300 undergraduate and 90 graduate students, including more than 50 doctoral students. Applicants must have outstanding leadership and administrative skills, and credentials (including a Ph.D. in computer science, computer engineering, or related field) commensurate with appointment as a full professor with tenure. Nomination letters should include statements regarding the nominee's relevant credentials. Applicants should submit a current resume, a statement of professional interests and vision, and the names, affiliations, and contact information of professional references. Applications will be accepted until the position is filled and should be sent by email to cse-chair-search@cec.sc.edu. The Department is particularly interested in receiving applications from minorities and women. The University of South Carolina is an affirmative action, equal opportunity employer.

IOWA STATE UNIVERSITY. The Department of Electrical and Computer Engineering at Iowa State University (www.ece.iastate.edu) is seeking a distinguished scholar with a record of excellence in research, education, and professional service to be the Department Chair in Electrical and Computer Engineering. Candidates are expected to have an international reputation for research accomplishment, to have spear-headed educational innovations, to have a strong commitment to diversity efforts, and to have demonstrated university and professional community leadership. Candidates also will be expected to demonstrate a commitment to continued excellence in discipline-leading education of undergraduate and graduate students and interdisciplinary research programs. We seek a dynamic, innovative, and collaborative leader with a bold vision for the future of Electrical and Computer Engineering at Iowa State as we expand our research programs in the strategic areas of Bioengineering, Cyber Infrastructure, Distributed Sensing and Decision-making, Energy Infrastructure, and Small-scale Technologies. For more information on the department, please visit www.ece.iastate.edu. The department strategic plan is located at www.ece.iastate.edu/research.html. The successful candidate will possess a PhD, or equivalent terminal degree, in Electrical Engineering, Computer Engineering, or a closely aligned field, and have an exemplary record of achievement in research, teaching, and service at a level commensurate with appointment as a tenured Full Professor. Preferred qualifications include prior budget management experience and demonstrated accomplishments in leadership, team-building, diversity and administration. All offers of employment, oral and written, are contingent upon the university's verification of credentials and other information required by federal and state law, ISU policies/procedures, and may include the completion of a background check. Application and all required materials must be submitted online by 12/15/09 at www.iastatejobs.com/applicants/Central?quickFind=77796. Direct questions and/or nominations to Dr. Gary Mirka, chair of the search committee, at ecpechairsearch@iastate.edu. Review of applicants will begin immediately and we anticipate having a successful candidate in place on July 1, 2010. Iowa State University of Science and Technology is a comprehensive, land grant, Carnegie Doctoral/Research Extensive University with an enrollment of over 25,000 students. Iowa State University is an Affirmative Action employer and will take action to ensure that employment practices are free of discrimination. Women and minorities are highly encouraged to apply for all employment opportunities. Inquiries or questions regarding our non-discrimination policy can be directed to Carla R. Espinoza at (515) 294-6458.

DUKE UNIVERSITY. The Department of Electrical and Computer Engineering at Duke University invites applications for tenure-track faculty positions at all levels. We are interested in strong candidates in all areas of computer engineering. Applications should be submitted online at www.ee.duke.edu/employment. Applications and letters of reference should be received by December 31, 2009. Duke University is an affirmative action, equal opportunity employer.

SOFTWARE ENGINEER (Unisys/Trenton): Design & develop new internal web application & migrations from Access & Oracle to .Net. Work done using VB.NET, Visual Studio.Net, ASP.Net & Visual Source Safe. Reqs: Bach deg in s/w engr'g or comp sci + 6 mos exp in job offered or 6 mos exp as a Web Developer. Exp must incl excellent .Net skills incl VB.NET, Visual Studio.Net, ASP.Net & Visual Source Safe. 40 hrs/wk; Salary commensurate w/ exp. Send resume to: IEEE Computer Society, 10662 Los Vaqueros Circle, Box # COM45, Los Alamitos, CA 90720.

THE HONG KONG POLYTECHNIC UNIVERSITY, Department of Computing. The Department invites applications for Professors in Database and Information Systems / AI and Knowledge Engineering / Computer System and Theory (Algorithms, OS, Computer Language, etc.). The appointees will be required to provide leadership in all aspects of academic activities, develop established or new research areas in the Department, take responsibility for the development of teaching programmes, and strengthen the international network of the Department and the University. Applicants should have a PhD degree in Computer Science and be conversant in other related disciplines, outstanding abilities with good administrative experience as an academic leader, and excellent track record in research and high quality publications. Please visit the website at http://www.comp.polyu.edu.hk for more information about the Department. Salary offered will be commensurate with qualifications and experience. Initial appointments will be made on a fixed-term gratuity-bearing contract. Re-engagement thereafter is subject to mutual agreement. Remuneration package will be highly competitive. Applicants should state their current and expected salary in the application. Please submit your application via email to hrstaff@polyu.edu.hk. Application forms can be downloaded from http://www.polyu.edu.hk/hro/job.htm. Deadline for application is 17 February 2010. Details of the University's Personal Information Collection Statement for recruitment can be found at http://www.polyu.edu.hk/hro/jobpics.htm.

BOISE STATE UNIVERSITY. The Department of Computer Science at Boise State University is seeking a tenure-track faculty member at the assistant-professor level. Candidates specializing in the areas of databases, software engineering, and visualization are especially encouraged to apply. Additional details are found online at: http://coen.boisestate.edu/cs/Opportunities.asp EEO/AA Institution, Veterans preference.

RUTGERS UNIVERSITY, Tenure-Track Position. The Department of Computer Science at Rutgers University invites applications for tenure-track faculty positions at the rank of Assistant, Associate or full Professor, with appointments starting in September 2010, subject to the availability of funds. All areas in experimental computer systems will be considered, but special emphasis will be given to pervasive computing and computer architecture. Applicants for this research/teaching position must, at minimum, be in the process of completing a dissertation in Computer Science or a closely related field, and should show evidence of exceptional research promise, potential for developing an externally funded research program, and commitment to quality advising and teaching at the graduate and undergraduate levels. Hired candidates who have not defended their Ph.D. by September will be hired at the rank of Instructor, and must complete the Ph.D. by December 31, 2010 to be eligible for tenure-track title retroactive to start date. Applicants should go to http://www.cs.rutgers.edu/employment/ and submit their curriculum vitae, a research statement addressing both past work and future plans and a teaching statement along with three letters of recommendation. Applications should be received by January 2, 2010 for full consideration. Rutgers subscribes to the value of academic diversity and encourages applications from individuals with varied experiences, perspectives, and backgrounds. Females, minorities, dual-career couples, and persons with disabilities are encouraged to apply. Rutgers is an affirmative action/equal opportunity employer.

UNIVERSITY OF CALGARY, Department of Computer Science, Assistant Professor Positions. The Department of Computer Science at the University of Calgary seeks an outstanding candidate for a tenure-track position at the Assistant Professor level, in the Information Security area. A second tenure-track position may be available, at the Assistant Professor level, in the area of Databases or Software Engineering. Details for the positions appear at: www.cpsc.ucalgary.ca/career. Applicants must possess a doctorate in Computer Science or a related discipline at the time of appointment, and have a strong potential to develop an excellent research record. The Department is one of Canada's leaders as evidenced by our commitment to excellence in research and teaching. It has an expansive graduate program and extensive state-of-the-art computing facilities. Calgary is a multicultural city that is the fastest growing city in Canada. Calgary enjoys a moderate climate located beside the natural beauty of the Rocky Mountains. Further information about the Department is available at www.cpsc.ucalgary.ca. Interested applicants should send a CV, a concise description of their research area and program, a statement of teaching philosophy, and arrange to have at least three reference letters sent to: Dr. Ken Barker, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, T2N 1N4 or via email to: search@cpsc.ucalgary.ca. The applications will be reviewed beginning November 2009 and continue until the positions are filled. All qualified candidates are encouraged to apply; however, Canadians and permanent residents will be given priority.

STEVENS INSTITUTE OF TECHNOLOGY, Assistant Professor of Computer Science. The Computer Science Department at Stevens Institute of Technology invites applications for a tenure-track position beginning in August 2010. Special consideration will be given to candidates in computer vision, computer graphics, and machine learning at the assistant professor level. However, outstanding applicants at other levels and/or in other areas of Computer Science may also be considered. Applicants are expected to have a Ph.D. in Computer Science or a closely related field, a demonstrated record of excellence in research, and a strong commitment to teaching. A successful candidate will be expected to conduct a vigorous, funded research program and to teach at both the undergraduate and graduate levels. Stevens Institute of Technology is a private university located in Hoboken, New Jersey. The 55-acre campus is on the Hudson river across from midtown Manhattan within a few minutes from NYC via public transportation. Hoboken is a small upscale city, the residence of New Jersey's governor, and the residence of choice for many professionals working in NYC. Faculty live in Hoboken, NYC, and in suburban communities in Northern New Jersey along commuter train lines to Hoboken and NYC. Stevens' location offers excellent opportunities for collaborations with nearby universities such as NYU, Princeton, Columbia, and Rutgers/DIMACS as well as industrial research laboratories such as Bell Labs, AT&T Labs, IBM Research, Google New York, Siemens, and the Sarnoff Corporation. Applications should be submitted electronically at http://www.cs.stevens.edu/Search. Applications should include a curriculum vitae, teaching and research statements, and contact information for at least three references. Candidates should ask their references to send letters directly to the search committee. PDF is preferred for all application materials and reference letters. Further information is provided at the web site. Review of applications will begin on December 1, 2009. Stevens is an Affirmative Action/Equal Opportunity employer.

WINSTON-SALEM STATE UNIVERSITY invites applications for three tenure-track Computer Science positions at the level of Assistant Professor, to begin August 15, 2010. Applicants must have a Ph.D. in computer science, information technology, computer engineering, systems engineering, electrical engineering, industrial engineering, or a related discipline. Teaching and research experience are required. Emphasis for positions include: [1] software engineering—including: programming languages, programming methodologies, software testing and quality; [2] information technology--including: systems administration, security, architecture, operating systems, database, and networking; [3] architecture--including: operating systems, parallel systems, programming languages, and high performance computing. Duties include: undergraduate and graduate teaching, academic advising, supervision of graduate projects/research; acquisition of funding from external sources; conducting research); assisting in curriculum assessment and development, conducting seminars and workshops, and serving on committees. Applications should be submitted via the university online application system at https://jobs.wssu.edu/.

CAREER OPPORTUNITIES

WEB ARCHITECT: Plan architecture of Web based applics to be deployed on Unix & WebSphere utilizing WebSphere Application & Portal Server, IBM DB2, HTML, DHTML, JSP, JDBC, ANT, Maven, K Shell, JACL & JYTHON Unix & Windows 98/00/NT. Reqs MS Comp Sci, Eng or rel. Mail resumes to Parsetek Inc., 13510 Lavender Mist Lane, Centreville, VA 20120.

