

COMMUNICATIONS

CACM.ACM.ORG OF THE ACM 01/2015 VOL.58 NO.01

Distributed Information Processing in Biological and Computational Systems

Data Brokers
Are Watching You

The Rise and Fall of
Industrial Research Labs

Does the Internet
Make Us Stupid?

Association for
Computing Machinery



Applicative 2015

February 26-27 2015
New York City

APPLICATIVE 2015 is ACM's first conference designed specifically for practitioners interested in the latest emerging technologies and techniques.

The conference consists of two tracks:

SYSTEMS will explore topics that enable systems-level practitioners to build better software for the modern world. The speakers participating in this track are involved in the design, implementation, and support of novel technologies and low-level software supporting some of today's most demanding workloads. Topics range from memory allocation, to multicore synchronization, time, distributed systems, and more.

APPLICATIONS will cover topics such as reactive programming, single-page application frameworks, and other tools and approaches for building robust applications more quickly. The speakers slated for this track represent leading technology companies and will share how they are applying new technologies to the products they deliver.

For more information about the conference and how to register, please visit:

<http://applicative.acm.org>



Association for
Computing Machinery



2015

ACM INTERNATIONAL CONFERENCE
ON INTERACTIVE EXPERIENCES FOR
TELEVISION AND ONLINE VIDEO

3-5 JUNE, 2015
BRUSSELS, BELGIUM

Paper Submissions by
12 January 2015

Work in Progress, Demos,
DC, & Industrial
Submissions by
2 March 2015

Welcoming Submissions on

Content Production

Systems & Infrastructures

Devices & Interaction Techniques

Experience Design & Evaluation

Media Studies

Data Science & Recommendations

Business Models & Marketing

Innovative Concepts & Media Art



Association for
Computing Machinery



iMinds

TVX2015.COM

INFO@TVX2015.COM

Departments

- 5 **Editor's Letter**
The Rise and Fall of Industrial Research Labs
By Moshe Y. Vardi
-
- 7 **Cerf's Up**
A Long Way to Have Come and Still to Go
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Toward a Map Interface Not Inherently Related to Geography
-
- 11 **ACM's FY14 Annual Report**
-
- 18 **BLOG@CACM**
A Valuable Lesson, and Whither Hadoop?
Valerie Barr considers outcomes from the Grace Hopper Celebration, while Michael Stonebraker ponders the past, present, and possible future of the "Hadoop stack."
-
- 49 **Calendar**
-
- 116 **Careers**

Last Byte

- 128 **Future Tense**
Wow!
Human life is filled with illusions, so virtual worlds are not especially unreal.
By William Sims Bainbridge

News



- 21 **In Search of Bayesian Inference**
Long relegated to the statistical backburner, Bayesian Inference is undergoing a renaissance.
By Erica Klarreich
-
- 25 **Smart Transportation Networks Drive Gains**
Transportation engineers and city planners are looking to information technology to redefine traffic management in urban areas. With the opportunity, however, comes risks.
By Samuel Greengard
-
- 28 **Data Brokers Are Watching You**
You would be surprised by how much they know about you, and what they are doing with your information.
By Gary Anthes
-
- 31 **Google Boosts ACM's Turing Award Prize to \$1 Million**
By Lawrence M. Fisher

Viewpoints

- 32 **Technology Strategy and Management**
How Traditional Firms Must Compete in the Sharing Economy
Considering the evolving relationship between established companies and their sharing-economy counterparts.
By Michael A. Cusumano
-
- 35 **Law and Technology**
A Right to Be Forgotten?
Searching for an answer to the question of how much responsibility search engine operators should bear for privacy-related issues connected to search engine results.
By Lee A. Bygrave
-
- 38 **The Business of Software**
A Little Queue Theory
When more work means less done.
By Phillip G. Armour
-
- 40 **Historical Reflections**
The Tears of Donald Knuth
Has the history of computing taken a tragic turn?
By Thomas Haigh
-
- 45 **Viewpoint**
Wearables: Has the Age of Smartwatches Finally Arrived?
Time will tell if smartwatches will find their niche.
By Reza Rawassizadeh, Blaine A. Price, and Marian Petre
-
- 48 **Viewpoint**
Does the Internet Make Us Stupid?
Yes, but this may not be as bad as it sounds...
By Hermann Maurer
-
- 52 **Viewpoint**
What It Means to Receive the Turing Award
By Silvio Micali



Association for Computing Machinery
Advancing Computing as a Science & Profession

Practice

54 **Disambiguating Databases**

Use the database built for your access model.

By Rick Richardson

62 **Internal Access Controls**

Trust, but verify.

By Geetanjali Sampemane

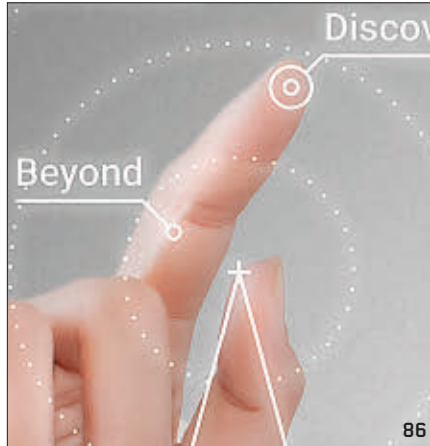
66 **Scalability Techniques for Practical Synchronization Primitives**

Designing locking primitives with performance in mind.

By Davidlohr Bueso

Q Articles' development led by **acmqueue**
queue.acm.org

Contributed Articles

76 **Analyzing Worldwide Research in Hardware Architecture, 1997–2011**

Results showed more published papers, collaboration with U.S. colleagues, preference for English, and uneven research impact.

By Virender Singh, Alicia Perdígones, José Luis García, Ignacio Cañas-Guerrero, and Fernando R. Mazarrón

86 **Interactive Intent Modeling: Information Discovery Beyond Search**

The system should let users incrementally direct their search toward relevant, though not initially obvious, information.

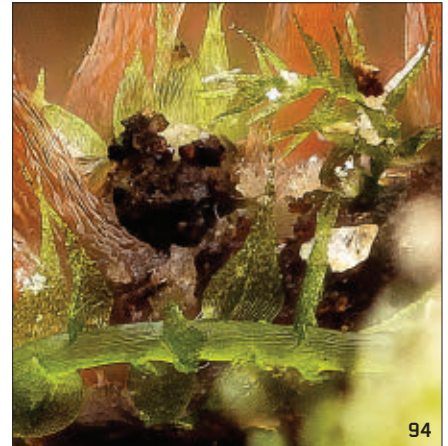
By Tuukka Ruotsalo, Giulio Jacucci, Petri Myllymäki, and Samuel Kaski

**About the Cover:**

Two converging technologies have led to increased interest in the relationship between information processing in biological and distributed computational systems. This month's cover story (p. 94) reviews some of the promising biological and computational problems to study that may improve our understanding of both fields.

Photograph courtesy of USGS Native Bee Inventory and Monitoring Program.

Review Articles

94 **Distributed Information Processing in Biological and Computational Systems**

Exploring the similarities and differences between distributed computations in biological and computational systems.

By Saket Navlakha and Ziv Bar-Joseph

Research Highlights

104 **Technical Perspective**
Big Data Needs Approximate Computing
*By Ravi Nair*105 **Neural Acceleration for General-Purpose Approximate Programs**
By Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger



ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Executive Director and CEO

John White

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Darren Ramdin

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Bernard Rous

Director, Office of Group Publishing

Scott E. Delman

ACM COUNCIL

President

Alexander L. Wolf

Vice-President

Vicki L. Hanson

Secretary/Treasurer

Erik Altman

Past President

Vinton G. Cerf

Chair, SGB Board

Patrick Madden

Co-Chairs, Publications Board

Jack Davidson and Joseph Konstan

Members-at-Large

Eric Allman; Ricardo Baeza-Yates;

Cherri Pancake; Radia Perlman;

Mary Lou Soffa; Eugene Spafford;

Per Stenström

SGB Council Representatives

Paul Beame; Barbara Boucher Owens;

Andrew Sears

BOARD CHAIRS

Education Board

Mehran Sahami and Jane Chu Prey

Practitioners Board

George Neville-Neil

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Fabrizio Gagliardi

ACM India Council

Srinivas Padmanabhuni

ACM China Council

Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs

Jack Davidson; Joseph Konstan

Board Members

Ronald F. Boisvert; Marie-Paule Cani;

Nikil Dutt; Roch Guerrin; Carol Hutchins;

Patrick Madden; Catherine McGeoch;

M. Tamer Ozsu; Mary Lou Soffa

ACM U.S. Public Policy Office

Renee Dopplick, Director

1828 L Street, N.W., Suite 800

Washington, DC 20036 USA

T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association

Lissa Clayborn, Acting Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

STAFF

DIRECTOR OF GROUP PUBLISHING

Scott E. Delman

cacm-publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Larry Fisher

Web Editor

David Roman

Editorial Assistant

Zarina Strakhan

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Designer

Iwona Usakiewicz

Production Manager

Lynn D'Addesio

Director of Media Sales

Jennifer Ruzicka

Public Relations Coordinator

Virginia Gold

Publications Assistant

Juliet Chance

Columnists

David Anderson; Phillip G. Armour;

Michael Cusumano; Peter J. Denning;

Mark Guzdial; Thomas Haigh;

Leah Hoffmann; Mari Sako;

Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission

permissions@cacm.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmhlp@cacm.acm.org

Letters to the Editor

letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY

10121-0701

T (212) 626-0686

F (212) 869-0481

Director of Media Sales

Jennifer Ruzicka

jen.ruzicka@hq.acm.org

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)

2 Penn Plaza, Suite 701

New York, NY 10121-0701 USA

T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF

Moshe Y. Vardi

eic@cacm.acm.org

NEWS

Co-Chairs

William Pulleyblank and Marc Snir

Board Members

Mei Kobayashi; Kurt Mehthorn;

Michael Mitzenmacher; Rajeev Rastogi

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;

John Leslie King

Board Members

William Aspray; Stefan Bechtold;

Michael L. Best; Judith Bishop;

Stuart I. Feldman; Peter Freeman;

Mark Guzdial; Rachelle Hollander;

Richard Ladner; Carl Landwehr;

Carlos Jose Pereira de Lucena;

Beng Chin Ooi; Loren Terveen;

Marshall Van Alstyne; Jeannette Wing



PRACTICE

Co-Chairs

Stephen Bourne

Board Members

Eric Allman; Charles Beeler; Bryan Cantrill;

Terry Coatta; Stuart Feldman; Benjamin Fried;

Pat Hanrahan; Tom Limoncelli;

Kate Matsudaira; Marshall Kirk McKusick;

Erik Meijer; George Neville-Neil;

Theo Schlossnagle; Jim Waldo

The Practice section of the CACM

Editorial Board also serves as

the Editorial Board of *queue*.

CONTRIBUTED ARTICLES

Co-Chairs

Al Aho and Andrew Chien

Board Members

William Aiello; Robert Austin; Elisa Bertino;

Gilles Brassard; Kim Bruce; Alan Bundy;

Peter Buneman; Peter Druschel;

Carlo Ghezzi; Carl Gutwin; Gal A. Kaminka;

James Larus; Igor Markov; Gail C. Murphy;

Shree Nayar; Bernhard Nebel;

Lionel M. Ni; Kenton O'Hara;

Sriram Rajamani; Marie-Christine Rousset;

Avi Rubin; Krishan Sabnani;

Ron Shamir; Yoav Shoham; Larry Snyder;

Michael Vitale; Wolfgang Wahlster;

Hannes Werthner; Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Gregory Morrisett

Board Members

Martin Abadi; Amir El Abbadi; Sanjeev Arora;

Dan Boneh; Andrei Broder; Stuart K. Card;

Jeff Chase; Jon Crowcroft; Matt Dwyer;

Alon Halevy; Maurice Herlihy; Norm Jouppi;

Andrew B. Kahng; Xavier Leroy; Kobbi Nissim;

Mendel Rosenblum; David Salesin;

Steve Seitz; Guy Steele, Jr.; David Wagner;

Margaret H. Wright

WEB

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;

Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2015 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$100.

ACM Media Advertising Policy

Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current Advertising Rates can be found by visiting <http://www.acm-media.org> or by contacting ACM Media Sales at (212) 626-0686.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhlp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM* 2 Penn Plaza, Suite 701 New York, NY 10121-0701 USA

Printed in the U.S.A.



Association for
Computing Machinery





DOI:10.1145/2687353

Moshe Y. Vardi

The Rise and Fall of Industrial Research Labs

I SPENT SOME of my most formative years as a researcher at the IBM Almaden Research Center. Those were magical years. IBM Research was then a Camelot of computing research, with the motto of being “Famous for its science, and vital to IBM for its technology.” IBM researchers received Nobel prizes for their discoveries of high-temperature superconductivity and the scanning tunneling microscope. IBM researchers invented the magnetic disk drive and relational databases. As a young researcher, I could not ask for a better research environment.

But the economic recession of the early 1990s hit IBM very hard, and the stress was acutely felt inside the Research Division. From an equal emphasis on science and technology, the balance was shifting more and more toward technology. When IBM reported its financial results in October 1992, its worst-ever quarterly loss of close to \$3 billion shook the corporation to its core. IBM’s CEO, John Akers, resigned in January 1993, and Louis Gerstner arrived as new CEO in April 1993, embarking on his storied transformation of IBM. The Research Division never fully recovered from the trauma of the early 1990s. Just as IBM lost the leadership of the computing world, passing the mantle to Cisco, Intel, and Microsoft, IBM Research lost the leadership of computing research. A mass exodus of talent took place throughout the 1990s. (I resigned in 1995.) IBM Research today is still a powerhouse of talent, with occasional outstanding research accomplishments, for example, Deep Blue and Watson; its main focus, however, is on supporting the near-term business of IBM’s business units. The mantle of computing-research

leadership passed to Microsoft Research, with close to 1,000 researchers and many research labs all over the world.

I was acutely reminded of those challenging times in the 1990s, when the news flashed last September that Microsoft Research closed down its Silicon Valley Lab, laying off dozens of researchers. Of course, Microsoft’s business challenges have been a topic of wide public discussion for the past few years, and the lab closure was part of a major strategy shift in Microsoft, which included layoffs of 18,000 employees. Yet, the swiftness of the lab closure, and the fact that Microsoft decided to close a lab located in Silicon Valley, widely considered the mecca of computing, shook the computing-research community.

This sense of nasty surprise led to an unusual public exchange of letters in October. Some 28 members of the theoretical computer science community posted an open letter to Microsoft Research management, expressing “the research community’s shock and disappointment at the sudden and harsh way in which the members of the Microsoft Research Silicon Valley Lab were dismissed,” and calling for “a dialogue about the ways in which Microsoft can try to restore the environment that enabled MSR to produce such great research in the past.” Microsoft Research’s management issued a response within a week, explaining the closure of the lab as a business necessity, and stating Microsoft’s continued commitment to fundamental research. (See story on the Computing Research Association’s Computing Policy Blog, <http://bit.ly/1xcVxKe/>.)

While the specifics of Microsoft Research’s actions may have been surprising, these actions are entirely con-

sistent with the historical pattern of the rise and fall of industrial research labs. While the rise and fall of IBM Research is discussed above, other research labs, such as Bell Labs, Xerox Palo Alto Research Center, and DEC’s Research Labs, went through similar phases. The technology business is conducive to a “winner-takes-all” pattern; whereby an early entrant develops a technology niche and goes on to dominate it, establishing an effective monopoly. Dominant corporations can afford to establish research divisions that focus not only on research that is directly relevant to the corporation’s business, but also on research of broader scope and longer term. Inevitably, however, the business environment changes, dominance is lost, and emphasis on research yields to more pressing business needs.

Microsoft Research after the closure of the Silicon Valley Lab is a changed place. Once a sense of invincibility is lost, it simply cannot be regained. The decision to close a single lab and lay off most of its researchers—including many top performers—rather than spread the pain across the whole research division, whatever its business rationale (which I cannot judge), means that Microsoft researchers are now acutely aware of their vulnerability. This sense of vulnerability may have serious consequences. Ultimately, the future of Microsoft Research is intimately tied to the future of Microsoft itself. Can Microsoft retain its dominance? Can Microsoft Research retain its commitment to fundamental research?

Follow me on Facebook, Google+, and Twitter.

Moshe Y. Vardi, EDITOR-IN-CHIEF

Copyright held by author.



Association for Computing Machinery (ACM) Chief Executive Officer

ACM, the Association for Computing Machinery, invites applications for the position of Chief Executive Officer (CEO).

ACM is the oldest and largest educational and scientific computing society with 108,000 members worldwide. The association has an annual budget of \$60 million, 75 full-time staff in New York and Washington DC, a rich publications program that includes 50 periodicals in computing and hundreds of conference proceedings, a dynamic set of special interest groups (SIGs) that run nearly 200 conferences/symposia/workshops each year, initiatives in India, China, and Europe, and educational and public policy initiatives. ACM is the world's premier computing society.

The ACM CEO serves as the primary executive responsible for the formulation and implementation of ACM strategic direction, for representing ACM in the worldwide computing community, and for overall management of the affairs of the association. The successful candidate will have a high professional standing in the computing field, executive experience, leadership skills, and a vision of the future of professional societies and computing. The CEO reports to the ACM President. The CEO is not required to work from ACM's New York headquarters, but he or she must be able to travel frequently to headquarters and other ACM meetings and venues. The full job description can be found at: ceosearch.acm.org

Interested applicants should contact the ACM CEO Search Committee: ceosearch@acm.org

The ACM is an equal opportunity employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, national origin, age, protected veteran status or status as an individual with disability.



**Association for
Computing Machinery**



Vinton G. Cerf

DOI:10.1145/2693199

A Long Way to Have Come and Still to Go

I AM a big fan of Robert Heinlein's stories and novels. His early and mid-career works generally appeal to me more than his later novels, which brings me to *Starman Jones*,^a written in 1953, and almost certainly intended for nerdy teenage guys. I actually had not read this novel until recently. I had read *Tunnel in the Sky*, which attracted a similar audience. Young kid, in his teens, thrust into situations that would challenge a capable and experienced adult, coming out a success. Not unlike the early Horatio Alger novels, Heinlein emphasized hard work, honesty, a strong moral compass, winning out against unfair and discriminatory odds. Makes me think of Harry Potter, too! Who doesn't like it when the underdog wins out?

But this is *not* the reason I wanted to draw attention to this novel. In 1953, computing was still very much in its infancy, but Heinlein has his protagonist using books of tabulated calculations and values and manually converting numbers into binary form to *program* the computers onboard a starship. Information is relayed *orally* and keyed in, in binary! All this under time-pressure to make a jump to light speed at exactly the right time.

In 1953, IBM released the model 701. From Wikipedia:^b

The system used vacuum tube logic circuitry and electrostatic storage, consisting of 72 Williams tubes with a capacity of 1024 bits each, giving a total memory of 2048 words of 36 bits each. Each of the 72 Williams tubes was 3

inches in diameter. Memory could be expanded to a maximum of 4096 words of 36 bits by the addition of a second set of 72 Williams tubes or by replacing the entire memory with magnetic core memory. The Williams tube memory and later core memory each had a memory cycle time of 12 microseconds. The Williams tube memory required periodic refreshing, mandating the insertion of refresh cycles into the 701's timing. An addition operation required five 12-microsecond cycles, two of which were refresh cycles, while a multiplication or division operation required 38 cycles (456 microseconds).

Heinlein's rendering of programming in the far future was even more primitive than programming the IBM 701, which was largely done in assembly language, as Marc Rochkind documents in his draft history.^c We have come a long way from 1953 and, ironically, oral interaction with computers has become increasingly common, not to enter binary codes, but to speak to the computers to get them to accept and execute tasks. For example, pretty much anywhere you use a keyboard, Google's applications now allow you to speak. We are still far from serious dialogue, but one can begin to imagine this possibility.

The history of programming is one of increasing abstraction and context. High-level frameworks and built-in libraries fashioned in layers together with parallel operation and remote procedure calls mediated by networking protocols have changed the programming landscape over the past 60 years. Anyone who has programmed spreadsheets or created presentations can appreciate the power of high-level constructs. Programming in languages

like Python, JavaScript, Ruby on Rails, or HTML5 seems to me rather different from FORTRAN, ALGOL, and C++. I look forward to hearing from readers with experience in some or all of these languages and applications to learn how they view the progress of programming over the decades.

I have written before about the problems we still face with regard to programming errors and I will not repeat my diatribe here, but one does wonder to what extent the mistakes we make are in part a consequence of the level of language in which we express our intentions. The lower the level of expression, perhaps the more generality we can achieve, but at the potential risk of making serious mistakes. One is reminded of Edsger Dijkstra's famous letter to *Communications* "Go To Statement Considered Harmful."^d This leads me to wonder whether it is possible to write significant programs by way of high-level oral (or written) interaction with a programming system. For this to work, I suppose it has to be possible to *discuss* with the programming system the objectives of the program, to enter into fairly specific descriptions of algorithms, and to interact with a growing body of program text that represents the programming system's understanding of the programmer's intent. One might have to be able to ask questions about the evolving software and its anticipated behavior. We have a long way to go to reach such an objective.

In a kind of Turing test variant, one might pit a really good programmer against an automatic system, with the party carrying out the negotiated programming effort trying to distinguish between the automatic and the manual programmers. □

^d *Commun. ACM* 11, 3 (Nov. 1968), 147–148

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

^a Robert A. Heinlein, *Starman Jones*, ISBN 0-345-32811-6, Del Rey published by Ballantine Publishing Group, 1953.

^b http://en.wikipedia.org/wiki/IBM_701

^c <http://basepath.com/new/701/Programming-the-IBM-701.pdf>

Toward a Map Interface Not Inherently Related to Geography

HANAN SAMET ET AL.'S article "Reading News with Maps by Exploiting Spatial Synonyms" (Oct. 2014) on maps as the basis for reading news was an impressive and informative description of how a map-query interface should work. The details were enlightening, with NewsStand described as "a general framework for enabling people to search for information with a map-query interface." I am not sure if the authors intend to broaden their scope to include even perhaps non-geographically related information but hope they do.

NewsStand is, I think, part of the research behind spatial interfaces in general. I was similarly impressed when first learning about the spatial data management implementations at MIT in the late 1970s led by Richard Bolt.¹ At the time, there was interest in exploiting spatial relationships as the main organizing interface. In the early 1980s, I became convinced the improving graphics of personal computers would allow spatial interfaces to become a useful and more mainstream way to access the full range of computer-stored information. I wrote two conference papers in 1984 and 1985 describing how graphical depictions of spatial orientations could be useful in finding information, including word maps for concepts. Several years later, I included screenshots of a geographic map interface using an early Macintosh to show how it might look when searching something as simple as a telephone directory.² I went on to suggest that, in the same way maps of real geography provide a natural interface to information involving geographical relationships, "information maps" could likewise be a useful and popular interface to content not inherently related to geography.

However, in spite of this potential, such an interface has not emerged; indeed, it seems there is a strong movement against relying on spatial orientation. The best example might be today's

fascination with putting everything in "the cloud"; it does not matter where it is, just that it is out there, somewhere. The popular Google search interface is another example, with no spatial arrangement other than a list, except for Google Maps and Google Earth, which are specifically designed to find geographic locations.

As impressive as NewsStand is, for map-query interfaces to go beyond geography-based systems, I assume the information maps would have to involve the same familiarity as traditional geographical maps, along with their relationship to real routes to a goal. I am perhaps misguided in thinking some sort of map interface could eventually be the dominant window into the enormous mass of digital bits being stored out of view. At least maybe NewsStand is a helpful step in that direction.

References

1. Bolt, R.A. *Spatial Data Management*. Massachusetts Institute of Technology, Cambridge, MA, 1979.
2. Veith, R.H. *Visual Information Systems: The Power of Graphics and Video*. G.K. Hall & Co., Boston, 1988.

Richard H. Veith, Port Murray, NJ

Realistic Parallel Programming Productivity

As a user and teacher of parallel programming, I question whether "[t]he time to ... the first successful parallel run" on two cores is indeed a useful measure of the productivity of a language, the metric used by John T. Richards et al. in the study they reported in their article "A Decade of Progress in Parallel Programming Productivity" (Nov. 2014).

First, any notion of productivity must take into consideration the time it takes to ensure a program is correct. A successful run on two cores is hardly a guarantee of correctness. A better criterion would require passing a battery of tests on a range of inputs and processor counts, perhaps supplemented by formal verification. In my own experience using MPI and various threading models, I often find I can

quickly "get something running" using threads, only to spend days debugging subtle race conditions. In contrast, using MPI may take longer at first, but subsequent debugging and verification is more straightforward.

The second issue with the metric is it does not address performance. Richards et al. said they did not measure performance. Why not? It would have been easy and provided insight into the trade-offs inherent in the choice of language. In parallel programming, a significant portion of the development effort goes to ensuring the program exhibits adequate runtime performance. Studies of productivity could take this time into account by, say, requiring the program to obtain some specified speedup using some specified (large) number of processors before it is deemed complete. (Richards et al. did mention X10 programs from another study that performed well, but presumably development of those programs did not end with their first successful two-core run.)

Without doubt, there is a need for parallel programming languages that are more productive, as well as more studies measuring productivity. But, to be persuasive, the studies must use metrics that address all phases of a realistic development process.

Stephen F. Siegel, Newark, DE

Authors' Response:

There is certainly more to developing a parallel application than getting it to run on two cores against a substantial enough set of data to reasonably ensure correctness. Our study found substantial productivity gains across six application types during this early coding, debugging, and testing, and we suspect, but cannot prove, the gains will hold during later scaling, tuning, and verification. As the X10 community grows, we hope to see additional aspects of productivity and performance investigated.

**John T. Richards, Jonathan Brezin,
Calvin B. Swart, and
Christine A. Halverson**

**'And we,
the producers
of the software,
are not responsible
for anything,
no matter what.'**

Long May the Empty Flag Wave

In his "Cerf's Up" column "Heidelberg Laureate Forum II" (Nov. 2014), Vinton G. Cerf explored Ivan Sutherland's idea for detecting "empty" when "nothing has been stored in the register," highlighting the existence of an "'empty' flag" (one bit) to carry the information that a register (or memory location) is empty. Such a flag would mean an N -bit register would require $N+1$ bits. As Cerf said, such "a register of N bits has 2^N values," or not using half its capacity. Another way to achieve the desired result (without wasting capacity) would be to deem a particular bit configuration as signifying "empty." On systems employing "sign and magnitude" representation of numbers there is no need for both a "+0" and a "-0," as they have the same arithmetic properties. One can be used for all arithmetic operations, the other as the "empty flag." Even on systems employing "two's complement arithmetic" the same bit configuration—leading bit "1," all other bits "0"—would signify an empty register. This bit configuration represents a negative number, one unit larger in magnitude than the largest positive number. Using it to signify "empty" removes the asymmetry in number range between positive and negative numbers. A high-speed "comparator" can easily provide an "interrupt" when a program fetches the bit configuration for an "empty" cell. A circuit consisting of an "inverter gate" (for the leading bit) and a "wired-OR" is all the system needs to generate a signal that the "empty flag" has been detected.

The concept of an "empty flag" is not new. Compilers have long generated code incorporating a compari-

son of a fetched variable with the bit configuration of an "empty flag" to detect the use of an uninitialized variable. A "compare" operation and a software "interrupt" on equality are all that is necessary.

Paul B. Schneck, Bala Cynwyd, PA

'Not My Fault' Not Good Enough

For as long as I can remember software being for sale, it has always come with a standard disclaimer, something like "This software is not guaranteed suitable for any purpose or to be free of defects. You use it at your own risk. And we, the producers of the software, are not responsible for anything, no matter what." Next comes a stipulation saying that in order to use the software, you must agree to these terms. This is all so the software producers can make more money, selling quickly developed, untested software full of bugs and security holes they claim is a good product, even when it is not.

Today we read about self-driving cars embedded with millions of lines of code for driving the car. The product liability lawyers of the world are unlikely to let software producers continue to have the luxury of not being responsible for anything, just because they say so or because a user agreed to their terms by clicking a button or by using the software.

People's lives and property are at stake. If a wreck or a person's death is caused by deficiencies or bugs in a car's software, then, agreement or no agreement, the producers of the software should be held accountable. This is how things should have been from day one of software development but never have been.

Software producers will not willingly accept such a change. Instead, it will be forced upon them by their own financial losses, when lawyers start winning lawsuits against them, thereby undoing the producers' self-granted "Get Out of Jail Free" card and negating their bogus claims of "I'm not responsible for anything."

Hal B. Lowe, Oswego, IL

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.

© 2015 ACM 0001-0782/15/01 \$15.00

Coming Next Month in **COMMUNICATIONS**

**Checking NFA
Equivalence with
Bisimulations Up
to Congruence**

**To Govern IT, or
Not to Govern IT?**

**Automated Support for
Diagnosis and Repair**

**Verifying
Computations without
Reexecuting Them**

**A Manifesto in Defense
of Soundness**

**Do-It-Yourself
Textbook Publishing**

Upstart Puzzles

Plus the latest news about recreating audio from visual information, privacy as a function of technology, and disseminating technology to help people.

Inviting Young Scientists



Association for
Computing Machinery

Meet Great Minds in Computer Science and Mathematics

As one of the founding organizations of the Heidelberg Laureate Forum <http://www.heidelberg-laureate-forum.org/>, ACM invites young computer science and mathematics researchers to meet some of the preeminent scientists in their field. These may be the very pioneering researchers who sparked your passion for research in computer science and/or mathematics.

These laureates include recipients of the ACM A.M. Turing Award, the Abel Prize, and the Fields Medal.

The Heidelberg Laureate Forum is **August 23–28, 2015** in Heidelberg, Germany.

This week-long event features presentations, workshops, panel discussions, and social events focusing on scientific inspiration and exchange among laureates and young scientists.

Who can participate?

New and recent Ph.Ds, doctoral candidates, other graduate students pursuing research, and undergraduate students with solid research experience and a commitment to computing research

How to apply:

Online: <https://application.heidelberg-laureate-forum.org/>
Materials to complete applications are listed on the site.

What is the schedule?

Application deadline—**February 28, 2015**.

We reserve the right to close the application website early depending on the volume

Successful applicants will be notified by **April 15, 2015**.

More information available on Heidelberg social media



DOI:10.1145/2691599

ACM's FY14 Annual Report

As I write this letter, ACM has just issued a landmark announcement revealing the monetary level for the ACM A.M. Turing Award will be raised this year to \$1 million,

with all funding provided by Google Inc. The news made the global rounds in record time, bringing worldwide visibility to the award and the Association. Long recognized as the equivalent to the Nobel Prize for computer science, ACM's Turing Award now carries the financial clout to stand on the same playing field with the most esteemed scientific and cultural prizes honoring game changers whose contributions have transformed the world.

This really is an extraordinary time to be part of the world's largest educational and scientific society in computing. Membership is at a record high for the 12th consecutive year, ACM continues to expand its global reach thanks to our thriving hubs in Europe, India, and China, and ACM's position at the forefront of computer science education has never been more prominent than it is today.

This year marked the stellar debut of the Heidelberg Laureate Forum, which brought together 200 students with 40 international recipients of the

This really is an extraordinary time to be part of the world's largest educational and scientific society in computing.

most prestigious awards in computer science and mathematics. I was honored to be a participant in this inaugural event; I found it an inspirational gathering of innovators and under-studies who will be the next award recipients.

This year also marked the publication of highly anticipated ACM-IEEE-CS collaboration *Computer Science Curricula 2013* and the acclaimed *Rebooting the Pathway to Success: Preparing Students for Computer Workforce Needs in the United States*, an exhaustive report from ACM's Education Policy Committee that focused on IT workforce needs at the U.S. state and national levels as well as the educational pathways to meet those needs.

As you read the following pages you will discover a year filled with ambitious initiatives and new services designed to increase the professional value of your ACM membership, wherever you may be in the world. Yet with all these achievements, ACM remains a work in progress and there is still much work to do. Issues of open access and digital library sustainability remain high priorities for ACM's leadership. Extending ACM's global influence and visibility even further will *always* be of utmost importance.

The coming year will see new opportunities and new challenges and ACM is ready to take on both with great determination. As always, we will look to our dedicated volunteers, members, and industry partnerships to help guide us. ACM's future is a journey we take together, and has been my great pleasure to serve as your president for this part of the trip.

Vinton G. Cerf, ACM PRESIDENT
JULY 2012-JUNE 2014

ACM Council

PRESIDENT

Vinton G. Cerf

VICE PRESIDENT

Alexander L. Wolf

SECRETARY/TREASURER

Vicki L. Hanson

PAST PRESIDENT

Alain Chesnais

SIG GOVERNING BOARD CHAIR

Erik Altman

PUBLICATIONS BOARD

CO-CHAIRS

Jack Davidson

Joseph A. Konstan

MEMBERS-AT-LARGE

Eric Allman

Ricardo Baeza-Yates

Cherri Pancake

Radia Perlman

Mary Lou Soffa

Eugene Spafford

Salil Vadhan

SGB COUNCIL

REPRESENTATIVES

Brent Hailpern

Andrew L. Sears

David A. Wood

REGIONAL COUNCIL CHAIRS

ACM India

PJ Narayanan, President

ACM Europe

Fabrizio Gagliardi

ACM China

Jianguang Sun

Younghao Lin

ACM-W

Valerie Barr

USACM

Eugene Spafford

Education Board

Andrew McGettrick

Practitioners Board

Stephen R. Bourne

ACM Headquarters

CHIEF EXECUTIVE OFFICER

John R. White

CHIEF OPERATING OFFICER

Patricia M. Ryan

Highlights of ACM Activities: July 1, 2013–June 30, 2014

ACM, the Association for Computing Machinery, is an international scientific and educational organization dedicated to advancing the arts, sciences, and applications of information technology.

Publications

The cornerstone of ACM publications is the ACM Digital Library (DL) serving as the primary distribution mechanism for all the Association's publications as well as host to scientific periodicals and a set of conference proceedings from external organizations. The DL, now available at 2,650 institutions in 190 countries, boasts an estimated five million users worldwide. An additional 31,000 individual subscribers in 196 countries have DL access. The result of this widespread availability led to more than 20 million full-text downloads in FY14.

ACM is committed to increasing the scope of material available via the DL. Last year, over 23,000 full-text articles were added, bringing total DL holdings to 425,000 articles. ACM's *Guide to Computing Literature* is also integrated within the DL. More than 105,000 works were added to the bibliographic database in FY14, bringing the total *Guide* coverage to more than 2.3 million works.

ACM is the publisher of 79 periodicals, including 44 journals and transactions, 8 magazines, and 32 newsletters as of year-end FY14. During the year, ACM added 450 volumes of conference and related workshop proceedings to its portfolio. The ACM International Conference Proceedings Series (ICPS) added 94 new volumes, a significant increase over FY13.

The ACM Publications Board continues to examine ways to offer authors and readers greater open access to our published material. In the past year, in addition to continuing current policies and experiments offering authors a choice of copyright options and a suite of author-retained rights, the Publications Board approved in con-

cept the creation of an umbrella Gold-OA ACM journal. Moreover, the board helped to found the Clearinghouse for the Open Research of the U.S. (CHORUS), a publishing industry effort to facilitate compliance with U.S. open access mandates.

ACM Press Book Series, in partnership with Morgan & Claypool publishers, finished its first full year of operation. Three titles were introduced in FY14, all available in the DL.

The Coalition to Diversify Computing presented its inaugural edition of *Celebration of Latinas in Computing* at the Grace Hopper Celebration in early FY14. This publication followed the CDC's *Celebration of Black Women in Computing*, which is now in its second edition.

The Computer Science Teachers Association (CSTA) worked with Microsoft to develop a set of curricular resources. *CS Principles: Computation in Action Curriculum* was designed to engage a variety of students in socially relevant, project-based learning.

SIGLOG debuted its first newsletter in FY14 and SIGMOBILE introduced an epub format for making content more easily accessible using popular mobile devices. Indeed, MobiSys 2014

provided epub files for all accepted papers and made them freely accessible to all conference attendees.

Education

ACM continues to lead the computer science education community through the work of the ACM Education Board, the ACM Education Council, ACM SIGCSE, Computer Science Teachers Association (CSTA), and ACM Education Policy Committee.

The CS2013 Final Report, a comprehensive joint activity of ACM and IEEE-CS, was published at year-end 2013 to great acclaim. The report contains guidance for undergraduate CS programs, including a revised Body of Knowledge and over 80 course exemplars.

The first Heidelberg Laureate Forum brought together 200 young researchers, 22 Turing laureates, and 11 Fields/Abel laureates. The weeklong event offered students a unique opportunity to share ideas with legends in the fields of computer science and mathematics.

ACM's Education Board was charged with overseeing the launch of the inaugural Learning@Scale conference in Atlanta last March. The extremely successful event marked the first time a conference was spawned by a Board within ACM.

The ACM Education Policy Committee (EPC) released its much-anticipated report: *Rebooting the Pathway to Success: Preparing Students for Computer Workforce Needs in the United States*. This latest report calls on education and business leaders and public policy officials nationwide to take immediate action aimed at filling the pipeline of qualified students pursuing computer science and other computer-related degrees, and to prepare them for the 21st century workforce.

ACM's partnering with Code.org represents a significant investment in seeing real computer science exist and count in U.S. high schools. It is an

ACM's partnering with Code.org represents a significant investment in seeing real computer science exist and count in U.S. high schools.

investment that has paid off substantially. At the end of FY14, 22 states had moved to enact legislation to make computer science count as a core graduation credit in science and math. Moreover, over 17 million students, parents, and teachers participated in the Hour of Code during Computer Science Education Week.

ACM, through the EPC, participated in the Leadership Council of the STEM Education Coalition. White House briefs on STEM policy allowed education stakeholders to learn about the Administration's proposed STEM reorganization plans and to discuss such issues with policymakers.

The CSTA continues to thrive as a key component in ACM's efforts to support computer science at the high school level. CSTA membership reached almost 17,000, an increase of 21% over FY13. Indeed, there are now 55 CSTA chapters in 25 U.S. and four Canadian provinces.

ACM Europe and Informatics Europe continued the work they began last year with their report "Informatics Education—Europe Cannot Afford to Miss the Boat," which highlights the need for increased investment in CS education for Europe to remain competitive in a global technology area. The group, newly named Committee on European Education, plans to supplement this report by mapping out informatics education both for curricula and for teaching methodologies across Europe.

ACM India established an Education Board with the goal of increasing knowledge and interest in computing throughout the nation's school system, improve teaching methods and content in engineering colleges, and encourage the use of online material for learning about computing.