VIRGINIA TECH, Artificial Intelligence/Machine Learning, Senior Position, Department of Computer Science. The Department of Computer Science at Virginia Tech (www.cs.vt.edu) invites applications for a full-time tenured position at the Professor or Associate Professor rank from candidates in artificial intelligence with particular interests in machine learning, knowledge representation, or data mining. Candidates should have an established record of scholarship, leadership, and collaboration in computing and interdisciplinary areas; demonstrated ability to contribute to teaching at the undergraduate and graduate levels in AI and related subjects; sensitivity to issues of diversity in the campus community; and the skills needed to establish and grow a multidisciplinary research group. CS@VT has over 40 tenure-track research-oriented faculty. PhD production is among the top 30 in the US and annual research expenditures exceed $6 million. There are rich opportunities in a highly collaborative department with strengths in HCI, HPC, CS education, digital libraries, computational biology and bioinformatics. Active interdisciplinary research also explores CyberArts, digital government, problem-solving environments. Emphases on security and personal health informatics are underway in collaboration with the newly formed VT-Carilion Research Institute associated with the VT-Carilion School of Medicine, opening in Fall 2010. CS@VT is part of the College of Engineering (www.eng.vt.edu) in a comprehensive research university with more than 26,000 students. The main campus is in Blacksburg, which is consistently ranked among the country's best places to live (http://www.vt.edu/where_we_are/blacksburg). Salary for suitably qualified applicants is competitive and commensurate with experience. Virginia Tech is an Equal Opportunity/Affirmative Action Institution. Applications must be submitted online to https://jobs.vt.edu for posting #090529. Applicant screening will begin January 15, 2010 and continue until the position is filled. Inquiries should be directed to Dennis Kafura, Hiring Committee Chair, kafura@cs.vt.edu.

UNIVERSITY OF NORTH TEXAS, Department of Computer Science and Engineering, Department Chair. Applications and nominations are invited for the Chair position in the Department of Computer Science and Engineering at the University of North Texas. Candidates must have an earned doctorate in Computer Science and Engineering or a closely related field with a record of significant and sustained research funding and scholarly output that qualifies them to the rank of full professor. Candidates must also demonstrate a record of teaching, research accomplishments, and professional leadership. Preferred: Administrative experience as a department chair or director of personnel working in computer science and engineering; experience in curriculum development; and demonstrated experience mentoring junior faculty. A record of strategic planning and organizational adaptation as well as knowledge of academic standards and procedures required of accrediting agencies is also preferred. The committee will begin its review of the applications on November 1, 2009 and will continue until the search is closed. For additional information and to apply please visit: http://facultyjobs.unt.edu/applicants/Central?quickFind=50503. Additional information about the department is available at www.cse.unt.edu. UNT is an AA/ADA/EOE.

CONSULTANT / DATABASE ADMINISTRATOR wanted f/t in Poughkeepsie, NY. Must have a Bach degree or equiv in Comp Sci or Engg or related & 1 yr exp performing d/base tuning using UDB, or 3 years of undergraduate education w/2 yrs exp performing d/base tuning using UDB. Send resume: Apollo Consulting Services Corp., Recruiting (SVK), 14 Catharine Street, Poughkeepsie, NY 12601.

ADVERTISER INFORMATION • NOVEMBER 2009 • *COMPUTER*

Advertiser	Page
Carnegie Mellon Qatar University	81
CCGrid 2010	27
Classified Advertising	80-86
Cool Chips 2010	19
Cray Inc.	Cover 4
IEEE	14, Cover 3
IEEE Computer Society	Cover 2
New York University Abu Dhabi	81
Services 2010	105
University of Guelph	82
University of Illinois	83
University of Kansas	82
University of Texas at Austin	83

Advertising Personnel

Marian Anderson
Sr. Advertising Coordinator
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: manderson@computer.org

Sandy Brown
Sr. Business Development Mgr.
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: sb.ieeemedia@ieee.org

Advertising Sales Representatives

Recruitment:

Mid Atlantic
Lisa Rinaldo
Phone: +1 732 772 0160
Fax: +1 732 772 0164
Email: lr.ieeemedia@ieee.org

New England
John Restchack
Phone: +1 212 419 7578
Fax: +1 212 419 7589
Email: j.restchack@ieee.org

Southeast
Thomas M. Flynn
Phone: +1 770 645 2944
Fax: +1 770 993 4423
Email: flynntom@mindspring.com

Midwest/Southwest
Darcy Giovingo
Phone: +1 847 498 4520
Fax: +1 847 498 5911
Email: dg.ieeemedia@ieee.org

Northwest/Southern CA
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Japan
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Europe
Hilary Turnbull
Phone: +44 1875 825700
Fax: +44 1875 825701
Email: impress@impressmedia.com

Product:

US East
Dawn Becker
Phone: +1 732 772 0160
Fax: +1 732 772 0164
Email: db.ieeemedia@ieee.org

US Central
Darcy Giovingo
Phone: +1 847 498 4520
Fax: +1 847 498 5911
Email: dg.ieeemedia@ieee.org

US West
Lynne Stickrod
Phone: +1 415 931 9782
Fax: +1 415 931 9782
Email: ls.ieeemedia@ieee.org

Europe
Sven Anacker
Phone: +49 202 27169 11
Fax: +49 202 27169 20
Email: sanacker@intermediapartners.de

Society Supports Software Engineering PE Examination

The IEEE Computer Society, in conjunction with the IEEE-USA, recently announced its partnership with the National Council of Examiners for Engineering and Surveying (NCEES) in support of establishing a Principles and Practice of Engineering examination for the software engineering discipline. The new software engineering PE exam will not be part of the currently existing electrical and computer engineering PE examination and does not replace the current computer engineering module of that exam. The exact specifications of the new software engineering PE exam will be finalized in coming months.

IDENTIFYING A NEED

The NCEES is an organization comprising all engineering and surveying licensing boards in the US and several territories. NCEES develops, scores, and, for many states, administers examinations used for engineering licensure, including the current electrical and computer engineering PE examinations.

For the NCEES to consider initiating a PE examination in a new discipline, at least 10 state licensing boards must submit written requests that demonstrate a need for the examination in their jurisdictions. In addition, no new discipline may be added to the examination program unless there is an Engineering Accreditation Commission (EAC)/ABET (formerly the Accreditation Board for Engineering and Technology)-accredited program in that discipline. ABET is the accrediting agency for all engineering and technology programs in the United States, and the EAC is responsible for engineering programs in particular.

The IEEE-USA Licensure and Registration Committee reports that the amount of examination knowledge content overlap between the existing computer engineering PE examination and the new software engineering examination will be at most 20 percent, since the existing computer engineering examination contains a significant amount of content related to hardware and data communications networking.

SOFTWARE ENGINEERING LICENSURE

Software engineering licensure offers IEEE members in the US a credential that is available to virtually all other engineering disciplines, ranging from mainstream electrical, civil, mechanical, and chemical fields to smaller disciplines such as control systems, fire protection, nuclear power, and naval engineering. The majority of respondents to a 2008 survey conducted by the Computer Society indicated they were in support of licensing software engineers.

Over the past decade, there have been several efforts to establish professional practice licensure for software engineers. In the past, a primary reason that these efforts were not successful was a lack of infrastructure to support licensure in accordance with NCEES policy. Specifically, the absence of a reasonable number of EAC/ABET-accredited programs offering an undergraduate degree in software engineering posed a significant challenge. However, according to ABET, there are now 17 EAC/ABET-accredited software engineering programs in the US. Therefore, the only remaining hurdle facing software engineering licensure is the creation and administration of a software engineering PE examination.

ESTABLISHING REQUIREMENTS

The next phase of developing the exam is a process known as a Professional Activities and Knowledge Study. The PAKS process includes surveys and meetings with licensed engineers who practice software engineering and who will work to create a specification of the content for the software engineering licensure examination. A committee of software engineers will then develop exam questions under the auspices of NCEES. After NCEES receives the committee's software engineering PE exam, each individual licensing board will decide whether or not it will license software engineers in its state or territory.

IEEE-USA will serve as the lead technical society sponsoring the examination, in cooperation with other organizations that include the IEEE Computer Society and the National Society of Professional Engineers.

MOORE ANSWERS SE LICENSING FAQS

With the advent of the licensing of software engineers in many states, a practicing software developer may wonder how his or her career will be affected. MITRE's James W. Moore, the 2009 IEEE Computer Society vice president for professional activities, answers some of those questions.

When will this happen?

Currently, it is estimated that the Principles and Practices exam for software engineering will become available in 2012.

Will my state license software engineers?

If your state is among the 10 states (Alabama, Delaware, Florida, Michigan, Missouri, New Mexico, New York, North Carolina, Texas, and Virginia) that requested the PE exam, it is likely that they will begin using it immediately for the purposes of licensing. Smaller states may fall in line quickly, while bigger states, with the resources to perform independent analysis, may delay adoption or pursue another course.

Will I be able to become licensed?

Each state has its own regulations for licensing, so no one answer is suitable for all. Many states require qualifications that include a BS in an ABET-accredited curriculum, successful completion of the Fundamentals of Engineering examination, verified experience (often four years), and the successful completion of a Principles and Practices exam. If you've already satisfied the first three requirements, then you may be a good candidate for licensure by sitting for the new exam.

Will all software developers need a software engineering license?

The principle behind licensing is to assure the public that those who claim expertise produce results that do not jeopardize public safety, health, and welfare. In the case of those working for sizable companies, the company's resources absorb any liability, and the employees of that company generally do not need to be licensed. Only software engineers offering their services directly to the public would need to be licensed.

Will all software projects require licensed software engineers?

Many software projects would not require the services of licensed engineers. Only software that affects the health, safety, and welfare of the public would require oversight by a licensed engineer.

Will I be able to call myself a "software engineer" if I'm not licensed?

Many states have laws in place to protect words like "engineer" or "architect" (or "realtor" or "cosmetologist"). Typically, such laws state that you *cannot offer your services to the public* using such occupational titles unless you are appropriately licensed.

What if I practice outside the US?

None of this is likely to affect you unless you offer products or services to the US public.

VOLUNTEER OPPORTUNITY

US IEEE members who wish to participate in the software engineering PE exam development effort can volunteer by filling out the online form located at www.ncees.org/volunteer.php. To learn more about licensure and registration, visit the IEEE Computer Society at www.computer.org; the IEEE-USA's Licensure and Registration Committee at www.ieeeusa.org/volunteers/committees/lrc; the National Society of Professional Engineers at www.nspe.org; and the NCEES at www.ncees.org. C

Society Publications Seek Editors in Chief for 2011-2013 Terms

The IEEE Computer Society seeks applicants for the position of editor in chief, serving two-year terms starting 1 January 2011. Prospective candidates are asked to provide (as PDF files) a complete curriculum vitae, a brief plan for the publication's future, and a letter of support from their institution or employer by **1 March 2010**.

For more information on the search process and to submit application materials for the following titles, please contact Hilda Carman (hcarman@computer.org) or Kathleen Henry (khenry@computer.org).

Computer
IEEE Internet Computing
IEEE Micro
IEEE Security & Privacy
IEEE Software
IEEE Transactions on Computers
IEEE Transactions on Visualization and Computer Graphics

QUALIFICATIONS AND REQUIREMENTS

Candidates for any Computer Society editor-in-chief position should possess a good understanding of industry, academic, and government aspects of the specific publication's field. In addition, candidates must demonstrate the managerial skills necessary to process manuscripts through the editorial cycle in a timely fashion. An editor in chief must be able to attract respected experts to the publication's editorial board. Major responsibilities include

- soliciting high-quality manuscripts from potential authors and, with support from publication staff, helping these authors get their manuscripts published;
- identifying and appointing editorial board members, with the concurrence of the Publications Board;
- selecting competent manuscript reviewers, with the help of editorial board members, and managing timely reviews of manuscripts;
- directing editorial board members to seek special-issue proposals and manuscripts in specific areas;
- providing a clear, broad focus through promotion of personal vision and guidance where appropriate; and
- resolving conflicts or problems as necessary.

Applicants should possess expertise recognized by the computer science and engineering community and must demonstrate clear employer support.

REAPPOINTMENTS

Other IEEE Computer Society publications have editors in chief who are currently standing for reappointment to a second two-year term. The IEEE Computer Society Publications Board invites comments upon the tenures of the individual editors.

Editors in chief standing for reappointment to terms in 2011-2012 are

- Isabel Beichl, *Computing in Science & Engineering*;
- Fei-Yue Wang, *IEEE Intelligent Systems*;
- Beng Chin Ooi, *IEEE Transactions on Knowledge & Data Engineering*;
- Ramin Zabih, *IEEE Transactions on Pattern Analysis & Machine Intelligence*; and
- Liang-Jie Zhang, *IEEE Transactions on Services Computing*.

Send comments to Hilda Carman (hcarman@computer.org) or Kathleen Henry (khenry@computer.org). C

Computer Society Transactions Name Four New EICs

Four top computer professionals will begin terms in January as editors in chief of IEEE Computer Society publications.

Kevin Skadron will assume the post of EIC of *IEEE Computer Architecture Letters*. Skadron cofounded *CAL* in 2001. He has served on the University of Virginia's computer science faculty since 1999 and served as a visiting professor for Nvidia Research.

Skadron holds a PhD in computer science from Princeton University, and has authored or coauthored more than 100 peer-reviewed articles. He is an associate editor for *IEEE Micro*.

Bashar Nuseibeh takes over as the new EIC of *IEEE Transactions on Software Engineering*. Nuseibeh received a PhD in software engineering from Imperial College London and is a professor of computing at the UK's Open University and a visiting professor at Imperial College London and the National Institute of Informatics in Japan.

Nuseibeh currently serves on the editorial boards of the *Requirements Engineering Journal* and several other international journals.

Ivan Stojmenovic, a professor of information technology and engineering at the University of Ottawa, was recently named EIC of *IEEE Transactions on Parallel and Distributed Systems*. He holds a PhD in mathematics from the University of Novi Sad and the University of Zagreb.

Stojmenovic, an IEEE Fellow, has published more than 250 papers and edited four books on wireless ad hoc and sensor networks and applied algorithms.

Ravi Sandhu is the incoming EIC of *IEEE Transactions on Dependable and Secure Computing*. Sandhu, who received a PhD in computer science from Rutgers University, is the founding executive director of the Institute for Cyber Security at the University of Texas at San Antonio.

Sandhu has written more than 180 technical papers, is the founding editor in chief of *ACM Transactions on Information and System Security*, and serves as chair of ACM Sigsac. C

CALLS FOR ARTICLES FOR IEEE CS PUBLICATIONS

IT Professional seeks papers related to all aspects of IT asset management.

The goals of ITAM are to uncover savings through process improvement, support strategic decision making, gain control of and manage inventory, and reduce risk through standardization and loss detection.

The guest editors of *IT Professional*'s July/August 2010 issue welcome case studies, articles on best practices, experience reports, and research summaries relating to ITAM.

Articles are due by **1 December**. Visit www.computer.org/portal/web/computingnow/itcfp4 to view the complete call for papers.

IEEE Internet Computing seeks articles for a November/December 2010 special issue on overcoming information overload issues.

Internet users today are inundated with information. They receive masses of e-mail, are interrupted by instant messages, and must remember to check social-networking sites, news sources, and company websites daily—or even many times each day. Web searches produce more hits than users can sift through.

Managing so much information is a very complex task. Syndication technology—such as RSS and Atom—and feed readers might provide some support, but issues related to the analysis, classification, evolution, retrieval, and other information are open problems.

This special issue seeks original articles examining the state of the art, open problems, research results, tool evaluation, and future research directions in overcoming information overload. Appropriate topics include building and managing information repositories; retrieving, aggregating, and visualizing information; extracting, matching, classifying, clustering, and measuring similarity; analyzing natural language and indexing applied to information; and syndication technology and feed readers.

Final submissions are due by **1 March 2010**. Visit www.computer.org/portal/web/computingnow/iccfp6 to view the complete call for papers.

CALLS FOR PAPERS

EMS 2010, Int'l Conf. on Eng. Management and Service Sciences, 19-21 September 2010, Shenzhen, China; abstracts due **10 March 2010**; www.scirp.org/conf/ems2010/CallForPapers.aspx

CALENDAR

DECEMBER 2009

1-4 Dec: CloudCom 2009, First Int'l Conf. on Cloud Computing, Beijing; www.cloudcom.org

1-4 Dec: RTSS 2009, IEEE Real-Time Systems Symp., Washington, D.C.; www.rtss.org

6-9 Dec: ICDM 2009, Int'l Conf. on Data Mining, Miami; www.cs.umbc.edu/ICDM09

7-10 Dec: WVM 2009, Winter Vision Meetings, Snowbird, Utah; http://vision.cs.byu.edu/wvm2009

7-11 Dec: APSCC 2009, IEEE Asia-Pacific Services Computing Conf., Singapore; http://apscc09.i2r.a-star.edu.sg

9-11 Dec: e-Science 2009, IEEE Int'l Conf. on E-Science, Snowbird, Utah; www.oerc.ox.ac.uk/ieee

13-14 Dec: ETT 2009, Int'l Conf. on Education Technology and Training, Sanya, China; www.isec-edu.hk/ett2009

14-15 Dec: SOCA 2009, IEEE Conf. on Service-Oriented Computing and Applications, Taipei, Taiwan; www.iis.sinica.edu.tw/soca09

14-16 Dec: ISM 2009, Int'l Symp. on Multimedia, San Diego; http://ism2009.eecs.uci.edu

14-17 Dec: ISSPIT 2009, IEEE Int'l Symp. on Signal Processing and Information Technology, Ajman, United Arab Emirates; www.isspit.org/isspit/2009

→ SUBMISSION INSTRUCTIONS

The Call and Calendar section lists conferences, symposia, and workshops that the IEEE Computer Society sponsors or cooperates in presenting.

Visit www.computer.org/conferences for instructions on how to submit conference or call listings as well as a more complete listing of upcoming computer-related conferences.

14-18 Dec: ADCOM 2009, IEEE Int'l Symp. on Signal Processing and Information Technology, Bangalore, India; www.adcom2009.com

16-19 Dec: HiPC 2009, IEEE Int'l Conf. on High-Performance Computing, Kochi, India; www.hipc.org/hipc2009

JANUARY 2010

9-14 Jan 2010: HPCA 2010, IEEE Int'l Symp. on High-Performance Computer Architecture, Bangalore, India; www.hpcaconf.org

MARCH 2010

1-6 Mar: ICDE 2010, Int'l Conf. on Data Engineering, Long Beach, California; www.icde2010.org

29 Mar-2 Apr: PerCom 2010, Int'l Conf. on Pervasive Computing and Communications, Mannheim, Germany; www.percom.org

APRIL 2010

12-16 Apr: DIGITEL 2010, IEEE Int'l Conf. on Digital Game and Intelligent Toy-Enhanced Learning (with WMUTE), Kaohsiung, Taiwan; http://digitel2010.cl.ncu.edu.tw

12-16 Apr: WMUTE 2010, IEEE Int'l Workshop on Wireless, Mobile, and Ubiquitous Technology in Education, Kaohsiung, Taiwan; http://wmute2010.cl.ncu.edu.tw

19-23 Apr: IPDPS 2010, IEEE Int'l Parallel & Distributed Processing Symp., Atlanta; www.ipdps.org

IPDPS 2010

IPDPS is an international forum for engineers and scientists from around the world to present their latest research findings in all aspects of parallel computation. In addition to technical sessions of submitted paper presentations, the meeting offers workshops, tutorials, a PhD forum, and commercial presentations and exhibits.

Topics set to be addressed at IPDPS 2010 include parallel and distributed algorithms, focusing on such issues as stability, scalability, and fault-tolerance of algorithms and data structures for parallel and distributed systems.

IPDPS is sponsored by the IEEE Computer Society Technical Committee on Parallel Processing. IPDPS takes place from 19-23 April 2010 in Atlanta. Visit www.ipdps.org for complete conference details.

EVENTS IN 2009

December 2009

1-4	CloudCom 2009
1-4	RTSS 2009
6-9	ICDM 2009
7-10	WVM 2009
7-11	APSCC 2009
9-11	e-Science 2009
13-14	ETT 2009
14-15	SOCA 2009
14-16	ISM 2009
14-17	ISSPIT 2009
14-18	ADCOM 2009
16-19	HiPC

January 2010

9-14	HPCA 2010

March 2010

1-6	ICDE 2010
29 Mar–2 Apr	PerCom 2010

April 2010

12-16	DIGITEL 2010
12-16	WMUTE 2010
19-23	IPDPS 2010

Computer and Information Security Handbook, John R. Vacca, ed. This book helps readers analyze risks to their networks and defines the steps needed to select and deploy the appropriate countermeasures for reducing exposure to physical and network threats. It also imparts the skills and knowledge needed to identify and counter some fundamental security risks and requirements, including Internet security threats and measures.

This book describes the essential knowledge and skills needed to select, design, and deploy a public-key infrastructure to secure existing and future applications. Chapters contributed by leaders in the field cover the theory and practice of computer security technology, helping the reader develop a new level of technical expertise. This book's up-to-date coverage of security issues facilitates learning and can help readers remain current and fully informed from multiple viewpoints.

Morgan Kaufmann; www.elsevierdirect.com; 978-0-12-374354-1, 844 pp.

The Business of IT: How to Improve Service and Lower Costs, Robert Ryan and Tim Raducha-Grace. IT organizations have achieved outstanding technological maturity, but many have been slower to adopt world-class business practices. This book provides IT and business executives with methods to achieve greater business discipline throughout IT, collaborate more effectively, sharpen focus on the customer, and derive greater value from IT investment.