Professional Development

The Practitioners Board and Professional Development Committee (PDC) directed many new products and initiatives designed for computing professionals and managers.

In FY14, ACM PDC continued the successful webinar series with consistent monthly webcasts, tighter integration with SIGs, and a more diverse set of topics. The committee added 13 webinars in FY14. Among the featured

The ACM-W Scholarship program further enhanced its support for women undergraduate and graduate students in computer science and related programs.

topics: IPv6 security, Intro to WebGL, Ruby for the Nuby, Hadoop/Big Data, and MOOCs.

For the fourth consecutive year, *ACM Queue*, the online practitioner's magazine spirited by the Practitioner Board, surpassed the million-pageview threshold, with 1,158,741 pages viewed over the last 12 months.

Public Policy

ACM's U.S. Public Policy Council (US-ACM) made significant progress this year in delivering on its mission to educate and inform policy leaders, ACM members, the computing community, and the public about U.S. policy issues related to IT and computing.

One of USACM's major projects this year was the establishment of the ACM Merit Award in Public Policy to honor efforts by individuals or groups that exhibit excellence in computing policy engagement. The ACM Awards committee approved USACM's proposal for this new policy award last May, with the agreement the award will be given every other year, beginning in 2016.

USACM also participated in several multistakeholder partnerships to help educate and raise awareness of technology issues among legislators and the public at large. Among those joint efforts were: working on the Identity Ecosystem Steering Group, USPTO software partnership, Internet policy task force, and Congressional Internet Caucus Advisory Committee.

ACM Europe established a EUACM body (similar to USACM) to address public policy concerns affecting the

European computing community. The first policy position of ACM Europe was issues regarding the Trans-Atlantic Trade Agreement.

The Committee on Computers and Public Policy assists ACM in a variety of internationally relevant issues pertaining to computers and public policy. CCpp's respected *ACM Forum on Risks to the Public in Computers and Related Systems*, designed to discuss potential and serious computer-related risks with a global audience, covers such issues as human safety, privacy, election integrity, and societal/legal responsibilities. Over this fiscal year, 63 issues of the Risks digest were produced.

Students

The 38th Annual ACM International Collegiate Programming Contest (ACM-ICPC) was hosted by Ural Federal University, Ekaterinburg, Russia, with 122 teams competing in the World Finals. Earlier rounds of the competition included nearly 32,000 contestants representing 2,200 universities from 94 countries. Financial and systems support for ICPC is provided by IBM. The top four teams won Gold medals as well as employment or internship offers from IBM.

The ACM Student Research Competition (SRC), sponsored by Microsoft Research, continues to offer a unique forum for undergraduate and graduate students to present their original research at well-known ACM-sponsored and co-sponsored conferences before a panel of judges and attendees. This year's SRC saw graduate and undergraduate winners compete against more than 215 participants in contests held at 17 ACM conferences.

The ACM-W Scholarship program further enhanced its support for women undergraduate and graduate students in CS and related programs. The committee awarded over 40 student scholarships in FY14 to students to attend research conferences around the world.

Internationalization

Representatives from government, industry, and academia considered Europe's future global competitiveness in research and innovation at a special open session at the ACM Europe

Council meeting last March. The panel explored “The Future of Computer Science Research in Europe,” particularly some of the weaknesses in the Horizon 2020 research program, for which the EU has invested nearly 80 billion euros. The meeting proved a unique opportunity to address the funding, educational, and research issues facing CS research in Europe.

ACM Europe also steered several activities to raise the visibility of ACM's mission and value throughout Europe. The Council of European Chapter Leaders (CECL) was created last year to promote networking between chapter leaders and to help grow more sustainable chapters. ACM Europe has been working tirelessly to increase the number of ACM conferences in Europe. They also further strengthened their collaboration with Informatics Europe to share resources and organizational insights.

ACM-W Europe held its first event at Manchester University last March. womENcourage Celebration of Women in Computing brought students, career researchers and practitioners to present, share, and celebrate their technical accomplishments and experiences working and studying computing.

The ACM China Council was reformulated this year; its new agenda focuses on growing the number of chapters, launching student chapters, starting new China-wide awards, bringing more conferences to China, and engaging in education work, particularly the translation of CS2013 to Chinese. In FY14, the number of chapters in China grew 77%.

ACM India is now in full operation as a separate legal entity. The Research Board of ACM India continues to develop; a highlight of its efforts included a major faculty summit organized by ACM India and MSR India. In FY14, the number of chapters in India grew 16%.

A series was organized by ACM-W India jointly with Madras Management Association. The talks were intended for company executives, professionals, administrators, consultants, and students. Of special interest was the first-of-its-kind CODHER Hackathon to promote participation among women in technical events.

Balance Sheet: June 30, 2014 (in Thousands)

ASSETS

Cash and cash equivalents	\$37,024
Investments	79,657
Accounts receivable and other current assets	7,593
Deferred conference expenses and other assets	7,661
Fixed assets, net of accumulated depreciation and amortization	1,046
Total Assets	\$132,981

LIABILITIES AND NET ASSETS

Liabilities:	
Accounts payable, accrued expenses, and other liabilities	\$13,996
Unearned conference, membership, and subscription revenue	24,627
Total liabilities	\$38,623

Net assets:	
Unrestricted	87,851
Temporarily restricted	6,507

Total net assets **94,358**

Total liabilities and net assets **\$132,981**

Optional Contributions Fund — Program Expense (\$000)	
Education Board accreditation	\$95
USACM Committee	20
Total expenses	\$115

Electronic Community

ACM's e-Rights transfer application system completed its first full year of operations. This comprehensive system gives authors new options for managing rights and permissions. The system, used by all ACM journals, proceedings, and magazines, completely automates the rights transfer process.

Statistics prove that more and more readers are accessing ACM's magazines via mobile devices. *Communications of the ACM*, *ACM Inroads*, *XRDS*, and *Interactions* are accessible as easy-to-use mobile apps for iPhones, iPads, and Android devices. These downloadable apps enable members to access their favorite ACM magazines in a new way.

ACM-W debuted a new website designed to act as an international hub for all of the council's events, announcements, programs, and online services. The site also introduced a

digital version of the popular ACM-W Connections newsletter.

ACM SIGs across the board continue to strengthen their online presence to build global awareness as well as incorporate social media into their operation at every opportunity. SIGHPC hosted two webinars during the year that broke all previous ACM records for participation. The SIG also chartered the first ACM “virtual chapter,” which focuses on education.

Conferences

SIGGRAPH 2013 welcomed 17,162 artists, research scientists, gaming experts and developers, filmmakers, students, and academics from 77 countries to Anaheim, CA. Over 1,350 speakers and contributors participated in the event and SIGGRAPH's exhibition hall drew 180 industry organizations from 15 countries.

Statement of Activities: Year ended June 30, 2014 (in Thousands)

REVENUE	Unrestricted	Temporarily Restricted	Total
Membership dues	\$8,323		\$8,323
Publications	20,375		20,375
Conferences and other meetings	27,161		27,161
Interests and dividends	1,870		1,870
Net appreciation of investments	6,813		6,813
Contributions and grants	3,994	\$1,556	5,550
Other revenue	253		253
Net assets released from restrictions	1,763	(1,763)	0
Total Revenue	70,552	(207)	70,345
EXPENSES			
Program:			
Membership processing and services	\$812		\$812
Publications	11,207		11,207
Conferences and other meetings	24,606		24,606
Program support and other	9,446		9,446
Total	46,071		46,071
Supporting services:			
General administration	11,263		11,263
Marketing	1,167		1,167
Total	12,430		12,430
Total expenses	58,501		58,501
Increase (decrease) in net assets	12,051	(207)	11,844
Net assets at the beginning of the year	75,800	6,714	82,514
Net assets at the end of the year	\$87,851	\$6,507	\$94,358*

* Includes SIG Fund balance of \$39,834K

Both SIGCOMM and SIGKDD drew record-high attendance to their flagship conferences. SIGCOMM 13 was held this year in Hong Kong, and KDD 2013 in Chicago noted a 16% growth in participants over the previous year.

Recognition

There were 147 new chapters chartered in FY14. Of the 36 new professional chapters, five were U.S.-based and 31 were based outside the U.S.; of the 111 new student chapters, 53 were established in the U.S. and 58 outside the U.S.

The ACM Fellows Program recognized 50 members for their contribu-

tions to computing and computer science in FY14.

ACM also named 40 new Distinguished Members in FY14, of which there were four Distinguished Educators, six Distinguished Engineers, and 30 Distinguished Scientists, bringing the total number of Distinguished Members to 371.

In the field of multimedia retrieval conferences, the Chinese Computing Federation Conference Rankings named ACM's International Conference on Multimedia Retrieval first in the world.

© 2015 ACM 0001-0782/15/01

2014 ACM Award Recipients

ACM A.M. TURING AWARD
Leslie Lamport

**ACM-INFOSYS FOUNDATION
AWARD IN THE COMPUTING
SCIENCES**
David Blei

GRACE MURRAY HOPPER AWARD
Pedro Felipe Felzenszwalb

**KARL V. KARLSTROM
OUTSTANDING EDUCATOR
AWARD**
Susan H. Rodger

**OUTSTANDING CONTRIBUTION
TO ACM AWARD**
Donna Cappel
Russell Harris

**DISTINGUISHED SERVICE
AWARD**
Gerhard T. Goos
Juris Hartmanis
Jan van Leeuwen

**PARIS KANELAKIS THEORY
AND PRACTICE AWARD**
Robert D. Blumofe
Charles E. Leiserson

SOFTWARE SYSTEM AWARD
Coq
Thierry Coquand
G rard Huet
Christine Paulin-Mohring
Bruno Barras
Jean-Christophe Filli tre
Hugo Herbelin
Chet Murthy
Yves Bertot
Pierre Cast ran

**2014 ACM-IEEE CS
ECKERT-MAUCHLY AWARD**
Trevor Mudge

**2014 ACM-IEEE CS KEN
KENNEDY AWARD**
Charles E. Leiserson

ACM PRESIDENTIAL AWARD
Mehran Sahami

**DOCTORAL DISSERTATION
AWARD**
Sanjam Garg

HONORABLE MENTION
Grey Ballard
Shayan Oveis Gharan

**THE 2014-2015 ACM-W
ATHENA LECTURER AWARD**
Susan T. Dumais

**ACM INDIA DOCTORAL
DISSERTATION AWARD**
Ramprasad Saptharishi

ACM

ON A MISSION TO SOLVE TOMORROW.



Dear Colleague,

Computing professionals like you are driving innovations and transforming technology across continents, changing the way we live and work. We applaud your success.

We believe in constantly redefining what computing can and should do, as online social networks actively reshape relationships among community stakeholders. We keep inventing to push computing technology forward in this rapidly evolving environment.

For over 50 years, ACM has helped computing professionals to be their most creative, connect to peers, and see what's next. We are creating a climate in which fresh ideas are generated and put into play.

Enhance your professional career with these exclusive ACM Member benefits:

- Subscription to ACM's flagship publication ***Communications of the ACM***
- Online books, courses, and webinars through the **ACM Learning Center**
- Local Chapters, Special Interest Groups, and conferences all over the world
- Savings on peer-driven specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

We're more than computational theorists, database engineers, UX mavens, coders and developers. Be a part of the dynamic changes that are transforming our world. Join ACM and dare to be the best computing professional you can be. Help us shape the future of computing.

Sincerely,

A stylized, handwritten signature in black ink, appearing to read 'Alexander Wolf'.

Alexander Wolf
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING. JOIN ACM TODAY.

ACM is the world's largest computing society, offering benefits that can advance your career and enrich your knowledge with life-long learning resources. We dare to be the best we can be, believing what we do is a force for good, and in joining together to shape the future of computing.

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

- ☐ Professional Membership: \$99 USD
- ☐ Professional Membership plus
ACM Digital Library: \$198 USD (\$99 dues + \$99 DL)
- ☐ ACM Digital Library: \$99 USD
(must be an ACM member)

ACM STUDENT MEMBERSHIP:

- ☐ Student Membership: \$19 USD
- ☐ Student Membership plus ACM Digital Library: \$42 USD
- ☐ Student Membership PLUS Print *CACM* Magazine: \$42 USD
- ☐ ACM Student Membership w/Digital Library
PLUS Print *CACM* Magazine: \$62 USD

- ☐ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in all aspects of the computing field. Available at no additional cost.

Priority Code: CAPP

Payment Information

Name

ACM Member #

Mailing Address

City/State/Province

ZIP/Postal Code/Country

Email

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc, in U.S. dollars or equivalent in foreign currency.

- ☐ AMEX ☐ VISA/MasterCard ☐ Check/money order

Total Amount Due

Credit Card #

Exp. Date

Signature

Return completed application to:
ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Satisfaction Guaranteed!

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)

Hours: 8:30AM - 4:30PM (US EST)
Fax: 212-944-1318

acmhelp@acm.org
acm.org/join/CAPP

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.



Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/2686591

<http://cacm.acm.org/blogs/blog-cacm>

A Valuable Lesson, and Whither Hadoop?

Valerie Barr considers outcomes from the Grace Hopper Celebration, while Michael Stonebraker ponders the past, present, and possible future of the "Hadoop stack."



Valerie Barr
"Satya Nadella
and Grace Hopper"
<http://bit.ly/1s0eTgd>
Oct. 16, 2014

I have just come home from the Grace Hopper Celebration of Women in Computing. Amazingly, I have now attended 13 of the 14 GHCs that have taken place, including the very first one in 1994. The conference was huge this year! There were 8,000 attendees, over 2,800 students from 441 schools, and attendees from 67 countries. The exhibition hall was a beehive of activity and I am happy to report we had a lot of traffic at the joint ACM/ACM-W/ACM CCECC/CSTA booth, making lots of great contacts among our various constituencies.

Of course, there has been a lot of commentary online about the remarks made by Satya Nadella, CEO of Microsoft. In case you missed it, in response to a question asked by Maria Klawe (president of Harvey Mudd College and member of Microsoft's Board of Directors), Nadella said women should not ask for pay raises (in truth, he said nobody should ask for raises, but Maria's

question was specifically about women). He said he trusted in the review system, and that if someone's work is good it would eventually be rewarded. As you can imagine, the conference and the Web lit up, followed soon thereafter by all other forms of media as well.

Here are my thoughts on this matter:

1. As a result of the ruckus, the Grace Hopper Celebration and the situation for women in computing got more press and more visibility than ever before. That is a very good outcome.

2. The situation provided a great opportunity for people to talk about the fact that meritocracy does not work when there is implicit bias. Nadella may think the Microsoft review process is fair and unbiased, but given that women in tech in the U.S. earn only \$0.86 for every \$1.00 earned by men, he should seriously research what the actual numbers are for Microsoft, and then adjust salaries accordingly.

3. This incident provided a very valuable lesson for the students at the conference. The companies recruiting at Hopper are trying very hard to improve their diversity statistics. The conference gives them access to a lot of women job can-

didates, and they treat the students very well (fancy swag, food, private events, interview booths, raffles, etc.). It would be easy for the students to be deluded into thinking everything is great now in the tech world and women are always well-treated. Nadella's comments serve as a reminder that women entering the field still have to be prepared to advocate for themselves when they negotiate starting salaries and subsequent raises.

4. I do not doubt for a minute that Nadella, along with many other tech CEOs right now, considers himself a strong advocate for women in computing. He is noteworthy for being the first tech CEO of that level to come to Hopper, and he spent a lot of time there. He still has some things to learn, as do many people in this field. As we know, there are many hearts and minds that need to be changed, and even some of our best allies have a lot to learn.



Michael Stonebraker
"What Does
'Big Data' Mean?"
<http://bit.ly/1pmVITk>
September 21, 2012

Since my last blog posting with Jeremy Kepner on this topic in 2012 (<http://bit.ly/1tF6vZ7>), a lot of water has gone under the bridge. I feel compelled to point out a few facts and opinions and report on a couple of announcements. I conclude with a prediction on where the "Hadoop stack" might be going.

The first announcement was Cloudera releasing a new DBMS, Impala (<http://bit.ly/13EAVRv>), which runs on HDFS.

Put simply, Impala is architected exactly like all of the shared-nothing parallel SQL DBMSs, serving the data warehouse market. Specifically, notice the MapReduce layer has been removed, and for good reason. As some of us have been pointing out for years, MapReduce is not a useful internal interface inside a SQL (or Hive) DBMS (<http://bit.ly/1x9MKYu> and <http://bit.ly/1AcBT5C>). Impala was architected by savvy DBMS developers, who know the above pragma. In fact, development activity similar to Impala is being done by both HortonWorks and Facebook. This, of course, presents the Hadoop vendors with a dilemma. Historically, “Hadoop” referred to the open source version of MapReduce written by Yahoo. However, Impala has thrown this layer out of the stack. How can one be a Hadoop vendor, when Hadoop is no longer in the mainstream stack?

The answer is simple: redefine “Hadoop,” and that is exactly what the Hadoop vendors have done. The word “Hadoop” is now used to mean the entire stack. In other words, HDFS is at the bottom, on top of which run Impala, Map-Reduce, and other systems. On top of these systems run higher-level software such as Mahout. The word “Hadoop” is used to refer to the entire collection.

The second recent announcement comes from Google, who announced MapReduce is yesterday’s news and they have moved on, building their software offerings on top of better systems such as Dremel, Big Table, and F1/Spanner (<http://bit.ly/1pi7QVC>). In fact, Google must be “laughing in their beer” about now. They invented MapReduce to support the Web crawl for their search engine in 2004. A few years ago they replaced MapReduce in this application with BigTable, because they wanted an interactive storage system and MapReduce was batch-only. Hence, the driving application behind MapReduce moved to a better platform a while ago. Now Google is reporting they see little-to-no future need for MapReduce.

It is indeed ironic that Hadoop is picking up support in the general community about five years after Google moved on to better things. Hence, the rest of the world followed Google into Hadoop with a delay of most of a decade. Google has long since abandoned it. I wonder how long it will take the rest of the world to follow Google’s direction and do likewise...

Notice the Hadoop vendors are now on a collision course with the data warehouse vendors. They are now implementing (or have implemented) the same architecture supported by the data warehouse folks. Once they have a few years to solidify their implementations, they will probably offer competitive performance. Meanwhile, most of the data warehouse vendors support HDFS, and many offer features to support semi-structured data. Hence, the data warehouse market and the Hadoop market will quickly converge. May the best systems win in the resulting head-to-head donnybrook!

Now I turn to HDFS, the only common building block left in the Hadoop stack. Notice clearly that HDFS is a file system, capable of storing bytes of data, a feature we have all come to expect on any computing platform. There are two possible world views of where HDFS might go in the future. If you take a file system view of the world, then users want a common distributed file system and HDFS is a perfectly reasonable alternative.

On the other hand, from the point of view of a parallel SQL/Hive DBMS, HDFS is a “fate worse than death.” A DBMS always wants to send the query (small kilobytes) to the data (lots of gigabytes) and never the other way around. Hence, hiding the location of data from the DBMS is death, and the DBMS will go to great lengths to circumvent this feature. All parallel DBMSs, either from the warehouse vendors or from the Hadoop vendors, will turn off location transparency, making HDFS look like a collection of Linux file systems, one per node. Likewise no DBMS wants file system replicas. See <http://bit.ly/1x4Blek> for an extensive discussion of this point. In short, load balancing, query optimization, and transaction considerations favor a DBMS-supplied replication system.

If it turns out the DBMS point of view prevails in the marketplace over time, then HDFS will atrophy as the DBMS vendors abandon its features. In such a world, there is a local file system at each node, a parallel DBMS supporting a high-level query language, and various tools on top or extensions defined via user-defined functions. In this scenario, Hadoop will morph into a standard shared-nothing DBMS with a collection of DBMS vendors competing for your software dollar.

On the other hand, if the file system point of view prevails, then HDFS will probably survive largely intact with a potpourri of tools running on top of it. Features users take for granted in a DBMS environment, such as load-balancing, auditing, resource governors, data independence, data integrity, high availability, concurrency control, and quality of service will be slow to come to file system users. There will be no higher-level standard interfaces in this scenario. In other words, a DBMS view of the world offers a bunch of useful services, and users would be well advised to consider carefully if they want to run lower-level interfaces.

In either scenario, the only common piece of software is a file system, and the Hadoop vendors will be selling file-system based tools, either DBMS ones or other stuff (or maybe both). In effect, they will join the ranks of the system software vendors selling software or services. May the best products win!

Comments

Great article, one of many in this series that have proven to be prophetic.

I think there is always tension in the marketplace for ideas when there is a choice of “extending” an existing technology vs. scrapping the old in favor of something revolutionary. In this case, large portions of the market chose the revolutionary, without considering all of the features that would eventually be needed to flesh out the technology for enterprise production purposes. That is not to say Hadoop and MapReduce are not useful or productive for some applications; they are. It is, however, a reach to say Hadoop and MapReduce make existing DBMS technology/products obsolete.

The significant aspects are the introduction of NoSQL databases, new forms of concurrency/transactions, for example, where “eventual consistency” is “good enough,” and “big data.” The particular products that introduce these will most likely not be the products that eventually make these features enterprise capable. Like COBOL programs, the applications using existing technology will probably never disappear until all the SQL programmers die off.

—C Rofer

Valerie Barr is professor at Union College. **Michael Stonebraker** is adjunct professor at the Massachusetts Institute of Technology.

© 2015 ACM 0001-0782/15/01 \$15.00



Glasgow June 22-25

ACM Creativity and Cognition 2015 will serve as a premier forum for presenting the world's best new research investigating computing's impact on human creativity in a broad range of disciplines including the arts, design, science, and engineering.

Creativity and Cognition will be hosted by The Glasgow School of Art and the City of Glasgow College. The 2015 conference theme is Computers | Arts | Data. The theme will serve as the basis for a curated art exhibition, as well as for research presentations.



Call for Papers, Posters and Demonstrations

Papers submission deadline: 6th January

Posters submission deadline: 6th March

Demonstrations submission deadline: 6th March 2015

Call for Workshops

Deadline for submission: 6th March 2015

We invite Workshops to be presented on the day preceding the full conference

Call for Artworks

Deadline for submission: 6th March 2015

We are calling for proposals for artworks, music, performances and installations to be presented in conjunction with the conference.

Computers

+ Art + Data

In Search of Bayesian Inference

Long relegated to the statistical backburner, Bayesian Inference is undergoing a renaissance.

IN THE EARLY morning of June 1, 2009, Air France flight AF 447, carrying 228 passengers and crew, disappeared over a remote section of the Atlantic Ocean. French authorities organized an international search; after about six days, aircraft and ships started finding debris and bodies from the crash, but could not find the airplane itself. A month-long search along the intended flight path to try to pick up signals from the airplane's underwater locator beacons turned up nothing; neither did a side-scan sonar search in August.

The French Bureau d'Enquêtes et d'Analyses pour la sécurité de l'aviation civile (in English, "Bureau of Enquiry and Analysis for Civil Aviation Safety," or BEA for short) turned to oceanographic experts to estimate the location of the wreckage based on how the recovered bodies and debris might have drifted along the surface currents. Yet side-scan sonar searches in April and May 2010 based on the predictions of the drift model failed to find the airplane.

Finally, the BEA asked Metron, a scientific consulting firm based in Reston, VA, to generate a probability map for the airplane's location using Bayesian inference, a statistical approach



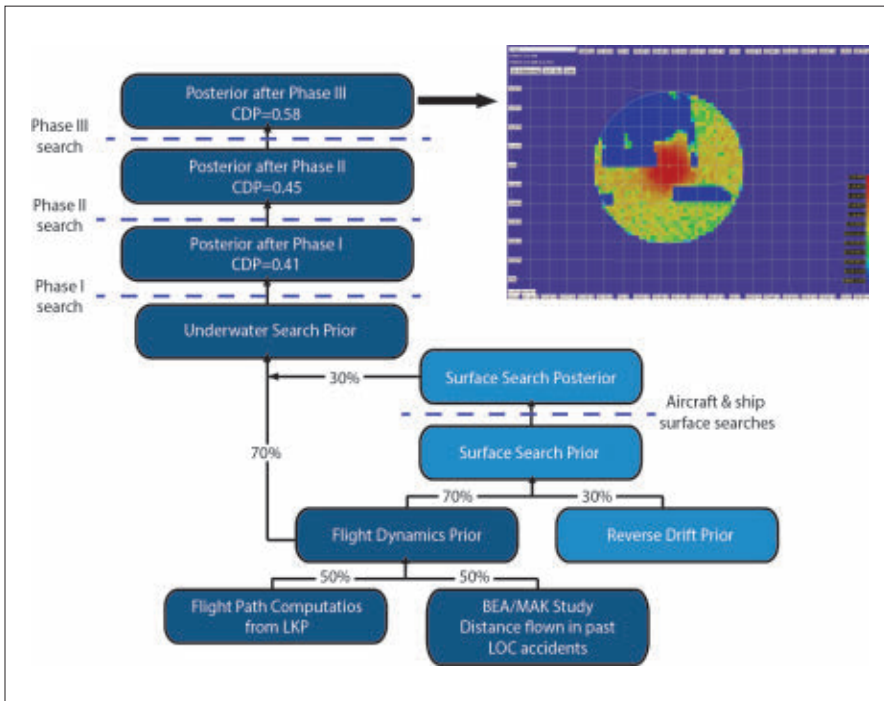
An Air France A330-203 landing at Paris-Charles de Gaulle Airport. This aircraft was destroyed during flight AF 447 in 2009 while carrying 228 passengers and crew.

to combining prior beliefs and experiences with new evidence. Metron started by constructing a probability map based on the initial data about the flight's disappearance, then used Bayes' Law to incorporate the evidence provided by the failures of the various search attempts.

A search in 2011 based on this new probability map found the wreckage within a week.

"Failure to use a Bayesian approach in planning the 2010 search delayed the discovery of the wreckage by up to one year," the Metron team wrote in the February 2014 issue of *Statistical Science*. The success of the Bayesian analysis, the team wrote, "provides a powerful illustration of the value of a methodical, Bayesian approach to search planning."

While Bayesian inference has been



A summary of how the Probability Distribution Function (a probability map) for the location of underwater wreckage of Air France flight AF 447 was computed, using Bayesian Inference.

around for over 250 years, it has swung in and out of favor over the centuries. During most of the 20th century, the approach was relegated to the statistical back burner, although even during that time it enjoyed some notable successes, such as Alan Turing's use of Bayesian statistics to help crack the Enigma encryption machines used by Germany in World War II.

In the past two decades, Bayesian inference has undergone a renaissance. It is now used in a wide range of applications, thanks in large part to increases in computing power and mathematical techniques that have made its large computations feasible. The approach was catapulted into the public eye when Nate Silver, on his FiveThirtyEight blog, used it to predict correctly which way every state would go in the 2012 U.S. Presidential election. The issue of *Statistical Science* featuring the account of AF 447—a special issue dedicated to Bayesian inference—describes its use in a wide range of applications, including finding distant quasars, estimating HIV prevalence in different regions, and explaining the phenomenon that richer people tend to vote Republican while richer states tend to vote Democrat.

“There are hundreds or thousands

of us [Bayesians] now, doing all sorts of things,” said Joseph B. Kadane of Carnegie Mellon University in Pittsburgh, who has used Bayesian inference, for instance, to demonstrate police bias in traffic stops along the New Jersey Turnpike.

Prior Knowledge

In its most basic form, Bayes' Law is a simple method for updating beliefs in the light of new evidence. Suppose there is some statement A that you initially believe has a probability $P(A)$ of being correct (what Bayesians call the “prior” probability). If a new piece of evidence, B, comes along, then the probability that A is true given that B has happened (what Bayesians call the “posterior” probability) is given by

$$P(A|B) = P(B|A)P(A)/P(B)$$

where $P(B|A)$ is the likelihood that B would occur if A is true, and $P(B)$ is the likelihood that B would occur under any circumstances.

Consider an example described in Silver's book *The Signal and the Noise*: A woman in her forties has a positive mammogram, and wants to know the probability she has breast cancer. Bayes' Law says that to answer this question, we need to know three

things: the probability that a woman in her forties will have breast cancer (about 1.4%); the probability that if a woman has breast cancer, the mammogram will detect it (about 75%); and the probability that any random woman in her forties will have a positive mammogram (about 11%). Putting these figures together, Bayes' Law—named after the Reverend Thomas Bayes, whose manuscript on the subject was published posthumously in 1763—says the probability the woman has cancer, given her positive mammogram result, is just under 10%; in other words, about 9 out of 10 such mammogram results are false positives.

In this simple setting, it is clear how to construct the prior, since there is plenty of data available on cancer rates. In such cases, the use of Bayes' Law is uncontroversial, and essentially a tautology—it simply says the woman's probability of having cancer, in light of her positive mammogram result, is given by the proportion of positive mammograms that are true positives. Things get murkier when statisticians use Bayes' rule to try to reason about one-time events, or other situations in which there is no clear consensus about what the prior probabilities are. For example, large passenger airplanes do not crash into the ocean very often, and when they do, the circumstances vary widely. In such cases, the very notion of prior probability is inherently subjective; it represents our best belief, based on previous experiences, about what is likely to be true in this particular case. If this initial belief is way off, we are likely to get bad inferences.

In the early 20th century, a group of statisticians led by Ronald Fisher tried to eliminate subjectivity from statistical inference. Their “frequentist” approach, which views probability not as a degree of belief but as the relative frequency of events that can be repeated many times, became the dominant statistical paradigm.

Bayesians, however, point out that many aspects of frequentist inference—for example, what confidence level you use to make a decision—are subjective. “It's not that you escape subjective judgments; it's that you don't explicitly acknowledge them,” said Lawrence Stone, chief scientist at Metron.

With time, more and more statisticians started arguing that completely rejecting the use of priors was throwing out the baby with the bathwater. "It's not good to ignore information just because it's not quantified by 1,000 tests," Stone said.

"People realized that the rigor was a bit of a straightjacket," said Michael Jordan, a statistician and computer scientist at the University of California, Berkeley. "You couldn't talk about subjective probabilities and expert opinions."

For decades, however, Bayesian analysis was too computationally intensive to carry out in many cases. The approach typically involves calculating high-dimensional integrals, whereas frequentist approaches more often involve optimization, which is easier from a computational standpoint. By the 1980s and 1990s, however, increases in computing power, combined with the development of Markov chain Monte Carlo methods for calculating numerical approximations to high-dimensional integrals "liberated Bayesian inference, and made it much more prominent," Jordan said.

Bayesian inference is most useful, Jordan said, in domains such as physics and biology, where experts can provide good models from which to construct the prior. It is often less suitable, he said, in big data arenas in which no clear model is known. "In the majority of big data applications that you see in technology and science, it's hard to start with models and prior probabilities for a lot of the exploratory analyses being done."

Bayesian analysis can sometimes be useful even in situations in which there are hundreds or thousands of parameters, said James Berger, a statistician at Duke University in Durham, NC. "One of my favorite examples is when we were doing an astronomy problem with 600 unknowns," he said. "One parameter was quite important, and we had a lot of knowledge about it." The team constructed a prior for that parameter based on its model, and used what Bayesians call an "objective" prior for the other 599 parameters.

Today, many statisticians use both the Bayesian and the frequentist paradigms. The two approaches are not so much opposing as complementary, said

Bayesian inference is most useful in domains where experts can provide good models from which to construct the prior.

statistician Bradley Efron of Stanford University. "They're plowing the same field, just in orthogonal directions."

Failed Beacons

In the case of flight AF 447, the BEA had determined the wreckage must lie within 40 nautical miles of the plane's last known position, which had been transmitted by satellite shortly before the plane disappeared. The Metron team constructed a prior probability distribution for the location of the wreckage that incorporated this information with data about the impact points of nine previous commercial airplane crashes and a drift model that reversed the paths of the recovered bodies back to the time of the crash. The drift model entailed many uncertainties, so the team subjectively decided to give it relatively low weight in the prior.

Next, the team used Bayes' Law to update this probability map in light of the four failed searches. Most statistical techniques cannot handle data that comes in so many different flavors—surface and underwater searches with different types of equipment, information about the plane's flight path, the drift model, and so forth—but Bayesian inference allows statisticians to easily combine many different types of measurements and data. Each measurement simply gets transformed into a likelihood function on the space of all possible locations for the airplane, representing the likelihood of obtaining that particular measurement if the airplane is in that particular spot. Bayes' Law then uses this likelihood function to update the prior, resulting in the posterior distribution.

ACM Member News

HAAS MAKES A SPLASH WITH PROTOTYPE PLATFORM



"I've had a 'split-brain' career," observes Peter J. Haas, a research staffer at IBM's Almaden

Research Center in San Jose, CA since 1987. Haas' current work on IBM's Splash project exemplifies that dichotomy. Splash is a prototype platform that facilitates collaboration across disciplines by combining heterogeneous datasets, simulation, and statistical and optimization models using information-integration techniques.

Haas graduated Harvard University with a B.S. in engineering and applied sciences, then went to Stanford University to earn a master's degree in environmental engineering and a Ph.D. in operations research.

Haas was named an IBM Master Inventor in 2012, helped by his work in stochastic simulation. "Real-world systems are uncertain and random," Haas observes. In stochastic simulation, researchers devise mathematical models to simulate behavior and predict probable outcomes. Haas has also applied statistical techniques from stochastic simulation to "deep dive database issues" like query optimization, database sampling, and probabilistic databases.

"Splash's goal is to support policy, planning and investment decisions to solve complex problems like population, health and safety," Haas says. Splash uses composite models to conduct deep predictive analytics, enabling "what-if" analyses, and allowing researchers to collaborate across disciplines and understand the trade-offs inherent in complex problems and proposed solutions. "Splash gives me a mechanism to tear down informational and conceptual model silos," Haas says, creating a broader picture of individual components, like why a water main break affects hospital emergency room operations.

In his spare time, Haas plays classical viola. "It takes my mind off simulation and requires a different kind of collaboration."

—Laura DiDio

“Bayesian analysis is the engine that allows you to combine different information and different uncertainties,” Berger said.

It even allows statisticians to combine conflicting data. In the case of AF 447, the prior distribution heavily favored the region close to the airplane’s last-known position, but the one underwater search of that region—the passive acoustic search for underwater locator beacons—had not found the airplane. The Metron team incorporated the failure of the search into its Bayesian model, but also incorporated the possibility that the locator beacons had been destroyed in the crash. The search based on Metron’s posterior probability map did, in fact, turn up the airplane in the region that the passive acoustic search had covered, and a later test of one of the beacons showed it was broken.

“It appears that the likely failure of the beacons to actuate resulted in a long and difficult search,” the Metron team wrote in *Statistical Science*.

To construct the prior and the likelihood functions, the team had to make many subjective decisions, using the best data available to them—about, for example, the probability both beacons would fail, the probability the side-scan sonar would have missed the wreck if it was searching in the right area, and so forth. “A substantial amount of art”

goes into a Bayesian analysis, said Metron’s Stone, who was also involved in Bayesian analyses that recovered the lost U.S. nuclear submarine *Scorpion* and the wreck of the *SS Central America*, a steamship that sank off the Atlantic coast in 1857. “You have to be thoughtful about how you construct the priors.”

That does not mean, however, that only highly experienced Bayesian statisticians can use these tools, Stone said. In 2007, Metron created a Bayesian system that is the basis for the U.S. Coast Guard’s Search and Rescue Optimal Planning System, now used routinely in search and rescue missions, including the successful search for fisherman John Aldridge, whose recovery after 12 hours floating in the Atlantic Ocean on July 24, 2013 was chronicled at length in *The New York Times Magazine* in January 2014. The system is straightforward enough to be used by trained Coast Guard commanders, Stone said. “You might think that only specialists like me who have been through this a zillion times can do it, but it’s not true,” he said. “Bayesian search methods really work, and they can be used by people who aren’t Ph.D. statisticians.”

Bayesian analysis is a crucial tool when decisions have to be made under uncertainty, “particularly when you don’t have the luxury of saying,

‘I need more information before I can decide,’” Stone said. “With both Bayesian and classical statistics, if you get enough data, you will make the same decision. But when you can’t afford to wait, you should use the prior, and you will make better decisions if you do.” 

Further Reading

Efron, B.
Bayes’ Theorem in the 21st Century, *Science*, Vol 340, 1177-1178, 7 June 2013.
<http://bit.ly/1zhCkrk>

Kratzke, T., Stone, L., Frost, J.
Search and Rescue Optimal Planning System, 2010 13th Conference on Information Fusion, 1-8, 26-29 July 2010.
<http://bit.ly/1v0pake>

Stone, L.
Search for the *SS Central America*: Mathematical Treasure Hunting, *Interfaces* 22:1, 32-54, January-February 1992.
<http://bit.ly/1xqc8u8>

Stone, L., Keller, C., Kratzke, T., Strumpfer, J.
Search for the Wreckage of Air France Flight AF 447, *Statistical Science*, Vol. 29 No.1, 69-80, February 2014.
<http://bit.ly/1xuRzh9>

Tough, P.
A Speck in the Sea, *The New York Times Magazine*, Jan. 2, 2014.
<http://nyti.ms/1wPTZH7>

Erica Klarreich is a mathematics and science journalist based in Berkeley, California.

© 2015 ACM 0001-0782/15/01 \$15.00

Milestones

Computer Science Awards, Appointments

EATCS NAMES 2014 FELLOWS

The European Association for Theoretical Computer Science (EATCS) has recognized 10 of its members for outstanding contributions to theoretical science with its first EATCS fellowships.

The new EATCS Fellows are:

- Susanne Albers, Technische Universität München, Germany, for “contributions to the design and analysis of algorithms, especially online algorithms, approximation algorithms, algorithmic game theory and algorithm engineering.”
- Giorgio Ausiello, Università di Roma “La Sapienza,” Italy, for “his scientific work

in the field of algorithms and computational complexity and for his service to the scientific community.”

► The late Wilfried Brauer of Technische Universität München, for “outstanding contributions to the foundation and organization of the European TCS community.”

► Herbert Edelsbrunner, Institute of Science and Technology, Austria, and Duke University, U.S., for “his tremendous impact on the field of computational geometry.”

► Mike Fellows, Charles Darwin University, Australia, for “his role in founding the field of parameterized complexity theory, which has become a major sub-

field of research in theoretical computer science, and for being a leader in computer science education.”

► Yuri Gurevich of Microsoft Research, U.S., for “his development of abstract state machines and for outstanding contributions to algebra, logic, game theory, complexity theory and software engineering.”

► Monika Henzinger of the University of Vienna, Austria, for “being one of the pioneers of Web algorithms, algorithms that deal with problems of the World Wide Web.”