The authors focus on four specific business practice areas that relate to improving IT service management, managing services' cost and value, measuring IT performance with a goal of improving service and lowering cost, and improving customer alignment. Drawing on their experience consulting with leading IT organizations, the authors help IT leaders make sense of alternative ways to improve IT service and lower cost.fl

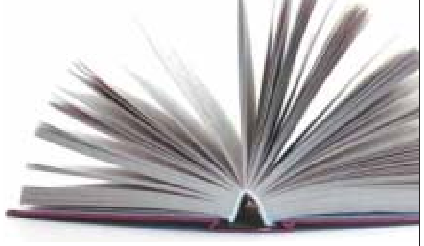
IBM Press; www.ibmpressbooks.com; 013-7-000-618; 292 pp.

The Art of Agent-Oriented Modeling, Leon S. Sterling and Kuldar Taveter. Today, when computing is pervasive and deployed over a range of devices and many users, developers must create computer software that interacts with both the ever-increasing complexity of the technical world and the growing uidity of social organizations. This book presents a new conceptual model for developing open, intelligent, and adaptive software systems. Its approach to modeling complex systems combines people, devices, and software agents in a changing environment sometimes called a distributed sociotechnical system.

Thinking in terms of agents changes how people think of software and the tasks it can perform. Offering an integrated and coherent set of concepts and models, the authors present three levels of abstraction that correspond to the motivation, design, and implementation layers. This book compares platforms by implementing the same models in four different languages and offers exercises suitable for class use or independent study.

MIT Press; mitpress.mit.edu; 978-0-26-201311-6; 408 pp.

Citizen Engineer: A Handbook for Socially Responsible Engineering, David Douglas and Greg Papadopoulos, with John Boutelle. Engineering today requires far more than just being an engineer. It requires considering not only projects' design requirements, but the full impact of one's work—from ecological, intellectual property, business, and sociological perspectives. Increasingly, engineers must also coordinate their efforts with as many as hundreds of other engineers.



This new age demands socially responsible engineering on a whole new scale. The citizen engineer focuses on two topics vitally important in engineers' day-to-day work: eco-engineering and intellectual property. It also examines how and why the world of engineering has changed, and provides practical advice to help engineers of all types master this new era.

Addison-Wesley Professional; www.informit.com; 013-7-143-923; 245 pp.

Mastering Unreal Technology, Volume I, Jason Busby, Zak Parrish, and Jeff Wilson. This introduction to level design with Unreal Engine 3 provides a start-to-finish guide for modding and level design with the engine, thanks to the authors' intimate knowledge of the training modules that shipped with the UT3 engine. Now, working with the full cooperation of Unreal Engine 3's creators, Epic Games, they introduce every facet of game development—from simple level creation to materials, lighting, and terrain.

With tips, hands-on tutorials, and expert techniques, this book can help readers create levels that look spectacular. It also includes tips on understanding the game development process from start to finish; planning projects for greater efficiency, faster delivery, and better quality; and crafting worlds with stunning beauty and clarity.

Addison-Wesley Professional; www.informit.com; 0-672-32991-3; 912 pp.

Send book announcements to newbooks@computer.org.

Managing Interns

Shel Finkelstein
SAP Labs

Internships help students determine their future career paths while providing companies with creative and energetic contributors who offer fresh perspectives and innovative skills.

Internships connect academia and industry. Students learn what it's like in industry by working as team members and contributing to projects. Successful internships can be terrific experiences for everyone—the interns, internship supervisors, universities, companies, and teams.

GOOD MATCHES

Successful internships begin with good matches. Good job descriptions specify job prerequisites clearly. Candidates who already know required languages, frameworks, tools, and technologies become productive much faster than those who need on-the-job training. Internships are relatively brief, perhaps only three months, although some companies can hire interns for up to six months.

Students should be honest about their skills; they will probably be quizzed during interviews by recruiters or potential supervisors about their experiences in courses and previous internships. Most technical interviewers can quickly determine when a candidate has exaggerated substantially, which sends a strong negative signal.

Personality matching—determining if the potential intern and supervisor will work well together—is even more important than specific technical skills. Successful interns are knowledgeable, smart, and creative, but they are also team-oriented, disciplined, enthusiastic, and adaptable. Sure, those are terrific attributes for any team member, whether an intern or not, but the short internship period implies that a strong working relationship must be established quickly.

ORIENTATION

Most companies offer orientation sessions that help new interns get started, perhaps pointing them to informational websites as soon as they accept job offers. All new employees, including interns, submit government and company paperwork (such as tax forms and confidentiality agreements), receive badges and equipment, and learn about the company and group they have joined. Orientations offer important information about company policies, organization, products, and websites, and may help new employees navigate the inevitable flood of corporate buzzwords and acronyms.

Interns are sometimes surprised by the scope of business activities that companies address. But the most crucial aspect of orientation involves fostering personal contacts. Orientation should include introductions to people in the same team and group, which makes it easier to talk to them later. When a company has interns throughout the year, as SAP does, existing interns are the best resources for new interns. These "old hands" recall the challenges they encountered in adapting to a new company and location and are usually eager to help others, just as they were helped when they arrived. Wiki pages describing past intern experiences (resources, activities, advice) also help.

SUPPORT

Every intern has a specific person identified as a supervisor or mentor. That person, who often isn't a manager, must supervise the intern's technical work. Often, the supervisor is one of the people who interviewed the intern when matching was evaluated. But sometimes assignments are modified for business reasons, such as changes in projects and personnel. The intern and supervisor should meet regularly, defining a project, examining approaches, reviewing progress, and discussing results. The supervisor should also explain the "big picture" of what the group is doing, so that the intern can understand how a particular project fits in.

Sometimes several interns work together on a project and help each other. Because communication and teamwork are so important, the intern might also consult other people on the supervisor's team and in the lab as a whole, creating a small network of contacts. Networks of collaborators, advisors, and supporters are vital to employees who work full-time in industry.

In addition to the supervisor, interns usually have a manager, who might be in a human resources group or in the same technical group as the supervisor. The intern manager is responsible for handling corporate matters for groups of interns, such as recruiting, hiring, orientation, payroll, and special intern activities. These activities can include outings, internal seminars, or seminars across multiple local companies. Such cross-company seminars help interns gather additional industry perspectives and learn about career opportunities in the same geographical area.

Ideally, interns and supervisors will share a strong relationship and can deal with any concerns directly, but the intern manager should assist if issues arise that can't be worked out. Both interns and supervisors should feel comfortable talking to intern managers when necessary, so that problems can be addressed early on.

EXPECTATIONS AND DELIVERABLES

Interns are expected to be professional team members who attend group meetings, define their deliverables and schedules with their supervisors and—sometimes—project managers, communicate their findings regularly, and help adjust schedules when necessary. In some cases, supervisors define what they believe are accomplishable units before interns arrive, while some supervisors define projects with interns on the fly, based on how quickly the interns learn and accomplish initial goals.

Projects may be experimental or directed. For experimental projects in research groups, goals might be defined loosely at first, with some room for independence and creativity. Interns should learn applicable internal and external technologies, define a project that achieves something or performs certain experiments, build a prototype (often based on some existing systems), understand and explain what they've learned from their experiments, then document and present their discoveries.

What an intern accomplishes contributes to the supervisor's and project's success, and even to the overall company.

There may be enough time to modify or extend the prototype based on experimental results to do a better job, or to contrast multiple approaches. Sometimes negative results might be as valuable as positive ones, as long as the experiment is a good one and teaches something insightful.

For directed research projects and for most development projects, goals will be more focused and schedule-driven. High-quality results can be crucial, with a series of intermediate milestones defined to monitor progress. It's appropriate to adjust scope and dates for intern assignments when delivery is late or results are disappointing, just as it is for anyone working on a project.

For either research or development, an important deliverable is the end-of-internship presentation. Such presentations give interns the chance to describe their results to the entire group, giving them experience preparing demonstrations and presentations, presenting their findings, and handling questions. Presentations also act as a driving force to deliver functional deliverables and well-grounded conclusions.

SUCCESS FOR EVERYONE

Everyone involved benefits from a successful internship. An intern is a member of a project team, just like other group members. Interns participate in the same meetings and activities as other team members and share responsibility for the success of their project. There are exceptions, however, because the intern is a temporary employee.

Both intern and supervisor learn new approaches and insights during internships. Interns must balance their own creativity with project needs. They must also recognize that they are being paid to contribute and their results affect others. A successful internship can improve the intern's résumé and result in a strong recommendation from the supervisor.

Although interns may focus on their personal achievements, an intern's work also reflects on the intern's university, particularly if the people on the project haven't hired many other interns from that university. Was the student well-educated and motivated? Will the company want to hire other students from that university? Similarly, will other students from the intern's university want to work at that company?

Moreover, the intern's success helps make the supervisor successful. Is the supervisor good at managing interns? If so, the supervisor might be a potential management candidate. Did the intern's project deliver useful results that contribute to the group and the company? If so, the supervisor might continue the project (perhaps with other interns) by writing additional code, internal documents, presentations, external papers, and sometimes patents, giving credit to the intern's contributions. A supervisor who collaborates well with productive interns enhances his or her own reputation, as well as the intern's.

When an intern works on a thesis during an internship, the supervisor must be careful to balance the intellectual property rights of the company with the obligation to help the intern complete a thesis. Intern accomplishments should help determine the company's directions and success, just as any other employee's contribution would. An intern who learns how to work on a team, contribute substantially, and make a difference has acquired experience that will be valued anywhere that intern works in the future, perhaps even at the company where he or she interned. And for companies, internships are an important recruiting technique since supervisors and interns learn firsthand what it's like to work together.

FEEDBACK

Interns are new to their companies and projects and must listen and learn, but different people have different learning curves. Sometimes an intern is highly independent and creative and just needs context, encouragement, and a sounding board to make strong, novel contributions; sometimes an intern has trouble delivering basic functionality and must shift to a more digestible problem and seek extensive support. Most internships fall somewhere in between, and both supervisors and interns need to be patient, open, and adaptable. Even though intern, supervisor, and project all seemed like a great match, problem complexity, technical challenges, or interpersonal issues might cause difficulties.

Such concerns should be addressed early on. The intern and supervisor need to communicate regularly and frankly—just as any employee should communicate directly with his or her manager—so that adjustments can be made as soon as possible. These adjustments can include education and explanations from the supervisor or others, breaking the problem down into simpler pieces, changing the intern's assigned problem, and—with help from management—assigning the intern to a different supervisor. Both the intern and supervisor should request assistance from the intern supervisor when discussing and resolving problems. Feedback should also come from other group members, not just at an end-of-internship presentation, but throughout the internship, as interns describe their results and challenges at team meetings, lunches, or informal discussions with colleagues.

Although discussion and course correction may be required in any supervisory relationship, the great majority of internships go very smoothly, and supervisors should make sure that interns know how much their work is appreciated, both in team meetings and directly. Supervisors can thank interns at end-of-internship presentations; in my group, supervisors provide desserts chosen by the interns as another way of expressing thanks.

Interns are ambassadors from their universities to the companies where they intern. Companies can learn a lot from their interns about what's going on at their universities. After internships end, interns also serve as ambassadors from their internship companies back to their universities. Other students and faculty might want to find out about the companies' and internships' projects and compare the experiences of different interns, which might help them determine if they want to work at those companies as interns or full-time employees.

Internships offer a terrific way for industry and academia to stay connected. Yes, there are many other ways, including exchanges between professors and industry researchers, industry-sponsored academic research programs, conferences, journals, and sabbaticals. But internships offer significantly different advantages. They give students a paid opportunity to learn industrial practices and cultures in what should be a highly supportive team setting geared for success, perhaps in a location that's new and exciting, where they can meet, get to know, and have fun with interns and other employees.

Internships help students determine what they want to do and where they want to do it. For companies, interns provide creative and energetic contributors who offer fresh perspectives and provide innovative skills that help projects meet their goals. When internships are successful—and they usually are—they are a win for everyone involved: students, universities, supervisors, projects, and industry. C

***Shel Finkelstein** is a director in the Office of the Chief Scientist at SAP Labs in Palo Alto. He thanks his colleagues from HP, IBM, PARC, and SAP for contributing thoughtful comments about their experiences. Contact him at shel.finkelstein@sap.com.*

Innovation for the Web 2.0 Era

Miguel Carrero, *Hewlett-Packard*

Open innovation enables today's companies to share the costs of research while capitalizing on the creativity they harness.

We all know the business mantra "grow or die." In technology terms we could translate this as "innovate or die." Throughout the tech industry, the cry to innovate resounds. It's espoused by analysts and demanded in our conference rooms. It leaps from the pages of the marketing collateral that we create and consume. We chase it, covet it, invest in it, all in the hope of meeting the demands of the evolving and ever more competitive marketplace.

Not so long ago, the major technological innovations typically came from large-scale industry operations. A group of bright people working in a well-equipped lab somewhere would investigate the potential of new products or of expanding the capabilities of those already existing. There would be test runs and trials, usually conducted within a closed system. If these went well, new products or services would be offered to the market, which hopefully recognized their merits and adopted them in sufficient numbers to ensure profitability.

However, nowadays innovation happens in many different ways. A key innovation can be based on a novel technology that breaks into new territories or results from intelligently compiling existing pieces in a unique way to provide distinctive value to the market.

Today's environment of scarce capital and extraordinary depth and breadth of technology mandates a new approach: *open innovation*. Open innovation can apply to collaboration across the internal boundaries of a corporation, or between corporations, or to suppliers and clients jointly trailblazing new spaces. In each case the goal is to find synergies in investments and applications of capital as much as brainpower.

Open innovation provides better returns and should minimize the chances of failure. However, it requires a very specific and cognitive approach. This open attitude can be counterintuitive for many brilliant innovators and engineers, and the concept has still not been fully internalized by many industry leaders.

EMBRACING OPEN INNOVATION

It's imperative to both support and practice open innovation. We need to do this not only in how we bring innovation to solutions and services, but also in how we drive and leverage innovation across the wider community.

Working within a corporation the size of HP presents operational complexities, but it also offers a vast array of internal sources of innovation. As an example of how a large company can drive efficiency, HP has instituted several critical practices, including

- tapping into innovative strategies that vertically integrate IT management technology into industry-specific solutions;
- fostering relationships between our business groups and HP Labs to ensure a clear line of communication surrounding technologies coming from R&D;
- collaborating with universities to accelerate HP Labs breakthrough research and bring new technologies to market at a faster pace;
- partnering with our hardware colleagues to get maximum value from the underlying solution hardware infrastructure; and
- collaborating closely with our Personal Systems Group to create solutions that go from the back end to the end user device.

However, no matter its size, no company can do it all. It's important to selectively invest in multiple partnerships; software and hardware houses, for example, make it possible to benefit from new developments coming from specialized players. But open innovation means going one step further—the closer to your customers you can innovate, the more meaningful this innovation will be.

HP uses a "dual path" model for development: one that happens primarily within the company and another that occurs in conjunction with specific customer implementations. We recognize that the answers to today's challenges and tomorrow's opportunities will come from both our own engineers and specialists, as well as from directly working alongside our global customer base to develop capabilities that answer their specific needs. We apply those customer-led innovations to the industry's broader challenges.

The benefits of this approach are evident in several key HP deployments. A good example is Pajama5, a mobile social networking service offering from SK Telecom (SKT), South Korea's leading mobile operator. This service lets a small group of friends instantly see the online status and emotional state of each member, which helps to strengthen their relationships while encouraging increased revenue through enhanced network usage.

The application's ease of use and fostering of closer ties to friends has proven a successful combination. SKT initially developed Pajama5 in-house. Now, HP and SKT are working together to evolve it with open technologies provided by HP to bring this type of innovation to the rest of the world.

ENABLING OPEN INNOVATION

The need for open innovation is especially critical in the communications and media industry, which harnesses a broad range of company types and sizes, from traditional carriers to viral social networking services. It's an industry that uses disparate business models and serves markets that address all customer segments.

This last characteristic—addressing all customer segments—pushes the need for open innovation to the extreme. In our industry, big corporations have been fairly successful sharing both the investment and return for several core aspects of the business. However, some of today's exciting innovators are small developer organizations, "garage" shops, and end users.

Innovation can occur anywhere. The next big ideas—as well as the countless small ones that add functionalities or fill market niches—are just as likely to come from a couple of friends working together in a cramped studio apartment as from a state-of-the-art research lab.

Technologies such as service-oriented architectures and service delivery platforms enable service providers to achieve open innovation.

What's different today is the reach and impact of these developers. The Internet has provided access to resources—knowledge, technology, and so on—as well as access to the market, traditionally a major barrier of entry for small businesses.

In addition, end users are imagining new capabilities for features and technologies. Service mash-ups, which combine service functionalities and whole services in unexpected ways, have expanded the possibilities for communications and media services. Enhanced revenue for service providers has followed. A new set of technologies enables this distributed way of innovating. In a sense, innovation is enabling open innovation.

Technologies such as service-oriented architectures (SOAs) and service delivery platforms (SDPs) enable service providers to achieve open innovation. The SDP serves as a controlled environment for services coming from the network, IT, the Web, or other applications. It provides governance and policy control, while increasing transparency and improving the management of converged services whether they are internal services or applications, from third-party developers, or created by nontraditional developers ("prosumers"). All of these parties can use the services and enablers governed by the SDP to create new services.

Empowering these developers and end users is a win-win for all involved—not only because of the fluid nature of technological advancement, but more importantly because it's the only way to fully tap into the specific needs of niche groups of people (the "long tail"). By the way, many of these supposed niches have proven to captivate the market. Just ask the folks on Facebook and Twitter.

THE EVOLVING SDP

A few years back, it became apparent that the legacy infrastructure used to support communications and media services was showing strains in keeping up with service demand. Utilization of precious network resources was among the key issues. Every time a new service was developed, it essentially required starting from scratch; there was no easy sharing of resources or work previously done. In terms of costs, time frames, and resource optimization, a better way was needed.

The evolution toward a service delivery platform using SOA technologies was a logical response. Providing the ability to abstract network assets to facilitate their sharing across applications, the SDP encouraged dynamic service development and dramatically reduced time to market. For the first time, developers (internal or external) leveraged a common architecture and could securely access network resources without having a deep knowledge of the underlying systems. This early SDP began to address third-party content and services, which was already being recognized as a key driver for customer satisfaction and revenue.

However, as the market continued to evolve, it quickly became apparent that the SDP would have to evolve along with it. Both professional users and consumers at large were demanding vast quantities of content and services. Resource sharing, while undoubtedly useful, was still fairly limited in that crop of SDP technologies. In addition, new critical capabilities were identified, including effective service governance, management, and quality for network, IT, and Web-based services.

Facing the certainty of needing to manage vast numbers of nontraditional mash-up applications, content, and services, HP and its service provider customers recognized that testing and quality assurance would be increasingly important. Related issues of security, privacy, and identity management tempered the opportunities presented by service enablers. Further, the SDP had to be integrated with SOA infrastructure enterprise-wide. Doing so required simplifying and more easily monitoring the end-to-end service life cycle and workflow across the operations/business support system.

In designing its second-generation SDPs, the industry also began to address enhanced services. This encompassed recognition of the digital device along with access networks, subscriber identity profiles, and relevant contextual data. The evolved SDP used this data to leverage the proper infrastructure, formats and protocols, and applications to deliver more relevant content and service offers with improved presentation and usability.

ENHANCING INNOVATION WITH WEB 2.0

Thriving in an evolving marketplace means getting there first with more, doing it consistently, and delivering the best possible service experience every time, all of which requires greater transparency and easier collaboration.

Web 2.0 provides the latest means for doing just that. Web 2.0 offers a powerful collection of capabilities for connecting and empowering individuals, communities, and enterprises; it's about taking open innovation to the nth degree.

Enabling service providers to draw upon prepackaged solutions offers unparalleled opportunities to monetize assets while reducing costs. To help customers capitalize on these opportunities, the industry needed to make sure that the newest generation of SDP allowed the use of multiple business models and enabled more effective collaboration with the developer and content-creation and -aggregation communities.

Achieving open innovation's enormous potential benefits requires tenacity and conviction, but in the end it must prevail.

The service mash-ups that the second-generation SDP began to address are now commonplace in our collaborative culture. We're in a time of incredible dynamism in which the walls that have separated technology innovators and those that use their innovations are truly coming down. Robust, easily integrated service enablers and widgets deliver new and enhanced services.

Increasing use of the Representational State Transfer (REST) architecture is further lowering the barriers to innovation. Environments such as Twitter and Facebook, among others, are built with the REST API, which is simpler than SOAP. Using REST enhances accessibility, expanding the pool of potential innovators.

The latest SDP evolution uses a "RESTful gateway" that automates access to Web tools to help simplify the creation of new services, thereby enabling developers to immediately become more productive. HP has chosen this path because we believe that it will greatly expand developer participation on service provider networks. The strong support of open standards such as REST significantly increases the numbers and types of engaging applications created, and will drive revenue and subscriber loyalty.

The newest crop of SDPs empowers developers and sophisticated end users to create and deploy tomorrow's revenue-producing services. As new marketplace opportunities and technical requirements crystallize, we'll continue to evolve these technologies.

Open innovation is the key to successful collaboration within companies, with partners, and with customers. This approach enables companies to share the costs and risks of research while capitalizing on the creativity they harness.

Make no mistake—convincing brilliant engineers to reuse somebody else's ideas isn't a simple task. But it's important to be able to retain an open attitude that embraces how people and organizations can collectively solve a problem, rather than squabbling over how to split the potential reward. Achieving open innovation's enormous potential benefits requires tenacity and conviction, but in the end it must prevail.