► Jean-Eric Pin, LIAFA, CNRS, and University Paris Diderot, France, for “outstanding contributions to the alge-

braic theory of automata and languages in connection with logic, topology, and combinatorics and service to the European TCS community.”

► Paul Spirakis, University of Liverpool, U.K., and University of Patras, Greece, for “seminal papers on Random Graphs and Population Protocols, Algorithmic Game Theory, as well as Robust Parallel Distributed Computing.”

► Wolfgang Thomas, RWTH Aachen University, Germany, for “foundational contributions to the development of automata theory as a framework for modelling, analyzing, verifying and synthesizing information processing systems.”

Smart Transportation Networks Drive Gains

Transportation engineers and city planners are looking to information technology to redefine traffic management in urban areas. With the opportunity, however, comes risks.

OVER THE LAST few decades, urban commutes have emerged as a genuine nightmare. In the U.S., motorists in the Washington, D.C. area waste an average of 67 hours a year stuck in traffic, while drivers in Los Angeles and the San Francisco Bay area squander 61 hours a year that way, according to a Bloomberg and Texas A&M Transportation Institute report. Meanwhile, in cities such as Mumbai, Beijing, and Sao Paulo, commutes can stretch to several hours per day and traffic backups can extend more than 100 miles (160 kilometers).

The result? Wasted time and fuel, higher levels of pollution, frayed nerves, safety concerns, and inefficiencies that rock the foundation of modern economies. A study by Texas A&M University found that in the U.S. alone, \$27 billion worth of time and fuel was wasted in 2011 as a consequence of all traffic-related delays, resulting in economic loss exceeding \$121 billion. In fact, despite telecommuting, home offices, and flexible work hours, the problem is growing worse. “We are facing extremely serious obstacles and a growing list of problems, says Dominique Bonte, a vice president and practice director at market research firm ABI Research.

Traffic engineers traditionally have focused on installing timed traffic lights and road sensors to detect vehicle flows, and developing algorithms to manage automobiles and infrastructure more effectively. While these systems help alleviate some congestion, they remain relatively unsophisticated. As a result, transportation engineers and computer scientists are now designing a smarter network of roads, traffic lights, signs, and vehicles. As Sei Kato, a managing consultant at IBM Global Business Services and an advisor for the City of Kyo-



to in Japan, puts it: “There is a need to maximize existing infrastructure.”

Make no mistake, smart traffic networks promise to redefine driving and urban environments by introducing smart systems that can react in real time to changing conditions. Yet, the technology is not without risks. Security researchers say these networks are vulnerable to hacks and attacks

that could wreak havoc and endanger public safety. “Successful hacks could seriously disturb traffic and cause widespread problems,” notes André Weimerskirch, a research scientist at the University of Michigan Transportation Research Institute.

Mixed Signals

The march toward traffic automation is nothing new. In 1914, the City of Cleveland, OH, installed the world’s first electric traffic signal, which ushered in a level of automation that ultimately transformed transportation. Over the ensuing years, green-yellow-red signals became ubiquitous, providing a reliable way to manage traffic flow and pedestrians. Engineers typically timed these stoplights to match traffic volumes. In recent years, they also have turned to sensors embedded in pavement and, in some cases, video cameras and other technologies, to determine when a light should switch to green and how long it should stay green.

Transportation engineers and computer scientists are designing a smarter network of roads, traffic lights, signs, and vehicles.

Along the way, cities have introduced increasingly sophisticated traffic networks. For example, the City of Los Angeles switched on its Automated Traffic Surveillance and Control System (ATSAC) system in preparation for the 1984 Summer Olympics. ATSAC has undergone significant upgrades since then and it now provides remote control capabilities for all of the city's 4,500-plus traffic lights. The system—which relies on sensors and closed-circuit cameras—reduces commute times by about 12%, according to the city. In addition, it is connected to rapid transit and bus networks so, for example, if a bus is running behind schedule, an engineer can alter the traffic light schedule so it can catch up.

Singapore has also forged ahead with traffic monitoring and control systems. It has already tested a system that relies on induction loops in roadways to monitor traffic. Later this year, working with IBM, the city will unveil a system that plugs in data from induction loops, video cameras, and taxi GPS systems to provide more accurate data about traffic patterns. The technology is also being used in Lyon and Montpellier, France, where it creates thousands of data points. The traffic control algorithm relies on a number of key factors, including road category, density of traffic on the road, speed limits along with traffic data and incident data, says Laura Wynter, director of the IBM Research Collaboratory in Singapore.

In the U.S., about 79% of vehicle detection systems currently use video or induction loops—also known as in-ground loops—to track the movement of vehicles on roads. Some communities also rely on microwave, radar, and ultrasonic sensors. Typically, a mix of technologies provides deeper insights into movements and behavior. According to a 2003 survey by the Institute of Transportation Engineers, 62% of traffic intersections across the U.S. operate in an interconnected fashion. However, these stoplights typically function in specific traffic corridors or as part of a network, and they lack the full spectrum of data to revolutionize traffic management.

"The problem with many of these systems is that they provide some benefits, and they help manage traffic better, but they do not tap into the efficiencies that are possible with a

fully networked infrastructure and vehicles," explains Bonte, who is responsible for ABI Research's global telematics, location, and navigation market coverage, as well as for the firm's coverage of smart cities. "It's one thing to monitor traffic and make adjustments; it's another to change the underlying dynamics of the overall system. There is a need for more highly networked and advanced technology and software that can work across an entire city and interface with other systems."

This notion appeals to Carolina Osorio, an assistant professor of civil and environmental engineering at the Massachusetts Institute of Technology. She and colleagues are developing more advanced technology to address traffic management on a broader scale. One of the problems with current traffic systems is that when engineers adjust one set of stoplights, it has a ripple effect on other intersections—along with driver behavior, she notes. For example, if wait times increase along a particular route, drivers will likely seek "alternative routes that feature fewer red lights." Too often, engineers wind up shifting problems from one set of motorists to another.

Osorio is currently testing a monitoring system and algorithms in Lausanne, Switzerland and New York City. She approaches the problem through the use of traffic simulators that analyze the behavior of drivers in response to changing conditions. In the Lausanne simulation, the approach has so far netted a 22% reduction in travel time compared to commercial traffic-light timing software. It encompasses 12,000

"There is a need for a more highly networked and advanced technology and software that can work across an entire city and interface with other systems."

individual drivers, approaches the task as a whole—taking into account the motion and movement of the entire traffic system, rather than focusing on a collection of individual motorists displaying distinct and complex behaviors. The initial simulation used 150 runs taking place during a typical 5 P.M. to 6 P.M. weekday period. City engineers use the simulation data to make changes to actual lights and other infrastructure.

While some cities already use high-resolution simulators, known as microscopic simulators, these systems often become too computationally intensive. Osorio's "homogenous" method approaches the accuracy of high-resolution models with the computational efficiency of a low-resolution model. Instead of merely reducing commute times, the algorithm provides data to improve fuel consumption and suggest where to place vehicle-sharing hubs and parking. It incorporates inputs such as a given time period, green signal phase durations, behavioral factors, and more.

Practical results occur at the intersection of connected infrastructure, vehicles, and algorithms, Osorio says. The traffic management systems she and others are developing attempt to embed a high-resolution description of both traveler behavior and network systems, including "prevailing traffic control strategies in the network," she notes. Key factors include: travel behavior, traffic flow theory, simulation, nonlinear optimization, and traffic control methodologies—including the response of motorists. "These strategies can lead to improvements in [traffic management] at a large scale, including the scale of a full city."

Of course, the problem is that drivers in Moscow are different than those in Auckland, New Zealand, or San Francisco, so there is no "one size fits all" algorithm. What is more, as conditions in a city change—say a new high-rise apartment building opens—traffic flow patterns may change with them. As a result, researchers increasingly examine two primary components: capturing data that identifies behavior and trends, while also accounting for deviations from trends.

Traffic: The Next Generation

Over the next few years, as the Internet of Things and connected machines

take hold, traffic management is likely to speed forward. Bonte says no infrastructure will likely be left untouched; using sensors, RFID, and other connected technologies, it is possible to transform everything from traffic lights and road signs to street and vehicle design. “Virtually everything can become part of a connected infrastructure,” he explains. This, in turn, will introduce other changes, including whether signs and other infrastructure (often referred to as “road furniture”) are actually necessary. Instead, a traffic network may rely on virtual traffic lights and cues delivered to drivers and pedestrians via smartphones, he says.

In addition, vehicle-to-vehicle (V2V) communications would allow autonomous or semiautonomous vehicles to connect to each other in real time, and slide the dial from a centrally managed system to a more distributed computing and communications model. With traffic data connected to stoplights and vehicles communicating with each other, “You quickly move past the incremental gains afforded by smarter traffic signals and unlock huge gains,” Bonte says. Within this scenario, cars can “platoon” a half a meter or less apart, thus stretching the capacity of existing roads and infrastructure. These networks will adjust traffic lights, subways, light rail systems, and bus routes across an entire city in real time in order to optimize conditions. Moreover, “You create a much safer environment. The number of collisions goes down to near zero,” Bonte explains.

The framework for this future is already taking shape. In Pittsburgh, PA, traffic engineers working with Carnegie Mellon University installed 49 smart signals in 2014. These stoplights use radar rather than street sensors to monitor the flow of traffic, and they communicate with each other in real time. Officials estimate the system will trim wait times in traffic by 42%, and total travel times by 24%. In Utah, the state’s Department of Transportation has installed a computerized traffic network that uses radar and video to manage traffic lights. The result? Only 28% of vehicles now hit a red light at any given intersection.

Meanwhile, technology companies such as IBM and Cisco Systems are developing software to run smart cities.

Practical results occur at the intersection of connected infrastructure, vehicles, and algorithms.

At Kyoto University and IBM’s Tokyo Research Laboratory, for instance, developers have introduced software that allows city officials to run simulations and adjust traffic systems in real time to respond to different variables, such as weather, sporting and entertainment events, and pedestrian flows. One simulation conducted by researchers at Carnegie Mellon University resulted in a 60% improvement in traffic flow in Porto, Portugal. Not surprisingly, smart vehicles connected to these networks could fuel further gains.

For now, Kato says one of the biggest technical challenges for traffic engineers and computer scientists revolves around analyzing and understanding “macroscopic behavior.” Current simulations can do a reasonably good job of understanding behavior involved with lane changes, merging traffic, and congestion, “but many problems remain when it comes to understanding route modeling and demand modeling. More sophisticated models and algorithms are now under development based on cognitive and big data approaches,” he explains.

There also is a need to address potential security and hacking risks associated with connected vehicles and traffic networks. Researchers have already demonstrated the ability to commandeer locks, brakes, and the steering wheel for existing automobiles. Moreover, in 2006, during a labor dispute, two traffic engineers in Los Angeles broke into the city’s transportation computer and clogged traffic at key intersections in the city for several days. Although experts say they are not aware of widespread system hacks to date, it is clear that connected and automated systems create new and bigger risks.

In fact, Cesar Cerrudo, an Argentinian security researcher, has identified vulnerabilities in traffic control systems in 40 U.S. cities and in nine other countries. He reported these systems lack key encryption and authentication, and it is possible to sniff out and intercept clear text and data running over 802.15.4 wireless networks. Breaking into these systems required only a few basic tools that cost less than \$100, he noted in a blog post in April 2014. “These [incidents] could cause real issues, even deadly ones, by causing accidents or blocking ambulances, firefighters, or police cars going to an emergency call,” he explained.

Nevertheless, smarter transportation networks are taking shape and the development of smart vehicles is speeding forward. Bonte says these initiatives will pick up steam in the years ahead and, in many cases, will fold into broader smart city initiatives that seek to reduce waste, pollution, and energy demands.

Says Kato: “Automobiles brought us rapid transportation and improvements in transportation efficiency. On the other hand, they introduced problems such as chronic traffic jams, noise, air pollution, and safety concerns. Smarter traffic networks help cities reduce the impact of these problems.” ■

Further Reading

Ghena, B., Beyer, W., Hillaker, A., Pevarnek, J., and Halderman, J.A.

Green Lights Forever: Analyzing the Security of Traffic Infrastructure, Electrical Engineering and Computer Science Department, University of Michigan, 8th USENIX Workshop on Offensive Technologies, August 2014

Cerrudo, C.

Hacking U.S. (and U.K., Australia, France, etc.) Traffic Control Systems, April 2014 blog.ioactive.com/2014/04/hacking-us-and-uk-australia-france-etc.html.

Zimmermann, T., Wirtz, H., Punal, O., & Wehrle, K.

Analyzing Metropolitan-Area Networking within Public Transportation Systems for Smart City Applications. In 6th International Conference on New Technologies, Mobility and Security 2014, (p. 1-5), IEEE, March 2014

Samuel Greengard is an author and journalist based in West Linn, OR.

© 2015 ACM 0001-0782/15/01 \$15.00

Data Brokers Are Watching You

You would be surprised by how much they know about you, and what they are doing with your information.

AIDED BY ADVANCES in data science and the increased digitization of analog information, an industry little known to the public is quietly compiling comprehensive dossiers on millions of Americans. The companies, called data brokers, say they operate within the law, but the unprecedented breadth and depth of the data files, the difficulty in correcting erroneous data, and the potential for abuse of personal information are raising alarms from privacy advocates, consumer groups, and government officials.

Data brokers compile information about individuals from a wide variety of online and offline sources, including email, personal websites, social media posts, U.S. Census records, retailers' systems, Department of Motor Vehicles records, and real property records. The data is often collected without the consent or knowledge of the individuals involved, integrated and synthesized using advanced analytic tools, then sold to other data brokers and businesses for a variety of purposes.

"You may not know them, but data brokers know you," said Edith Ramirez, chairwoman of the U.S. Federal Trade Commission (FTC) in a statement last May. "They know where you live, what you buy, your income, your ethnicity, how old your kids are, your health conditions, and your interests and hobbies."

Ramirez' statement accompanied the release of a report, "Data Brokers: A Call for Transparency and Accountability" which, in turn, was based on information supplied by nine such brokers: Axiom, CoreLogic, DataLogix, eBureau, ID Analytics, Intelius, PeekYou, RapLeaf, and Recorded Future. One of them reported having a

Digitization of analog data, along with advances in algorithms behind data analytics, has enabled a dramatic leap in the ability of data brokers to track individuals.

database of 700 billion data elements culled from 1.4 billion consumer transactions; a second had information from \$1 trillion worth of consumer transactions, and a third said it was adding three billion new records to its database each month.

The FTC has called for legislation that would give consumers greater access to data brokerage practices and more control over their own information. The agency suggested Congress should consider requiring brokers to create a centralized portal where consumers could look at all their data and opt out of having it used.

While personal data is sold for a variety of purposes, many of the brokers' customers use the information for targeted marketing. The FTC said one broker segments consumers into handy buckets with labels such as "Urban Scramble" (heavily populated with low-income Latinos and African Americans), "Rural Everlasting" (single men and women over the age of 66 with little education and small net worths), and "Married Sophisticates" (upper-middle-class young adults with no chil-

dren). More narrowly defined groups included "Expectant Parent," "Diabetes Interest," and "Cholesterol Focus."

It is easy to imagine a financial company aiming sub-prime loans at "Urban Scramblers" while pitching platinum credit cards to "Married Sophisticates," privacy advocates warn. Even worse, how can anyone be sure that membership in one of the medical cohorts doesn't affect one's insurance rates? "The companies see what website you visit, then they add offline information to that," said Ed Mierzwinski, program director at U.S. PIRG, the federation of state Public Interest Research Groups. "Then the website you go to next might look different than it otherwise would."

The process described by Mierzwinski, known as "onboarding," has drawn special scrutiny and concern from the FTC, Congress, and the White House. In onboarding, a data broker will add offline information—data from manual sources or from other systems such as loyalty cards, warranty registrations, and stores' point-of-sale terminals—into the cookies of computers used by individuals to access websites monitored by the broker. Once in place, the cookies can track the Web activity of the person from place to place, serving up targeted ads at each site. "Data brokers are helping to blur the line between online and offline behavior," said the FTC's Ramirez. "[They] use your offline purchases and information to find and target you online."

With so much information coming from so many sources, it is inevitable that errors arise in digital dossiers; moreover, the errors can be difficult or impossible to correct. In a study published in 2013, the FTC reported one in five consumers had an error in one or more of their credit reports. These errors can lead to an unfairly poor credit

rating that can prevent someone from getting a much-needed loan. The FTC also found more than 10% of consumers saw a change in credit score after getting errors corrected.

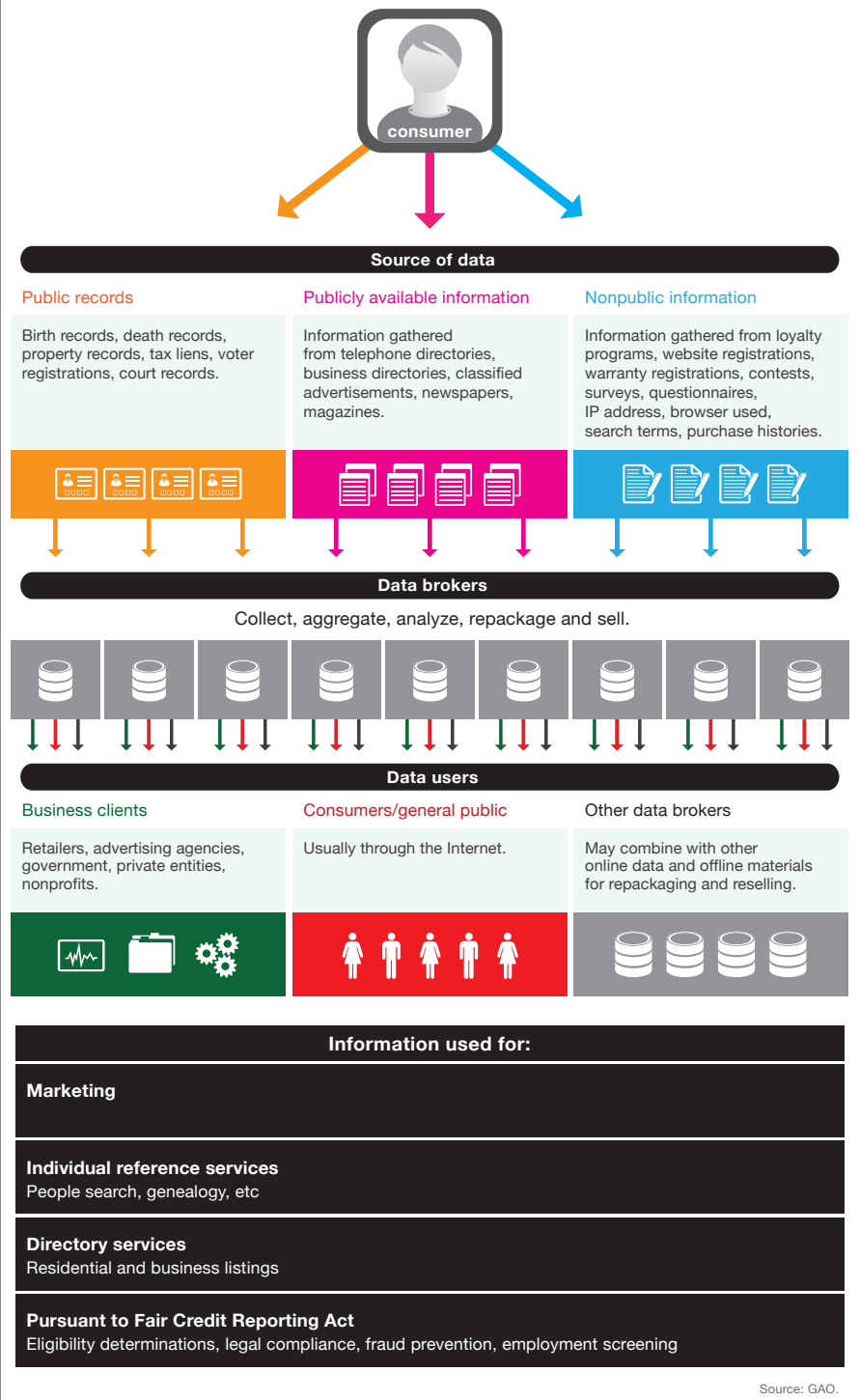
Many of the gatherers of online information offer ways to correct the personal data they hold, but the methods for doing so are often not well publicized, or are difficult to follow. In any case, consumers often do not know who is collecting or using their data, and there is no central clearinghouse for verifying all of one's online data. Even the most diligent, tech-savvy people cannot ensure the integrity of all their data, anymore than they can control its use.

The rapid increase in the number of available sources of data is especially troubling when it involves the capture of analog information, said Susan Graham, a computer science professor at the University of California, Berkeley (UC Berkeley). That is because the devices that capture this information—surveillance cameras and various kinds of sensors—for valid technical reasons sweep up more data than is needed for the immediate purpose. For example, she said, dramatic improvements in high-resolution digital cameras and facial recognition software will soon make it possible to identify people gathering at news events or in cars passing traffic law-enforcement cameras. With such abilities come obvious possibilities for abuse, she said.

The digitization of vast amounts of previously analog data, plus advancements in the algorithms behind data analytics, have enabled a dramatic leap in the ability of data brokers to track and understand the day-to-day activities of individuals, Graham said. “You can put together very disparate pieces of information and create a profile of a person. It’s not necessarily correct, but you can do it very quickly, and you can do it with information that the individual may not have provided. That’s a fairly profound shift.”

Much of this shift has been hidden from the public, according to a year-long investigation of data brokers by the U.S. Senate Committee on Commerce, Science, and Transportation. “The refusal by several major data broker companies to provide the committee complete responses regarding data

Flow of consumer data through data brokers to third-party users



sources and customers only reinforces the aura of secrecy surrounding the industry,” the committee said in its final report in late 2013.

Industry Response

The Direct Marketing Association, which calls itself “the world’s largest trade association for data-driven

marketers,” said its members operate entirely within the law. Asked if the actions of data brokers nevertheless raise legitimate privacy concerns, DMA senior vice president Stephanie Miller responded, “No. Marketing data used responsibly for marketing purposes is a consumer benefit, providing much of the value in our data-driven

lifestyles.” She cited an independent study that showed the “data-driven marketing economy” contributed \$156 billion to the U.S. economy and supported 675,000 jobs in 2012. Miller also said a number of data brokers, as well as vendors such as Google and Yahoo, already offer consumers ways to opt out of targeted marketing.

Indeed, the FTC cited a number of benefits that flow from data-driven marketing. “Data broker products help to prevent fraud, improve product offerings, and deliver tailored advertisements to consumers,” the FTC said in its report. The brokers also foster competition by enabling small businesses to pitch innovative products to consumers they could not otherwise reach. The agency also acknowledged the existence of laws that protect the data of consumers in certain industries, such as finance and health care.

For example, the Fair Credit Reporting Act (FCRA) of 1970 regulates and restricts the use of consumer data when such data may be used for “eligibility” determinations in credit, employment, insurance and housing. Last April, the FTC settled with two data brokers for selling data in violation of the FCRA. The FTC found the companies had sold data to employers and landlords without ensuring the accuracy of the data, or that the buyers had legitimate reasons for wanting the information, as required by law.

David LeDuc, a senior director at the Software & Information Industry Association, said he is not sympathetic to those calling for legislative reforms because they find what data brokers do “creepy.” “We oppose the imposition of unnecessary barriers to the collection and use of data,” he said. “The focus should be on real harm, not crystal ball gazing about what makes certain people uncomfortable.”

Solutions

The use of personal information by data brokers and others has become so pervasive that limiting its collection by existing means has become unworkable. Asking consumers to “opt out” of data collection at myriad companies they have never heard of is unrealistic, and the existing online “notice and consent” forms—in which users “agree” to the collection and use of

personal data—are ineffective because they are mostly ignored by consumers.

Instead, said UC Berkeley’s Graham, government policy as well as technology should focus more on the use of personal data, and less on its collection. Her views and those of 10 other information technologists, businesspeople, and policy experts appear in a report submitted to President Obama last May, “Big Data and Privacy: A Technological Perspective.” In the report, a working group of the President’s Council of Advisors on Science and Technology (PCAST) argued policies that focus on data “collection, storage, applications, and analysis” are not scalable, as it becomes increasingly difficult to ascertain what personal information may be latent in a particular dataset or its fusion with other data. Also, PCAST said, policies limiting collection and retention will become increasingly unenforceable by other than draconian means.

To control use, individuals might choose a privacy preference profile offered by third parties, PCAST said. For example, “Jane” might choose one offered by the American Civil Liberties Union that gives special weight to privacy, while “John” might prefer one from *Consumer Reports* that emphasizes economic value to the consumer. Market forces or government regulation would compel the users of personal data to conform to the profiles.

Technology is already moving to enable that kind of control. For example, commercial privacy systems have been developed by firms such as Booz Allen Hamilton and IBM and are in use by a few government agencies, financial services firms, and pharmaceutical firms. These systems are based on the

Expecting consumers to understand and specify their privacy preferences may be unrealistic at present.

Trusted Data Format (TDF) for file-level tagging and security.

Expecting consumers to understand and specify their privacy preferences may be unrealistic at present, said Mark Gorenberg, managing director of Zetta Venture Partners and a member of the PCAST working group. “You need to create a market for this,” he said. “You’d see products and systems and cloud-based services with usage-based components in them.” If major vendors such as Google and Amazon began offering file tagging and tracking services, perhaps based on TDF, then the public might come to use them, he said. If not, Congress would have to decide whether to mandate some kind of usage-based controls, he said.

“It would be a mistake to stop collecting data which, when combined with other data, gives much better results to the public,” Gorenberg said. “We are at a golden age of being able to use data to get better results for companies and individuals.”

Graham sees the glass as half-full. “Things are becoming technically feasible today that would have been crazy 10 years ago—for the good guys and the bad guys.” 

Further Reading

U.S. Federal Trade Commission, *Data brokers: a call for transparency and accountability* May 2014 <http://1.usa.gov/1kXR5g0>

Fertik, Michael

The Rich See a Different Internet Than the Poor, *Scientific American*, Jan. 15, 2013 <http://bit.ly/1qtmpQt>

Executive Office of the President, *Big data and privacy: a technological perspective*, President’s Council of Advisors on Science and Technology, Big Data and Privacy Working Group, May 2014 <http://1.usa.gov/1ro5aq5>

Russell, Matthew A.

Mining the social web, 2nd edition, O’Reilly Media, October 2013 <http://oreil.ly/1ljIwX>

U.S. Senate Committee on Commerce, Science, and Transportation, *A review of the data broker industry: collection, use, and sale of consumer data for marketing purposes*, Staff report for Chairman Jay Rockefeller, Dec. 18, 2013 <http://1.usa.gov/1vlVESn>

Gary Anthes is a technology writer and editor based in Arlington, VA.

©2015 ACM 0001-0782/15/01 \$15.00

Google Boosts ACM's Turing Award Prize To \$1 Million

The increase reflects the escalating impact of computing on daily life, through the innovations and technologies it enables.

ACM AND SEARCH ENGINE giant Google recently raised the funding level for the ACM A.M. Turing Award to \$1 million.

The organizations say this quadrupling of the cash award for Turing Award recipients (it previously was \$250,000) is intended to raise the visibility of the Turing Award as the premier avenue of recognition for computer scientists and engineers who have made contributions of major technical and lasting importance to the computing field. The substantially larger cash award also reflects the continuing impact of computing on many aspects of daily life, through the innovations and technologies it enables.

The new amount will be awarded to Turing Award laureates beginning with the 2014 ACM A.M. Turing Award, which will be announced later this year, and which will be presented at the subsequent ACM Awards Banquet.

In a press conference to announce the funding increase, ACM president Alexander L. Wolf observed that thanks to “the generous support of Google, we can celebrate the mainstream role of computing in transforming the world and the way in which we connect, shop, and socialize. We can also commemorate the pioneering fundamental contributions of our ACM Turing Award recipients in advancing computing as a science and profession.”

In attendance at the press conference were a number of Turing laureates, including:

Silvio Micali (2012), Barbara Liskov (2008), Edmund M. Clarke and Joseph Sifakis (2007), Butler W. Lampson (1992), Robert Tarjan (1986), and Michael O. Rabin (1976).



Turing laureates joining the press conference announcing Google's increased funding support for the ACM A.M. Turing Award were treated to a short video on the life and accomplishments of pioneering computer scientist Alan Turing.

Another Turing laureate (and former ACM president), Vint Cerf (2004), appeared on video from Australia as “virtual Vint” to offer his congratulations and support.

Google vice president of engineering Stuart Feldman, a Fellow (and former president) of ACM, said Google “is proud to provide increased support for ACM’s Turing Award and its unique role in recognizing fundamental contributions in computing and computing science. Our funding reflects our continuing commitment to foster technology breakthroughs that drive progress in the digital age.”

Google depends on computing and on advances in computing, Feldman said, and “the Turing Award has represented the finest of thought and the broadest of impact.” He added, “We appreciate this opportunity to help ACM raise awareness of the innova-

tors who have shaped the lives of people around the world with their imagination and inspiration.”

Since 1966, ACM has bestowed the A.M. Turing Award to honor the computer scientists and engineers who created the systems and underlying theoretical foundations that have propelled the information technology industry. The award is named for British mathematician Alan Turing, who envisioned the power of the thinking machine, leading the way to innovations that have changed the world, including programmable computers, mobile devices, cryptology, artificial intelligence, robotics, genomics, and the philosophy of science. 

Lawrence M. Fisher is Senior Editor/News for ACM magazines.

© 2015 ACM 0001-0782/15/01 \$15.00



DOI:10.1145/2688487

Michael A. Cusumano

Technology Strategy and Management

How Traditional Firms Must Compete in the Sharing Economy

Considering the evolving relationship between established companies and their sharing-economy counterparts.

TRADITIONAL COMPANIES NOW have another set of competitors to worry about—Internet startups in the “sharing economy.” These new companies are actually Web platforms that bring together individuals who have underutilized assets with people who would like to rent those assets short-term. The assets targeted by these startups range from spare time for everyday tasks (TaskRabbit, Fiverr) and spare time and automobiles to drive people around (Uber, Lyft) to extra rooms (Airbnb, Flipkey, Roomba) and occasionally used tools and other household items (Streetbank, Snap-Goods/Simplist).^{9,12}

Sharing-economy startups are in many ways a logical outgrowth of social media platforms such as Facebook, Pinterist, and Trip Advisor, which bring together people with common interests to share ideas, information, or personal observations. They threaten established companies to the extent

that peer-to-peer networks can grow exponentially through the power of platform dynamics and network effects (see my previous *Communications* columns “The Evolution of Platform Thinking,” January 2010, and “Platform Wars Come to Social Media,” April 2011).

Sharing-economy startups also reflect the broader “servitization” trend. Here, instead of selling products outright, companies can expand their potential markets by renting access to products that people used to buy. Examples include everything from Salesforce.com selling software as a service instead of as a product to automakers like Daimler and BMW, following the lead of Zipcar (now owned by Avis Rental Cars) and offering transportation as a service instead of selling automobiles.^{2,4}

Not all has been smooth sailing in the sharing economy, however. Some startups have already run into legal and regulatory hurdles from city governments, courts, and traditional unions

or lobbies that want to restrict them or shut them down. Airbnb has faced problems in New York, San Francisco, and other cities because of regulations regarding private hosting and subletting. New York City officials are still grappling with the issue of who should pay hotel taxes. But the big question is really how should traditional companies compete with startups in the sharing economy?

Let’s take the hotel business as an example. Airbnb started in 2007 in San Francisco when the founders had an extra room to rent and decided to offer a low-cost “air mattress” and “bed and breakfast experience” to attendees at a local conference.⁶ They set up a website, targeted cities with conferences, and signed up other people with spare rooms. Progress was slow until 2009, when the company entered the Y-Combinator accelerator and then got venture funding. As of September 2014, Airbnb had expanded to 800,000 room listings in 190 countries and claimed that 17 million people had



used its services.^a The firm has also raised \$826 million in venture capital, with a most recent valuation of \$10 billion, about 40 times 2013 estimated revenues of \$250 million.^{8,13}

By comparison, the largest hotel chains had fewer rooms, much slower growth rates, and much lower valuations compared to their revenues. Marriott (which includes Ritz-Carlton, Bulgari, and Fairfield Inn) had 675,000 rooms in 74 countries, with 2013 revenues of \$12.7 billion (not including franchisee revenues) and a September 2014 market value of \$20 billion (1.6 times prior year revenues). Hilton (which includes the Waldorf-Astoria, Embassy Suites, and Hampton Inns) had 679,000 rooms in 91 countries, 2013 revenues of \$9.7 billion, and a market value of \$25 billion (about 2.5x). InterContinental Hotels Group (which includes Crown Plaza and Holiday Inn) in 2014 had 674,000 rooms in 100 countries, 2013 revenues of \$1.9 billion, and a market value of \$9 billion (about 5x).^b

^a See <http://www.airbnb.com/about/about-us>.

^b See <http://listdose.com/top-10-biggest-hotel-chains-in-the-world/> for a list of hotel chains. More specific data on rooms and revenues can be found in the company annual reports.

For hotel chains to add rooms, they have three options. They can seek relationships with investors who want to construct new hotels and turn them over to their management, get existing hotels to switch their management companies (such as from Hilton to Marriott or vice versa), or build new hotels themselves. Most hotel chains do not own the vast majority of properties they manage, so they do not need a lot of their own capital to expand. Nonetheless, building new hotels is financially risky for the investors, and getting existing property owners to switch allegiances and brand names can be difficult and risky as well.

By contrast, to grow its portfolio, which ranges from cheap spare bedrooms to luxury homes for vacation rentals, Airbnb relies mostly on the power of network effects. It needed some inventory to get started and lure the first customers. But, as word-of-mouth has increased, and as the company has advertised more widely (for example, you can now find Airbnb ads on Expedia.com), network effects took over. More people have offered their rooms and more people have used the service, which has led to more available rooms, more users, more rooms,

and so forth. Will Airbnb someday be as dominant in the lodging business as Amazon now is in books and electronics retailing? Probably not, but the potential for growth is limited only by the power of the Internet as a network—and that is pretty big—as well as by how competitors respond.

So how do you compete with Airbnb if you are a conventional hotel chain? The first thing you do is go to your local government officials and make sure that Airbnb is not violating the rules for subletting, hotel taxes, insurance, and other regulations. The playing field should be level. In the long run, though, traditional hotels can compete by providing a type and level of service that Airbnb cannot match at any price. For example, hotel rooms, reservations, and other services can be much more standardized and reliable as well as localized in taste and design. Hotel chains can cater to individuals or businesses of different sizes and types. They can host major events and small gatherings. They can connect customers to a broad variety of vendors for tourism, transportation, and other personal or business services, tailored to the locality and the occasion. They can leverage long histories of relationships

with their customers and offer extensive loyalty programs and discounts.

Consider the taxi business as another example. Here, Uber poses a bigger threat to existing companies than Airbnb does to hotel chains, except perhaps for their lower-end properties. Taxis are highly regulated and restricted as well as not universally available, such as in small or rural communities, whereas individually owned cars and potential drivers are everywhere. Uber got its start in 2009, also in San Francisco, as a private luxury car service. It then began raising venture capital and launched a mobile smartphone app in 2010 that enables potential customers to call for a ride, get a price quote, and then accept or reject it. The company expanded to UberX in 2012, enabling customers to arrange for rides in smaller, less expensive cars. Uber also now helps individual drivers get loans to buy new cars.¹⁰ In June 2014, Uber raised an impressive \$1.2 billion at a valuation that, with the new money, totals \$18.2 billion.⁷ As of September 2014, Uber had accumulated over \$1.5 billion in venture capital and operates in more than 70 cities in 45 countries.^c

So how do you compete with Uber if you are a local taxi company? Again, the first thing you do is see if Uber is violating regulations for transportation services—insurance, training of drivers, and licenses. In some cases, such as in Germany, you may file a claim in court and win a nationwide ban.¹¹ But most people like the concept of the sharing economy because it means greater access to goods and services at lower prices. In Boston, for example, the number of taxi cab licenses (“medallions”) was established in 1934 at 1,525 and has increased only by 300 in subsequent decades.¹ The system has created an artificial shortage of cabs, even though drivers generally work in shifts of three per medallion. It took pressure from average citizens and intervention by state government officials and the courts to overcome temporary legal actions that restricted Uber’s operations.^{3,5}

What options other than regulation are there? Like hotel chains, taxi companies can offer a standardized level of service and price that Uber will not always be able to match. True, the

The sharing economy has ushered in a new age where underutilized assets become peer-to-peer services for hire.

challenge for differentiation is tougher in the taxi market, since a car ride is a car ride, and does not require the same level of discretion as choosing a hotel. Nonetheless, traditional enterprises should be able to provide more reliable, consistent, broader, and safer services than sharing-economy competitors.

There is also nothing to stop traditional companies from becoming more like their sharing-economy counterparts. Marriott, for example, is partnering with boutique hotels around the world. It offers access to these “Autograph Collection” properties through its reservation system and loyalty program, which customers can access through the Web on PCs or mobile devices.^d Maybe this part of its business should grow—a lot. In another case, MetroCab Boston has introduced a smartphone app that enables potential customers to order taxis and pay via credit card.^e Taxi companies can also work with local governments to ease regulations, expand the number of licenses, and hire more drivers.

Sharing-economy startups also have weaknesses. For example, there can be lags in confirming reservations on Airbnb that make it difficult to make definite travel plans. And Uber, like airlines and some hotels, relies on “dynamic pricing.” This means prices go up when demand is high, such as during rush hours or rainstorms. Uber drivers can also decline to provide service, such as when they do not like the requested destination or the rating of the customer (yes, both customers and drivers are rated on Uber). These are features

of the Uber service that taxi companies can exploit since they are obligated to offer standardized prices and provide service to anyone who calls.