HP continues to develop technologies and solutions that enable open innovation. The newest crop of SDPs, both products and enablers of open innovation, is just entering the marketplace. The quest for innovation is a constant journey, so stay tuned. C

Miguel Carrero *is the director of WW Applications & Emerging Business Service Delivery Infrastructure & Application, Communications and Media Solutions, Hewlett-Packard Company. Contact him at miguel_carrero@hp.com.*

Editor: Simon S.Y. Shim, Dept. of Computer Engineering, San Jose State Univ., San Jose, CA; simon.shim@sjsu.edu

My IT Carbon Footprint

Kirk W. Cameron, *Virginia Tech*

Self-awareness is the first step toward reducing our carbon footprint.

As this is the last Green IT column in 2009, I can't help but reflect on the computing community's achievements toward energy efficiency and sustainability this year. Given all the attention green IT has garnered in the global media, in economic stimulus packages across the world, and in corporate marketing of IT hardware and software, progress clearly has been made even if only to elevate the discourse.

This column's goals are to inform and educate readers in the areas of IT design, deployment and use, and retirement and recycling. With the help of my guest columnists, I have sought to lay a foundation this year upon which we can build in 2010.

As a researcher by trade, however, I wanted to evaluate the past year more quantitatively. I could have interviewed industry leaders or conducted a thorough study of empirical data, but for a change I decided to take a slightly less scientific, yet reasonably insightful, approach.

As a consumer myself, I thought it would be useful to track my own IT carbon footprint. Other researchers have tried to ascertain the carbon footprint of reading a webpage or using a particular system, but I was curious as to how an individual's use of IT in everyday life contributes to consumption. Though my carbon footprint may not be representative of everyone's, it's hardly anomalous either.

My goal was to quantify the IT equipment I use in a typical week at work as well as in my leisure, to answer questions such as: Will modifying my own carbon footprint matter in the grand scheme of things? What potential difference can I make by changing my footprint, or if the industry can improve the technologies I use regularly in the coming years? Is my footprint more behavioral or systemic—am I responsible for more carbon release because of my behavior or my poor choice in the systems I use?

MEASURING MY CARBON FOOTPRINT

How do you measure your personal IT carbon footprint? One way is to monitor the power and time usage of the computer systems you use and the amount of time you use them.

For nearly a decade, our laboratory has been measuring and dissecting power and energy use in computer systems. For this exercise, I simply made a series of direct power measurements using a midrange multimeter, the Watts up? Pro ES. Since my experience has taught me such meters can be inaccurate at times, I took several measurements and performed numerous baseline tests against more accurate meters we own. Nonetheless, without the statistical rigor we normally apply in the laboratory, my measurements are arguably more useful for qualitative discussions than quantitative conclusions.

I mainly tracked usage in the extremes such as idle and max power used. Benchmarking for a particular application was beyond the scope of this study (see the March 2009 Green IT column for details on the art of benchmarking with SPECPower). I was simply trying to gauge typical usage to analyze my carbon footprint.

Note that converting to carbon credits is a simple calculation wrought with assumptions that you *should* mistrust. While I use widely accepted conversion techniques, there are competing methods that could also be applied.

Finally, take this study for what it is: a single data point along the spectrum of IT usage that I think is a reasonable approximation of average for computer practitioners. It's relatively straightforward to extrapolate from this data the use of additional systems and calculate your own IT carbon footprint.

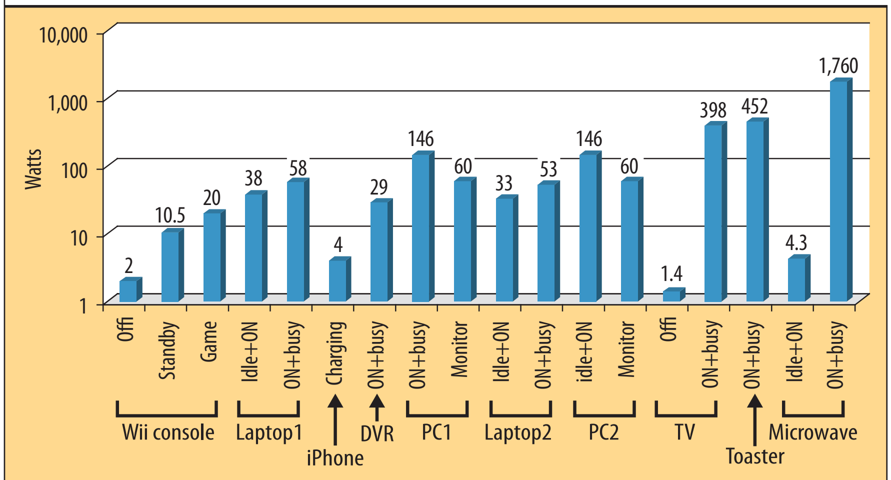


Figure 1. Power consumption of common IT and household devices. System mode clearly infiuences power consumption.

MY SPHERE OF INFLUENCE

In a typical day I use multiple systems for various needs. Figure 1 illustrates the power consumption (in watts, $\log_{10}$ scale) of each of the devices in my sphere of infiuence on a daily basis for several power modes.

Among the ensemble of IT equipment I use periodically are two laptops and two desktop systems.

Laptop1 is my personal laptop for everyday use; it's large, rather bulky, and a bit of a power hog as far as laptops go. I keep this laptop on when it's not in use so I can access information quickly on demand. Laptop2 is lightweight, and I use it primarily for teaching a few hours a week and when on the road. I keep this laptop off when not in use.

PC1 and PC2 are my business and home PCs, respectively, and are physically identical. PC1 gets normal everyday offlce use and is turned off overnight and on weekends. PC2 is rarely used directly but serves as a flle and application server at home—it's mostly on for on-demand access.

I also use my DVR and iPhone daily and the family's Wii console on weekends.

Measuring the power consumption of hundreds of embedded consumer electronics systems from my coffee maker to my refrigerator to my car was beyond the scope of this exercise, but I did evaluate a few devices that I use daily, including my TV, toaster, and microwave, for comparison with my IT equipment.

As Figure 1 shows, system mode clearly infiuences power consumption. For example, my Wii console has three power modes: When off, it draws about 4 watts; in standby, it draws 10.5 watts; and when playing a game, it draws 20 watts on average. My game console isn't alone, however, as my plasma TV also consumes power when off, though only about 1.4 watts. When in use, my TV and DVR consume nearly as much power as my toaster.

As far as the IT equipment, at idle my laptops and PCs consume considerable power as well. Note that idle power (Idle+ON) is not that much less than power under load (ON+busy). As previous Green IT columns have pointed out, efforts are under way to reduce this disparity. I purposely separated out the power for the PC monitor, which accounts for about a third of PC power use.

As for my DVR, there's no noticeable difference between watching TV, watching a recorded movie, or recording a movie. Luckily, none of the IT equipment's power usage comes close to that of my microwave when set on high—a whopping 1,760 watts.

IMPACT AND ENERGY USE

To calculate my energy use (energy = power × time) for the equipment listed in Figure 1, I analyzed my own behavior over several weeks to determine how long I used each device in each mode. Table 1 shows the devices, modes, and power consumption as well as my average usage per week in hours.

As expected, my behavior significantly influences my carbon footprint.

First, I spend too much time—29 hours on average—in front of the boob tube; although this includes time playing the Wii, it's still a sad commentary on my all-too-American lifestyle. Second, the added convenience of having on-demand access to my data on Laptop1 and PC2 causes me to keep these systems in fairly high power modes for much longer than necessary. Third, allowing my Wii console to sit in standby mode during weekend play is wasteful.

Across all the equipment measured, I am using over 3,200 kilowatt-hours per year. Multiplying this by the average rate of electricity ($.1123 per kWh) in the state of Virginia, where I live, I spend about $365 annually powering these machines. Excluding the TV, toaster, and microwave, I spend about $275 per year on the IT equipment I use.

According to the US Department of Energy (www.eia.doe.gov), a coal-generated kWh produces about 2 pounds of CO_2. As 42 percent of energy in Virginia comes from coal sources (see www.americaspower.org for other US states), I multiply kWh × 2 × .42 to convert from kWh to pounds CO_2. As Table 1 shows, I'm personally responsible for about 2,700 pounds of CO_2—more than one metric ton (2,204 pounds); my IT equipment use alone generates over 2,050 pounds of CO_2 annually.

It's estimated that a typical American has about a 20-ton annual carbon footprint (higher than any other country). Thus, my IT equipment use

Table 1. My IT carbon footprint.

Device	Mode	Power (watts)	Usage per week (hours)	My kWh/year	My annual cost (dollars)	My pounds CO_2/year
Wii console	Off	2	120	12.51	1.41	10.51
	Standby	10.5	44	24.09	2.71	20.24
	Game	20	4	4.17	0.47	3.50
Laptop1	Idle+ON	38	149	295.23	33.15	248.00
	ON+busy	58	19	57.46	6.45	48.27
iPhone	Charging	4	7	1.46	0.16	1.23
DVR	ON+busy	29	168	254.04	28.53	213.39
PC1	ON+busy	146	45	342.58	38.47	287.77
	Monitor	60	45	140.79	15.81	118.26
Laptop2	Idle+ON	33	0	0.00	0.00	0.00
	ON+busy	53	3	8.29	0.93	6.96
PC2	Idle+ON	146	168	1,278.96	143.63	1,074.33
	Monitor	60	7	21.90	2.46	18.40
Plasma TV	Off	1.4	143	10.44	1.17	8.77
	ON+busy	398	29	601.83	67.59	505.54
Toaster	ON+busy	452	0.25	5.89	0.66	4.95
Microwave	Idle+ON	4.3	166.25	37.28	4.19	31.31
	ON+busy	1,760	1.75	160.60	18.04	134.90
Total				**3,257.52**	**365.82**	**2,736.32**
Total (IT use only)				**2,441.49**	**274.18**	**2,050.85**

accounts for nearly 5 percent of my annual carbon footprint.

Is my IT carbon footprint significant? It's obviously much bigger than it was in years past. An at-home data server and DVR are fairly recent additions to my carbon footprint, and I have more machines performing more tasks all the time. While individual devices may become more energy-efficient over time, my use of the devices, especially my constant use in the case of the DVR and file server, is on the rise.

I have no reason to think this trend won't continue. The raw numbers of my IT carbon footprint may not be worrisome, but the increased use of devices is troubling.

Although my IT equipment use probably exceeds that of the average American, I suspect it's typical for computing practitioners. I consider myself a fairly energy-conscious person, but clearly there are ways I can substantially decrease my carbon footprint. For example, I could sacrifice the convenience of data on demand from my file server and primary laptop. Turning off and unplugging devices such as my Wii console and DVR could also increase efficiency. And as mobile systems are becoming much more powerful and have smaller carbon footprints than traditional PCs, I could consolidate many devices in a single machine. Finally, I really do need to stop watching so much TV.