In short, the sharing economy has ushered in a new age where underutilized assets become peer-to-peer services for hire, enabled by the Internet and smartphones. But there are still many uncertainties. Web startups are easy to launch but many will not survive once their funding runs out. Moreover, network effects lead to positive feedback loops, increasing returns to scale, and winner-take-all shakeouts that favor the bigger platforms—much like Amazon and Google have come to dominate Internet retailing and search. Airbnb and Uber are already big and probably here to stay. They are likely to become bigger, better, and more varied in the services they offer. Traditional companies in these markets are not likely to go out of business, but they cannot stand still. They must adapt and compete based on their own unique advantages—or they will become much diminished versions of what they used to be. **C**

References

1. Boston Globe Spotlight Team. For Boston cabbies, a losing battle against the numbers. *Boston Globe* (Apr. 1, 2013).
2. Cusumano, M.A. *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Uncertain World*. Oxford University Press, 2010), 68–113.
3. Farrell, M. Uber battle goes to heart of Cambridge. *Boston Globe* (June 21, 2014).
4. Gardiner, B. Big European players embrace the car-sharing trend. *The New York Times* (Nov. 19, 2013).
5. Herman, C. Court rules state can give OK to Uber. *Boston Globe* (June 26, 2013).
6. Kessler, A. Brian Chesky: The ‘sharing economy’ and its enemies. *The Wall Street Journal* (Jan. 17, 2014).
7. Kovach, S. Uber raises \$1.2 billion at \$17 billion valuation. *Business Insider* (June 6, 2014); <http://www.businessinsider.com>.
8. Luden, I. Airbnb has closed its \$500M round of funding at a \$10B valuation. *TechCrunch* (Apr. 18, 2014).
9. Malhotra, A. and Van Alstyne, M. The dark side of the sharing economy...and how to lighten it. *Commun. ACM* 57, 11 (Nov. 2014), 24–27.
10. Roose, K. Uber might be more valuable than Facebook someday. Here’s why. *New York Magazine* (Dec. 6, 2013).
11. Scott, M. and Eddy, M. German court bans Uber service nationwide. *The New York Times* (Sept. 3, 2014), B3.
12. Singer, N. In the sharing economy, workers find both freedom and uncertainty. *The New York Times* (Aug. 16, 2014).
13. Spector, M., MacMillan, D., and Rusli, E. TPG-led group closes \$450 million investment in Airbnb. *The Wall Street Journal* (Apr. 18, 2014).

Michael A. Cusumano (cusumano@mit.edu) is a professor at the MIT Sloan School of Management and School of Engineering and author of *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Uncertain World* (Oxford University Press, 2010).

The author thanks CEO Arne Sorenson and SVP Shafiq Khan for the opportunity to learn about Marriott.

Copyright held by author.

d See <http://www.autograph-hotels.marriott.com/>.

e See <http://www.boston-cab.com> and <http://www.metro-cab.com/>.

c See <https://www.uber.com/about>.

Law and Technology

A Right to Be Forgotten?

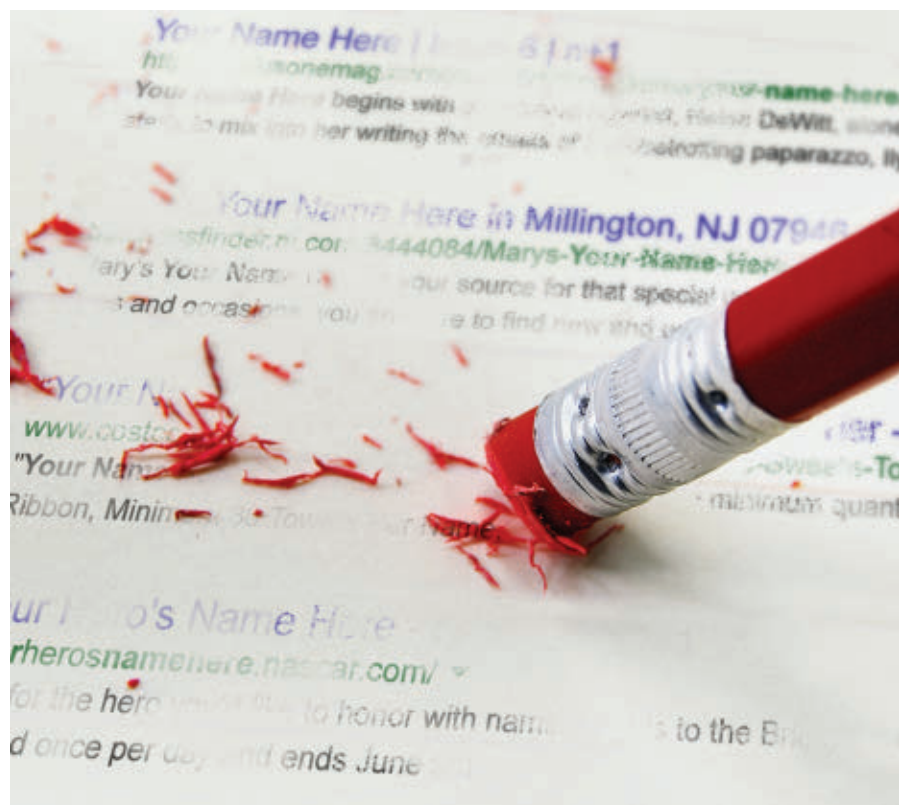
Searching for an answer to the question of how much responsibility search engine operators should bear for privacy-related issues connected to search engine results.

COURT DECISIONS in the field of data privacy rarely make a durable impression on the general public. And European court decisions in any field rarely cause a stir outside Europe. An exception on both counts is the May 2014 decision by the Court of Justice of the European Union (EU) ordering Google to suppress certain Internet search results for privacy-related reasons.¹ Popularly characterized as establishing a “right to be forgotten,” the decision is unlikely to fade quickly from public memory, either in Europe or many other regions. Since the decision was handed down, its merits have been debated worldwide.

Lines of Debate

Some of the debate has been unnecessarily fueled by misunderstanding of the decision, not least regarding the right it is said to uphold. Although the decision has been commonly flagged as establishing a right to be forgotten, this is a misnomer. The decision does not establish a right to be forgotten. Rather, it upholds a qualified right to be de-indexed, or, more specifically, a qualified right not to figure in a public index of search results. Nonetheless, when put into effect, the right makes certain personal information more difficult to find and thus serves, indirectly, an interest in being forgotten.

At a very general level, the lines of debate over the decision revolve around the balance to be drawn between privacy interests and free speech interests. More particularly,



the primary issue concerns the degree to which information about a person's past life ought to be made easily accessible online even when the information has diminished relevance for assessing the person's present life and would be difficult to find offline. Less salient but also significant is the issue as to what extent online mechanisms for data search and retrieval should emulate the standards of the (past) offline world. The importance of both issues is sharpened in an era when the spread and relative indelibility of on-

line data makes it extremely difficult to “let bygones be bygones.”

Both issues obviously call into question the role played by Internet search engines, particularly the legal responsibility the search engine operators ought to bear for the search results they generate. Google has usually portrayed itself in this respect as merely a medium for locating and presenting data. And, as such, it has tended to argue that it should escape legal liability for privacy-related harms arising from the information its search engine

helps to disseminate. In May last year, the EU court decided otherwise.

Super Mario vs. Googledom

The factual background for the court's decision is briefly as follows: a Spanish lawyer, Mario Costeja González, had to sell off some of his property in insolvency proceedings in the late 1990s. A Spanish newspaper reported on the proceedings at the time, and was indeed required to do so under Spanish law. González resolved his financial problems, but when the newspaper later established an online service, its reports of the property sale became accessible by googling González' name. In 2010, he asked the newspaper to remove the information from its online service but the newspaper refused—a stance subsequently vindicated by Spanish authorities. He also requested that Google remove search results relating to the information concerned. Google's refusal to do so led to the litigation that eventually ended in it sustaining a serious blow before the EU court.

The essence of the court's decision is that European data privacy law obliges Google to suppress search results revealing personal data that is,

in the court's words, "inadequate, irrelevant or no longer relevant, or excessive in relation to the purposes of the processing at issue carried out by the operator of the search engine." This is so even when the data remains lawfully published by the websites to which the search results point.

Google as Mere Medium?

A key question in the litigation was Google's status as a search engine operator under European data privacy law. In particular, was Google to be regarded as a "controller" of the personal data inhering in the links indexed by its search engine? The controller role kicks in when the entity concerned helps to determine the means and ends of data processing, and it enhances the entity's legal responsibilities for the data. Google argued it could not be a controller since it is, in effect, merely acting in a robotic capacity and exercises negligible control over the content of search results. The court disagreed. It thereby cast upon Google a raft of duties under European data privacy law. One such duty, the court held, is to effectuate justified requests to suppress search results in which certain types of personal data inhere.

Competing Interests

In holding Google to account, the court rightly paid little regard to the company's successful business model. Google's economic interests were judged as secondary to the data privacy rights of the persons who figure in search engine results.

As for free speech interests, the court acknowledged that these must be taken into due regard when assessing whether search results ought to be suppressed. And it stated that suppression requests may be trumped by "the preponderant interest of the general public in having ... access to the information in question." Yet, apart from observing the heightened risks to data privacy owing to the Internet's capacity to make search results ubiquitous, the court was remarkably silent about the broad implications of its decision for use of the Internet as a global communications network. For this, the court has justifiably attracted much criticism.

A key question in the litigation was Google's status as a search engine operator under European data privacy law.

One such implication is the decision's potential to exacerbate 'Balkanization' of the Internet by accentuating regional differences in what information can be easily accessed online. Indeed, following the decision, Google has been suppressing search results generated through its European domains but not through google.com or its other non-European domains.² It is questionable, though, whether the decision permits this practice.

The court also failed to address the practical consequences of the de-indexation right. Over the first five months after the decision was handed down, Google received approximately 143,000 de-indexation requests related to 491,000 links.² Yet, the court's decision ignores the problem of how to effectuate efficiently but fairly these requests and the thousands more that will follow. It gives scant guidance on the assessment criteria that ought to apply. And it overlooks the issue as to whether a search engine operator is suitably placed to engage in the delicate balancing exercises that such assessments frequently require. These omissions have rightly amplified irritation with the decision.

Nonetheless, Google and other major search engine operators are far from being 'babes in the woods' in this area. They already have experience in handling massive amounts of de-indexation requests related to alleged infringement of intellectual property rights. While the assessments required by those sorts of requests are qualitatively different from privacy-

**Have a question
about advertising
opportunities?**

CONTACT US

212 • 626 • 0686

acmm mediasales@acm.org



Association for
Computing Machinery

related assessments, the logistics are similar. And European data protection authorities have shown willingness to assist—for instance, in drawing up guidelines to ensure consistency in assessing de-indexation requests.

The Decision in Context

The Gonzáles case is not the first time the EU court has fired across the path of Internet-related technology deployment. Two years earlier, it shot down two proposals in Belgium to employ deep packet inspection aimed at preventing copyright infringement through peer-to-peer file sharing.^a The Gonzáles case, though, is the first time the court has fired directly at a commonly used Internet mechanism. It is also the first time it has fired, albeit more indirectly, at a basic element of the eminently successful “Internet economy.”

Yet, for those who have followed the development of the court’s jurisprudence on data privacy over the last decade, the decision in the Gonzáles case is not unexpected. The court has been increasingly prepared to subject data-processing operations that affect privacy to stringent proportionality assessments. This reflects the fact that, under EU law, protection of personal data is a fundamental right in itself.^b Thus, data privacy must be treated on an equal par with other important human rights, such as freedom of expression. Another factor, though one less recognized, is a long-running strand of scepticism within European data privacy law toward fully automated decisions that impinge on personal interests. We see this scepticism manifested, for example, in the 1995 EU Data Protection Directive, which gives a person the qualified right to object to certain types of fully automated decision-making processes that are based on profiles of their character.^c

a See its judgments of November 24, 2011 in Case C-70/10, *Scarlet Extended v Société belge des auteurs, compositeurs et éditeurs SCRL (SABAM)* and its judgment of February 16, 2012 in Case C-360/10, *SABAM v Netlog*.

b See Article 8 of the Charter of Fundamental Rights of the European Union [2010] OJ C83/389; Article 16 of the Treaty on the Functioning of the European Union [2010] OJ C83/47.

c See Article 15 of the Directive.

Such scepticism made it more difficult for Google to legitimize its search engine operations as value-neutral, robotic applications of algorithms.

The rhetorical battlelines over the decision’s pros and cons have largely followed the transatlantic divide. This far from surprising. The relationship between the U.S. and Europe has long been plagued by disagreement over regulatory policy on data privacy. Although sharing some common ground, U.S. and European approaches to data privacy regulation diverge significantly. The U.S. approach is generally less restrictive than the European. Transatlantic tensions in the field have simmered over the last 30 years. They are now close to a boil as the EU considers introducing a new General Data Protection Regulation to replace the 1995 Directive. The proposed Regulation, which is intended to be adopted later this year, contains a number of controversial rights, including an enhanced right of data erasure that has been trumpeted (somewhat misleadingly) by the European Commission as a ‘right to be forgotten’. Some U.S. government officials have warned of a new trade war if certain rights in the proposed Regulation, such as the putative right to be forgotten, are not watered down.³

However, the fate of the proposed Regulation is tied to the mast of the EU court’s case law on fundamental rights. And the judgment in the Gonzáles case has tightened those ropes. Thus, we are unlikely to see in the near future any significant diminishment—at least formally—of Europe’s de-indexation dig-in. 

References

1. Judgment of May 13, 2014 in Case C-131/12, *Google Spain SL and Google Inc v Agencia Española de Protección de Datos (AEPD) and Mario Costeja González*.
2. Scott, M. Google details requests in Europe ‘to be forgotten.’ *International New York Times* (Oct. 10, 2014), 16.
3. U.S. Diplomat Warns of “Trade War” if “Right to be Forgotten” Proposals are Followed Through. Pinsent Masons: Out-Law.com (February 4, 2013); <http://bit.ly/1sNMlry>.

Lee A. Bygrave (lee.bygrave@jus.uio.no) is Professor of Law and Director of the Norwegian Research Center for Computers and Law at the University of Oslo. His most recent books are *Data Privacy Law: An International Perspective* (2014) and *Internet Governance by Contract* (2015), both published by Oxford University Press.

Copyright held by author.



Association for
Computing Machinery

ACM Conference Proceedings Now Available via Print-on-Demand!

Did you know that you can now order many popular ACM conference proceedings via print-on-demand?

Institutions, libraries and individuals can choose from more than 100 titles on a continually updated list through Amazon, Barnes & Noble, Baker & Taylor, Ingram and NACSCORP: CHI, KDD, Multimedia, SIGIR, SIGCOMM, SIGCSE, SIGMOD/PODS, and many more.

For available titles and ordering info, visit:
librarians.acm.org/pod





The Business of Software

A Little Queue Theory

When more work means less done.

SO, HOW FULL is your in-box? How many tasks on your to-do list? Take a look at that calendar—any open spaces? No? That's good, isn't it? The more work that is in the hopper, the more must be getting done, right? Empty spaces in the plan means less is being achieved, right? Well, maybe not.

It seems to be an axiom of modern projects that for every project team member, every minute of every day should be filled with assigned tasks. I know project managers who boast of loading themselves and their subordinates with way more than 100% of available time. It is not uncommon for people to be juggling 10 or 20 tasks concurrently. As one who admits to poor multitasking capability, I have always thought there is a downside to having too much on one's plate at one time. But it is difficult to argue against this overloading of effort when the equation is more work on the task list means more gets done. But is that true?

If You Do This, You Don't Do That

One of the simplest ways to look at this is: if I am working on *this* task, I am not working on *that* task. *That* task is waiting. Of course, I am working on *this* task because it is the most important, it is the most critical. To me. But what are the consequences of the pending work? Who and what is waiting on *that*? And what is happening while they wait?

In any project, tasks and products have dependencies; some work cannot be properly done until prior work



is under way or completed. Attempting the dependent work before its upstream dependency is resolved can result in rework ranging from a small percentage of the original task effort to more than 100% when the erroneous work has to be undone and the correct work redone.

The Random Plan

In software development, project planning is often viewed as a definitive pre-

diction that charts the future of the project, rather like laying down track for a railroad. If the plan is “good” the project will follow it unerringly like a train on rails. Project failures are often laid at the door of planning—project failure must be planning failure. This philosophy holds that, if planning is good, more planning is better and more planning will make your project succeed. This might be true for highly deterministic activities, tasks that are

predictable and do not vary from their expected course, and tasks that are not subject to volatile internal and external forces. Unfortunately, that does not describe most software projects.

While a plan might provide an overall roadmap for a project, and is certainly necessary for project staffing and resource allocation, to some extent all projects deviate from their initial plan. Elements of the plan and the dependencies between tasks often operate somewhat unpredictably and even randomly.

Driving to Work

Rather than a train on tracks, a better metaphor is driving to work. While we might have a chosen route, we may also encounter road construction, heavy traffic, or an accident. When this happens we do not just stick to our original plan, we adjust our route based on the latest information. Construction and traffic might be predictable but accidents are not. So it is with projects: some things we can know in advance if we do sufficient research, for other issues we might make an educated probabilistic guess, while other stuff just happens and must be dealt with as best we can if and when it happens.

Items in Queue

When some percentage of our workload is driven by random events, we can suggest certain characteristics of the way our tasks are done and not done. Operating at high levels of committed time (what queuing theory calls “capacity utilization”) causes negative effects. One of these is the number of tasks that will typically be in queues waiting to be worked.

If the capacity utilization is given by ρ (the Greek letter rho), then the percentage of time work will “arrive” but not be able to be worked is simply $(1 - \rho)$. So if we have 95% utilization and something urgent comes along, there is only one chance in 20 that it can be worked right away. It will have to go into the work queue and wait for its turn. Simple work queues can be considered to have: random work arrival (as unexpected work tends to be); variable time to complete the newly arrived work; single tasking, where a person can really only work on one thing at a time; and a really

Project failures are often laid at the door of planning—project failure must be planning failure.

large in-box where we could have an arbitrarily large number of things waiting to be done. These are all approximations of real project situations and I will deal with the differences in a later column. But even this simple queue model gives us insights into how queues affect our productivity and our throughput. For instance, the number of items in the queue is given by $\rho^2/(1 - \rho)$. With a utilization of 80% of available capacity, we might have one thing being worked and around three things *not* being worked. Push this utilization to 95% and there will be 18 things not being worked. At 98% unworked items are up around 48. That is a lot of things not being done. When tasks are dependent in projects and work queues are large, each of these things not being done may have downstream effects—there is work in the project that is simply waiting on work that is currently stuck in someone’s email in-box.

Less Important Tasks

When important tasks with downstream dependencies are deferred, people work on less-important tasks. This means their more-important tasks are also delayed and there is a snowball effect on the project. Bit by bit, these more-important tasks are sidelined, critical paths lengthened, and the project slips. Unfortunately, the classic project management response to a project that is slipping is to pile on the work under the assumption that more work means more is getting done. But when piling on causes the work queues to get bigger, the opposite may be true. This is a factor that contributes to Brooks’ Law: *Adding manpower to a late software project makes it later.*¹

The very high utilization levels to

which project managers load their people can only work in fully deterministic systems—systems where the work that needs to be done, when it will be done, by whom, and how long it will take to complete is completely predefined and invariant.² This does not describe the business of software at all.

The solutions that projects try may actually compound the problem. Queuing theory shows that loading up on the tasks makes the work queues worse. Throwing more people at a project is ineffective or detrimental. Replanning can work, but in volatile situations the constant replanning effort itself only adds to the workload and the planning churn itself slows down the project. There are other things we can try.

Measure Queues, Relax Loading

Most mature projects measure when tasks are done and what they cost to do; but these are trailing indicators. If we expect a task to take 40 hours we usually learn it actually takes 80 hours after the task has overrun and the downstream damage has already occurred. Using two cooperative approaches we should be able to better manage our throughput on projects:

- Intentionally plan for less-than-100% task loading. Effectively managed, the unblocked time a project would allocate is a workload buffer that can absorb the variation in tasks effort and task priority when unpredicted events occur.

- Measure our work queues; keep track, not only of what is being done, but what is not being done and how long the uncompleted task lists are becoming.

The measurement and control of our nondeterministic work queues might be one aspect of project management that traditional approaches have traditionally neglected and that would help balance an unbalanced work flow. ■

References

1. Brooks, F.P. *The Mythical Man-Month*. Addison-Wesley Publishing Company, Reading, MA, 1975, 25.
2. Reinertsen, D.G. *The Principles of Product Development Flow*. Celeritas Publishing, Redondo Beach, CA, 2009, 60.

Phillip G. Armour (armour@corvusintl.com) is a senior consultant at Corvus International Inc., Deer Park, IL, and a consultant at QSM Inc., McLean, VA.

Copyright held by author.

Historical Reflections

The Tears of Donald Knuth

Has the history of computing taken a tragic turn?

IN THIS COLUMN I will be looking at the changing relationship between the discipline of computer science and the growing body of scholarly work on the history of computing, beginning with a recent plea made by renowned computer scientist Donald Knuth. This provides an opportunity to point you toward some interesting recent work on the history of computer science and to think more broadly about what the history of computing is, who is writing it, and for whom they are writing.

Last year historians of computing heard an odd rumor: that Knuth had given the Kailath lecture at Stanford University and spent the whole time talking about us. Its title, “Let’s Not Dumb Down the History of Computer Science,” was certainly intriguing, and its abstract confirmed that some forceful positions were being taken.^a The online video eventually showed something remarkable: his lecture focused on a single paper, Martin Campbell-Kelly’s 2007 “The History of the History of Software.”^b Reading it had deeply saddened Knuth, who “finished reading it only with great difficulty” through his tear-stained glasses.

What Knuth Said

Knuth began by announcing that, despite an aversion to confrontation, he

would be “flaming” historians of computing. This, he worried “could turn out to be the biggest mistake of my life.” The bout might nevertheless be seen as a mismatch. Knuth is among the world’s most celebrated computer scientists, renowned for his ongoing project to classify and document families of algorithms in *The Art of Computer Programming* and for his creation of the TeX computerized typesetting system ubiquitous within computer science and mathematics. Campbell-Kelly has a similar prominence within

the much smaller community of historians of computing but, even by Google Scholar’s generous definitions, the paper that saddened Knuth has been cited only nine times.

Knuth then enumerated his motivations, as a computer scientist, to read the history of science. First, reading history helped him to understand the process of discovery. Second, understanding the difficulty and false starts experienced by brilliant historical scientists in making discoveries that specialists now find obvious helped him to



Donald Knuth.

^a See http://kailathlecture.stanford.edu/featured_speaker.html#abstract_bio.

^b The video is posted at <http://kailathlecture.stanford.edu/2014KailathLecture.html>.

see what made concepts challenging to students and thus to become a “much better writer and teacher.” Third, appreciating the historical contribution of non-Western scientists helped in “celebrating the contributions of many cultures.” Fourth, history is the craft of telling stories, which is “the best way to teach, to explain something.” Fifth, the biographies of scientists teach tactics for a successful and rewarding career. Sixth, history teaches how human experience has changed over time. As humans we should care about that.

Knuth also identified some special contributions to the history of science that professionally trained historians are uniquely well placed to make. We are good at “smoking out” primary sources and putting historical activities in the context of broader timelines. He also appreciates our ability to translate papers written in languages that he cannot himself read. He finds attempts at historical analysis “probably the least interesting” aspects of our papers but appreciates lengthy quotations from primary sources.

Things then headed in a less positive direction. Knuth explained that Campbell-Kelly had centered his paper on a table of important works related to the history of software published between 1967 and 2004. It coded the predominant approaches into four categories—one of which was technical—to demonstrate the technical approach had been dominant until about 1990, dwindling thereafter and vanishing altogether after 1997. Campbell-Kelly characterized this as an “evolution” away from “technical histories” of the “low-hanging-fruit variety” written by Knuth and other “outstanding technical experts” that were “constrained, excessively technical, and lacking in breadth of vision.”

Knuth had previously viewed Campbell-Kelly as a kindred spirit but had now been granted a glimpse of “what historians say when they’re talking to historians instead of when they’re talking to people like me.” Without pausing to dry his glasses he had written to Campbell-Kelly to accuse him of having “lost faith in the notion that computer science is actually scientific.”

The shift described by Campbell-Kelly reflected a change in the population of scholars writing the history of

So why is the history of computer science not being written in the volume it deserves, or the manner favored by Knuth?

computing. Many of the senior computing figures of the 1970s worked to preserve the history of the 1940s and early 1950s, starting with a number of “pioneer days” and workshops organized. The most important of these was held at Los Alamos National Laboratory in 1976.¹⁵ Most of the 90 participants included in the group photograph of attendees were computer pioneers of the 1940s. Knuth himself contributed a detailed history of the first tools for “automatic programming” (assemblers and compilers). He was one of a handful of interested younger computer scientists who entered the field in the 1950s, which also included Edsger Dijkstra and Brian Randell, a systems programmer turned academic who had assembled an important collection of reprinted historical documents. At the conference were only a handful of trained historians. The editorial board of *Annals of the History of Computing*, which began in 1979 as a publication of AFIPS, a long-defunct umbrella group for professional computing societies, had a similar makeup. As graduate students in history and history of science programs began to write dissertations on computer-related topics they eventually inverted the ratio of trained historians to computer scientists, though the journal continues to publish a significant number of papers by computer scientists and technical experts.

In his lecture Knuth worried that a “dismal trend” in historical work meant that “all we get nowadays is dumbed down” through the elimination of technical detail. According to Knuth “historians of math have always faced the fact that they won’t be able to please everybody.” He feels that other historians of science have succumbed

to “the delusion that ... an ordinary person can understand physics ...”

I am going to tell you why Knuth’s tears were misguided, or at least misdirected, but first let me stress that historians of computing deeply appreciate his conviction that our mission is of profound importance. Indeed, one distinguished historian of computing recently asked me what he could do to get flamed by Knuth. Knuth has been engaged for decades with history. This is not one of his passionate interests outside computer science, such as his project reading verses 3:16 of different books of the Bible. Knuth’s core work on computer programming reflects a historical sensibility, as he tracks down the origin and development of algorithms and reconstructs the development of thought in specific areas. For years advertisements for *IEEE Annals of the History of Computing*, where Campbell-Kelly’s paper was published, relied on a quote from Knuth that it was the only publication he read from cover to cover. With the freedom to choose a vital topic for a distinguished lecture Knuth chose to focus on history rather than one of his better-known scientific enthusiasms such as literate programming or his progress with *The Art of Computer Programming*.

Computing vs. Computer Science

Here is where I part ways with Knuth’s interpretation. Campbell-Kelly’s article was “The History of the History of Software,” not “The History of the History of Computer Science.” Knuth’s complaint that historians have been led astray by fads and pursuit of a mass audience into “dumbed down” history reflects an assumption that computer science is the whole of computing, or at least the only part in which historians can find important questions about software. This conflated the history of computing with the history of computer science. Distinguished computer scientists are prone to blur their own discipline, and in particular few dozen elite programs, with the much broader field of computing. The tools and ideas produced by computer scientists underpin all areas of IT and make possible the work carried out by network technicians, business analysts, help desk workers, and Excel programmers. That does not make those workers computer scientists.

The U.S. alone is estimated to have more than 10 million “information technology workers,” which is about a hundred times more than the ACM’s membership. Vint Cerf has warned in *Communications* that even the population of “professional programmers” dwarfs the association’s membership.⁷ ACM’s share of the IT workforce has been in decline for a half-century, despite efforts begun back in the 1960s and 1970s by leaders such as Walter Carlson and Herb Grosch to broaden its appeal.

Computing is much bigger than computer science, and so the history of computing is much bigger than the history of computer science. Yet Knuth treated Campbell-Kelly’s book on the business history of the software industry (accurately subtitled “a history of the software industry”) and all the rest of the history of computing as part of “the history of computer science.”⁴ Others have written about the history of computer use in life insurance and other areas of business, the history of cybernetics, the history of the semiconductor industry, the history of punched card machines, the history of the IT workforce, the history of computer-producing companies such as IBM, the use and development of computers in particular countries, the history of the personal computer, and the history of computer usage in particular areas of scientific practice such as biomedicine. To call such work “dumbed down” history of computer science, rather than smart history of many other things, is to misunderstand both the intentions and the accomplishments of its authors.

The truth is that regrettably little history of computer science, whether dumb or deep, has been written by trained historians even though the history of computing literature as a whole has been expanding rapidly. Consider our output between 1990 and 2010. Michael Mahoney, a historian of science and mathematics at Princeton University, worked on a narrative history of theoretical computer science but ultimately produced only a set of provocative but schematic papers.¹³ Mahoney was also interested in the history of software engineering, and several other historians have discussed the 1968 NATO Conference on Soft-

ware Engineering at which that field was launched. Eminent sociologist of science Donald MacKenzie worked on the history of formal methods and its relationship to the development of computer technology.^{11,12} Two books explored the history of DARPA and its role in shaping the development of computer science and technology, though Knuth would not approve of their institutional focus.^{17,19} William Aspray wrote several papers on the history of NSF support for computing² and a book on John von Neumann.¹ A complete list would be longer, but not that much longer.

Historical Careers in Computer Science

So why is the history of computer science not being written in the volume it deserves, or the manner favored by Knuth? I am, at heart, a social historian of science and technology and so my analysis of the situation is grounded in disciplinary and institutional factors. Books of this kind would demand years of expert research and sell a few hundred copies. They would thus be authored by those not expected to support themselves with royalties, primarily academics.

Academic careers are profoundly shaped by the disciplinary communities in which they develop. Throughout their training, scholars are socialized into the culture of their field and pick up a wealth of tacit and explicit knowledge on what is expected of them. They learn how to select a research project, what kinds of work are noticed and which are ignored, what style to write in, how to structure a paper, which professors are respected, what search committees and grant review panels are looking for. This continues throughout their careers, as they aspire to prestigious awards, named chairs, or favors from the Dean. Whether they realize it or not, successful academics have internalized the rules of the game played in their particular field.

The history of computer science might be undertaken from two disciplinary base camps within academia: computer science and the history of science. Someone whose primary training is in history will naturally see the history of computing differently from someone whose disciplinary loy-

alty is to computer science. They will choose different topics and explore them in different ways for different audiences. For different reasons, outlined below, neither group has shown much interest in supporting work of the kind favored by Knuth. That is why it has rarely been written.

Prospects within the History of Science

The history of science is a kind of history, which is in turn part of the humanities. Some historians of science are specialists within broad history departments, and others work in specialized programs devoted to science studies or to the history of science, technology, or medicine. In both settings, historians judge the work of prospective colleagues by the standards of history, not those of computer science. There are no faculty jobs earmarked for scholars with doctoral training in the history of computing, still less in the history of computer science. The persistently brutal state of the humanities job market means that search committees can shortlist candidates precisely fitting whatever obscure combination of geographical area, time period, and methodological approaches are desired. So a bright young scholar aspiring to a career teaching and researching the history of computer science would need to appear to a humanities search committee as an exceptionally well qualified historian of the variety being sought (perhaps a specialist in gender studies or the history of capitalism) who happens to work on topics related to computing.

This, more than anything else, explains the rise of the broad and non-technical approaches decried by Knuth. Work in the history of computing has been seen by most in the humanities as dull and provincial, excessively technical and devoid of big historical ideas. Whereas fields such as environmental history have produced widely recognized classics that convince non-specialists of the scholarly potential, historians of computing are still inching toward broad acceptance of their relevance. The roles Knuth outlined for them would not serve them well as they were essentially those of the research assistant: gather primary materials, translate them if necessary,

Historical work on computing as a contribution to computer science versus as a contribution to a subfield of history or social science studies.

Historical work on computing framed primarily as a contribution to computer science.	Common in the 1970s and 1980s, less so today.	Almost impossible to accomplish (Campbell-Kelly's early work being an exception).
Historical work on computing framed primarily as a contribution to a subfield of history or science studies.	Fairly common, particularly for trained historians of computing working outside academia or in fields where history is not seen as central.	Increasingly common, with hopeful signs for further growth.
	Historical work on computing as a secondary interest or activity during retirement.	Historical work on computing as the major focus of a scholarly career, for which one is hired or promoted.

historical interests from tenured positions or to dabble after retirement. Despite some worthy initiatives, such as the efforts of the ACM History Committee to encourage historical projects, the impulse to write technical history has not spread widely among younger generations of distinguished and secure computer scientists.

To summarize, the upper-right quadrant in the accompanying table is essentially empty. It reflects historical work forming the backbone of a scholarly career and intended as a contribution to computer science. I share Knuth's regret that the technical history of computer science is greatly understudied. The main cause is that computer scientists have lost interest in preserving the intellectual heritage of their own discipline. It is not, as Knuth implies, that Campbell-Kelly is representative of a broader trend of individual researchers deciding to stop writing one kind of history and to devote a fixed pool of talent to writing another kind instead. There is no zero sum game here. More work by professionally trained historians on social, institutional, and cultural aspects of computing does not have to mean less work by computer scientists themselves. They cannot count on history departments to do this for them, and I hope Knuth's lament motivates a few to follow his lead in this area. Not simply because Knuth did it—few computer scientists have emulated him by procuring their own domestic pipe organs—but because his commitment to the intellectual history of computer science makes a powerful argument that historical knowledge of a particular kind is a prerequisite for deep technical understanding.

Reopening the Black Box

I will end on a positive note. In his paper, Campbell-Kelly offered a "biographical mea culpa" for his own early work that he now reads with a "mild flush of embarrassment." He came to see his erstwhile enthusiasm for technical history as a youthful indiscretion and his conversion to business history as an act of redemption, paralleling his own development and that of the field in a way that relied implicitly on a rather unfashionable conceptualization of history as progress along a fixed trajectory.

and make them available to computer scientists who will do the analysis.

Current enthusiasm for the "digital humanities" and the inescapable importance of computing to the modern world could provide opportunities. One day humanities search committees might even seek out historians of computing, but only those whose work engages with and appeals to scholars who themselves know nothing of computer science. In the meantime many scholars with doctorates in the history of computing have found work in museums or in academic employment outside both history and computer science, for example, in business schools, information schools, or specialist programs such as engineering education. These positions pose their own disciplinary challenges, but for obvious reasons provide few incentives to study the history of computer science.

Prospects within Computer Science

Thus the kind of historical work Knuth would like to read would have to be written by computer scientists themselves. Some disciplines support careers spent teaching history to their students and writing history for their practitioners. Knuth himself holds up the history of mathematics as an example of what the history of computing should be. It is possible to earn a Ph.D. within some mathematics departments by writing a historical thesis (euphemistically referred to as an "expository" approach). Such departments have also been known to hire, tenure, and promote scholars whose research is primarily historical. Likewise medical schools, law schools, and a few busi-

ness schools have hired and trained historians. A friend involved in a history of medicine program recently told me that its Ph.D. students are helped to shape their work and market themselves differently depending on whether they are seeking jobs in medical schools or in history programs. In other words, some medical schools and mathematics departments have created a demand for scholars working on the history of their disciplines and in response a supply of such scholars has arisen.

As Knuth himself noted toward the end of his talk, computer science does not offer such possibilities. As far as I am aware no computer science department in the U.S. has ever hired as a faculty member someone who wrote a Ph.D. on a historical topic within computer science, still less someone with a Ph.D. in history. I am also not aware of anyone in the U.S. having been tenured or promoted within a computer science department on the basis of work on the history of computer science. Campbell-Kelly, now retired, did both things (earning his Ph.D. in computer science under Randell's direction) but he worked in England where reputable computer science departments have been more open to "fuzzy" topics than their American counterparts. Neither are the review processes and presentation formats at prestigious computer conferences well suited for the presentation of historical work. Nobody can reasonably expect to build a career within computer science by researching its history.

In its early days the history of computing was studied primarily by those who had already made their careers and could afford to indulge pursuing

Contrary both to Knuth's despair and to Campbell-Kelly's story of a march of progress away from technical history, some scholars with formal training in history and philosophy have been turning to topics with more direct connections to computer science over the past few years. Liesbeth De Mol and Maarten Bullynck have been working to engage the history and philosophy of mathematics with issues raised by early computing practice and to bring computer scientists into more contact with historical work.³ Working with like-minded colleagues, they helped to establish a new Commission for the History and Philosophy of Computing within the International Union of the History and Philosophy of Science. Edgar Daylight has been interviewing famous computer scientists, Knuth included, and weaving their remarks into fragments of a broader history of computer science.⁸ Matti Tedre has been working on the historical shaping of computer science and its development as a discipline.²² The history of Algol was a major focus of the recent European Science Foundation project *Software for Europe*. Algol, as its developers themselves have observed, was important not only for pioneering new capabilities such as recursive functions and block structures, but as a project bringing together a number of brilliant research-minded systems programmers from different countries at a time when computer science had yet to coalesce as a discipline.^c Pierre Mounier-Kuhn has looked deeply into the institutional history of computer science in France and its relationship to the development of the computer industry.¹⁶

Stephanie Dick, who recently earned her Ph.D. from Harvard, has been exploring the history of artificial intelligence with close attention to technical aspects such as the development and significance of the linked list data structure.^d Rebecca Slayton, another Harvard Ph.D., has written about the engagement of

The history of computer science retains an important place within the diverse and growing field of the history of computing.

prominent computer scientists with the debate on the feasibility of the "Star Wars" missile defense system; her thesis has been published as an MIT Press book.²⁰ At Princeton, Ksenia Tatarchenko recently completed a dissertation on the USSR's flagship Akademgorodok Computer Center and its relationship to Western computer science.²¹ British researcher Mark Priestley has written a deep and careful exploration of the history of computer architecture and its relationship to ideas about computation and logic.¹⁸ I have worked with Priestly to explore the history of ENIAC, looking in great detail at the functioning and development of what we believe to be the first modern computer program ever executed.⁹ Our research engaged with some of the earliest historical work on computing, including Knuth's own examination of John von Neumann's first sketch of a modern computer program¹⁰ and Campbell-Kelly's technical papers on early programming techniques.⁵

Most of this new work is aimed primarily at historians, philosophers, or science studies specialists rather than computer scientists. However, it does not shy away from engagement with the specifics of computer technology or the detailed workings of the computer science community, re-introducing technical analysis along with continued attention to social, cultural, and institutional factors. Some of it may confirm Campbell-Kelly's prediction that the field will move toward "holistic" work integrating different approaches.

The history of computer science retains an important place within the diverse and growing field of the history of computing. Work of the particular kind preferred by Knuth will flourish only if his colleagues in computer science are willing to produce, reward, or commission it. I nevertheless hope he will continue to find much value in the work of historians and that we will rarely give him cause to reach for his handkerchief. **C**

References

1. Aspray, W. *John von Neumann and the Origins of Modern Computing*. MIT Press, Cambridge, MA, 1990.
2. Aspray, W. and Williams, B.O. Arming American scientists: NSF and the provision of scientific computing facilities for universities, 1950–73. *IEEE Annals of the History of Computing* 16, 4 (Winter 1994), 60–74.
3. Bullynck, M. and De Mol, L. Setting-up early computer programs: D.H. Lehmer's ENIAC computation. *Archive of Mathematical Logic* 49 (2010), 123–146.
4. Campbell-Kelly, M. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. MIT Press, Cambridge, MA, 2003.
5. Campbell-Kelly, M. Programming the EDSAC: Early programming activity at the University of Cambridge. *Annals of the History of Computing* 2, 1 (Jan. 1980), 7–36.
6. Campbell-Kelly, M. The history of the history of software. *IEEE Annals of the History of Computing* 29, 4 (Oct.–Dec. 2007), 40–51.
7. Cerf, V. ACM and the professional programmer. *Commun. ACM* 57, 8 (Aug. 2014), 7.
8. Daylight, E.G. *The Dawn of Software Engineering: From Turing to Dijkstra*. Lonely Scholar, Heverlee, Belgium, 2012.
9. Haigh, T., Priestley, M., and Rope, C. Los Alamos bets on ENIAC: Nuclear Monte Carlo simulations, 1947–48. *IEEE Annals of the History of Computing* 36, 2 (Jan.–Mar. 2014), 42–63.
10. Knuth, D.E. Von Neumann's first computer program. *ACM Computing Surveys* 2, 4 (Dec. 1970), 247–260.
11. MacKenzie, D. *Knowing Machines*. MIT Press, Cambridge, MA, 1998.
12. MacKenzie, D. *Mechanizing Proof*. MIT Press, Cambridge, MA, 2001.
13. Mahoney, M.S. and Haigh, T., Eds. *Histories of Computing*. Harvard University Press, Cambridge, MA, 2011.
14. Matti, T. *The Science of Computing: Shaping a Discipline*. CRC Press/Taylor & Francis, 2014.
15. Metropolis, N., Howlett, J. and Rota, G.-C. Eds. *A History of Computing in the Twentieth Century: A Collection of Papers*. Academic Press, New York, 1980.
16. Mounier-Kuhn, P. Logic and computing in France: A late convergence. In *Proceedings of the Symposium on the History and Philosophy of Programming* (Birmingham, July 2012); <http://bit.ly/15ZRPLx>.
17. Norberg, A.L. and O'Neill, J.E. *Transforming Computer Technology: Information Processing for the Pentagon, 1962–1986*. Johns Hopkins University Press, Baltimore, MD, 1996.
18. Priestley, M. *A Science of Operations: Machines, Logic, and the Invention of Programming*. Springer, New York, 2011.
19. Roland, A. and Shiman, P. *Strategic Computing: DARPA and the Quest for Machine Intelligence*. MIT Press, Cambridge, MA, 2002.
20. Slayton, R. *Arguments that Count: Physics, Computing, and Missile Defense, 1949–2012*. MIT Press, Cambridge, MA, 2013.
21. Tatarchenko, K. *A House With the Window to the West: The Akademgorodok Computer Center (1958–1993)*. Princeton, 2013.
22. Tedre, M. *The Science of Computing: Shaping a Discipline*. CRC Press/Taylor & Francis, 2014.

Thomas Haigh (thaigh@computer.org) is an associate professor of information studies at the University of Wisconsin, Milwaukee, and immediate past chair of the SIGCIS group for historians of computing.

Copyright held by author.

c *IEEE Annals of the History of Computing* 36, 4 (Oct.–Dec. 2014) is a special issue based on this work.

d Dick had earlier published "AfterMath: The Work of Proof in the Age of HumanMachine Collaboration," *Isis* 102, 3 (Sept. 2011), 494–505.

Viewpoint

Wearables: Has the Age of Smartwatches Finally Arrived?

Time will tell if smartwatches will find their niche.

THE IDEA OF strapping a computer to one's wrist is not new. By the middle of the 20th century there were plenty of examples of this in science fiction and the 1970s saw the first LED and LCD watches such as Pulsar NL C81. Intel's Gordon Moore famously bought the Microma digital watch company, hoping to develop a more powerful wrist-mounted computer. However, the technology was expensive at the time and not powerful enough to provide much more functionality than non-digital watches. Therefore, similar efforts, such as the Fossil wrist PDA and IBM/Citizen WatchPad also failed.

In the last century, the wristwatch replaced the pocket watch as a more convenient place to get at time information quickly. More recently, the ubiquity and functionality of mobile phones (which can also display the time) caused many to abandon the wristwatch as redundant jewelry and go back to what is effectively a pocket watch.

Now after more than 40 years of technological improvements, smartwatches seem to be heading in the same direction as mobile phones: multiple functionalities. Smartwatches are not only a chronograph, but can also be a general-purpose, networked computer with an array of sensors. Moreover, they have followed a familiar pattern in technological trends: the concept has been around for long time, but it took longer for the technology to advance enough for proper implementation.



Recent advances in miniaturization and diminishing costs in battery, processor, sensor, and communication technologies are now enabling smartwatches to move from the specialist market to the mainstream.

To gain an overview of the research potential and technological impacts of these devices, we provide a brief description of the current market. Based on the current market status, literature reviews, and reviews of smartphone technology, we can identify both restrictions and advantages of smartwatches—and from this highlight the research challenges and opportunities associated with smartwatches.

Market Demand and Direction

In May 2012, the Pebble watch designers had one of the most success-

ful campaigns in the history of crowd funding at the time.^a The Pebble story is evidence of a huge market demand for such technology. Generator research predicts this market will grow to 214 million units by 2018.^b

The Pebble is not the first modern smartwatch; it was preceded by Sony's SmartWatch. However, because Sony's device relied on its Bluetooth connection to act as a complementary interface to compatible Sony smartphones, the Pebble could be considered the first fully independent smartwatch.

Large-scale consumer electronics manufacturers have begun to release their own smartwatches or define specifications for their future smartwatches.

^a See <http://kck.st/18KLHqM>

^b See <http://bit.ly/1ybRoEm>

Some, such as Samsung and Sony, have started to launch smartwatches as a complementary interface for their mobile devices (smartphone or tablet) while others are adopting a hybrid approach. For example, the Android Wear watch requires an Android phone to install apps, but the watch can then operate independently without a smartphone connection. Both the Apple Watch and the Microsoft Band also function independently of a smartphone but make use of a smartphone for some communications. It may be too early to judge if smartwatches are truly independent computing devices, or if their restricted capabilities and competition in the market make them little more than remote interfaces or limited-function fitness devices. Nevertheless, this property provides a crucial focus for analyzing their constraints and advantages.

Constraints

Smartwatches suffer from two major constraints to do with keeping them small enough to wear on a wrist: their small screen size results in restricted I/O and their small hardware results in weaker computing capability and especially limited battery capacity in comparison larger devices. Although mobile devices have advanced in recent years, the challenge of reducing size while maintaining features and battery life continues.

The small screen size restricts input and output capabilities. For instance, fitting a keyboard on a smartwatch screen or wristband is even more challenging than fitting one on a smartphone. Although there have been efforts to enable smartwatches to display multimedia, their small screen size makes them a poor choice as a media player for video or images. Micro projectors are a tantalizing solution, but several years after their announcement they are still too expensive to consider, and thus wearable projectors³ do not seem to be a realistic prospect in the near future. A larger screen, which can be achieved through curving the screen around the wrist, seems to be an option to begin to address the size issue, but still not enough for keyboard integration. Several brands, such as LG, have launched their curved screen for TVs and Samsung also recently released Samsung Gear S smartwatch with a

Smartwatches have two strong advantages over other devices: their mount location and (probably more important) the continual connection to the skin.

curved screen. Most traditional watch screens are lightweight, waterproof/resistant and antiscratch. How well curved screens maintain these characteristics remains to be seen. The screen restriction will also require fresh thinking on user interface (UI) designs and for new interaction techniques such as 3D ultrasonic gesture recognition.^c Google's Android Wear relies heavily on voice input for simple user interaction but this may restrict users to performing more complicated tasks.

Small hardware means less computing power, smaller battery capacity, and less-precise sensors. These challenges exist in smartphones as well as smartwatches, although to a lesser extent. Software and hardware providers have been working, slowly but continuously, to address resource-related issues. For example, GPS systems typically have a high power requirement. Localization services try to overcome this problem by relying on a combination of sensors such as GSM, Wi-Fi, and GPS—hence reducing the power consumption.

Although hardware components get smaller through advances in sensor technologies and electronics,⁷ the smaller footprint means smartwatches have fewer ubiquitous computing features compared to smartphones. Most of the smartphone-dependent smartwatches overcome this by offloading power-consuming sensing and computing operations to the phone and use low-power Bluetooth to communicate. This enables smartwatches to communicate with the smartphone

and to rely on its superior computing capabilities. This way, power-hungry chips such as GPS can draw on the larger battery of the phone and simply use the watch as a convenient user interface, but the phone still has most of the “smarts.” The fully independent smartwatches, which incorporate their own 3G or GPS chips, tend to have a much shorter battery life.

Advantages

On the other hand, smartwatches have two strong advantages over other devices: their mount location, and (probably more important) the continual connection to the skin. Similar to augmented reality glasses, their interactions do not always require both hands. In particular, users do not need to hold the device, which is required by smartphones and one hand is completely occupied. Nevertheless, users need to change wrist position and, if a touch or voice input is needed, use the other hand for the interaction.

Smartwatches are body mounted, with a standard, known location. This helps activity recognition research by removing the burden of identifying the location of the device. Much research has been done to quantify health information via mobile devices (mHealth). The standard location overcomes challenges associated with smartphones; for instance, when measuring users' activities based on accelerometer data, the location of the smartphone (if in a shirt pocket, in a bag, or in a trouser pocket) affects the data.

A smartwatch is typically in constant contact with its owner, and thus capable of recognizing its owner's physical activities and location. In contrast, a limitation associated with smartphones is that, when users are not holding them, they can sense only the users' environment (outward) and not the users' specific condition (inward). When smartphones are carried, they are in various locations in bags and pockets. Often, in an indoor environment such as an office or home, a smartphone is not moving and may not be co-located with its owner. The smartwatch's continual connection to the skin could revolutionize mHealth studies. Enterprise vendors recognize this and are already including slots for this data in their health kit toolsets, such as Google Fit,

^c Chirp gestural interface: <http://bit.ly/1zM3tSP>

Apple HealthBook, Samsung S.A.M.I, and Microsoft Healthvault.

The location also permits easy recording of heart rate, heart rate variability, temperature, blood oxygen, and galvanic skin response (GSR). GSR can be used to identify physiological arousal, especially when combined with heart rate and heart rate variability. Emotion has two dimensions: mood valence (positive vs. negative) and arousal (high vs. low).⁴ Automatic emotion quantification approaches (except those based on image recognition) are restricted to arousal and not valence. Therefore, in order to collect valence, existing affective recognition systems rely either on image recognition or manual user input. Because user input is subjective, its accuracy is often questioned. With the advent of smartwatches and effective multisensor data collection, we might develop algorithms (for sensor data fusion) that can identify valence without the need for processing a facial image.

Research Opportunities and Future Visions

Based on the constraints and advantages discussed, we can identify several research opportunities for advances in smartwatches.

► Battery life and cost are probably the most important success factors for smartwatch acceptance in the market. The failure of Microsoft's SPOT (Smart Personal Object Technology),¹ because of cost and battery limitations, and the delays in Apple's Watch release show the importance of considering these two factors. The challenge of optimizing resources, especially power, is an ongoing research topic. This is a multidisciplinary challenge. End-user applications, internal hardware (light-weight battery, hardware miniaturization) and external hardware (wireless inductive charging or substituting lithium-ion with other materials such as solid electrolytes) could be investigated further. Research on harvesting kinetic motion and body heat for electricity generation could also have an impact. However, device makers continue to focus on reinventing the battery and its charging process, such as charging wirelessly with magnetic induction. We have yet to see any revolutionary improvements.

► Smartwatches could be a significant boon to mHealth technologies. Physical activity sensing is already being used in many smartwatches and fitness trackers, such as the Nike+ SportWatch, Fitbit, JawBone, Basis, and Microsoft Band, to persuade users toward a more active lifestyle. Smartwatches could also host more bio-sensors and take more accurate measurements than the simple accelerometer-based wristbands. Algorithms that focus on energy-efficient activity recognition and convert raw biological data into higher-level data will need further development.

► With capability comes responsibility. Smartwatches can easily sense and record most private information such as sexual activities, sensitive medical information, and knowledge of the owner's fundamental weaknesses. Given the potential of smartwatches to sense and record individuals' behavior and physiological responses, maintaining the privacy of this information is another research challenge.

► Existing smartwatches have very small touchscreens with no or few buttons on the side, suggesting the need for new interaction techniques such as voice, haptic, gestural interfaces, projection, and near-field communication (NFC). Some brands, such as Android Wear, provide highly reliable voice commands for interaction. Some other brands are working on interaction through the accelerometer as well as haptic feedback. Therefore, it is likely that new interaction algorithms and also new UI patterns will be introduced, and existing approaches will be optimized toward better precision on a small screen and lower resource usage. For smartwatches that are smartphone dependent, interaction techniques are distributed between both the devices, for example, supporting keyboard input via the smartphone and voice input via the smartwatch.

► There are many discussions about the negative effect of new technologies on memory,^{2,5,6} and smartwatches are another attention- and cognition-consuming device. Although it may be that their restricted I/O capabilities will prevent users from interacting with them too much, their impact on activity and memory and cognition is still an open question to explore.

► What is the best operating system for smartwatches? Some vendors such as Samsung and Qualcomm launched their first devices with the Android OS and WIMM lab was developing Android-based smartwatches before Google acquired them in 2012. Google released a lighter Android version, Android Wear, for smartwatches and other wearables. Although Android is currently the most popular OS for smartphones, it is clear that an optimized OS is needed for smaller devices. Samsung is using a wearable-optimized OS, Tizen, for some of their wearables. Apple and Microsoft have taken a similar approach in creating a new OS for their wrist devices. Clearly there are research opportunities for OS developers to either create a new OS distribution or customize an existing one to make it light enough for smartwatches.

Smartwatches are likely to find their own place in the market. Just as tablets have not replaced the laptop and smartphones have not replaced TV remotes, so smartwatches may find their niche without replacing other devices. A new arena for pervasive computing research is emerging, one that is likely to attract many researchers and entrepreneurs, with interests from data analytics and self-quantification to hardware design, human interfaces, and manufacturing. **C**

References

1. Baker, C. How Microsoft lost the wrist-top. *Wired* 18, 7 (July 2010), 27.
2. Carr, N. *The Shallows: What the Internet Is Doing to Our Brains*. W.W. Norton, 2010.
3. Mistry, P. and Maes, P. SixthSense: A wearable gestural interface. In *ACM SIGGRAPH ASIA 2009 Sketches* (2009), 11.
4. Russell, J.A. A circumplex model of affect. *Journal of Personality and Social Psychology* 39, 6 (1980), 1161–1178.
5. Sparrow, B. Liu, J., and Wegner, D.M. Google effects on memory: Cognitive consequences of having information at our fingertips. *Science* 333, 6043, (2011), 776–778.
6. Spitzer, M. *Digital Dementia 2.0 Arguments about Risks and Side Effects of Digital Information Technology*. Schattauer GmbH, 2012.
7. Wright, A. Patient, heal thyself. *Commun. ACM* 56, 8 (Aug. 2013), 16–18.

Reza Rawassizadeh (rrawassizadeh@acm.org) is a project scientist in the Department of Computer Science and Engineering at the University of California, Riverside.

Blaine A. Price (b.a.price@open.ac.uk) is a senior lecturer in computing in the Centre for Research in Computing at the Open University, Milton Keynes, U.K.

Marian Petre (m.petre@open.ac.uk) is a professor of computing in the Centre for Research in Computing at the Open University, Milton Keynes, U.K.

Copyright held by authors.

Viewpoint

Does the Internet Make Us Stupid?

Yes, but this may not be as bad as it sounds...

ACCORDING TO FARHOOD Manjoo in *Slate* magazine, almost nobody finishes reading papers online.⁵ Jakob Nielsen has shown that superficial reading carries over from screens to printed material.^a Bauerlein,¹ Brabazon,² Carr,⁴ and others have argued convincingly that the Internet and other information and communication technologies (ICTs) are changing our reading habits to “skimming” rather than careful reading. Putting this together, I am concerned you will not read this Viewpoint carefully to the end. Hence, I better present my conclusions right away: ICTs are indeed reducing many of our cherished cognitive facilities, much as our physical fitness has been reduced by all kinds of machinery for physical work and locomotion. However, in my opinion, this is not too bad, as long as our reduced facilities are overcompensated by appropriate technology, and provided we make sure of two things: that we are not completely lost in case of large-scale breakdowns of technology, and that use of ICTs does not endanger our creativity. Both provisos are starting to receive attention: the first will hopefully be solved by introducing systems with sufficient redundancy; the second attracts varying opinions: some, like Carr, see mainly dangers, others like Thompson,⁷ see our future in a grow-



ing man-machine symbiosis. My own opinion is that creativity is not endangered if the new technologies are used with some caution.

If you stop reading here, you have read the important part of the message. If you continue reading, I hope I can drive home the message with emotional emphasis.

Over the last six years or so, numerous papers and books have claimed the Internet and related technologies are reduc-

ing our cognitive abilities. Here are some better-known examples and quotes.

Brabazon writes: *Looking at schools and universities, it is difficult to pinpoint when education, teaching and learning started to haemorrhage purpose, aspiration and function. As the Internet offers a glut of information, bored surfers fill their cursors and minds with irrelevancies, losing the capacity to sift, discard and judge.*² Brabazon is particularly worried by evidence she collected

^a See <http://www.nngroup.com/articles/how-users-read-on-the-web/>.

herself that *reading with understanding* and *creative writing* is markedly reduced in students who use ICTs intensively, and that concentrated thinking and attention spans appear much reduced. This is echoed by many later publications and books, including the ones discussed here.

The title of Bauerlein's book *The Dumbest Generation*¹ and its subtitle *How the Digital Age Stupefies Young Americans and Jeopardizes Our Future (Or, Don't Trust Anyone Under 30)* clearly indicate he is thinking along the same lines. Bauerlein was probably also the first to diagnose that ICTs are increasing the generation gap, since young people, in their effort to be "in," learn increasingly more from peers than from adults.

Carr states very strongly that new technologies including the Net *fragment contents and disrupt our concentration*.⁴ He emphasizes one important aspect that has been recognized by neuroscience and brain science for some time: use of new media like the Net creates new habits and changes the brain: *the plasticity of the brain can work against us by reinforcing certain behavior*. Carr gives many examples that *the way people read and write has already been changed by the Net that ... contributes to the ecosystem of interruption technologies*,⁵ even causing newspapers to change their style, by turning to shorter stories, or such. He quotes the neuroscientist Michael Merzenich, professor emeritus at the University of California in San Francisco, as saying *he was profoundly worried about the cognitive consequences of the constant distractions and interruptions the Net bombards us with. The long-term effect on the quality of our intellectual lives could be "deadly."*

Weber's book, *The Google-Copy-Paste Syndrome*, with statements such as: *The real danger is not that plagiarism is used fraudulently for personal gain, but that the copy/paste mentality destroys thinking* (translated from German by the author) emphasizes another aspect.

The title of Spitzer's book *Digital Dementia: How We Drive Ourselves and Our Children Mad* (translated from German by the author) says it all, even if you do not read to the point where he claims: *Computers for teaching subjects like history are as important as bicycles for teaching how to swim* (translated from German by the author).

Creativity is not endangered if the new technologies are used with some caution.

Why So Negative?

The previously mentioned authors share a number of concerns: We are so inundated by information that our attention span has become very small. We are heading (my terminology) toward a global attention deficit syndrome: we cannot be inactive any more, yet we are losing the power to concentrate. If we are not drowned by email, by text messages, or tweets telling us to look at some site or YouTube clip, or browsing our friends' updates on a social network, listening to mp3 music, zapping through 100 TV shows, answering the phone, or calling someone, we somehow *must* find another way to occupy ourselves. Many people waste so much time keeping their "friends" in the network happy they have little time left for productive work. They have externalized much of their knowledge into the cloud and their smartphones. They do not need to remember many facts, threatening the functioning of their memory. Students do not write essays these days. Rather, using Google search, copy, and paste, they glue pieces of information together, hardly understanding what they are producing. They are no longer able to read complicated texts, spoiled by bite-sized pieces of information contained in tweets or in text messages, often written in a new shorthand: "sms r good 4u." A blind trust that e-learning can replace good teachers often leads to less-educated children; apps are substituting thinking and cognitive capacity is shrinking.

These are some of the concerns often mentioned. If they were simply concerns, we could brush them off. However, almost all of them are based on solid quantitative research and experiments. Hence, we must take them seriously. The technologies involved

Calendar of Events

January 8–10

The 9th International Conference on Ubiquitous Information Management and Communication, Bali, Indonesia, Sponsored: SIGAPP, Contact: Sukhan Lee, Email: lsh@ece.skku.ac.kr

January 15–17

The 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Mumbai, India, Sponsored: SIGPLAN, Contact: Sriram Rajamani, Email: sriram@microsoft.com

January 16–19

Ninth Conference on Tangible, Embedded, and Embodied Interaction, Stanford, CT, Sponsored: SIGCHI, Contact: Wendy Ju, Email: wendyju@gmail.com

January 17–20

Foundations of Genetic Algorithms XIII, Aberystwyth, United Kingdom, Sponsored: SIGEVO, Contact: Thomas Jansen, Email: thj10@aber.ac.uk

January 19–22

20th Asia and South Pacific Design Automation Conference, Chiba, Japan, Sponsored: SIGDA, Contact: Kunio Uchiyama, Email: kunio.uchiuyama.xh@hitachi.com

January 26

SIGAI Career Network Conference, Austin, TX, Sponsored: SIGAI, Contact: Sanmay Das, Email: sanmay@seas.wustl.edu

January 31–February 4

ACM/SPEC International Conference on Performance Engineering, Austin, TX, Sponsored: SIGMETRICS and SIGSOFT, Contact: Connie U. Smith, Email: regester@spe-ed.com

ACM Transactions on Accessible Computing



This quarterly publication is a quarterly journal that publishes refereed articles addressing issues of computing as it impacts the lives of people with disabilities. The journal will be of particular interest to SIGACCESS members and delegates to its affiliated conference (i.e., ASSETS), as well as other international accessibility conferences.

www.acm.org/taccess
www.acm.org/subscribe



Association for
Computing Machinery

do indeed make us, or upcoming generations, more stupid, measured by the cognitive strength of brains.

An Unpleasant Surprise

All of us working in computer science have realized ICT helps us with very good access to information, research papers, and communication with colleagues, ensuring we do not end up in cul-de-sacs and are exposed to new ideas rapidly. ICT is above all a powerful and positive force in many areas like medicine, transportation, and production, just to name a few. It has also made many mundane tasks easier, such as booking a hotel, an event, or a trip. But it has also produced serious problems of privacy, of indirect control over us by others, of increased violence through violent games (as Bushman states convincingly³). And it is possibly creating new kinds of warfare, and yes, in a few aspects it has made us lazier.

Or maybe it has just relieved us of some tasks to create room for new challenges? Some superficial arguments concerning cognitive tasks might be: Why should I do complicated calculations when my smartphone has a built-in calculator? Why worry about counting change, if I pay by credit card anyway? Why bother about spelling mistakes when my spell-checker makes fewer mistakes than my teachers did? Why remember phone numbers when my smartphone has speech activation? I do not worry if I cannot find my phone at home: I use my wife's phone and the ringing of my phone makes sure I can locate it. Too bad my shoes do not ring yet, but soon NFC devices will help me find them, or anything else of interest to me, for that matter. Handwriting—what the heck! I dictate most things these days, or else use a keyboard. With the language app on my phone, I can converse on a simple level in any language of the world. I forgot where I found the app, but I am sure you will be able to locate it. Even before I used the English-Japanese language app, it was easy for me to order in a Japanese restaurant: they all have plastic replicas of the meals in the window, so I just took a picture with my digital camera and showed it to the waiter. I find it convenient when hiking that I no longer have to memorize details of a map. What does it matter that my sense of orienta-

tion may have deteriorated, my smartphone can find any place in any city or on any hike with a few taps.

In addition, there are all the benefits from using ICT in important applications like medicine, transportation, production, and so forth, as previously described.

And now I am supposed to believe all those great achievements come at the price of increasing stupidity! Does this mean we technologists will soon have to ask ourselves the same question physicists had to ask themselves in connection with nuclear weapons: Do we contribute positively to mankind or do we threaten it, because we are reducing the capacity of humans for deep logical thinking?

Do We Need to Worry?

One can argue there is no need to worry if some of our cognitive facilities are reduced due to technology, as long as the loss is overcompensated by technology and as long as we can assure two crucial points: independent and creative thinking must not be threatened and we must still be able to function in a reasonable “basic mode” if technologies fail. The difficulty is that we do not know at this point how much knowledge we can “outsource” into the Net and computers without reducing creativity. Also, our infrastructure is inadequate for a massive breakdown of the Net or the electric grid. The first issue requires serious research in neurosciences, the second research and attention by engineers in a number of disciplines to provide

The difficulty is that we do not know at this point how much knowledge we can “outsource” into the Net and computers without reducing creativity.

enough redundancy. Unfortunately, providing redundancy will have its cost, hence will meet resistance.

We must stop looking at humans as naturally, biological grown beings. Rather, we must understand ourselves as organic beings in symbiosis with technology. I myself am a good example. I am middle-ear deaf. That is, without very special hearing aids I would not hear a thing. I wear eyeglasses or I would see everything blurred. My pacemaker keeps my heart beating properly. And the metal plate in my replacement hip is perfect; well, when I go through airport security, I sometimes have to show a medical statement about that piece of metal. If you were to take away all this technology, I would be physically impaired at best, but probably dead. As is, I can hike, scuba dive, go to concerts, do research, and even make it into *Communications* once in 20 years.

In other words, we should not judge persons now and in the future without the technological tools they are using, whether those tools are built-in (pacemaker) or external (hearing aid, smartphone, reading software for visually challenged persons, tablet PCs, Google Glass). We have long accepted this for physical properties: my grandfather was strong: he could carry 50kg 20 kilometres in four hours! Well, I can do better: I can carry 250kg 200 kilometres in two hours with my car. If I were to encounter an adversary, I would still prefer it to be an unarmed body-builder rather than someone skinny with a machine gun. What is happening now is that technology is starting to also replace some cognitive functions, reducing our very own capabilities like our memory or orientation facilities. This raises an important concern: Does our increasing dependency on technology sabotage our ability to think for ourselves? Surely, looking up some facts is not the same as coherent logical thinking. Is the latter in danger? The answer is frustrating: none of us knows. As an optimist I hope we can make good use of the possibility offered to us by easy access to high-quality research and arguments and discussion with colleagues, without losing our power of thinking.

Despite my positive attitude, I am aware of the two important aspects hinted at before. The first is that if we

We must understand ourselves as organic beings in symbiosis with technology.

rely on technologies, we should also ensure we have a backup when those technologies fail. I do not think this issue has been taken seriously enough in the past: we should be more careful to have solutions if, for example, electricity, transportation systems, water supply, or other services fail for an extended period over a large area. With ICT's enormous influence on our lives and cognitive capabilities, this issue is becoming more pressing.

Secondly, repeated actions change our brain and hence how we think. As such, this is nothing new: physical work also changes how we act, since our muscles get stronger. Yet the danger that creativity is threatened because we "outsource" too much into the Net or delegate it to computer algorithms is real: we must not empty our brain or it seems likely we might lose the capability to think clearly and bring together important facts: no links in the Net can do this for us. Knowing how to navigate the Net does not make up for synapses generated in our brain. We will be using algorithms doing some job for us (like calculating something, speech translations, finding a route, and other functions). Yet it is clear some basic information and the facility to do serious logical thinking must not disappear. Whether logical thinking can be best achieved by learning mathematics, or by some other means like learning how to play chess or bridge (might be more fun) will still have to be determined. To find good answers requires more neuroscience and brain research: what capabilities do we need in our brains to remain creative? What do we lose and what do we gain, if instead of retaining all the details of one book on a particular topic, we retain a few details of many books with different views on that topic in our brain? At this point in time nobody seems to have valid an-

swers, hence this is an important and crucial research topic but more for neuroscience than computer science. When these important questions have been answered we will know whether we need a personal trainer for our minds as we need one for our bodies!

Conclusion

There is no doubt ICT has had a positive influence on many aspects of our life. Looking specifically at thinking and at the discovery of new results and doing research, it is clearly positive that we can easily access new research results and communicate worldwide as a basis for own imaginative thinking, possibly added by the ability to deal with foreign languages or getting better insights through new kinds of visualization. However, the Net (and other ICTs) will also "make us more stupid" as far as some of our cognitive capabilities are concerned. This should be accepted by us if two important points are not forgotten: that we must not be completely dependent on technology and that we must retain the capability for logical thinking and creativity. Only then can we judge the balance of what we gain against what we lose.

Before we understand the full impact of ICT on our brain and our thinking, caution is essential. Thus, major challenges for further research and the study of behavioral patterns are to find out what we can outsource and what we better retain in our own brains. 

References

1. Bauerlein, M. *The Dumbest Generation*. Tarcher, 2008.
2. Brabazon, T. *The University of Google*. Ashgate, 2007.
3. Bushman, B.J. *The Effects of Violent Video Games. Do They Affect Our Behavior?* (2012); <http://ithp.org/articles/violentvideogames.html>.
4. Carr, N. *The Shallows: What the Internet Is Doing to Our Brains*. W.W. Norton, 2010.
5. Manjoo, F. *You Won't Finish This Article*. Slate Magazine (2013); http://slate.com/articles/technology/technology/2013/06/how_people_read_online_why_you_won_t_finish_this_article.single.html.
6. Spitzer, M. *Digitale Demenz. Wie wir uns und unsere Kinder um den Verstand bringen*. Droemer Verlag [in German], 2012.
7. Thompson, C. *Smarter Than You Think. How Technology is Changing Our Minds to the Better*. Collins, 2013.
8. Weber, S. *Das Google-Copy-Paste-Syndrom*. dpunkt Verlag [in German], 2008.

Hermann Maurer (hmaurer@iicm.tu-graz.ac.at) is emeritus professor of Computer Science at Graz University of Technology, Austria, and a member of the Board of Academia Europaea.

The author thanks Keith Andrews and the anonymous reviewers for their valuable input.

Copyright held by author.

Viewpoint

What It Means to Receive the Turing Award

ACM HELD A press conference on Nov. 13, 2014 to announce Google's \$1 million funding of the ACM A.M. Turing Award (see p. 31). In attendance were seven Turing laureates: Michael Rabin, Robert Tarjan, Butler Lampson, Edmund Clarke, Joseph Sifakis, Barbara Liskov, and Silvio Micali. At the event, Micali was asked to share what it means to receive the Turing Award. Here are his comments.

I'll be brief.

As we enter life, we all struggle to understand the world. Some of us continue this struggle with *dogged* determination. These are the *scientists*. Some of them realize that computation provides a privileged perspective to understand the world outside and the one within. These are the *computer scientists*. To some of them, once a year, the ACM confers the *Turing Award*.

The end. ☺

It is wonderful that six Turing awardees could attend this event. Personally, I am particularly happy, because they are not only my role models and my mentors, but my *friends* as well. On behalf of this uniquely distinguished group, I wish to share with you my view on what it means to *receive* the Turing Award.

Let me start by saying that the Turing Award gives us a unique opportunity to become ambassadors of our wonderful field. Within our computer science community, everyone always knew they could go to Michael, Bob, Butler, Ed, Joseph, and Barbara for an idea, a suggestion, or an opinion. But the Turing Award increases our *outside*

visibility and enables us to build bridges to *other* disciplines. There is already a vibrant cooperation between computer science and mathematics, economics, quantum physics, and biology. But it is becoming clear that computation underlies many more fields, and it is a duty and a privilege for a Turing Award winner to provide an initial point of contact and to facilitate joint exploration with other fields.

My second meaning is more personal. But since the more personal we get the more universal we become, I suspect that my friends also share this meaning. As scientists, we strive not only for utmost *innovation* and but also for broadest *recognition*. Strife for

innovation is certainly good, but so is strife for recognition: the world would be *extremely* dangerous if scientists no longer craved social approval! Innovation and recognition, however, can be antagonistic forces. The safest way to gather broad approval is to advance an *already accepted* research direction. It is thus a crucial role of the Turing Award to strike a better balance between these two innate forces, and give a louder voice to "the better angels of our nature." Each and every one of my friends here has earned the Turing Award by going against the then prevailing wisdom, by treading where everyone else feared to go, by rolling a bright red carpet on what used to be



Turing laureates at the ACM press conference (front row, from left): Silvio Micali (2012), Michael O. Rabin (1976), Edmund M. Clarke (2007). Back row, from left: Joseph Sifakis (2007), Barbara Liskov (2008), Butler W. Lampson (1992), Robert Tarjan (1986).

a dangerous pass. Since March 2013, by satisfying my need for recognition (temporarily!), the Turing Award has spurred me to take on additional scientific risk. A few months ago, my colleague Nir Shavit proposed that we have a look at computational neurobiology. I was two-minded about his proposal. On one hand, I knew *absolutely nothing* about neurobiology. On the other, I *absolutely knew* that of course computation does help it. (a) Risky, (b) *Tempting*... What might I have answered just two years before? Who knows? I answered Yes. I am glad I did.

Finally, after the interdisciplinary and the personal, I wish to point out the third and most important significance of receiving the Turing Award: *encouraging our own discipline*. The impact of computer science on human history may only be comparable to that of the “invention of fire.” Instantaneous communication and our ability of analyzing and sharing *enormous* amounts of data are all children of the computer. Our cars, jet planes, and all our sophisticated appliances are sophisticated *because* they have computers on board. Science is more and more *computation-based*. Our generation did well. But the next one is called to do *much better*. For, it is no longer the point of revolutionizing and improving the way we live: our very *survival* as a species will depend on our ability to advance computation. To each generation its own challenge. Fortunately, we are surrounded by enormously talented young people. Yet, the tasks before them are so daunting that they might be tempted to unleash their tremendous potential onto lesser and easier targets. It is our responsibility to encourage them to take the *harder* road; to reassure them they have the *hearts* and the *minds* to prevail.

So, after this wonderful celebration, let's go home and encourage our young scientists. Let's tell them *you are NOT alone. WE are with you. THE ENTIRE FIELD is with you. YOU WILL WIN. And when you do, wherever we might be, we shall share your pleasure.*

Long Live Computer Science! 

Silvio Micali (silvio@csail.mit.edu) is the Ford Professor of Engineering at MIT, Cambridge, MA. He is the co-recipient (along with Shafi Goldwasser) of the 2012 ACM A.M. Turing Award.

Copyright held by author.



"The Turing Award gives us a unique opportunity to become ambassadors of our wonderful field," said Silvio Micali at the ACM press conference on November 13.



Seven Turing laureates participated in ACM's announcement of Google's \$1 million funding of the Turing Award.



New York Times reporter Steve Lohr (left) with Turing laureates Butler Lampson and Barbara Liskov.

Use the database built for your access model.**BY RICK RICHARDSON**

Disambiguating Databases

THE TOPIC OF data storage is one that does not need to be well understood until something goes wrong (data disappears) or something goes really right (too many customers). Because databases can be treated as black boxes with an API, their inner workings are often overlooked. They are often treated as magic things that just take data when offered and supply it when asked. Since these two operations are the only understood activities of the technology, they are often the only features presented when comparing different technologies.

Benchmarks are often provided in *operations* per second, but what exactly is an operation? Within the realm of databases, this could mean any number of things. Is that operation a transaction? Is it an indexing of data? A retrieval from an index? Does it store the data to a durable medium such as a hard disk, or does it beam it by laser toward Alpha Centauri?

It is this ambiguity that causes havoc in the software industry. Misunderstanding the features and guarantees of a database system can cause, at best, user consternation due to slowness or unavailability. At worst, it could result in fiscal damage—or even jail time due to data loss.

The scope of the term *database* is vast. Technically speaking, anything that stores data for later retrieval is a database. Even by that broad definition, there is functionality that is common to most databases. This article enumerates those features at a high level. The intent is to provide readers with a toolset with which they might evaluate databases on their relative merits. Because the topics cannot be covered here in the detail they deserve, references to additional reading have been included. These topics may be the subjects for future articles.

This feature-driven approach should allow readers to assess their own needs and to compare technologies by pairing up like features. When viewed through this lens, comparative benchmarks are valid only on data bases that are performing equal work and providing the same guarantees.

Before digging into the features of databases, let's discuss why you would not just take all of the features. The short answer is that each feature typically comes with a performance cost, if not a complexity cost.

Most of the functions performed by a database, as well as the algorithms that implement them, are built to work around the performance bottleneck that is the hard disk. If you have a requirement that your data (and meta data) be durable, then you must pay this penalty one way or another.

The Hard Disk

The serial ATA (SATA) bus of a typical server (Ivy Bridge Architecture) has a theoretical maximum bandwidth of 750MB per second. That seems high, but compare that with the PCI 3.0 bus, which has a maximum of 40GB per second, or the memory bus, which can do 14.9GB per second per channel (with at least four channels). The SATA bus has the lowest-bandwidth data path within a modern server (excluding peripherals).⁵

In addition to the bandwidth bottleneck, there is latency to consider. The highest-latency operation encountered within a data center is a seek to a ran-



dom location on a hard disk. At present, a 7200RPM disk has a seek time of about four milliseconds. That means it can find and read new locations on disk about 250 times a second. If a server application relies on finding something on disk at every request, it will be capped at 250 requests per second per disk.⁴

Once a location has been found, successive append-to or read-from operations at that same location are significantly cheaper. This is called a sequential read or write. Algorithms regarding data storage and retrieval have been optimized against this fact since magnetic rotating disks were invented. Typically, people refer to file operations as either random or sequential, with the understanding the latter comes at a far lower cost than the former.

Solid-state drives (SSDs) have brought massive latency and throughput improvements to disks. A seek on an SSD is about 60 times faster than a hard disk. SSDs bring their own challenges, however. For example, the storage cells within an SSD have a fixed lifetime—that is, they can handle only so many writes to them before they fail. For this reason, they have specialized firmware that spreads writes around the disk, garbage collects, and performs other bookkeeping operations. Thus, they have fewer predictable performance characteristics (though they are predictably faster than hard disks).

The Page Cache

Because of the high latency and low throughput of hard drives, one optimization found in nearly every operating system is the page cache, or buffer cache. As its name implies, the page cache is meant to transparently optimize away the cost of disk access by storing contents of files in memory pages mapped to the disk by the operating system's kernel. The idea is the same local parts of a disk or a file will be read or written many times in a short period of time. This is usually true for databases.

When a read occurs, if the contents of the page-cache are synchronized with the disk, it will return that content from memory. Conversely, a write will modify the contents of the cache, but

not necessarily write to the hard disk itself. This is to eliminate as many disk accesses as possible.