***Kirk W. Cameron**, Green IT column editor, is an associate professor in the Department of Computer Science at Virginia Tech. Contact him at greenit@computer.org.*

Global Trends in Computing Accreditation

Harry L. Reif and Richard G. Mathieu
James Madison University

As computing accreditation increases, three trends are emerging: clarified definitions of disciplines, a unified approach to accreditation, and a focus on graduates' long-term competencies.

Members of the international computing community are actively working to facilitate the mobility of computing and IT-related professionals through the recognition of equivalency of accredited academic programs leading to a degree in a computing or IT-related discipline.

As of December 2008, eight organizations—ABET (US), ABEEK (Korea), ACS (Australia), BCS (United Kingdom), CIPS (Canada), HKIE (Hong Kong), IEET (Chinese Taipei), and JABEE (Japan)—had signed the Seoul Accord (www.seoulaccord.com) agreeing to work together to become recognized as the international authority for quality assurance for education in computing and IT-related professions and to promote and develop best practices for the improvement of education in computing and IT-related disciplines.

Within the EU, participants in the multiyear Bologna Process are adopting a system of comparable higher education degrees to overcome obstacles to the free movement of students, teachers, and researchers and to promote European cooperation in the quality assurance of higher education programs (www.ond.vlaanderen.be/hogeronderwijs/bologna).

At the same time, the International Federation for Information Processing (IFIP) has initiated the International Professional Practice Program (IP3) with the goal of creating a set of globally recognized and trusted professional certification schemes that represent the hallmark of true IT professionalism (www.ifip.or.at/projects/ITProf_Report.pdf).

INCREASE IN ACCREDITATION

Accreditation of computer science, information systems, IT, computer engineering, software engineering, and other related programs is undeniably growing.

For example, ABET's Computing Accreditation Commission (http://abet.org/statistics.shtml)—which accredits computer science, information systems, and IT programs—saw a steady increase from 1996 (140 accredited programs) to 2000 (163 accredited programs) to 2004 (215 accredited programs) to 2008 (309 accredited programs). If programs in computer engineering, software engineering, and telecommunications are included, ABET accredited 621 computer-related programs in 2008.

Many institutions are achieving high placement rates for their graduates. There is widespread recognition that accreditation helps students and their parents choose quality college programs; enables employers to recruit graduates they know are well-prepared; is used by registration, licensure, and certification boards to screen applicants; and provides institutions a structured mechanism to assess, evaluate, and improve their programs' quality.

Unlike many of the traditional and longer-lived engineering fields, the computing disciplines are a relatively new phenomenon. As they have become essential components of scientific and business environments, the number of computing-related disciplines has grown beyond computer science and computer engineering to include software engineering, information systems, IT, health informatics, and library science to name but a few.

Over the past decades, organizations, including the IEEE Computer Society (www2.computer.org/portal/web/education) and the ACM (www.acm.org/education/curricula-recommendations), have made efforts to codify model curricula in computer science, information systems, and IT programs, but there is no universal agreement on what competencies graduates of these programs should be able to demonstrate. This contrasts with the explicit licensure requirements found in accounting, medicine, and many engineering fields.

However, there is increasing recognition that computing professionals require continuing education and in some cases verification of their credentials. For example, the Computer Society has created an e-Learning Campus (www2.computer.org/portal/web/e-learning/home) with access to more than 3,000 online courses and online books from leading technical publishers, as well as the CSDA (Certified Software Development Associate) and CSDP (Certified Software Development Professional) certifications in software engineering and development.

For these reasons it makes sense to examine the state of accreditation and the work being done to unify accreditation and professionalization processes worldwide.

ACCREDITATION TRENDS

Internationally, many bodies accredit computing programs. In the US, the predominant organization that has emerged as the accrediting body for computing programs is ABET, in particular the Computing Accrediting Commission (CAC) that operates under the ABET umbrella.

Accreditation is a means of formally involving two groups of stakeholders, peer educators and IT professionals, in the assessment of an institution's computing program. It is also a method for sharing and adopting appropriate best practices among all computing programs. As we examined ABET's work, three significant trends emerged.

First, in the process of defining appropriate criteria to use in evaluating computing programs, the CAC has defined three distinct disciplines: computer science, information systems, and IT. In addition, ABET's Engineering Accrediting Commission (EAC) has clearly defined academic programs in computer engineering, software engineering, and telecommunications. While a wide diversity of program names is still being offered, as time moves forward, there appears to be greater commonality of program names centered on these disciplines.

There is increasing recognition that computing professionals require continuing education and in some cases verification of their credentials.

Second, as institutions embrace accreditation and as accrediting bodies provide greater services, there is increased demand for the harmonization of evaluation criteria so that institutions can repurpose and reuse as much data as possible to support the accreditation of multiple programs without having to recast, reformat, and otherwise revise the data to suit each accrediting body's seemingly unique requirements.

Third, accreditation requirements have evolved from a prescriptive set of courses and course content requirements to a set of mission-driven requirements that contain some common elements but are uniquely shaped by each institution through a process that involves alumni, advisory boards, faculty, and students. This evaluation methodology requires each institution to set expectations for graduates that extend beyond the classroom and the temporal duration of each student's formal degree program.

CLARIFIED DEFINITIONS OF DISCIPLINES

As the field of computing continues to evolve, definitions of computing-related disciplines are becoming more clarified.

Perhaps the greatest advance has occurred with the use of "computer science" as a program name. While variations such as "computing science" and "computer software engineering" still exist, many potential students and employers have a good idea of what competencies to expect from a computer science graduate. The use of "information systems" and "information technology" as core elements in programs' names is becoming increasingly common, and work continues on establishing a common set of expectations regarding the competencies that graduates of these degree programs should possess.

As part of establishing its criteria for evaluating computer-related programs, ABET/CAC has defined specific criteria that distinguish computer science, information systems, and IT degree programs. In seeking ABET accreditation, schools must align the names of their programs with these criteria.

The criteria do not preclude programs specifying additional competencies beyond the minimal criteria that their graduates will possess. This encourages programs to focus their curricula on meeting the needs of specific sectors of future employers while simultaneously presenting a unified expectation about the baseline qualifications that all computer scientists, information systems professionals, and information technologists will possess. Concurrently, it lets educators concentrate on making their courses and curricula consistent with the common criteria used to define and measure the success of each program discipline.

In the end, potential students and employers will be able to distinguish between the expected competencies that graduates of these programs should demonstrate.

UNIFIED APPROACH TO ACCREDITATION

With the globalization of business and government relationships, it is rational to expect a more unified approach to recognizing quality programs in the computing disciplines to evolve. Within an accreditation agency, the term "harmonization" is often used to define a unified approach to accreditation.

For ABET this means creating common languages that reduce confusion for institutions being visited by multiple commissions, including the CAC, the EAC, the Applied Science Accreditation Commission (ASAC), and the Technology Accreditation Commission (TAC). In addition, it simplifies training for ABET volunteer program evaluators and increases efficiencies by eliminating duplicate efforts, forms, and processes.

The Seoul Accord and the Bologna Process are mechanisms to create transparency and coordination between the different computing accreditation bodies throughout the world. These agreements identify comparable academic programs and request that signatories make a reasonable effort to ensure that any bodies responsible for registering or licensing computing and IT-related professionals to practice in its country accept the equivalence of academic computing and IT-related programs accredited by the signatories.

LONG-TERM COMPETENCIES OF GRADUATES

One unique characteristic of computer-related technologies is their uncharacteristically short half-life. Consequently, practitioners' knowledge must be updated continuously and evolve over time. Institutions must recognize this focus on graduates' long-term competencies and capabilities and entwine it with the curricula and evaluation processes.

The IFIP's IP3 program (www.ipthree.org) is working to establish internationally recognized professional standards. A key component of this will be work done by professional societies seeking to credential qualified individuals. For example, the Computer Society developed the CSDA and CSDP programs through a job-analysis process that provided an industry-accepted, systematic procedure for identifying and validating the performance domain of a job and the knowledge and skills necessary to perform that job.

Taken together, these three trends—clarified definitions of computing-related disciplines, a unified global approach to accreditation, and a focus on graduates' long-term competencies and capabilities—combine to create an environment where educational institutions and employers alike can qualitatively measure each accredited institution's graduates for the purpose of best matching graduates with needs and selecting programs and students that best meet an organization's specific requirements for talented computing professionals.

Academics and professionals should remain aware of the increased importance placed on the credentialing of computing professionals. The global computing community is looking for professional standards that are vendor neutral, independent, and maintained through continuing professional development.

Harry L. Reif is an associate professor of computer information systems and management science in the College of Business at James Madison University as well as an ABET commissioner and a founding director of the International Telecommunications Education and Research Association. Contact him at reifhl@jmu.edu.

Richard G. Mathieu is the ManTech Fellow and head of the Department of Computer Information Systems and Management Science in the College of Business at James Madison University, as well as a program evaluator for ABET/CAC. Contact him at mathierg@jmu.edu.

Editor: Richard G. Mathieu, Dept. of Computer Information Systems and Management Science, College of Business, James Madison Univ., Harrisonburg, VA; mathierg@jmu.edu

IEEE 6th World Congress on Services (SERVICES 2010)

July 5-10, 2010, Miami, Florida, USA, http://www.servicescongress.org/2010

Modernization of all vertical services industries including finance, government, media, communication, healthcare, insurance, energy and …

IEEE 7th International Conference on Services Computing (SCC 2010)

In the modern services and software industry, Services Computing has become a cross-discipline that covers the science and technology of bridging the gap between Business Services and IT Services. The scope of Services Computing covers the whole lifecycle of services innovation research that includes business componentization, services modeling, services creation, services realization, services annotation, services deployment, services discovery, services composition, services delivery, service-to-service collaboration, services monitoring, services optimization, as well as services management. The goal of Services Computing is to enable IT services and computing technology to perform business services more efficiently and effectively. Visit http://conferences.computer.org/scc.

IEEE 8th International Conference on Web Services (ICWS 2010)

As a major implementation technology for modernizing services industry, Web services are Internet-based application components published using standard interface description languages and universally available via uniform communication protocols. The program of ICWS 2010 will continue to feature research papers with a wide range of topics focusing on various aspects of implementation and infrastructure of Web-based services. ICWS has been a prime international forum for both researchers and industry practitioners to exchange the latest fundamental advances in the state of the art on Web services. Visit icws.org.

IEEE 3rd International Conference on Cloud Computing (CLOUD 2010)

Cloud Computing is becoming a scalable services delivery and consumption platform in the field of Services Computing. The technical foundations of Cloud Computing include Service-Oriented Architecture (SOA) and Virtualizations of hardware and software. The goal of Cloud Computing is to share resources among the cloud service consumers, cloud partners, and cloud vendors in the cloud value chain. Major topics cover Infrastructure Cloud, Software Cloud, Application Cloud, and Business Cloud. Visit http://thecloudcomputing.org.