Assuming that writing a record of data takes five milliseconds, and you have to write 20 different records to disk, performing these operations in the page cache and then flushing to disk would cost only a single disk access, rather than 20. Considering that accessing main memory on a machine is about 40,000 times faster than finding data on disk, the performance savings add up quickly.

Every operating system has a different model for how it flushes its changes to disk, but almost all work with the scheduler to find appropriate points to silently sync the data in memory onto disk. Files and pages can also be manually flushed to disk. This is useful when you need to guarantee that data changes are made permanent.⁸

Be aware the page cache is a significant source of optimization, but it can also be a source of danger. If writes to the page cache are not flushed to disk, and a power, disk, or kernel failure occurs, you will lose your data. Be mindful of this when analyzing database solutions that leverage the page cache exclusively for their durability operations.

Database Features

Databases have dozens of classifications. Each of the hundreds of commercially or freely available database systems likely fall into several of these classes. This article skips past the classifications and instead provides a framework through which each database can be evaluated by its features.

The five categories of features explored here are: data model, API, transactions, persistence, and indexing.

Data model. There are fundamentally three categories of data models: relational, key value, and hierarchical. Most database systems fall distinctly into one camp but might offer features of the other two.

Relational model. Relational databases have enjoyed popularity in recent history. Throughout the 1980s and 1990s, the chief requirement of databases was to conserve a rare and expensive resource: the hard disk. This is where relational databases shine. They allow a database designer to minimize data

duplication within a database through a process called normalization.⁷

Lately, however, the cost of disk storage has fallen considerably,³ making the economic factor of relational databases less relevant. Despite this, they are still widely used today because of their flexibility and well-understood models. Also, SQL—the lingua franca of relational databases—is commonly known among programmers.

Relational databases work by allowing the creation of arbitrary tables, which organize data into a collection of columns. Each row of the table contains a field from each column. It is customary to organize data into logically separate tables, then relate those tables to one another. This allows constituent parts of a greater whole to be modified independently.

A major downside of relational databases is their storage models do not lend themselves well to storing or retrieving huge amounts of data. Query operations against relational tables typically require accessing multiple indexes and joining and sorting result vectors from multiple tables. These sophisticated schemes work well for 1GB of data but not so well for 1TB of data.

The fundamental trade-off a relational database makes is saving disk space in return for greater CPU and disk load.

The benefits of this model are many: it uses the lowest amount of disk space; it is a well-understood model and query language; it can support a wide variety of use cases; it has schema-enforced data consistency.

The downsides of this model are that it is typically the slowest; its schemas mean a higher programmer overhead for iterating changes; and it has a high degree of complexity with many tuning knobs.

Key-value model. Key-value stores have been around since the beginning of persistent storage. They are used when the complexity and overhead of relational systems are not required. Because of their simplicity, efficient storage models, and low runtime overhead, they can usually manage orders-of-magnitude more operations per second than relational databases. Lately, they are being used as event-log collectors. Also, because of their simplicity,

they are often embedded into applications as internal data stores.

Key-value stores operate by associating a key (typically a chunk of bytes) to a value (typically another chunk of bytes). Also, because records are often homogeneous in size and have replicated data, they can be heavily compressed before being stored on disk. This can drastically reduce the bandwidth required across the SATA bus, which can provide performance gains.

Through clever row and column creation, and even schema application, some key-value stores can offer a subset of relational features, but they typically offer far fewer features for data modeling than a relational system. If multiple indexes are needed, they are simulated by using additional key-value lookups.

This is a fast, fairly flexible, and easily understood storage model; however, it often has no schema support, so no consistency checks, and its application logic is more complicated.

Hierarchical model. The hierarchical, or document data, model has achieved popularity relatively recently. Its major advantage is ergonomics. The data is stored and retrieved from the database in the way it is stored within objects in an application.

The hierarchical model tends to store all relevant data in a single record, which has delineations for multiple keys and values, where the values could be additional associations of keys and values.

In the general case, all of the data of a real-world object is found within a single record. This means it will necessarily use more storage space than the relational model because it is replicating the data instead of referencing it. It also simplifies the query model since only a single record needs to be retrieved from a single table.

Because the data being stored is heterogeneous in nature, compression can provide limited gains and is typically not used.

Hierarchical databases typically offer some relational features, such as foreign references and multiple indexes. Many such databases do not offer any schema support, as the data structure is arbitrary.

This is the most flexible model. Its arbitrary indexes support easy access



Because of the high latency and low throughput of hard drives, one optimization found in nearly every operating system is the page cache, or buffer cache.



to data and it has the highest fidelity between application data structures and on-disk data structures.

On the downside, this model has the highest disk-space usage; and without a schema, data layout is arbitrary, so there are no schema or consistency checks.

API

The application programming interface (API) is, in short, how you and your program interact with a database. The interface can be diced in many different dimensions, but let's start with two:

In process vs. out of process. If the database is running in the same process (at least partially) as the client application, then typically there is a library of function calls that invoke methods in the database engine directly. This tight coupling results in the lowest possible latency and highest possible bandwidth (memory). It reduces flexibility, however, since it means only a single client application can access the data at one time. It also poses an additional risk: if the client application crashes, so does the database, since they share the same process.

If the database runs in a separate process, a protocol over TCP/IP is typically used. Many RDBMSs (relational database management systems), and recently, other types of databases, support either the ODBC (open database connectivity) or JDBC (Java database connectivity) protocols. This simplifies the creation of client applications, as the libraries that can leverage these protocols are plentiful. A network protocol does drastically improve flexibility of a database, but TCP carries with it latency and bandwidth penalties versus direct memory access.

SQL vs. not. SQL is a declarative language that was designed originally as a mechanism to simplify storage and retrieval of relational data. Its usage is ubiquitous, and as such, many developers speak the language fluently. This can aid the adoption of a database.

The biggest “innovation” touted by most NoSQL databases was simply achieving faster operations by removing transactions and relational tables. Many of those databases began to support SQL as an API language, even though they did not use its relational

features. Some SQL features such as querying, filtering, and aggregating were quite useful. Therefore, it was said that NoSQL databases should be renamed NoACID (atomicity, consistency, isolation, and durability) because of their lack of transaction support. In 2015, many of those same databases now have transactional support. These days, NoSQL might be more accurately called NoRelational, but NoSQL sounds better and is close enough.

One challenge of SQL is it must be parsed and compiled by the database engine in order to be used. This imposes a runtime overhead. Most database engines or client APIs work around this by precompiling, or compiling on the first run, the SQL-based function calls into prepared statements. Then the compiled version is saved and used for future calls.

SQL cannot effectively describe all data relationships. For example, hierarchical relationships are difficult to describe in SQL. In addition, because of SQL's declarative nature, iterations or other imperative operations are not describable in the core SQL specification. The specification has been expanded to include recursion to address both iteration and hierarchical relationships. In addition, vendors have provided nonstandard extensions of their own. Support for these extensions is not widely prevalent, however, and neither is the understanding of how to leverage them.⁹

In many cases the features of databases are so sparse, lacking features such as indexing or aggregation, there is simply no reason to support the complexity of a SQL parsing and execution engine. Key-value stores often fall into this category.

Transactions

A database transaction, by definition, is a unit of work treated in a coherent and reliable way. The most common recipe for database transactions is ACID. Many database systems claim support for transactions or “light-weight” transactions, but they may provide only the features of ACID that are convenient and efficient to support. For example, many distributed databases offer the concept of transactions without the isolation step. This means the data is being modified

in place, and other transactions see that data while it is being modified. You can work around this if this behavior is expected. If not, the results could be disastrous.

Let's briefly look at the ACID guarantees, and then what a database might do to provide them.

Atomicity. Within a transaction, there could be multiple operations. Atomicity guarantees all operations will either succeed or fail together. An operation could fail for a number of reasons:

- **Constraints.** A logical constraint is violated, such as foreign keys or uniqueness.

- **Concurrency.** Another process completes modification of a field that your process was going to modify, and to continue doing so would violate the atomicity guarantee of their transaction.

- **Failure.** Something in the hardware or software stack fails, causing one of the operations to fail.

In a busy, concurrent database, failures can happen often. Without atomicity, data can get into an inconsistent state very quickly. Thus, atomicity is a key component of the next property of ACID.

Consistency. This guarantee means the state of the database will be valid to all users before, during, and after the transaction. Databases may make certain guarantees about the data itself. Basic guarantees such as serializability mean all operations will be processed in the order in which they are applied. This might sound easy, but when many applications with many threads are operating on a system concurrently, (expensive) steps must be taken to ensure this is possible.

Relational databases often make an even larger set of consistency guarantees, including foreign-key constraints, cascading operations on dependent types, or triggers that might be executed as part of this operation. In terms of performance, this means all of these operations might be running while rows and/or pages are locked for editing, so no other clients will be able to use those parts of the system during that time. It also, clearly affects the round-trip time of the request.

Isolation. Transactions do not happen immediately. They occur in steps, and, as in the atomicity example, if

an outsider were to see a partial set of completed steps, results would range from “amusing” to “horribly wrong.” Isolation is the guarantee that says this will not happen. It hides all of the operations from others until the transaction completes successfully.

Durability. An important trait indeed, durability simply promises that when the transaction completes the results of the operations will be successfully persisted on the specified storage medium (typically the hard disk).

Implementation of transactions. Six steps are common to ACID transactions:

1. Log the incoming request to persistent storage in a transaction log (also known as a write-ahead log). This will protect the data in case of a system failure. In the worst-case scenario, this transaction will be able to be restarted from the log upon startup.

2. Serialize the new values to the index and table data structures in a way that does not interfere with existing operations.

3. Obtain write locks on all cells that need to be modified. Depending on the operation in question and the database, this might mean locking the entire table, the row, or possibly the memory page.

4. Move the new values into place.

5. Flush all changes to disk.

6. Record the transaction as completed in the transaction log.

Transactions have performance implications. They can lead to speed-ups over performing the operations piecemeal, since all of the disk operations are batched into a single set of operations. Also, if ACID, transactions are a form of concurrency control. Since they sit at the data itself, they can often be more efficient than custom-built concurrency solutions in the application itself.

On the downside, transactions are not good for highly concurrent applications. Highly contentious operations will generate excessive replays and aborts (which result in more replays). They are also complex—all of the moving parts required to provide transactions add to larger and less maintainable code bases.

Persistence Models

As previously stated, transactions and even indexing are completely optional

within databases. Persistence, however, is their *raison d'être*.

The performance costs associated with disks (and the risk of data loss associated with the page cache) mean trade-offs with respect to how data is stored and retrieved. A multitude of highly specialized data structures are tailored to different access models, and, typically, if a data structure excels in one area, it will perform poorly in another area. A scheme for inserting large amounts of incoming events in a sequential manner will likely not offer great performance for random updates (or may not even offer that capability at all).

Across all of the potential schemes for storage and retrieval of data, four of the broadest categories are: row based, columnar, memory only, and distributed.

Row based. The most common storage scheme is to store data, row by row, in a tree or some other compact data structure on a local hard disk. Although the exact data structures and access models vary, this mechanism is fairly universal.

In row-based storage, the rows themselves are contiguous in memory. This usually means the storage model itself is optimized for fetching regions of entire rows of data at one time.

There are two common data structures for storing rows. The B+ tree is optimized for random retrieval, and the log-structured merge (LSM) tree is optimized for high-volume sequential writes.

B+ tree. A B+ tree is a B-tree-style index data structure optimized for, you guessed it, minimizing disk seeks. It is one of the most common storage mechanisms in databases for table storage. It is also the data structure of choice for almost all modern file systems. The B+ tree is typically a search tree with a high branching factor, and each node is a contiguous chunk of memory containing more than one key. This is specifically designed to maximize the probability that multiple keys can be compared with only a single retrieval of data from disk.¹

Figure 1 shows how B-tree-based row storage is laid out in memory. Each leaf node has space for four keys, reducing the amount of disk lookups that need to be executed per row. The

key in the tree points to a region on disk or memory that stores the row, which is arranged serially by column. Also note that at each node, not every cell needs to be filled; they can remain free for future values.

Log structured. The LSM tree is a newer disk-storage structure optimized for a high volume of sequential writes. It is designed to handle massive amounts of streaming events, such as for receiving Web-server access logs in real time for later analysis.

Despite its origins in log-style event collection, the LSM tree is beginning

to be used in relational databases as well. It has a major trade-off, however, in that you cannot delete or update in an LSM data structure as part of the standard data path. Such events are recorded as new records in the log. When reading an LSM tree, you typically start from the back to read the newest version of the data.

Periodically, the records that have been made obsolete by subsequent deletes or updates must be garbage collected. This is typically called a compaction process. Some LSM systems compact in separate threads at run-

Figure 1. The layout of a B-tree-based row storage in memory.

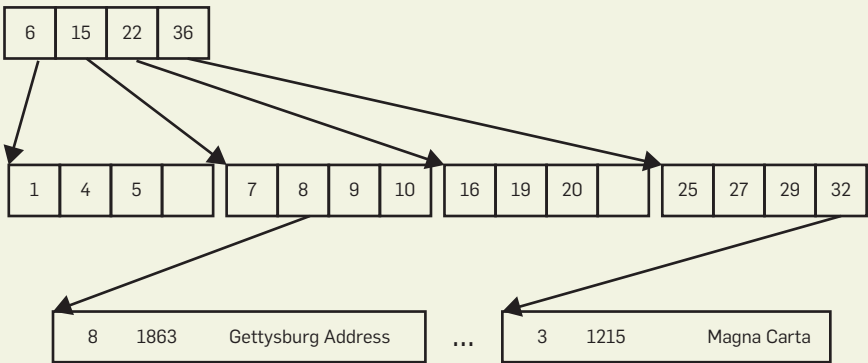
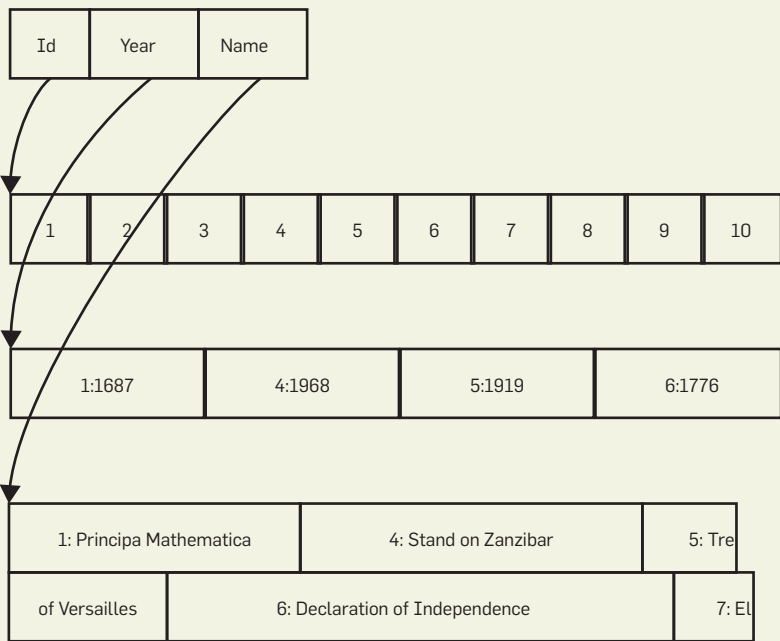


Figure 2. The layout of a columnar data file.



time; other systems attempt to incrementally compact in place.⁶

Columnar. Column-based data stores optimize for retrieving regions of the same column of data, rather than rows of data. For this reason, successive columns are stored contiguously in memory.

Because all data types in a column are necessarily the same, compression can have a huge positive impact, thus increasing the amount of data that can be stored and retrieved over the bus. Also, breaking up the data into multiple files, one per column, can take advantage of parallel reads and writes across multiple disks simultaneously.

The downside of column-based databases is they are often inflexible. A simple insert or update requires a significant amount of coordination and calculation. Because data is so typically tightly packed (and compressed) in columns, it is not easy to find and update the data in place.

To help keep “rows” in sync across column files, many times a column field will also contain a copy of the primary key (or row ID, if there are no keys). This aids in reassembly of the data into rows, but it reduces the efficiency of storage and retrieval.²

Figure 2 shows a columnar data file. Each data type is laid out in its own contiguous region, whose offset is indicated in the master column. Column files are typically built and rebuilt in batches to serve data-warehousing applications for massive datasets.

Memory only. In many cases, durability is simply not a requirement. This is common for systems such as caches, which update frequently and are optimized for nothing other than access speed. Because data in caches is typically short lived, one may not need to persist to disk. This is where in-memory databases shine. Without the requirement to store and retrieve from the disk, a much wider variety of sophisticated trees can be leveraged.

Distributed. The topic of distributed databases is vast and would require its own series of articles for proper coverage. In the context of persistence, however, there is one relevant fact: it is faster to copy a dataset across the network within a data center than it is to store it onto a local disk.

Distributed databases can provide an interesting option when establish-



All of the data of a real-world object is generally found within a single record. This means it will necessarily use more storage space than the relational model, because it is replicating the data instead of referencing it.



ing the balance between speed and persistence. It might be too risky to store data only in memory on a local machine, as data loss would be complete if the machine crashed. If you copy the data across many machines, then your risk of total data loss is reduced. It is up to the application developer to determine the probability of machine failure and the level of acceptable risk.

Page-Cache Considerations

When you opt to store to disk, the page cache is likely to be involved. Accessing the disk directly would be far too cumbersome and would hamper the performance of any other applications running on the system, as you would be monopolizing the disks unnecessarily.

The question of when to flush from the page cache to disk is perhaps the most important of all when designing a database, as it tells you exactly how much risk you have for data loss.

Many databases purport many thousands of operations per second. These databases often operate entirely on data structures on memory-mapped pages in the page cache. That is how they achieve their speed and throughput—by working on in-memory data structures. They do this by deferring all flushing and syncing operations to the operating system itself. This means it is up to the kernel to decide when data in the cache should be persisted to disk. It will likely take into account not only the database, but also all applications running on the system. This means the actual persistence of the data is being left to the operating system, which does not understand the application domain or data reliability requirements.

If durability is not a strong requirement, then deferring to the operating system is probably fine. In most cases, though, it is important to be clear about the behavior of the database with respect to the page cache.

For systems that sync automatically:

- ▶ If it flushes too frequently, it will have poor performance.
- ▶ If it flushes infrequently, it will be faster, but there is a risk of data loss.

A better approach might be to leverage a manual syncing scheme for the database, since that will provide the control to match the guarantees an application requires. This increases the complexity of applications. For highly

concurrent systems, the difficulty level increases, as a disk operation serving one application might interfere excessively with another application.

Systems such as transactions and batch operations that sync at the end can be beneficial. This will reduce the number of disk accesses, but there is a very clear guarantee as to when the data is flushed to disk.

Indexing

Data is rarely stored as an isolated value. It is typically a heterogeneous collection of fields that make up a record. In relational databases, those fields are called columns, and they are fixed to the schema that defines the tables.

In nonrelational databases, heterogeneous fields are still often accommodated and even indexed. When you want to look up a table by specifying one of the fields in a record, that field needs to be part of an index. An index is just a data structure for performing random lookups, given a specified field (or, where supported, a tuple of specified fields).

To tree, or not to tree. Since a B-tree is an on-disk data structure as well, it is the tool of choice for most lookup indexes, since it efficiently supports hard disks. Unlike the B+ tree that stores data, a lookup index is optimized for storing references to data. A B-tree can accommodate inserts efficiently without having to allocate storage cells for each operation. It also tends to be flat in structure, reducing the number of nodes that need to be searched, therefore reducing the number of potential disk seeks.

There are other options, however. For example, a bitmap index is a data structure that provides efficient join queries of multiple tables.

A tree-style index grows linearly for the number of items in the index, and search time grows with the depth of the tree (a logarithmic function of the total depth).

A bitmap index, on the other hand, grows with the number of different items in the index. As the name implies, it builds a bitmap that represents the membership of values for all relevant columns. Multiple Boolean operations against bitmap indexes are very fast, and they produce new bitmaps that can be cached efficiently as search results.

One of the other major innovations of the bitmap index is it can be compressed, and it can even perform query operations while compressed. This makes storage retrieval faster. It also makes the bitmap index more CPU cache friendly, which can further reduce latencies. Because of its more complicated update process, the bitmap index tends to be used in read-heavy systems, especially those with multidimensional queries such as OLAP (online analytical processing) cubes.

Indexing performance summary. Unless your only data-access model is a full scan of large regions of data, you will probably need indexes. Every additional index that your dataset leverages, however, will add increased disk and CPU load, as well as increasing the latency of inserts.

If your system is read-heavy, and it possesses a relatively low variety of data in the columns (known as low cardinality), you can take advantage of bitmap indexes. For everything else, there are tree indexes.

Many indexes require a unique key to point to a record. If it is the only unique index for that record, it is referred to as the primary key. Even schema-less databases often support such an indexing constraint. They can help ensure consistency and detect errors when loading data.

For performance, there is one basic rule to follow: index as little as possible. Almost every database that supports adding indexes will allow you to add them after the data is loaded. So add them later, once you are sure you need them. Using a unique index can provide a double benefit of ensuring data consistency.

If all of your data is loaded at once (in a data-mart model, perhaps), you might benefit from creating indexes afterward. This can even result in a more efficient index, as many indexes suffer negative effects from fragmentation caused by many inserts and updates.

Pulling It All Together

The trade-offs between performance and safety revolve around the disk. You might just get the best of both worlds if you choose the database that is built exactly for your access model. Take the time to understand your access model

thoroughly and to know which features you require and which you are willing to forgo in the name of performance.

If you do not need guaranteed and immediate durability for every operation, you can delay persisting the operation to disk by leveraging a memory-mapped data structure. Understand the risk of data loss is present any time you rely on memory to speed things up. If a failure occurs, those pending writes can disappear.

Regardless of your application, take time to understand the page cache in your operating system. Writes that you think are safe may not be. It, too, has many settings for fine-tuning performance. It can be set to be highly paranoid but busy, or carefree and fast. You should be very clear about how and when your database writes to disk. If it defers to the operating system, then take steps to ensure it behaves correctly for your use case.

It is worth verifying that your expectations match reality. It may just save your data. 

Related articles on queue.acm.org

Bridging the Object-Relational Divide

Craig Russell

<http://queue.acm.org/detail.cfm?id=1394139>

Sentient Data Access via a Diverse Society of Devices

George W. Fitzmaurice et al.

<http://queue.acm.org/detail.cfm?id=966721>

Distributed Computing Economics

Jim Gray

<http://queue.acm.org/detail.cfm?id=1394131>

References

1. B-trees; http://www.scholarpedia.org/article/B-tree_and_UB-tree.
2. Column-oriented databases; http://en.wikipedia.org/wiki/Column-oriented_DBMS.
3. Hard-drive costs; <http://www.mkomo.com/cost-per-gigabyte-update>.
4. Latency numbers; http://www.eecs.berkeley.edu/~rcs/research/interactive_latency.html.
5. LGA; http://en.wikipedia.org/wiki/LGA_2011.
6. LSM trees; <http://dl.acm.org/citation.cfm?id=230826>; and <http://www.eecs.harvard.edu/~margo/cs165/papers/gp-lsm.pdf>.
7. Normalization; http://en.wikipedia.org/wiki/Database_normalization.
8. Page cache; <http://www.westnet.com/~gsmith/content/linux-pdflush.htm>.
9. SQL; http://en.wikipedia.org/wiki/Hierarchical_and_recursive_queries_in_SQL.

Rick Richardson is a systems architect for 12Sided Technology where he is helping to reinvent market structure and forge the next generation of trading systems for the financial world.

Copyright held by author. Publication rights licensed to ACM, \$15.00.

 Article development led by [acmqueue](http://acmqueue.queue.acm.org)
queue.acm.org

Trust, but verify.

BY GEETANJALI SAMPEMANE

Internal Access Controls

EVERY DAY SEEMS to bring news of another dramatic and high-profile security incident, whether it is the discovery of longstanding vulnerabilities in widely used software such as OpenSSL or Bash, or celebrity photographs stolen and publicized. There seems to be an infinite supply of zero-day vulnerabilities and powerful state-sponsored attackers. In the face of such threats, is it even worth trying to protect your systems and data? What can systems security designers and administrators do?

While these threats are very real, they are not the biggest ones faced by most organizations. Most organizations do not face targeted attacks from hostile governments or criminals intent on stealing users' data; their systems are more likely to be unavailable

because of ill-timed software updates or misconfiguration.²⁻⁴

People tend to overreact to dramatic events like terrorist attacks, but they underestimate mundane threats. This is made worse by the fact the threat landscape is evolving; security advice that was once reasonable becomes obsolete. For example, users are routinely advised to use long, complex passwords, but account compromise caused by password reuse is probably a bigger threat these days than brute-force password cracking, so choosing different passwords for different sites is a better strategy than creating a complex password, memorizing it, and using it everywhere.

In a former life, I helped organizations connect to the Internet, and, as part of that process, warned administrators of new threats they now faced. Those conversations convinced me that practical systems security was still too difficult for most people to get right. In the years since, Internet connectivity has become more routine, but methods for securing systems have not kept pace.

This article argues in favor of relatively mundane tools that systems security designers and administrators can use to protect their systems and detect attacks. The principles proposed here are good *internal* access controls: regular automated monitoring and verifying of access configurations, and auditing user access to data. At Google, we use these techniques as part of our security strategy, but the principles are applicable to any organization with data to protect.

The Problem

Systems security administrators, who have more incentive than the average user to get security right, have a difficult job. With the increasing proliferation of mobile devices, and increased expectation of anytime/anywhere access, there are only a few high-security environments where users can be prohibited from bringing their personal phones or devices into the corporate environ-



ment. Keyboard loggers and malware on personal machines can thus be a path to attack enterprise systems. These devices can be used to exfiltrate data, deliberately or accidentally.

Even when users are restricted to using corporate-owned and -managed devices for work, they still tend to reuse passwords on different systems, and this can provide a vector of attack. Stashes of username/passwords stolen from compromised servers can be retried on other sites, so users who have reused a username/password on multiple sites can contribute to a bigger problem. People remain vulnerable to social engineering or phishing attacks. Improved authentication systems, such as having a second factor or one-time passwords, help some, but the vast majority of systems do not use those yet.

It is therefore reasonable to assume that some user accounts will get com-

promised, and it is important to design a system to be resilient to that. Such a system also offers the benefit of providing some protection against malicious insiders. Insider attacks have the potential to cause great damage, since people cause them with authorized access and, often, knowledge of systems and processes. Designing protections against insider attacks, however, can be difficult without making the system very cumbersome to use or making users feel untrusted and, therefore, uncooperative with security measures.

Users of the system often do not understand the threat models, so they end up viewing security measures as hoops they have to jump through. Better explanations of the rationale for restrictions may make users more cooperative and dissuade them from looking for ways around the hoops.

Another common problem is mis-

configured security controls. As sys-

tems and security software grow more complex, the chance of administrators misunderstanding them increases. This can lead to an increase in successful attacks based on such flaws as overlooked default passwords or misconfigured firewall rules.

Why Have Internal Access Controls?

The case for good internal access controls, also called defense in depth, is easy to understand but surprisingly difficult to get right in practice. Internal access controls make it harder for attackers to break in (it is not just the firewall that needs to be breached) and limits damage if a system is attacked (one phished password will at most get the attackers what that user has access to, not necessarily everything on the internal network). Given that a common way systems are attacked is via compromised legitimate user accounts, limit-

ing the damage a single compromised (or malicious) user can get away with undetected is a useful goal.

The problem is that systems typically start out small, with little or no valuable data, and internal access controls seem like overkill. A good firewall and unrestricted access to (the small number of) authorized users seems like more than enough. People get used to that unrestricted internal access, and processes and tools are developed under that assumption, so adding internal security barriers as the system grows can be disruptive and meet with resistance from users. Removing permissions can also break systems, often in unexpected ways. Retrofitting security into systems is difficult.

Most organizations have different kinds of valuable information that needs protecting—company-confidential code and documents, customer information, or data entrusted to them by their users (in the case of cloud service providers). Different employees need access to different subsets of this information, either for development and debugging services, or to provide customer service, or for routine activities such as indexing or backup. How does the organization ensure people have the right level of access they need and no more?

Achieving the Right Granularity of Permissions

Administrative usability is often overlooked while designing access schemes. Very fine-grained permissions seem like a good idea, since they can grant exactly the necessary access, but it can easily become too much work to manage. Too many or too low-level permissions can also result in clutter and can be difficult to understand and reason about.

On the other hand, the problem with access that is too coarse-grained is it can grant too much access. One of the bigger problems with granting too much access is not malicious use but accidental use. Many systems do not enable permissions on an as-needed basis but, rather, have all the permissions a user is granted; this is the equivalent of always running as a superuser rather than as a regular user. Again the problem is one of granularity—having to specify every permission needed be-



Most organizations have different kinds of valuable information that needs protecting—company-confidential code and documents, customer information, or data entrusted to them by their users.



comes tedious, so the tendency is just to leave permissions enabled.

Role-based access control systems¹ help with this by grouping related sets of permissions, but people who perform different roles still end up with a lot of access and not-always great ways of using the least-privileged access possible.

What can be done about this? Try to understand the system well enough to set up access controls at the right places, but also recognize that you will sometimes get this wrong and will grant more or less access than is needed. This may be because you want to simplify administration or because your mental model of permissions and usage is wrong. It is thus useful to have a system in place to review and monitor permissions, and correct the access configuration as appropriate.

Monitoring Access Configurations

Too often, access requests are reviewed at grant time and never again. People in an organization move across roles and projects, but old permissions do not always expire. Removing unused permissions rarely seems that urgent, and guessing wrong about whether something is unused can break running systems. Unused permissions are not dangerous as long as they remain unused, but they do make the access configuration more difficult to understand.

At Google, we use regular monitoring of access configurations to identify unexpected or unwanted permission behavior. The principle of access-configuration monitoring is much like unit testing for code. Like any type of verification, this is most useful if the verification uses a different approach from the configuration—for example, viewing the permissions in the live production configuration rather than just viewing them as configured.

Administrators specify invariants about the access configuration that should be maintained, and automated test infrastructure periodically verifies these invariants hold. Preconfigured alerts can be raised if any problems are detected.

Access-configuration monitoring is useful for a few different purposes:

- *Catching differences between static and live configurations.* Some access

systems require configuration changes to be reviewed by administrators and then “pushed” to take effect. Occasionally, changes are pushed to live systems without changing the static configuration, or the configuration is changed and not pushed. This sort of situation can lead to unpleasant surprises when long-running systems are restarted.

► *Verifying the configuration is behaving as expected.* Most configuration languages have their quirks, so it is good to have tests to confirm they are doing what you expect them to do. A common example is firewall rules that block too much or too little traffic.

► *Tripwire-like monitoring to notify people of changes.* Typically, these are expected changes, but this can catch unauthorized or unexpected changes. It is important that these not be too noisy, or people who receive them will tune them out.

► *Catching drifts such as sudden (or even gradual) increases in the number of authorized people.* People often create an ACL (access-control list) for a particular reason, and, over time, tend to use it for other reasons, and the size grows. This sort of monitoring can be useful for recognizing when a group has grown too large, contains too many permissions, and should be split.

► *Verifying that separation of permissions holds.* For example, you may want to prevent any one person from having certain combinations of permissions (like being able to make changes to code and push them to production without review).

Auditing to Understand Access

Audit logs are a common part of systems security. Typically, all configuration changes and any access to sensitive data generate audit logs, which are hard to subvert. These are often a requirement for regulatory compliance.

Many systems, however, stop at generating the audit logs, using them only for postmortem analysis when something goes wrong. An “audit” in these systems is a sign of trouble. Therefore, access audits should be much more routine, and not a hostile process. Whenever an employee performs a nonroutine access, perhaps for troubleshooting or debugging, the access will be audited. In most cases, this may involve just documenting

the reason for access. This develops a culture of accountability, where users expect to have to justify access to sensitive data.

Knowing that all accesses are audited makes granting permissions a little easier. Restricting access to very few people can make a system fragile. It would be more robust if more people were granted emergency access but did not have to use it. Having overbroad permissions, however, is generally a problem. Users could accidentally or maliciously misuse their accesses or become targets for social-engineering attacks because of it. Having good audit logs at the time of use of permissions mitigates this risk somewhat, since inappropriate access is unlikely to go undetected.

Routine access audits also help identify access patterns and can help tune access configuration. If all access is logged, it becomes possible to identify unused permissions reliably and prune them safely if needed. This catches the cases where people move jobs or roles without explicitly giving up permissions.

Auditing accesses that are actually used provides visibility into which accesses are needed for people to do their jobs. This allows for the development of better tools, sometimes reducing the amount of access that needs to be granted for a particular task.

Good tools are needed to prevent access audits from becoming bureaucratic nightmares. Routine access can be recognized, based on job roles or access history, and only unusual access patterns can be flagged for extra or manual review.

It is also worth noting that auditing accesses is not a substitute for good access controls; audits can recognize inappropriate access only after it has happened, unlike access controls, which prevent it. As just described, however, auditing all accesses can help tune access configurations. Having to justify access also helps prevent inappropriate access by authorized users. Further, in the unfortunate event of inappropriate access, audit logs can help administrators assess the damage.

Conclusion

While high-profile targeted attacks will continue, organizations can do a lot

to protect their systems. Internal access controls at the right granularity, combined with access logging and auditing, can help detect and prevent unwanted access. Access configurations suffer from “bit rot,” and users often accumulate unnecessary permissions over time; therefore, regular monitoring, a la unit tests for code, can help detect unwanted situations.

Making security goals and threats clear to system users may encourage their cooperation, rather than leaving them to view security as a nuisance to be worked around. Making the system and security configuration easy for administrators to understand will likely lead to fewer configuration errors, and well-designed monitoring can catch any remaining ones. Finally, making access audits routine can help system administrators understand access patterns and notice unusual access, whether it is a result of some nonroutine event or because a user account has been compromised. 

Related articles on queue.acm.org

A Decade of OS Access-control Extensibility

Robert N. M. Watson

<http://queue.acm.org/detail.cfm?id=2430732>

Standardizing Storage Clusters

Garth Goodson, Sai Susarla, and Rahul Iyere

<http://queue.acm.org/detail.cfm?id=1317402>

Monitoring and Control of Large Systems with MonALISA

Iosif Legrand, Ramiro Voicu, Catalin Cirstoiu,

Costin Grigoras, Latchezar Betev,

and Alexandru Costan

<http://queue.acm.org/detail.cfm?id=1577839>

References

1. Computer Security Resource Center. Role based access control and role based security. National Institute of Standards and Technology, Computer Security Division, 2014; <http://csrc.nist.gov/groups/SNS/rbac/>.
2. Hockenson, L. Facebook explains the cause behind its early Thursday downtime. Gigaom; <https://gigaom.com/2014/06/19/facebook-explains-the-cause-behind-its-early-thursday-downtime/>.
3. Moscaritolo, A. Verizon billing system hit by major outage. *PC Mag UK*, 2014; <http://uk.pcmag.com/news/33726/verizon-billing-system-hit-by-major-outage>.
4. Wikipedia. RBS Group computer system problems, 2012; http://en.wikipedia.org/wiki/2012_RBS_Group_computer_system_problems.

Geetanjali Sampemane (geta@google.com) belongs to the Infrastructure Security and Privacy group at Google. She started her career administering India's first connection to the Internet and then spent a few years working for the United Nations Development Program, helping developing countries connect to the Internet.

Copyright held by author. Publication rights licensed to ACM. \$15.00.

Article development led by **acmqueue**
queue.acm.org

Designing locking primitives with performance in mind.

BY DAVID LOHR BUESO

Scalability Techniques for Practical Synchronization Primitives

IN AN IDEAL WORLD, applications are expected to scale automatically when executed on increasingly larger systems. In practice, however, not only does this scaling not occur, but also it is common to see performance actually worsen on those larger-scale systems.

While *performance* and *scalability* can be ambiguous terms, they become less so when problems present themselves at the lower end of the software stack. This is simply because the number of factors to consider when evaluating a performance problem decreases. As such, concurrent multithreaded programs such as operating-system kernels, hypervisors, and database engines can pay a high price when misusing hardware resources.

This translates into performance issues for applications executing higher up in the stack. One clear example is the design and implementation of synchronization primitives (locks) for shared memory systems. Locks are a way of allowing multiple threads to execute concurrently, providing safe and correct execution context through mutual exclusion. To achieve serialization, locks typically require hardware support through the use of atomic operations such as compare-and-swap (CAS), fetch-and-add, and arithmetic instructions. While details vary across different cache-coherent architectures, atomic operations will broadcast changes across the memory bus, updating the value of the shared variable for every core, forcing cache-line invalidations and, therefore, more cache-line misses. Software engineers often abuse these primitives, leading to significant performance degradation caused by poor lock granularity or high latency.

Both the correctness and the performance of locks depend on the underlying hardware architecture. That is why scalability and the hardware implications are so important in the design of locking algorithms. Unfortunately, these are rare considerations in real-world software.

With the advent of increasingly larger multi- and many-core NUMA (non-uniform memory access) systems, the performance penalties of poor locking implementations become painfully evident. These penalties apply to the actual primitive's implementation, as well as its usage, the latter of which many developers directly control by designing locking schemes for data serialization. After decades of research, this is a well-known fact and has never been truer than today. Despite recent technologies such as lock elision and transactional memory, however, concurrency, parallel programming, and synchronization are still challenging topics for practitioners.¹⁰

Furthermore, because a transactional memory system such as Trans-

actional Synchronization Extensions (TSX) still needs a fallback path with a regular lock when the transaction does not succeed, these challenges are not going to disappear anytime soon. In addition, a transactional system does not guarantee specific progress beyond starvation freedom; therefore, this is not always a viable option for regular locking schemes. As a result of the complexities, not only can systems lack in scalability, but also their overall performance can be brought to its knees. Previous work has demonstrated that the cost of a poor, nonscalable, locking implementation grows as the number of cores in the system increases.² The onset of the performance collapse can, in fact, happen very suddenly, with just a few more cores added to the equation. The issues with performance and scalability go well beyond synthetic workloads and benchmarks, affecting real-world software.

There have recently been significant efforts in addressing lock-scaling issues in the Linux kernel on large high-end servers.³ Many of the problems and solutions apply to similar system software. This article applies general ideas and lessons learned to a wider systems audience, in the hope it can be helpful to people who are encountering similar scaling problems. Of course, locks are important on any shared memory system, but optimizing them does not imply ignoring the more important aspects: how those locks are used and what is being serialized.

Hybrid Locking Models and Optimistic Spinning

Locking primitives are traditionally classified as busy-waiting or blocking, depending on what occurs when a lock is not immediately available. If lock hold times are small, such as only serializing a reference count operation, then it makes more sense to avoid all blocking overhead and just burn CPU cycles for a short period of time. This is typically implemented by looping CAS calls until an unlocked condition is met. The alternative is to block un-

til that condition becomes true. Apart from the higher overhead and latency, blocking primitives can commonly have a strict dependency on the operating-system kernel's scheduler. As such, the thread-scheduling policies will increasingly depend on the previous level of execution in the software stack—for example, at hardware, kernel, or multiple user-space layers. Multiple sched-

tors. The Linux kernel, for example, has primarily three kinds of sleeping semaphores: mutexes (binary) and two counting semaphores, which includes a widely used reader/writer variant.

The use of a hybrid model is a common way of dealing with the trade-offs of each lock type. The goal is to delay blocking as much as possible and optimistically busy-wait until the lock



uling policies can impact both lock fairness and determinism.

Thrashing is another factor to keep in mind when building sleeping locks. Practitioners must consider the system consequences under heavy lock contention. Popular examples of busy-wait primitives are memory barriers and spinlocks. Blocking mechanisms normally include semaphores and moni-

is available. Determining when the lock sleeps, however, has an effect on performance. The algorithm needs to perform equally well for workloads that might benefit differently. In other words, users who explicitly want the characteristics of sleeping locks cannot pay performance penalties when a CPU will first spin on a lock for a determined period of time—often, not even at all.

In addition, users of general-purpose locking primitives should never be allowed to influence their algorithmic behavior. Misdesigning locking APIs can lead to unexpected consequences later, such as when the lock becomes heavily used. Simplicity is very much a virtue—and a consequence of a well-designed locking interface.

By using the notion of lock ownership, the Linux kernel keeps a pointer to the task that is currently holding the lock. The benefits of knowing the lock owner are twofold: it is a key piece of data when determining when to stop spinning; and it serves a debugging purpose—for example, deadlock detection. Similar strategies date back to 1975, proposing the notion of lock ownership in databases.⁸ Because of the overhead of maintaining the lock ownership, implementers can decide against reentrant locks, which also typically use a counter field.¹⁵

The rationale behind optimistic spinning is if the thread that owns the lock is running, then it is likely to release the lock soon. In practice, a Linux kernel mutex or rw_semaphore (reader-writer semaphore), the two most

commonly used locks throughout the system, can follow up to three possible paths when acquiring the lock, depending on its current state:¹²

► *Fastpath*. It tries to acquire the lock atomically by modifying an internal counter such as `fetch_and_add` or atomic decrementing. This logic is architecture specific. As shown in Figure 1, with x86-64, the mutex-locking fastpath has only two instructions for lock and unlock calls:

► *Midpath* (aka optimistic spinning). It tries to spin for acquisition while the lock owner is running and there are no other higher-priority tasks ready to run, needing to reschedule. Spinner threads are queued up using an MCS lock so that only one spinner can compete for the lock.

► *Slowpath*. As a last resort, if the lock still cannot be acquired, the task is added to the wait queue and sleeps until the unlock path wakes it up.

Because hybrid locks can still block, these primitives need to be safely used in a sleeping context. Optimistic spinning has proved its value in operating-system kernels such as Linux and Solaris. Even to this day,

simply delaying any kind of blocking overhead can have important impacts on system software. On a Linux Virtual File System (VFS) `create+unlink` microbenchmark to stress mutexes, optimistic spinning gave a boost in throughput of about 3.5 times on a commodity desktop system, as opposed to immediately sleeping. Similarly, for `rw_semaphores` on AIM7 workloads, hybrid locking provided an increase in throughput of about 1.8 times.³

One notable difference between `rw_semaphores` and mutexes in the Linux kernel is how they deal with lock ownership. When a lock is shared, the notion of ownership becomes more ambiguous than with exclusive locks. When workloads combine both readers and writers, there is a chance optimistic spinning can make writers spin excessively since there is no owner to spin on when readers hold it. Strategies to overcome this problem exist, such as using heuristics and magic numbers to determine when to stop spinning for readers. Reader ownership can be particularly tricky, however, and tends to add extra complexity and overhead in optimistic-spinning fastpaths. Furthermore, the use of magic numbers in locking primitives can bring unexpected consequences and must not be used lightly. By their very nature, heuristics can help particular scenarios while actually hurting performance in the common cases. Scalability is not about optimizing for the 1% and not thinking of the other 99%, but quite the opposite. In addition, solving the actual source of contention can be a valid alternative before considering overcomplicated primitives.

Contributing Factors to Poor Lock Scaling

It is not uncommon to find multiple factors contributing to poor lock performance at any given time. Naturally, the impact will vary depending on each system and workload. The factors and lock properties described here can be divided at a software-engineering level: between implementers, who typically design and implement the locking primitives, and users, who strictly apply them to their parallel or concurrent workloads and algorithms.

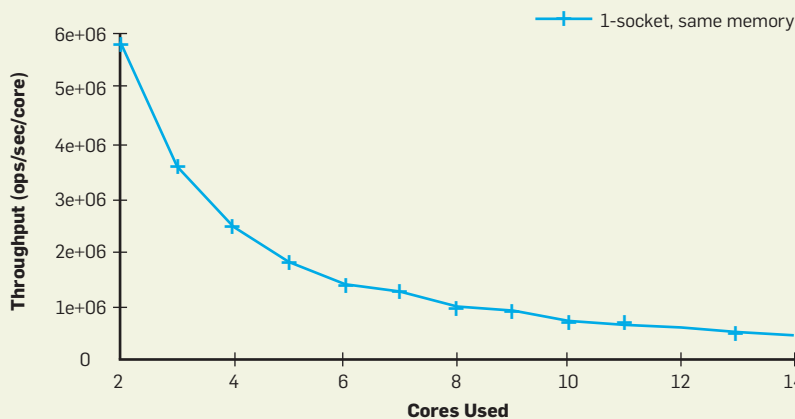
► *Length of critical region*. Reducing

Figure 1. Mutex lock and unlock fastpath code for x86-64..

```
00000000000000e10 <mutex_lock>:
e21:  f0 ff 0b          lock decl (%rbx)
e24:  79 08              jns     e2e <mutex_lock+0x1e>

00000000000000bc0 <mutex_unlock>:
bc8:  f0 ff 07          lock incl (%rdi)
bcb:  7f 0a              jg      bd7 <mutex_unlock+0x17>
```

Figure 2. Effects of cache-line contention within a single socket.



the length of a critical region can certainly help alleviate lock contention. In addition, the type of primitive used to serialize concurrent threads in lock implementations can play a key role in performance. When holding a lock in a slowpath, such as those that handle contended scenarios when acquiring or releasing a lock, practitioners often need to manage internal wait queues for threads that are waiting to take some action. In these cases, implementers must ensure the critical regions are short enough so as not to cause unnecessary internal contention. For example, issuing prechecks or wake-ups (for sleeping primitives) can easily be done asynchronously, without any additional serialization. Most recently, in the Linux kernel, efforts to shorten critical regions in mutexes and SysV semaphores (as well as other forms of inter-process communication) provided important performance benefits.^{3,13}

► *Lock overhead.* This is the resource cost of using a particular lock in terms of both size and latency. Locks embedded in data structures, for example, will bloat that type. Larger structure sizes mean more CPU cache and memory footprint. Thus, size is an important factor when a structure becomes frequently used throughout the system. Implementers also need to consider lock overhead when enlarging a lock type, after some nontrivial modification; this can lead to performance issues in unexpected places. For example, Linux kernel file-system and memory-management developers must take particular care of the size of VFS struct inode (index node) and struct page, optimizing as much as possible.⁴ These data structures represent, respectively, information about each file on the system and each of the physical page frames. As such, the more files or memory present, the more instances of these structures are handled by the kernel. It is not uncommon to see machines with tens of millions of cached inodes, so increasing the size of the inode by 4% is significant. That is enough to go from having a well-balanced workload to not being able to fit the working set of inodes in memory. Implementers must always keep in mind the size of the locking primitives.

Because of their simple nature,



It is not uncommon to find multiple factors contributing to poor lock performance at any given time. Naturally, the impact will vary depending on each system and workload.



busy-wait primitives have better latency than more complex locks. Calls to initialize, but particularly to acquire or release the lock, need to be cheap, incurring the fewest CPU cycles. The internal logic that governs the primitive must not be confused with other factors that affect call latency, such as hold times and contention. Blocking (or sleeping) locks are expected to be more expensive, as any algorithm must at least take into account events such as putting threads to sleep and waking them up when the lock becomes available. The trade-off, of course, is that users choose blocking locks only when dealing with large critical regions or executing in a context that requires sleeping, such as when allocating memory. Thus, if the average time a thread expects to wait is less than twice the context-switch time, then spinning will actually be faster than blocking.¹⁵ The quality-of-service guarantee is another factor to consider when choosing between spinning and sleeping locks, particularly in realtime systems. Blocking on larger NUMA systems can ultimately starve the system of resources.

In addition, reader-writer primitives may have different latencies when dealing with shared or exclusive paths. This is not an ideal situation, as users will have to consider the extra overhead penalties when, for example, sharing the lock. Making matters worse, users might not even realize this difference, and, thus, sharing the lock will in fact result in poorer performance than using an exclusive lock. The read:write ratio (discussed later) is an important factor when determining if sharing the lock is actually worth the extra cost of using a reader lock. In general, choosing the wrong type of lock can impact performance, incurring unnecessary overhead.

► *Lock granularity.* This refers to the amount of data a lock protects, normally with a trade-off between complexity and performance. Coarse granularity tends to be simpler, using fewer locks to protect large critical regions. Fine granularity, on the other hand, can improve performance at the cost of more involved locking schemes, particularly beneficial when a lock is contended. When designing concurrent algorithms, coarse-grained locks might be misused only because they can be sim-

pler and, initially, more obvious than the finer-grained alternatives. Because bugs related to synchronization—such as deadlocks, race conditions, and general corruption—can be particularly difficult to debug, programmers might prefer them only because of fear and uncertainty, assuming that protecting too much is better than not enough.

Making matters worse, these issues can easily make entire software systems unreliable—thus, practically useless. Even experienced lock practitioners can overlook the potential performance benefits of fine-grained locking, not noticing scaling problems until reported. Perhaps the most famous coarse-grained lock in the Linux kernel was the now-replaced Big Kernel Lock (BLK), serializing threads that enter kernel space to service system calls. Because the lock protects so much data, such primitives are also referred to as giant locks. Futexes are another area in the kernel that can suffer from coarse-grained locking schemes. With a chained hash-table architecture, spinlocks protecting the chain can become heavily contended only because of collisions. Finer graining and parallelizing these paths can be done by simply increasing the size of the hash table. This approach improved the hashing throughput by a factor of up to eight times.³ This fine-grained technique is also known as lock stripping, improving concurrency by having multiple locks to serialize different, nondependent parts of an array, or list-based data structure.

Coarse-grained locks certainly do have their place, nonetheless. One important factor to consider when addressing coarse-grained locks is the overhead of an individual lock. In some cases, the extra memory footprint of embedding additional locks will shadow the benefits of alleviating the contention. Abusing fine-grained locking can have just as negative an effect on performance as abusing coarse granularity. Furthermore, coarse granularity can be particularly good when contention is not a problem and critical regions and hold times are short, albeit rare.

Both practice and research have shown the performance benefits of combining coarse and fine granularity, although it is usually more complex.



When holding a lock in a slowpath, practitioners often need to manage internal wait queues for threads that are waiting to take some action.



Hybrid strategies have been proposed for locking in the Hurricane kernel, combining the benefits of both kinds of granularity.¹⁶ For more than 20 years, the Linux kernel's SysV semaphore implementation suffered from coarse-grained locking when dealing with `semtimedop(2)` system calls. When a finer-grained locking model was introduced for these calls to handle the common case of a task waiting to manipulate a single semaphore within an array containing multiple semaphores, benchmark throughput improved by more than nine times.¹³ This hybrid strategy also directly impacts important RDBMS (relational database management system) workloads that rely heavily on these semaphores for internal locking, with contention decreasing from approximately 85% to 7%. Coarse-grained locking in IPC (inter-process communication) can still occur when manipulating more than a single semaphore. Choosing lock granularity must be a well-informed decision. Changing the granularity later in a program life cycle can be an expensive and error-prone task.

► *Read/write ratios.* This is the ratio between the number of read-only critical regions and the number of regions where the data in question is modified. Reader/writer locks will take advantage of these scenarios by allowing multiple readers to hold the lock while acquiring it exclusively when modifying protected data. Multiple research and development efforts have tried to optimize primitives for read-mostly situations, making the cost of reader synchronization as minimal as possible—normally at a higher cost, or overhead, for writer threads. Examples include variations of the RCU (read-copy update) mechanism, sequence locks (seqlocks), and read-mostly locks (rmlocks) in FreeBSD.

It is well known the Linux kernel makes heavy use of RCU, allowing lockless readers to coexist with writers. Because readers do not actually hold a lock, it is a particularly fast mechanism that avoids the overhead and hardware implications that regular reader/writer locks incur. RCU handles updates by: making them visible to readers by single-pointer read and write, ensuring readers execute before or after the modification, depend-

ing on whether they see the update in time; and delaying the reclaim of the data structure until all readers are done with their critical region, as it is guaranteed that no readers hold references to the data structure, similar to garbage-collection approaches. Introduced in the 2.5 era, in the early 2000s, it is no coincidence the adoption of RCU within the kernel has grown substantially, including, among many examples, scaling of the dentry (directory entry) cache, NMI (nonmaskable interrupt), and process ID handling.¹¹ Most recently, converting the epoll control interface from using a global mutex to RCU permitted significant performance improvements by allowing file-descriptor addition and removal to occur concurrently. Specifically, important Java-based workloads were boosted with throughput improvements of up to 2.5 times on large HP and SGI NUMA systems.

► *Fairness.* Most importantly, fairness avoids lock starvation by using strict semantics to choose which task is next in line to acquire the lock in contended scenarios. A common example of unfair spinlocks is any thread acquiring the lock without respecting if other threads were already waiting for it. This includes the same task constantly reacquiring the lock. Unfair locks tend to maximize throughput but incur higher call latencies. If such a scenario becomes pathological, it is a real concern for lock starvation and lack of progress—unacceptable in real-world software. A widely used solution to this is different variations of ticket spinlocks. Fair locks address starvation issues at the cost of increased preemption sensitivity,^{1,6} making them at times unsuitable when preemption cannot be controlled, such as in user-space applications. If the kernel's CPU scheduler preempts a task that is next to acquire the lock and the lock is released during this time, then it will cause the rest of the contending threads to wait until the preempted task is rescheduled. Similarly, the greater the cost of lock acquisition or release, the greater the chance of excessive queuing contributing to poor performance.

Experiments conclude that unfair locks are particularly useful when running more than one thread per core, as they can outperform fair alterna-

tives in highly contended scenarios.⁷ Of course, since threads have to wait longer to acquire the lock, this problem can become significantly more pronounced when dealing with NUMA systems, particularly if the fair lock incurs expensive cache-line issues, as described later.

Statistically it is possible to have different degrees of fairness in NUMA systems. For example, because of CPU node locality, threads can have a better chance of acquiring the lock if it is on the local memory node. By transforming any kind of busy-wait lock into a NUMA-aware primitive, cohort locks were developed to address some of these issues. In this scheme, writer locks are passed among contending threads within the same NUMA node, while readers maintain shared resources within the same node.

Fairness takes a different turn when dealing with read/write locks, depending on the context, workload, and reader:writer ratios. There will be occasions where it is more suitable to give preference to readers, and vice versa. Either way, practitioners should take special care the primitive does not starve reader or writer threads to the point it makes the lock perform poorly in some use cases. One alternative is implementing different variations of the same read/write primitive with particular fairness preferences, but this can also lead to developer misuse in different contexts. For performance reasons, the Linux kernel uses the concept of writer-lock stealing for rw semaphores, breaking the strict FIFO (first-in first-out) fairness for writers. Writers can atomically acquire the lock, even if other writers are already queued up.

► *Cache-line mishandling.* Cache-line bouncing and contention are probably the two worst forms of performance degradations on large NUMA systems when it comes to low-level locking primitives. Tasks spinning on a contended lock will try to fetch the lock cache line repeatedly in some form of tight CAS loop. For every iteration, usually in an atomic context, the line containing the lock is moved from one CPU cache to another. It is easy to see how this bouncing will degrade performance with increasing CPU counts, incurring expensive memory-bus and

interconnect usage penalties. Furthermore, if the lock-protected data structure is in the same cache line, it can significantly slow down the progress of the lock holder, leading to much longer lock hold times.

A straightforward way of addressing cache-line bouncing was proposed 30 years ago¹⁴ with the compare compare-and-swap (CCAS) technique. The idea is to do a simple read of the lock state and incur the read-modify-write CAS operation only when the lock becomes available. The Linux kernel now relies on CCAS techniques for the internal counter checks of both mutexes and read/write semaphores when attempting to acquire a lock exclusively (note that sharing the lock does not require such CAS storms). Concretely, for mutexes, Java-based benchmarks experienced up to a 90% throughput increase³ on a 16-socket, 240-core system. AIM7 benchmark workloads that are designed to execute mostly in kernel space also saw noticeable improvements on eight-node, 80-core Westmere systems with throughput increases of up to three times. As for read/write semaphores, CCAS results from pgbench on smaller, quad-core desktop systems showed improvements of up to 40% on 1GB PostgreSQL databases. Most noticeably improving the cache-line bouncing that occurs when the mmap_sem is heavily contended, this lock is designed, among other things, to serialize concurrent address-space operations.

In general, experimentation shows that CCAS techniques will help on large high-end systems with four or more sockets, or NUMA nodes. Normally, the overhead of checking the counter is so low in noncontended cases that it will not impact performance negatively on smaller systems. Performance work must always ensure no regressions are introduced in lower-end machines, particularly in the Linux kernel, which spans an enormous user base.

Alternatively, the use of back-off algorithms, which address expensive CAS operations when spinning on a lock, can help alleviate cache-line bouncing and memory-interconnect overhead. As with CCAS techniques, the idea is to delay read-modify-write calls in tight spinning loops, the main difference being the delay factor when the lock is

not immediately available. Significant research has attempted to estimate the optimal delay factor across multiple systems and workloads; the different algorithms and heuristics can be classified as static and dynamic. Delaying for a static amount of time can be optimized for a particular workload but, of course, will not necessarily perform well when generic locking solutions are needed. To this end, dynamic delays are far more realistic but at the cost of extra overhead in the heuristics and the risk of miscalculating the back-off strategy: threads should not back off for too long, as this can shadow any performance benefits these strategies try to introduce.

Good heuristics for back-off algorithms are based on the number of CPUs trying to acquire the lock, such as proportional and exponential, with analogies drawn from CSMA (carrier sense multiple access) networks such as Ethernet. Delaying for the optimal time requires estimating the length of the critical region and the latency of the lock holder to release the lock so another thread can acquire it.^{6,15} Needless to say, this sort of information is rarely available in real-world workloads. In addition, back-off algorithms do not address the fact that, just as with regular CAS and CCAS spinlocks, all threads spin on the same shared location, causing cache-coherence traffic on every successful lock acquisition. With these caveats, it is not surprising that back-off algorithms are not suitable for the Linux kernel, even though promising results have been seen with ticket spinlocks and proportional back-off strategies.⁵

Demonstrating the Effects of Cache-Line Contention

The sole purpose of locking and concurrency is improving system performance by allowing correct parallel executions of threads. Even on uniprocessor systems, concurrency permits the illusion of multitasking in preemptive scheduling environments. If locking primitives get in the way of performance, then they are simply broken. Of course, unlike correctness attributes, locks that perform poorly will matter more on larger systems.

In a modern large-scale, multicore NUMA system, the effects of cache-line contention can go from incurring

moderately acceptable overhead to being the culprit of severe system issues. Table 1 shows three high-end HP x86-64 servers used as test systems.³ All these machines follow a normal NUMA topology in which cores and memory resources are evenly distributed among nodes. There are, therefore, no exotic NUMA use cases.⁹ Each socket, or node, has 15 cores, and memory is equally divided as well. A simple spinlock microbenchmark is enough to measure cost of cache-line contention in large NUMA systems—in this case, using a Linux 3.0-based spinlock implementation, iterating a million times in a tight loop just acquiring and releasing the lock, similar to what a torturing program such as *rcutorture* would do.

While this pathological behavior does not fall into the real-world category, it exemplifies nicely theoretical issues such as lock contention. Furthermore, real software with very similar behavior does exist, leading to horrendous contention. Based on the number of threads and the loop iteration count, the average number of operations per second per CPU can be calculated when $<N>$ CPUs are involved in the cache-line contention. This serves as the benchmark throughput.

For a full understanding of the effects of cache-line contention, the examination must include a single socket. Otherwise, factors such as NUMA awareness and interconnect cost could not be differentiated from the results. Furthermore, it should provide the least performance impact and therefore can provide a level ground to compare against other results that involve multsocket communication. Figure 2 shows the microbenchmark throughput regression by increasing core counts within a single socket.

It is easy to see how performance smoothly degrades as more cores contribute to cache-line contention. By maximizing resources and using all cores in a single socket (that is, by a factor of seven), the microbenchmark throughput decreases by 91.8% when compared with using only two cores. The situation ends up, therefore, being the exact opposite of what naive developers will normally intuit, expecting performance at least to improve “some” by blindly throwing in more

cores. Furthermore, the 91.8% regression can be considered a maximum performance penalty when evaluating systems that will logically partition their software architecture around NUMA topologies. This is commonly done to avoid latency penalties when dealing with remote memory, particularly on large-end machines.

Of course, the situation does not get any better. In fact, it gets a great deal worse. Table 2 compares the costs of cache-line contention in a single socket versus two sockets. Most noticeably, there is more than an 80% decrease in throughput when going from 15 to 30 cores.

Furthermore, the cost of having the lock's memory location on a remote node is clearly worse when using two sockets, reducing throughput by more than 90% when compared with a single socket. In this case, benchmark runtimes are even more concerning, going from two to 21 seconds to complete the million iterations when the memory is completely remote.

The number of cores is not the only factor to consider when measuring this sort of contention. The distribution method plays a key role in performance, where cache-line contention is always better when all the contention is contained in as few sockets as possible.³ As core counts increase, the FF (first fill) method will always try to use the cores available in the same socket, while an RR (round robin) distribution will use cores from increasing socket counts.

Figure 3 exhibits the probability of inter- versus intra-cache-line contention, estimated by computing $100/\text{num_sockets}$. Increasingly large multsocket systems will thus have a much lower chance that two random cores participating in the same cache-line contention will reside on the same socket.

This can have catastrophic effects on performance and demonstrates contention in a two-socket system is extremely different from that in a 16-socket system. As such, applications tuned for smaller systems cannot be blindly executed on larger systems and immediately expect good results without previous careful thought and analysis. As previously seen, scaling based only on the number of CPUs is likely to introduce significant lock and cache-line contention inside the Linux kernel.

Queued/MCS: Scalable Locking

As already discussed, even busy-wait locks with CCAS or back-off strategies can incur enough cache-line contention to impact overall system performance significantly on large machines. As such, their inherently unfair nature exposes them to an irregular number of cache misses when the lock becomes contended. On the other hand, a truly scalable lock is one that generates a constant number of cache misses, or remote accesses, per acquisition,² thus avoiding the sudden performance collapse that occurs with nonscalable alternatives. Queued locks achieve this by ensuring each contending thread spins for a lock on the CPU's local memory, rather than the lock word. As a consequence of this determinism, queued locks are fair, commonly granting lock ownership to waiting threads in FIFO order. Another attractive property of these locks is the overhead, requiring only $O(n+j)$ space for n threads and j locks,¹⁵ as opposed to $O(nj)$ for nonscalable primitives. Most forms of queued locks—MCS, K42, and CLH, to name a few popular ones—maintain a queue of waiters, each spinning on its own queue entry. The differences in these locks are how the queue is maintained and the changes necessary for the lock's interfaces.

A common experiment is to replace a regular spinlock with an MCS lock,^{2,3} always with quite superb results in kernel space (normally using the Linux kernel). Figure 4 compares a vanilla 2.6 Linux spinlock with a prototype MCS implementation on an AIM7 file-system benchmark, serializing concurrent linked-lists operations with a single lock. Contention on this global spinlock is the cause of all the cache-line contention.

As more users are added to the workload, throughput (jobs per minute) in the spinlock case quickly flatlines, while with the MCS lock it steadily improves by a factor of 2.4. Similarly, system time drops from 54% to a mere 2%; thus, the bottleneck is entirely removed. As no additional optimization efforts are made, such as finer-graining the lock, it is safe to conclude the MCS lock alone makes a huge impact by minimizing only the intersocket cache-line traffic.

To this end, MCS locks have been added to the Linux kernel,² particularly

Table 1. Testing three high-end HP x86-64 servers.

System	Sockets/Cores	Memory
HP DL580 Gen8 (Intel Xeon E7 - 4890 v2)	4 sockets/60 cores	2 TB
HP CS900 (Intel Xeon E7 - 4890 v2)	8 sockets/120 cores	6 TB
HP CS900 (Intel Xeon E7 - 4890 v2)	16 sockets/240 cores	12 TB

Table 2. Comparison of cache-line contention costs in a single socket with two sockets.

Execution Nodes	Memory Node	Sockets Used	Cores Used	ops/sec/core
Node 0	Node 1	1 socket	15 cores	474,519
—	—	—	—	—
Nodes 0-1	Node 1	2 sockets	30 cores	69,200
Nodes 1-2	Node 1	2 sockets	30 cores	67,126
Nodes 2-3	Node 1	2 sockets	30 cores	45,993

Figure 3. Chances of two random cores participating in the cache-line contention.

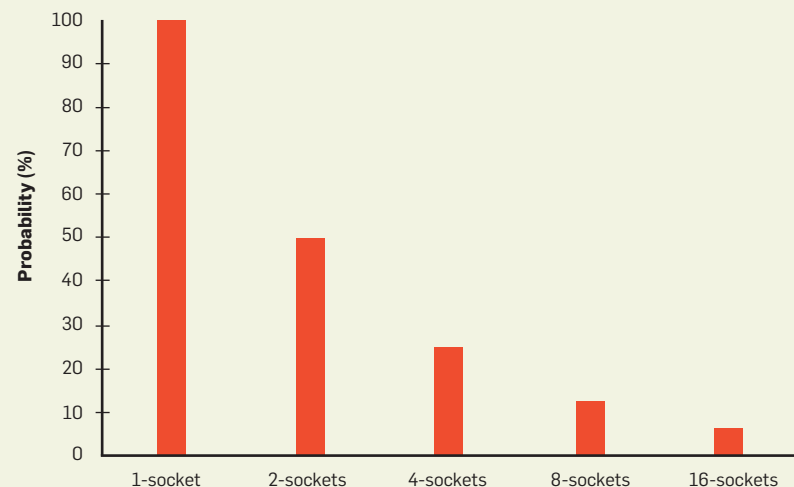


Figure 4. Eight-socket/80-core HT-enabled Xeon.

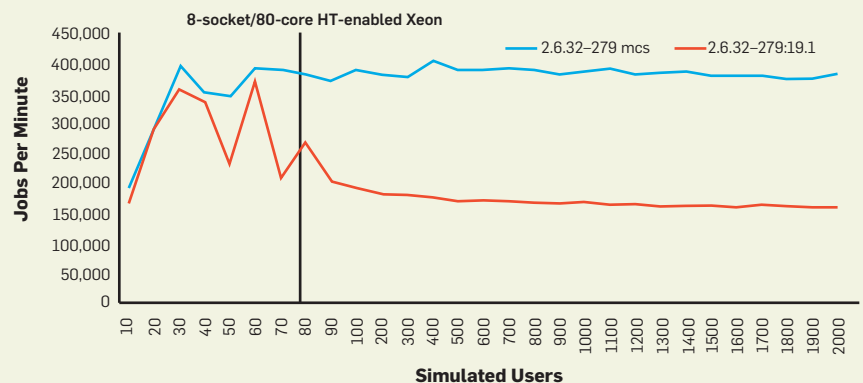


Figure 1. Kernel interfaces for MCS lock primitives.

```

struct mcs_spinlock {
    struct mcs_spinlock *next;
    int locked;
};

void mcs_spin_lock(struct mcs_spinlock **lock, struct mcs_spinlock *node);
void mcs_spin_unlock(struct mcs_spinlock **lock, struct mcs_spinlock
    *node);

```

in the spinning logic of sleeping semaphores. In the case of mutexes, this provides a throughput boost of around 248% in a popular Java workload on a 16-socket, 240-core HP Converged-System 900 for SAP HANA. The code in Figure 5 shows the primitive's interfaces.

The idea is that when a thread needs to acquire the contended lock, it will pass its own local node as an argument, add itself to the queue in a wait-free manner, and spin on its own cache line (specifically on `node->locked`) until its turn comes to acquire the lock. The FIFO fairness comes naturally as the current lock holder will pass the lock along to the next CPU in the linked list when it releases the lock. One drawback of this algorithm is that, in order to add itself to the wait queue, a task must store the pointer of its local node in the `->next` field of its predecessor, potentially incurring extra overhead if the predecessor task needs to release the lock.

Unfortunately, using MCS locks in regular ticket spinlocks cannot be done without enlarging the size of the lock. Spinlocks are used throughout the kernel and cannot be larger than a 32-bit word size. To this end, other queued locking approaches¹⁷ offer valid alternatives to the current ticketing system.

Conclusion

Designing locking primitives with performance in mind is not only good practice in general, but also can mitigate problems in the future. This article was born from experience with larger-scale servers and on real issues that impacted real clients using Linux. While Donald E. Knuth stated “premature optimization is the root of all evil,” it is quite the opposite when implementing locking primitives. Empirical data and experiments have shown the cost of misusing, or ignoring, the

underlying hardware implications of locking. In addition, both users and implementers must take special care in how these locks are used and the consequences of particular decisions during the life cycle of a lock, such as different degrees of contention.

In the future, as computer systems evolve and increase their processing capabilities, the practice and theory of locking primitives will need to adapt accordingly, ultimately making better use of the hardware architecture. Even today, very specialized locks are available, such as cohort and hierarchical primitives like HCLH that optimize for memory locality, addressing the fairness issues of back-off-based locks, while keeping the benefits of queued-based locks.

Of course, there is no single recipe for locking performance and scalability topics, and there are many more related topics the reader can follow up on, including lockless data structures and addressing particular constraints for specialized systems such as priority inversion and inheritance for real-time environments. An article of this nature should, at least, serve as a way of creating awareness among practitioners when dealing with severe scaling problems on large systems.

Acknowledgments

The scaling work in the Linux kernel has been a team effort, both internally in the Hewlett-Packard Linux performance group and the upstream kernel community. Many thanks to Scott J. Norton, Paul E. McKenney, and Samy Al Bahra.

This work represents the views of the author and does not necessarily represent the view of SUSE LLC. Linux is a registered trademark of Linus Torvalds. Other company, product, and service names may be trademarks or service marks of others. 

Related articles on queue.acm.org

Nonblocking Algorithms and Scalable Multicore Programming

Samy Al Bahra

<http://queue.acm.org/detail.cfm?id=2492433>

Tackling Architectural Complexity with Modeling

Kevin Montagne

<http://queue.acm.org/detail.cfm?id=1862187>

Structured Deferral: Synchronization via Procrastination

Paul E. McKenney

<http://queue.acm.org/detail.cfm?id=2488549>

References

1. Al Bahra, S. Nonblocking algorithms and scalable multicore programming. *ACM Queue* 11, 5 (2013).
2. Boyd-Wickizer, S., Kaashoek, F.M., Morris, R., and Zeldovich, N. Non-scalable locks are dangerous. In *Proceedings of the Linux Symp.*, 2012, Ottawa, Canada.
3. Bueso, D. and Norton, S.J. An overview of kernel lock improvements. *LinuxCon North America*, Chicago, IL, 2014; <http://events.linuxfoundation.org/sites/events/files/slides/linuxcon-2014-locking-final.pdf>.
4. Corbet, J. Cramming more into struct page. *LWN.net*, 2013; <http://lwn.net/Articles/565097/>.
5. Corbet, J. Improving ticket spinlocks. *LWN.net*, 2013; <http://lwn.net/Articles/531254/>.
6. Crummey-Mellor, J.M., Scott, M.L. Algorithms for scalable synchronization on shared-memory multiprocessors. *ACM Transactions of Computer Systems* 9, 1 (1991), 21–65.
7. Fuerst, S. Unfairness and locking. *Lockless Inc*, 2014; <http://locklessinc.com/articles/unfairness/>.
8. Gray, J.N., Lorie, R.A., Putzolu, G.R., Traiger, I.L. Granularity of locks and degrees of consistency in a shared data base. *IBM Research Laboratory*, San Jose, CA, 1975.
9. Lameter, C. Normal and exotic use cases for NUMA features. *Linux Foundation Collaboration Summit*, Napa, CA, 2014.
10. McKenney, P.E. Is parallel programming hard, and, if so, what can you do about it? 2014; <https://www.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html>.
11. McKenney, P.E., Boyd-Wickizer, S., Walpole, J. RCU usage in the Linux kernel: one decade later, 2013; <http://www2.rdrop.com/users/paulmck/techreports/RCUUsage.2013.02.24a.pdf>.
12. Molnar, I. Bueso, D. Design of the generic mutex subsystem. *Linux kernel source code*, 2014; [documentation/mutex-design.txt](https://lwn.net/Articles/543659/).
13. van Riel, R., Bueso, D. 2013. ipc.sem: sysv semaphore scalability. *LWN.net*, 2013; <http://lwn.net/Articles/543659/>.
14. Rudolph, L., Segall, Z. Dynamic decentralized cache schemes for MIMD parallel processors. In *Proceedings of the 11th Annual International Symposium on Computer Architecture*: 340–347.
15. Scott, M.L. Shared-memory synchronization. *Synthesis Lectures on Computer Architecture*. Morgan & Claypool Publishers, San Rafael, CA, 2013.
16. Unrau, R.C. Krieger, O. Gamsa, B., Stumm, M. Experiences with locking in a NUMA multiprocessor operating system kernel. *Symposium on Operating Systems Design and Implementation*, 1994; https://www.usenix.org/legacy/publications/library/proceedings/osdi/full_papers/unrau.a.
17. Zijlstra, P. Longman, W. locking: qspinlock. *LWN.net*, 2014; <http://lwn.net/Articles/590189/>.

Davidlohr Bueso is a performance engineer at SUSE Labs. He is an active Linux kernel contributor in areas such as synchronization primitives, memory management, and IPC—all with a special focus on scalability.



We invite you to contribute to the

**17th International Conference on Human-Computer
Interaction with Mobile Devices and Services**

in Copenhagen, Denmark

Sebastian Boring & Enrico Rukzio
(General Chairs)

MobileHCI 2015

Mobile Human-Computer-Interaction

Upcoming Submission Deadlines

February 6th 2015: Paper submission deadline

February 13th 2015: Workshop proposals

May 1st 2015: Contributions for doctoral consortium and
industry case studies

May 8th 2015: Submission of posters, demos, panels and tutorials



@ACMMobileHCI

MobileHCI2015



Tivoli Hotel
& Congress Center

**Copenhagen, Denmark
August 24 to 27, 2015**

<http://mobilehci.acm.org>



Photos: Saket Vora (left) | Chris Brown (right)



Association for
Computing Machinery



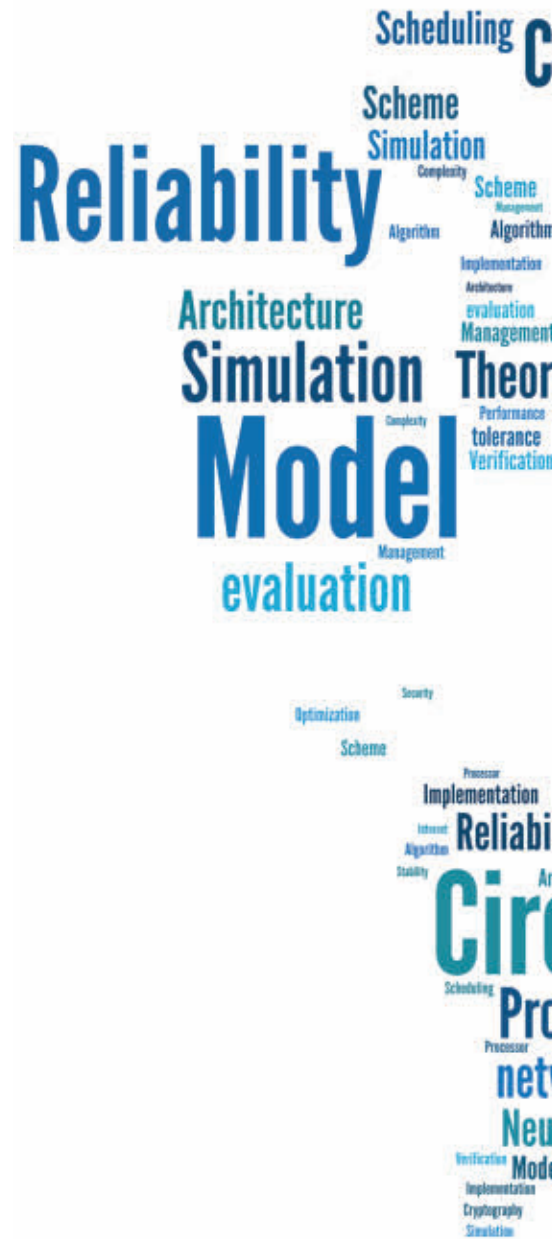
DOI:10.1145/2688498.2688499

Results showed more published papers, collaboration with U.S. colleagues, preference for English, and uneven research impact.