Sponsored by IEEE Technical Committee on Services Computing (TC-SVC, tab.computer.org/tcsc)

IEEE computer society

IEEE Celebrating 125 Years *of Engineering the Future*

Services Computing

IBM

IBM Research

Submission Deadlines For Abstracts and Papers

ICWS 2010: **Feb. 1, 2010**
SCC 2010: **Feb. 15, 2010**
SERVICES 2010: **March 6, 2010**
CLOUD 2010: **March 6, 2010**

Contact: Liang-Jie Zhang (LJ) at zhanglj@ieee.org (Steering Committee Chair)

IEEE TRANSACTIONS ON SERVICES COMPUTING

IT Professional
Web Services
International Journal of Web Services Research
Business Process Integration and Management

Continued from page 108

76d54m32—have interest but need special treatment in the arithmetic.

Many useful functions can be represented by symbolically modifying basic functions. In the exact calculator, the two special symbols were used for this, but a larger keyboard allows a larger set of distinctive symbols to be used, and modifiers can be placed like accents above the function symbol. Thus monadic $\tilde{\times}$ squares its argument, while dyadic $\tilde{\div}$ reverses its arguments, dividing its first argument *into* its second.

The simplest notation for combining functions is juxtaposition.

Another way to augment a function is to use an integer as superscript to have the function repeatedly applied, so that $\times^2$ multiplies its first argument by the square of its second. A zero superscript leaves the first or only argument unaltered while a negative superscript inverts the function, so that monadic $\tilde{\times}^{\nabla 1}\ \div$ takes the square root of its argument.

BREADTH

To add breadth, the formulator works on lists of numbers. If a simple arithmetic function has two list arguments, then they must be of the same length. A dyadic function with one argument a list and the other a single item applies the single item to each item of the list.

Lists of numbers are awkward to display, particularly on a handheld device, so a suite of graphical display options is used to help the user. Keying lists of numbers in can also be awkward, but abbreviation conventions help. For example, subscripted replication as in H_2O and CO_2 allows 99_{10} as a list of ten 99s, and 6[7]89 provides integers between 6 and 89 spaced by 7. The subscript is useful on display as monadic $\tilde{=}$ immediately shows the number of items in its argument, at least if there is neither an infinity $1^{\nabla}0$ nor an indeterminacy $0^{\nabla}0$.

Much of the handling of lists can be done by providing an edit capability, for example to extend or combine lists or to replace, add, or remove items. But when a result depends on the values within the list, a formal functional approach works best, with structural functions alongside but distinct from arithmetic ones. Such functions strictly preserve the values of list items so that different kinds of numbers can be mixed within a list.

Arithmetic functions can change the list's structure, however. The acute accent signals the arithmetic reduction of a list, so that monadic $\acute{+}$ would total a list while dyadic $\acute{+}$ would add a list up in groups of a size given by the single item argument. Monadically, $\acute{-}$ gives the alternating sum and $\acute{\div}$ the alternating product.

Arithmetic functions can produce more than one valid result from a single item—such as analytical functions that extract the factors of an exact value or the roots of a complex value or a polynomial—by allowing items of a list to be sets. This adds a good deal of simple richness to the arithmetic, especially for students. Notationally, a set is enclosed in parentheses, and converting a list to a set removes duplicates and puts the items in sequence.

CLARITY

Thus far, the formulator works like an operational calculator in that one or two arguments are selected and a single function, simple or modified, then operates on the arguments to produce an immediate result. This means that complex calculations are procedurally complex and their nature hidden behind that complexity.

Clarity is achieved by providing for functions to be combined notationally, as in traditional algebra. Operationally, this means maintaining two stacks: one for potential arguments and another for potential functions. Editing can be done in either stack, and then any calculation is a kind of anticlimax to developing an algebraic function, testing it, and eventually applying it. Calculation is done by selecting one or two arguments from their stack, then selecting their function from the formula stack.

The simplest notation for combining functions is juxtaposition, in the same way that digits are juxtaposed to form numbers. All functions in such a compound are used monadically except the lowest-order one, which is used whichever way the compound function is used. Thus $\tilde{\times}\div$ is monadically the square of the reciprocal of its argument, and dyadically the square of the first argument divided by the second. If part or all of a compound function is to be modified, it is enclosed in parentheses with the modifier placed over one parenthesis.

A basic compound function applies each component successively to the result coming from its right. Only the rightmost function of a compound sees the compound's argument(s). However, the decimal-point symbol $_{\Delta}$ is also used in compound functions as a dyadic point. Functions to the right of the $_{\Delta}$ are monadically applied to each argument, with their results joined by the first function to the left of the point. Thus dyadic $+_{\Delta}\div$ is the sum of the reciprocals of its arguments.

The next level of notation is the list of functions, called a *train*. Items in the train can be simple or compound functions, and parentheses can be used to enclose a train to make it an item within a compound or train.

A train of two items is called a *hook*; its first item is a dyad and the other a monad. The second argument of the dyad is the result of the monad that is applied to the second or only argument. The first argument of the dyad is the first argument of a dyadic hook, or the only argument of a monadic hook. So monadic + ÷ adds the argument to its reciprocal.

A train of three items is called a *fork*; its center item is dyadic with its arguments the results of its neighboring items. The neighboring items each take the argument or arguments of the fork. Thus monadic +́ ÷ +́≅ calculates the mean of its argument, though the formulator would probably have a primitive symbol for the +́≅ compound.

In a longer train with an odd number of items, the first two items form a fork with the rest of the train. In a train with an even number of items, the first item forms a hook with the rest of the train.

DEPTH

Clarity in the formulator is supported by the ability to construct formulas by putting functions together in various ways to define a sequence of evaluation. For depth, such construction is supported by templates used to select members of a family of functions in a process of "contemplation."

A template uses placeholder symbols to define where in the template one or two figments will be used. Figments are to a template much like what arguments are to a function. Templates are kept in the function stack but cannot be used directly for calculation.

A new function is produced by selecting as figments one or two functions from the function stack or keyboard, then selecting the template to be used. The new function joins the function stack.

THE FORMULATOR

The design I have outlined is constrained in several important ways. First, the numerical data for calculation are either items or lists of items, where an item can be a set but not a list. This is not as limiting as it might seem. A dyadic function with arguments of different length—to calculate a polynomial for a list of arguments, for example, can be taken to need its second argument dribbled into the function item by item to be worked on by the function with its entire first argument to reduce it to a single item of the result. The ~ modifier can be used to have the items of the first argument dribbled in instead.

Second, the notation is entirely symbolic, free of alphabetic characters. This is inspired by the thoughts of the late great Kenneth Iverson, as expressed in his Turing Award essay, "Notation as a Tool of Thought" (elliscave.com/APL_J/tool.pdf), which describes the principles behind his APL (A Programming Language).

Iverson later surrendered to the tragic typographical tyranny of the computer industry and its profession and led a redesign of APL called J (jsoftware.com) based on the ASCII character set and the QWERTY keyboard. Incidentally, the idea of trains is used in J and occurred to Iverson as he flew back to Canada from an APL conference in Sydney, Australia.

APL, J, and several related systems are splendid for programming. However, the formulator is not designed for programming. The commodity calculator is a device for ordinary people and students to use for ad hoc calculation. Similarly, the formulator, as a commodity device, is for ordinary people to use for ad hoc algebra, and for students to use to learn algebra.

The decline of literacy and numeracy is well attested. Digital technology can be used in early education to counteract this (The Profession, Mar. 2008, pp. 102-104), and this is starting to happen.

But the decline in numeracy and thence the study of mathematics in later education continues (see tinyurl.com/ye5q5yc, for example). This can't be counteracted by the use of mathematical packages like APL and Mathematica in schools because their proper use must be based on an understanding of the mathematics involved. An expert user of such packages is not necessarily an expert mathematician.

What is needed is a mathematical tool like the formulator. Development of the necessary standard for such a tool, and training teachers to use it, is an important challenge to the computing and mathematics professions. C

Neville Holmes *is an honorary research associate at the University of Tasmania's School of Computing and Information Systems. Contact him at neville.holmes@utas.edu.au.*

Truth and Breadth, Clarity and Depth in Algebra

Neville Holmes, *University of Tasmania*

The formulator does to the calculator what the calculator did to the abacus.

In an essay titled "Truth and Clarity in Arithmetic" (The Profession, Feb. 2003, pp. 126-128), I outlined a simple calculator design that avoided the several unfortunate faults in the commodity calculator. Although I got some laudatory e-mails, I was bemused by one from a calculator designer who told me I had no idea how a calculator should be designed, but who failed to point out any specific fault in my design.

Since then, digital technology has brought in devices like the iPod and BlackBerry, about the size of the commodity calculator, which makes it interesting to consider what might be done to the calculator to exploit their technology. I will call this extended design a *formulator* and hope thus to avoid blanket condemnation.

THE EXACT CALCULATOR

The design of the exact calculator was motivated by the unmet need, especially in early education, for truth and clarity, and is the basis for the formulator's design.

Truth was mainly achieved by providing only exact arithmetic. Combined integer and fractional arithmetic was thus the basis, and this required a notation that allowed both decimal and other fractions to be represented exactly. While exactitude ruled out functions such as the square root, and values such as multiples of π, it also made desirable very simple functions such as quotient and remainder.

Clarity was mainly achieved by requiring a minimum of four lines of display so that at least the two or three numbers involved in the immediately prior calculation, and a number being keyed in for the next, would be clearly visible. Also, as befits a calculator, the tapping of a function key caused the calculation with that function to be carried out, and the result and its sources displayed together with the function symbol. In this context, a function is literally an operation.

The representation of numbers was enriched by the use of two symbols: $^{\triangledown}$ for the negative sign and fraction point and $_{\vartriangle}$ for the decimal point—provided within the usual three-by-four digital key matrix, with four basic function keys alongside. The number of distinct functions provided was greatly expanded by being able to use the two special symbols as prefixes to the basic function symbols, and to use all functions as either monads or dyads.

TRUTH

The next step up from exact arithmetic is inexact arithmetic, and the challenge is to stay truthful.

Truth in this case resides in proclaiming inexactitude for numbers both on the way in and on the way out. Italic representation, with a shift key for it, is one way of meeting this need. And of course exactness is maintained if possible, at least internally, and the greatest practical accuracy assured otherwise.

Handling inexactness makes many extra functions useful, such as for exponentiation and trigonometry, thus handling imaginary and complex numbers. An interesting notation for this would use the $ symbol for the imaginary point, so that 2$3 would represent the number traditionally and confusedly represented as an arithmetic expression: *2+3i*. Nondecimal values such as dates, times, and angles—represented maybe as *2009y10m19 7w6d5h12*

Continued on page 106

The Mighty "Mini"
Born out of petascale technology. Backed by the name in "super" computing. The Cray XT5m – Powered by industry leading AMD Opteron ™ processors – delivering exceptional performance, manageability, and stability in a midsized supercomputer.

Learn more today at www.cray.com/mighty.

CRAY
THE SUPERCOMPUTER COMPANY

AMD Opteron 64