BY VIRENDER SINGH, ALICIA PERDIGONES, JOSÉ LUIS GARCIA, IGNACIO CAÑAS-GUERRERO, AND FERNANDO R. MAZARRÓN

Analyzing Worldwide Research in Hardware Architecture, 1997–2011

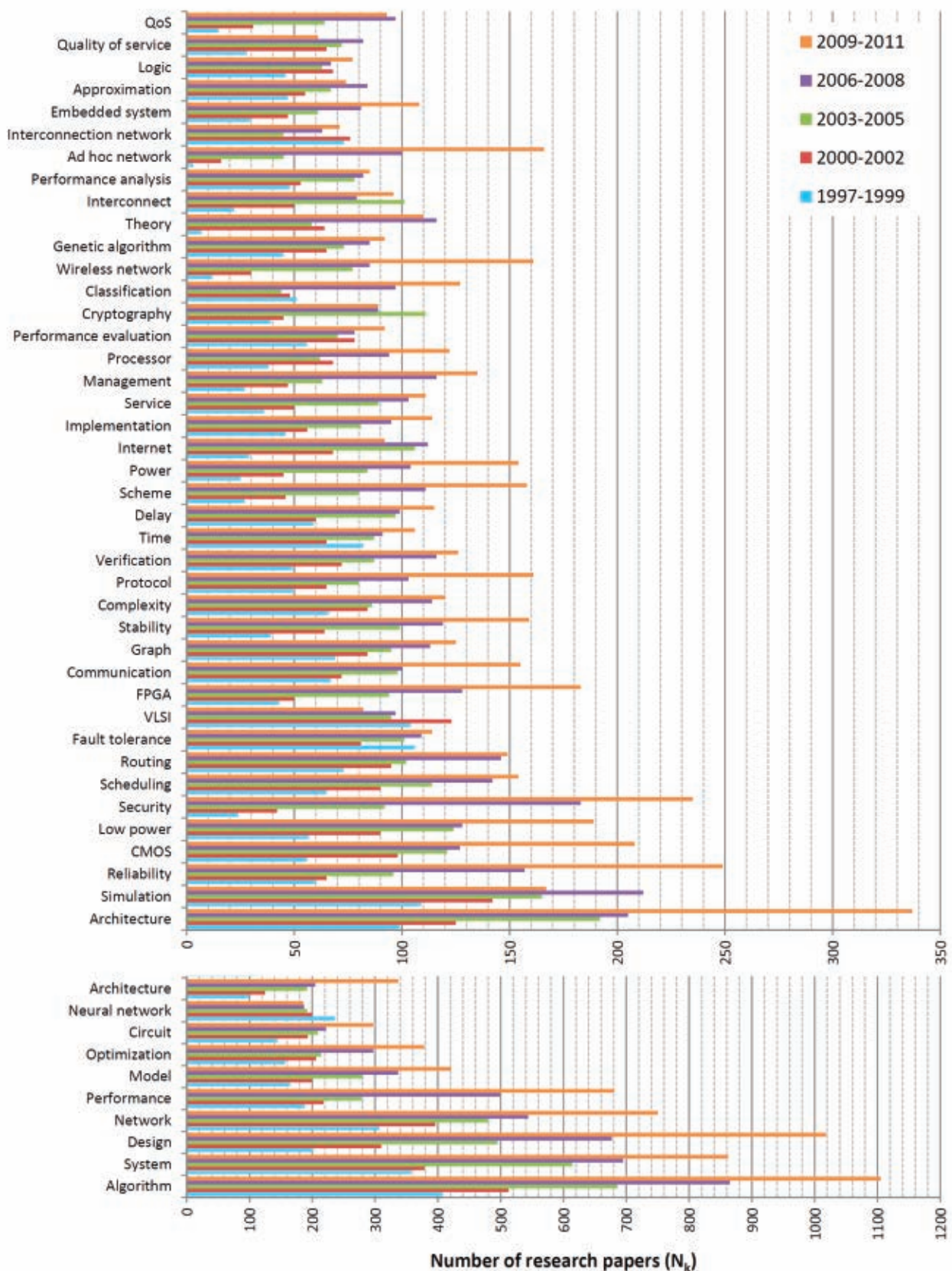
THIS WORK COVERS research in the field of hardware architecture in the years 1997 to 2011. More than 51,000 journal articles (conference articles excluded) were analyzed using a specially developed in-house software application able to analyze keyword evolution, estimate an article's impact, determine collaborations established at different scales, and calculate collaboration-weighted productivity of countries and research centers. Keyword analysis allows rapidly growing research topics (such as networks, sensors, security, stability, energy, and chips), as well as topics of lesser interest (such as neural networks, ATM, and VLS),



» key insights

- Analysis of keywords identifies evolution of the most important research topics in hardware architecture, as well as research topics with lesser interest.
- Analysis of collaborations reveals great differences in terms of internationalization and complexity of research networks in hardware architecture.
- These results should be very useful for characterization and decision-making by research institutions in the field of hardware architecture.

Figure 1. Change in use of the most frequently used keywords (as they appeared in the selected papers), 1997–2011.



laborative networks,⁷ competitiveness of universities,⁹ and impact of publications.^{2,6} The literature had not previously included studies on global scientific production in the field of hardware architecture or its development in recent years.

This article analyzes the state of global research into hardware architecture and its development over the period 1997–2011. Keyword analysis was used to identify the research topics that showed the most growth over the study period, as well as those of lesser interest. The scientific production, collaborations, and research impact of countries and research centers were also examined, along with distribution of national and international networks, change in the collaboration-weighted productivity of countries and research centers, and journals in which they published. This was done through the analysis application developed by our group, the PADOC-Research Group at Technical University of Madrid (<http://www.grupoinvestigacionpadoc.com>), analyzing keyword evolution, computing the collaboration-weighted productivity of research centers and countries, estimating the impact of each article, and determining the collaborations established at different scales.

Materials and Methods

The WoS Core Collection, a Thomson Reuters product consisting of seven databases including thousands of scholarly articles, reviews, journals, book series, and reports provided the raw data used in the study.

Following the methodology in numerous research papers,^{3,5,11} all traditional journal articles, or “article” and “review” document types, published 1997–2011 in the category “computer science, hardware architecture” ($n=51,474$) were downloaded (December 2012) from the WoS Core Collection for analysis. Proceedings papers, reprints, notes, letters, and corrections were excluded; 1997 was selected as the first year of the study period since it was when information on impact factors (IF) became available from the Institute for Scientific Information (ISI).

The author(s), editor(s), title, source, addresses, citation time, keywords, language, and WoS categories

were recorded for each article and analyzed through the analysis application (written in Visual Basic). It was able to identify countries and research centers associated with research papers (3.4% had no information on origin in the WoS field “addresses”), tracked development of the most important research topics, and analyzed changes in the geographic and institutional distribution of publications.

Statistical analysis of author keywords is useful for revealing science trends and was proved important for monitoring development of science and programs.¹² The keywords were provided by paper authors; keywords or “keywords plus” were generated by the ISI. The analysis application obtained an annual count of the number of research papers (N_K) in which each detected keyword appeared; it was able to identify singular and plural forms to provide an overall figure. Compound keywords were also analyzed. These analyses were then combined to provide a more realistic and representative account of the research topics that had been studied.

The geographic and institutional distribution of publications was examined by determining the number of collaboration-weighted articles (N_P) produced by each country and research center. This ensured the sum of research papers published by all countries and research centers corresponded to the total number of published papers. For any specific period, it was calculated as follows (Equation 1)

$$N_P = \sum_{i=1}^N \frac{1}{N_{C,i}}; N_P = \sum_{i=1}^N \frac{1}{N_{RI,i}}$$

where N is the total number of research papers produced by a country or research center over a set period, $N_{C,i}$ is the number of participating countries associated with each research paper “i,” and $N_{RI,i}$ is the number of research institutions participating in each paper “i.”

To analyze the complexity of the research network structures, the analysis application examined the degree of international collaboration involved in each article, as well as the number of research centers and authors. This was achieved by recording the connections between participating countries

and their research centers, then generating corresponding matrices. The percentage of international collaboration, or Col (%), for the various countries and centers was then calculated as the number of research papers published in collaboration with at least one center of another country over their total production.

The average number of authors per article from each country or research center (N_A) was calculated as follows (Equation 2)

$$N_A = (\sum_{i=1}^N N_{A,i})/N$$

where $N_{A,i}$ is the number of authors (of any nationality or institution) in each paper “i.”

The average number of research institutions (N_{RI}) participating in each paper was calculated for each country and research center (Equation 3)

$$N_{RI} = (\sum_{i=1}^N N_{RI,i})/N$$

where $N_{RI,i}$ is the number of research institutions participating in each paper “i.”

To determine the impact of a particular piece of research, the application examined the IF and the number of citations per article. First, an impact indicator, or “year impact factor” (YIF), the IF of a journal for the year in which the article was published⁴ was assigned to each paper. The average YIF for each country and research center was then calculated as follows for different periods of time (Equation 4)

$$YIF = (\sum_{i=1}^N YIF_i)/N$$

where YIF_i is the YIF of a paper “i.”

The average number of citations for each country and research institution (NCI) was then calculated as follows for different periods of time (Equation 5)

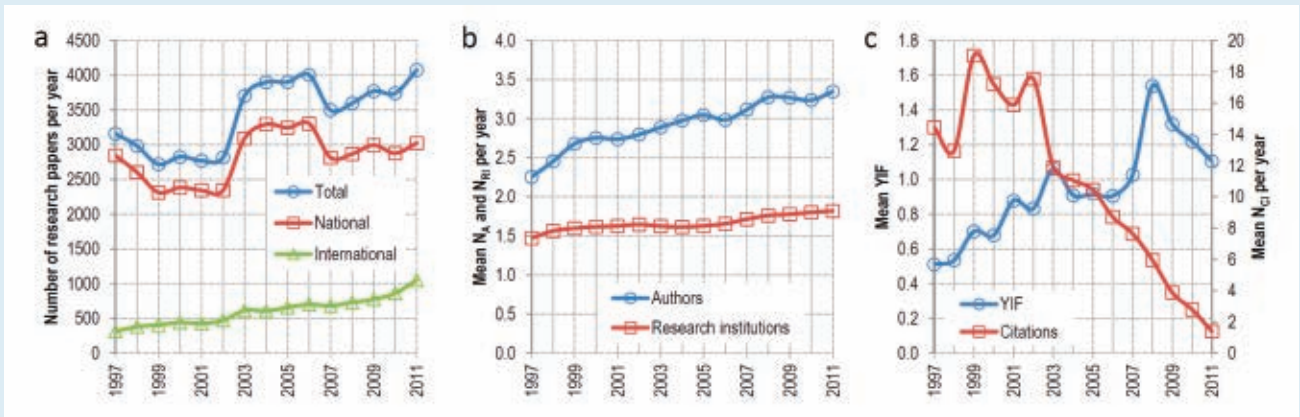
$$N_{CI} = (\sum_{i=1}^N N_{CI,i})/N$$

where $N_{CI,i}$ is the number of citations received by each paper “i” until the moment it is downloaded.

Results

Development of the most important research topics. The analysis application detected more than 77,000 different keywords in the category “computer science, hardware archi-

Figure 2. Change in number of national and international research papers, mean number of authors per paper (N_A), mean number of research centers participating in papers (N_{RI}), mean year impact factor (YIF), and mean number of citations per article (N_C).



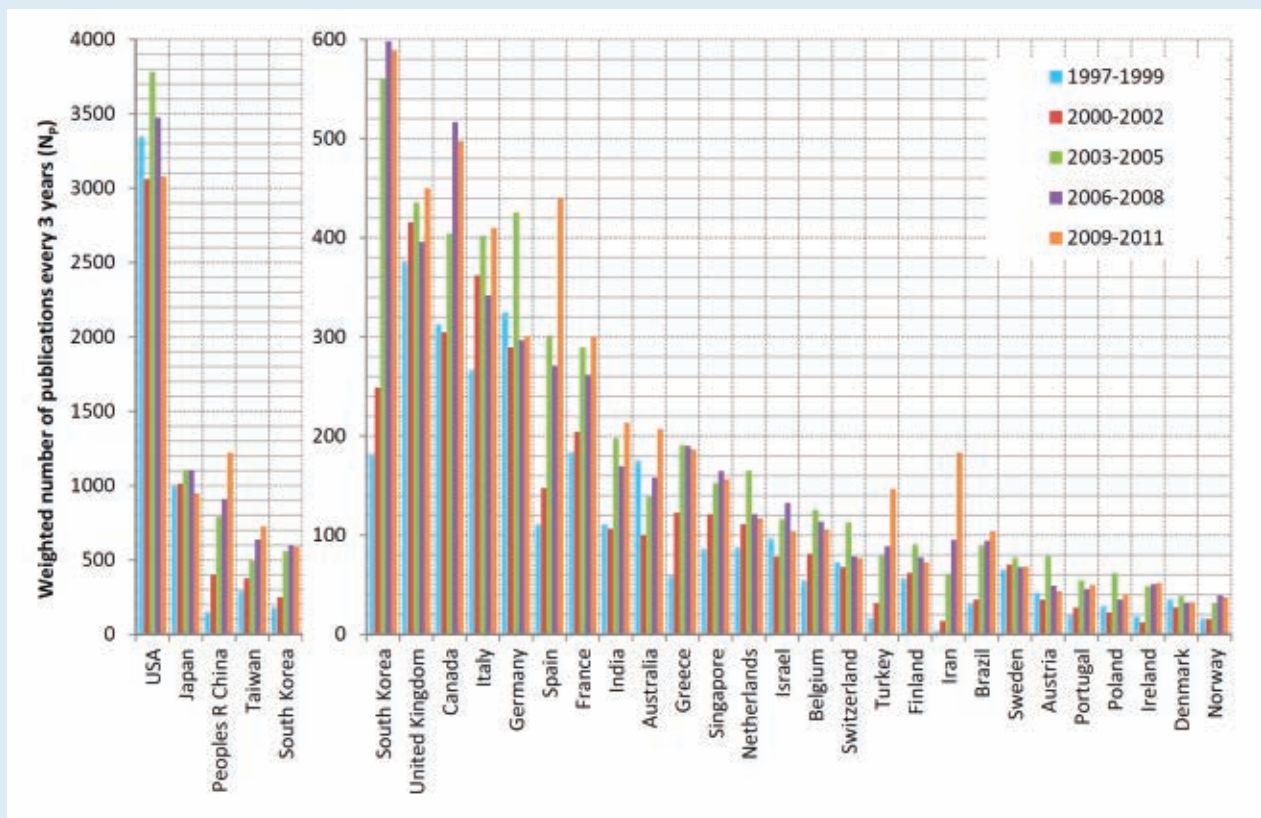
texture,” appearing 265,000 times. Some 28% were used in more than one research article but only 4% in more than 10. The 100 most used keywords accounted for 18% of the total, and the 5,000 most used keywords accounted for 61%, revealing the heterogeneity of research topics in this field of study.

Curiously, however, few terms used in the definition of “computer science, hardware architecture” were also used as keywords. Even when all occurrences (or as a single keyword or part of compound keyword) of board, bridge, SPARC, RISC, or CISC were calculated, their use was found to be limited (<100 articles each). Other terms,

including bus, interface, storage, router, and scalable were found in 100–400 papers. Hardware was mentioned in just over 700 articles. Some keywords were, however, used heavily, including network (12,643), power (3,288), memory (1,798), processor (1,236), and parallel (1,460).

The keywords in the top positions

Figure 3. Change in number of collaboration-weighted papers (N_p) published in each three-year period by the main research countries.



were relatively general, including algorithm, system, design, network, performance, model, optimization, circuit, and architecture.

Among the 50 most common keywords (see Figure 1), the rate of growth in the use of some of them stagnated or declined, including neural network, simulation, scheduling, routing, fault tolerance, VLSI, Internet, performance evaluation, cryptography, interconnection network, and quality of service. Notable declines were also seen in ATM, distributed system, low-power design, high-level synthesis, parallel, petri net, hypercube, multiprocessor, multimedia, learning, logic synthesis, partitioning, and Boolean function.

The most notable growth in the study's final three years, 2008–2011, was associated with network terms, including wireless sensor network (position 14), ad hoc network (position 18), wireless network (position 20), sensor network (position 27), reliability, security, CMOS, low power, FPGA, protocol, stability, scheme, communication, power, management, classification, processor, theory, technology, reduction, measurement, authentication, framework, wireless, signal, transmission, process variation, privacy, oscillator, throughput, network on chip, energy efficiency, peer to peer, and phase noise. The words used in compound keywords were analyzed separately; for more, see the Appendix accompanying this article in ACM's Digital Library.

Change in global research activity. The number of research papers on “computer science, hardware architecture” experienced irregular incremental change over the study period. Article production experienced a decline over the first four of these years, from 3,159 publications in 1997 to 2,770 publications in 2001. Production, then held steady at 3,500–4,000 papers per year. The total increase in production over the study period was approximately 30%. It appears to be an area of irregularly increasing research publishing.

The percentage of papers involving international collaboration more than doubled from 10% in 1997 to 26% in 2011 (see Figure 2a). The number of authors per article also gradually increased from 2.3 in 1997 to 3.3 in 2011, a gain of 43%. The average number of

research centers involved per paper also increased, from 1.5 in 1997 to 1.8 in 2011, a gain of 23% (see Figure 2b), reflecting the increasingly global nature of scientific research in the field.

The YIF value also saw modest growth, increasing from 0.51 in 1997 to 1.10 in 2011 (see Figure 2c). However, the number of citations per article decreased gradually (see Figure 2c); newer articles naturally receive fewer citations within a given period, as less time elapsed over which other authors are able to cite them. This was taken into account when assessing researcher merit.

Analysis of the languages used in the articles found English was dominant, with 99.85% of all papers examined in it. This finding underscores the importance of English as the language of scientific communication.

Change in geographic and institutional distribution of publications. The 10 most productive countries were responsible for 76% of scientific production 1997–2011 in this research area; the 30 most productive countries were

responsible for 93%. The U.S. alone was responsible for 32.5%, followed by several Asian countries (4%–10%). Surprisingly, of these 30 most productive countries, Denmark and Norway lagged far behind, with just 0.3% of the total production each (see Table 1).

U.S. and Asian dominance was reflected in the ranking of the most productive research centers (see Table 2). The production of the first 30 centers ranged from 0.4% to 1.2% of all articles. Among the top 100 centers, 44 were in the U.S., 11 each from China and Japan, eight from Taiwan, six from Canada, five from South Korea, four from Italy, two from Singapore, Israel, and Greece, and one from the U.K., Switzerland, the Netherlands, India, and Belgium, respectively. Curiously, countries that achieved a high ranking, including France, Germany, and Spain, had no outstanding, highly productive centers. Rather, their national output represented the sum of many centers.

The change in the number of collaboration-weighted articles (N_p)

Table 1. Scientific production indicators for the main countries (average values 1997–2011).

Country	1997–2011						
	N_p	N_p (%)	Col (%)	YIF	N_{ci}	N_A	N_{RI}
USA	16743	32,5%	26,0%	1,22	16	3,1	1,9
Japan	5169	10,0%	14,1%	0,48	4	3,2	1,6
Peoples Republic of China	3474	6,8%	36,2%	1,03	7	3,4	1,9
Taiwan	2527	4,9%	14,6%	0,84	7	2,8	1,7
South Korea	2179	4,2%	25,6%	0,72	4	3,2	1,8
Unite Kingdom	2073	4,0%	41,9%	1,06	10	3,1	1,9
Canada	2035	4,0%	45,7%	1,04	9	3,0	1,9
Italy	1783	3,5%	35,9%	1,04	10	3,5	1,9
Germany	1637	3,2%	43,5%	1,08	11	3,5	2,0
Spain	1270	2,5%	35,4%	1,01	6	4,0	1,9
France	1239	2,4%	50,6%	1,05	10	3,4	2,2
India	798	1,6%	34,6%	0,90	7	3,1	1,9
Australia	779	1,5%	47,0%	1,07	10	2,9	1,9
Greece	749	1,5%	34,3%	1,09	6	3,5	1,9
Singapore	682	1,3%	41,8%	1,19	11	3,0	1,8
Netherlands	602	1,2%	49,0%	0,99	12	3,3	2,1
Israel	528	1,0%	49,9%	1,33	15	3,1	2,1
Belgium	480	0,9%	46,3%	0,95	9	3,8	2,1
Switzerland	409	0,8%	58,0%	1,35	14	3,8	2,2
Turkey	363	0,7%	35,3%	0,90	8	2,5	1,7
Finland	359	0,7%	33,3%	1,01	15	3,4	1,8
Iran	357	0,7%	30,8%	0,61	3	2,8	1,7
Brazil	355	0,7%	42,7%	1,02	7	3,5	2,0
Sweden	349	0,7%	50,3%	0,91	12	3,4	2,1
Austria	248	0,5%	47,8%	1,00	11	3,3	2,0
Portugal	196	0,4%	37,7%	0,94	9	3,1	2,0
Poland	188	0,4%	45,2%	0,99	8	3,0	1,9
Ireland	181	0,4%	44,9%	1,17	11	3,4	1,9
Denmark	165	0,3%	50,9%	1,09	9	3,2	2,0
Norway	140	0,3%	48,5%	0,98	7	3,2	2,1

produced by different countries was notably different. When the study period was divided into consecutive three-year slots—1997–1999, 2000–2002, 2003–2005, 2006–2008, and 2009–2011—the U.S., Japan, the U.K., Italy, and Germany showed rather constant N_p values (see Figure 3). In contrast, some countries experienced tremendous growth; for example, N_p multiplied eight times for China, raising it from 11th to second position in our ranking.

China’s increase in total R&D expenditure—\$14,733 million in 1997 to \$208,172 million in 2011¹⁰—clearly seemed to reflect its production and impact in the field. N_p also grew in Taiwan and South Korea, which in the final three-year period both ranked among the five top producers. Other countries that increased their N_p in-

clude Canada, Iran, and Spain.

Developments at the national level were reflected in national research centers. The Tsinghua University (China) thus achieved first place in the N_p ranking in the final three-year period, the Chinese Academy of Sciences third, and the National Tsing Hua University, the National Cheng Kung University, the National Chiao Tung University, and the National Taiwan University (all from Taiwan) fourth, eighth, ninth, and 10th place, respectively. On a smaller scale, South Korea’s Seoul National University, Korean Advanced Institute of Science & Technology, and Korea University all entered the top 20. The most prominent positions still belonged to U.S. research centers, including the University of Texas, second, the University of Illinois, sixth, Purdue University, seventh, and the Georgia In-

stitute of Technology, 14th. Singapore’s Nanyang Technological University was fifth, the U.K.’s Imperial College of Science, Technology and Medicine, 11th, Japan’s Tokyo Institute of Technology, 13th, Waseda University, 17th, Canada’s University of Waterloo, 16th, and Italy’s Politecnico di Torino, 19th.

The analysis application also found a significant fall in N_p for IBM Corporation, with the company sliding out of the top 10 due largely to its division into several specific research centers (such as IBM Systems & Technology Group and IBM T.J. Watson Research Center). A striking decline was also seen for Intel Corporation (U.S.), the Massachusetts Institute of Technology (U.S.), the University of California, Berkeley (U.S.), Carnegie Mellon University (U.S.), Stanford University (U.S.), the Indian Institute of Technology (India), and the University of Maryland (U.S.), none in the top 20 during the study’s final three-year period.

Collaborations. The analysis application also helped weigh the complexity and size of research networks, analyzing indicators of research collaboration at all scales. The number of authors per article varied significantly, depending on country. In general, European countries had more authors per article (N_A), with the most in Spain (4.0), Belgium (3.8), and Switzerland (3.8). Others averaged fewer than three authors per article, including Australia, Iran, Taiwan, and Turkey (see Table 1). Moreover, the average number of research institutions per article (N_{RI}) was similar in most countries, close to two.

Most of the most-productive centers included from three to four authors per article (N_A) and between 1.5 and 2.5 research institutions per article (N_{RI}) (see Table 2). The U.S. and Chinese centers often reflected the highest values for collaboration. In absolute terms, the highest number of collaborations involved major U.S. research centers (such as IBM and Intel) and major U.S. universities (see Table 3), with the strongest between Purdue University and the University of Iowa, which together produced 90 papers. Papers involving IBM had a high average number of authors but low mean number of participating research centers, indicating involvement of a large number of company researchers (see Table 2).

Table 2. Scientific production indicators for the main research institutions (average values 1997–2011).

Research Institution	Country	1997–2011						
		N_p	N_p (%)	Col (%)	YIF	N_{CI}	N_A	N_{RI}
IBM Corp.	USA	633	1.2%	22.6%	1.55	22	4.7	2.1
University of Texas	USA	465	0.9%	21.6%	1.25	12	3.2	2.1
University of Illinois	USA	388	0.8%	22.4%	1.26	17	3.4	2.3
Nanyang Technological University	Singapore	338	0.7%	29.8%	1.02	10	2.9	1.6
Tokyo Institute of Technology	Japan	323	0.6%	8.9%	0.45	3	3.0	1.6
Tsinghua University	Peoples Republic of China	318	0.6%	38.2%	0.89	4	3.8	1.9
National Chiao Tung University	Taiwan	310	0.6%	13.0%	0.90	9	2.9	1.8
Intel Corp.	USA	289	0.6%	21.5%	1.08	13	3.6	2.4
MIT	USA	287	0.6%	21.6%	1.42	35	3.3	2.2
Georgia Institute of Technology	USA	287	0.6%	18.4%	1.28	30	3.0	1.9
Purdue University	USA	285	0.6%	19.5%	1.32	13	3.2	2.2
University of California, Berkeley	USA	277	0.5%	28.7%	1.29	32	3.6	2.4
Chinese Academy of Sciences	Peoples Republic of China	277	0.5%	25.6%	0.77	5	3.5	2.0
Carnegie Mellon University	USA	277	0.5%	20.9%	1.30	21	3.4	2.2
Stanford University	USA	260	0.5%	27.6%	1.39	29	3.5	2.2
Indian Institute of Technology	India	259	0.5%	33.1%	0.87	7	3.0	1.8
National Taiwan University	Taiwan	255	0.5%	20.2%	1.01	13	3.1	1.9
University of Maryland	USA	251	0.5%	19.8%	1.23	18	3.1	2.1
University of California, San Diego	USA	245	0.5%	27.3%	1.16	15	3.7	2.4
University of Southern California	USA	245	0.5%	22.7%	1.19	28	3.2	2.1
University of California, Los Angeles	USA	242	0.5%	24.9%	1.13	25	3.6	2.2
Korea Advanced Institute of Science & Technology	South Korea	237	0.5%	20.2%	0.86	5	3.2	1.9
University of Michigan	USA	236	0.5%	15.6%	1.25	17	3.6	2.1
University of Waterloo	Canada	231	0.4%	48.1%	1.28	10	2.9	2.0
Osaka University	Japan	226	0.4%	8.5%	0.48	4	3.6	1.9
National Cheng Kung University	Taiwan	216	0.4%	14.9%	0.91	5	3.0	1.7
National Tsing Hua University	Taiwan	214	0.4%	19.1%	0.97	7	3.1	1.9
National University of Singapore	Singapore	211	0.4%	47.0%	1.39	15	3.1	1.9
University of Tokyo	Japan	205	0.4%	22.6%	0.63	6	3.2	2.1
Seoul National University	South Korea	203	0.4%	26.5%	0.84	7	3.5	1.9

Each country's local conditions influenced the amount of its international collaboration. The countries with the most international research collaboration, or Col (%), were Switzerland (58.0%), Denmark (50.9%), France (50.6%), Israel (49.9%), and Sweden (44.6%). Some countries, including Japan and Taiwan, were below 20% (see Table 1).

The U.S. had the highest number of collaborations, producing very large total output. Its associates were all over the world, but special relationships

were seen with China (776 papers in common), Canada (588), South Korea (445), and Germany (399); for more, see the online Appendix. For some countries, work with the U.S. represented most of their international collaborations. Other important collaborations, though on a smaller scale, included those between China and Canada (182), the U.K. (169), Japan (150), and Singapore (141).

International collaboration accounted for approximately 50% of the total number of research papers pro-

duced by some centers (such as the University of Waterloo and National University of Singapore) but less than 10% of the production of others (such as Tokyo Institute of Technology and Osaka University).

Impact. Substantial differences were seen between countries in terms of mean YIF value and number of citations received (NCI). The huge production by the U.S. also had high impact; it was indeed one of the countries with the highest average YIF (1.22) and NCI (16), well above the values recorded for

Table 3. Number of collaboration-unweighted articles by the most productive research centers.

Research Institution	Ibm Corp	Univ Texas	Univ Illinois	Nanyang Technol Univ	Tokyo Inst Technol	Tsinghua Univ	Natl Chiao Tung Univ	Intel Corp	Mit	Georgia Inst Technol	Purdue Univ	Univ Calif Berkeley	Chinese Acad Sci	Carnegie Mellon Univ	Stanford Univ	Indian Inst Technol	Natl Taiwan Univ	Univ Maryland	Univ Calif San Diego	Univ So Calif	Univ Calif Los Angeles	Korea Adv Inst Sci & Tech	Univ Michigan	Univ Waterloo	Osaka Univ	Natl Cheng Kung Univ	Natl Tsing Hua Univ	Natl Univ Singapore	Univ Tokyo	Seoul Natl Univ
IBM Corp.	26	35					1	19	14	9	12	19		26	14	4	3	7	8	5	18	27			2	2		1	3	
University of Texas	26		17	2	1	6	7	32	6	5	5	11	1	11	14	3	5	3	3	5	4	1	6		1		1	2		
University of Illinois	35	17		1		2	2	31	8	4	15	4	4	10	11	2	3	3	14	6	5	1	5	2	1		1	3	7	
Nanyang Technological University		2	1			2	1		2	1	1		6	1		2					2		2				10			
Tokyo Institute of Technology		1								2		1									1				7				9	
Tsinghua University		6	2	2				2		2	3	2	15	3	5		1	2	17	4	12		10			1	2			
National Chiao Tung University	1	7	2	1				1		1	2	1	1				22	2		2						3	7	1		
Intel Corp.	19	32	31			2	1		2	19	39	11	1	15	7	2	2	4	5	10	7	2	18	10			7		1	
MIT	14	6	8	2				2		10		17		10	14	1		5	4	5	6	4	5	3				4		
Georgia Institute of Technology	9	5	4	1	2	2	1	19	10		3	4		2	4	2		2	4	1	4	3	1			1			1	
Purdue University	12	5	15	1		3	2	39		3		1	1	3	4	2		1	4	1	6		1	9			3	1		
University of California, Berkeley	19	11	4		1	2	1	11	17	4	1		1	14	21	4	4	8	12	6	17		9	2			1		2	
Chinese Academy of Sciences		1	4	6		15	1	1			1	1					1		1	3		1	2				5	1		
Carnegie Mellon University	26	11	10	1		3		15	10	2	3	14			9	2	1	7	4	6	3	3	10		2			3	3	1
Stanford University	14	14	11			5		7	14	4	4	21				3	6	1	7	10	6		4	1	1			1	1	
Indian Institute of Technology	4	3	2	2				2	1	2	2	4		2	3				2	7		1	1	3				4		
National Taiwan University	3	5	3			1	22	2				4	1	1	6			1	3	2		1				3	7		1	1
University of Maryland	7	3	3			2	2	4	5	2	1	8		7	1	2	1		4	8	10		12	3				1		
University of California, San Diego	8	3	14			17		5	4	4	4	12	1	4	7	7	3	4		8	12	2	6	1	1		1	1	2	
University of Southern California	5	5	6			4	2	10	5	1	1	6		6	10		2	8	8		15		3	1		2		3		
University of California, Los Angeles	18	4	5	2	1	12		7	6	4	6	17	3	3	6	1		10	12	15		9	1							
Korea Advanced Institute of Science & Technology		1	1					2	4	3				3		1	1		2			1					4	1	2	9
University of Michigan	27	6	5					18	5	1	1	9	1	10	4	3		12	6	3	9	1		3				2	4	
University of Waterloo			2	2		10		10	3		9	2	2		1			3	1	1	1		3				1	2		
Osaka University		1	1		7									2	1				1										4	1
National Cheng Kung University	2					1	3			1							3			2							5			
National Tsing Hua University	2	1	1				7	7			3	1					7		1			4		1		5				
National University of Singapore		2	3	10		2	1		4		1		5	3	1	4		1	1	3		1	2	2						
University of Tokyo	1				9								1	3			1					2			4				2	
Seoul National University	3		7					1		1		2		1	1		1		2			9	4		1				2	

the other most productive countries. Israel and Switzerland also managed reasonable figures (see Table 1). In contrast, Iran, Japan, and South Korea had below-average YIF values and N_{CI} .

Though YIF values of the countries improved over time (see Figure 4), rates differed. YIF for China, the U.K., Germany, Singapore, the Netherlands, Israel, and Switzerland grew more strongly, exceeding the world average. In contrast, Japan and Iran had below-average YIF for the entire study period, experiencing only limited growth toward the end. The average number of citations (N_{CI}) decreased over time for all countries; for more, see the online Appendix. The high citation values of Finland and Ireland are explained by their production of a small number of high-impact articles.

Though many U.S. research centers failed to maintain their growth in production and lost ground in the productivity ranking, their research work still had the greatest scientific impact. High YIF values and NCI were noteworthy for IBM, MIT, Stanford, Carnegie

Mellon, the University of California, Berkeley, the Georgia Institute of Technology, the University of Southern California, and the University of California, Los Angeles (see Table 2). The National University of Singapore also had a high YIF and NCI. Conversely, the lowest values were returned by research centers in Japan and South Korea.

The difference in research impact could be explained by the preference of the journals in the category. In the final five years of the study period, the U.S. was the main contributor to 75% of the related journals, with many more articles published in high-impact journals than in low-impact journals, explaining the high values of the country's quality indicators. The high-impact values achieved by several other countries over the same period can be explained the same way. The low-impact values of Iran, Japan, South Korea, and Turkey are explained by a significant percentage of their articles being published in low-impact journals.

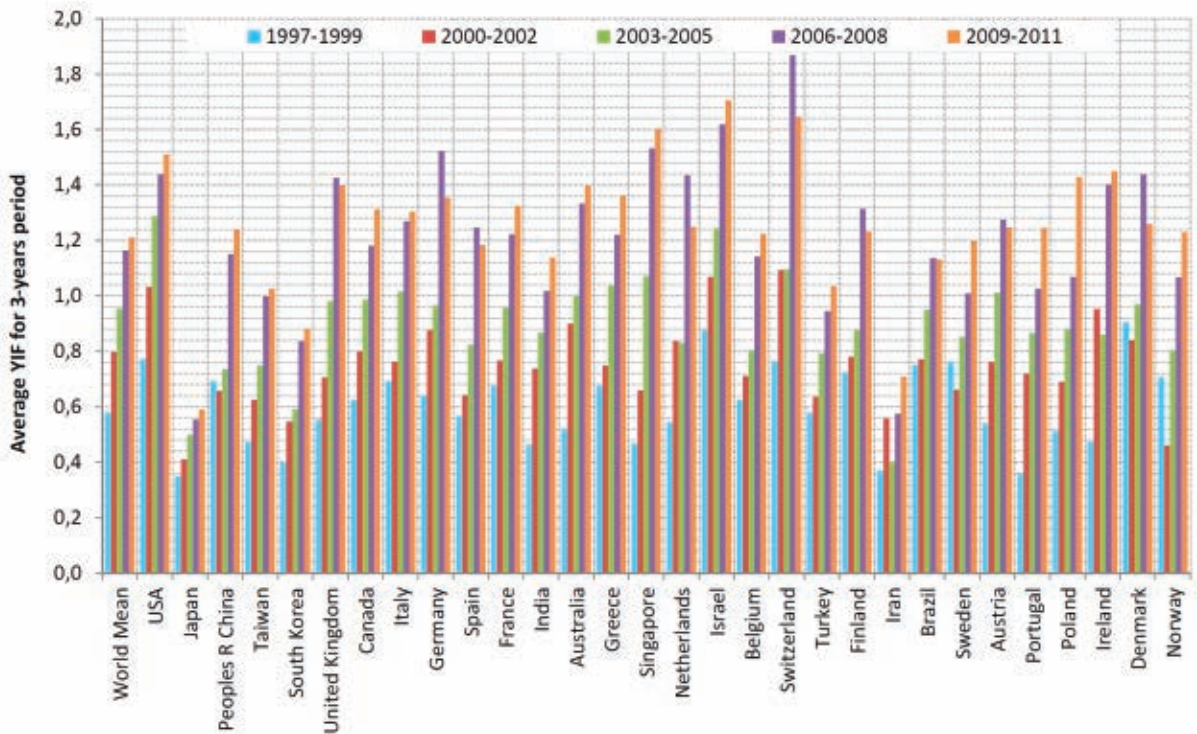
Among research centers, approximately 50% of the papers published

by IBM appeared in the *IBM Journal of Research and Development*. Over 20% of the production of MIT, the University of California, Berkeley, and Stanford appeared in *Communications*, while Purdue, Carnegie Mellon, the National Taiwan University, the University of California, San Diego, the University of California, Los Angeles, and the University of Michigan commonly published in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. These rates of publication in these and other important journals justify the high impact rankings of the research centers. In contrast, many papers produced by Japanese research centers, including Tokyo Institute of Technology, 86%, Osaka University, 78%, University of Tokyo, 48%, and Waseda University, 86%, appeared in *IEICE Transactions on Fundamentals of Electronics*, which has limited international visibility.

Conclusion

The study was an overview of the research activity in the field of hard-

Figure 4. Change in average YIF for the most productive countries.



ware architecture for the years 1997 to 2011. More than 51,000 articles were analyzed using specially developed in-house software, including keyword evolution, weighted productivity with respect to number of research centers per article, impact of each article, and collaborations at multiple scales.

Keyword analysis found that the topics of interest evolve. Most technical terms used in the definition of “computer science, hardware architecture,” including board, bridge, SPARC, RISC, CISC, bus, and router interface, were used only rarely by researchers during the study period. The years toward the end of the period saw a remarkable increase in the amount of research in networks and sensors, including wireless sensor networks and ad hoc networks; also showing notable increase were security, stability, energy, chips, optical, and education. The study also identified keywords of lesser interest, including neural networks, ATM, and VLSI.

The global average values for productivity, collaborations, and impact indicators provide a framework for assessing the research activity of different centers, and thereby the merits of the researchers themselves, in a realistic context.

Scientific production and the impact of the research grew unevenly. Scientific production was also concentrated, with the 30 most productive countries responsible for 93% of world production.

The U.S. dominated the field (32.5% of the total collaboration-weighted production) with high research impact, much higher than most other countries.

Of the 100 most productive research centers, 44 were in the U.S. Although many U.S. centers failed to maintain the growth of their production, their research work generally had greater scientific impact.

Italy, Germany, Japan, and the U.K. all returned relatively constant levels of output, while China was found to be a growing force. However, the impact of research by these countries and their research centers showed differences due to the internationalization and impact of the journals in which they published.

The study also produced detailed information on existing collaborations between countries and research



The years toward the end of the period saw a remarkable increase in the amount of research in networks and sensors, including wireless sensor networks and ad hoc networks; also showing notable increase were security, stability, energy, chips, optical, and education.



centers. Collaboration levels increased at all scales, indicating more complex research networks. Local conditions influenced the amount of international collaboration in which each country engaged, with U.S. research centers collaborating most with the rest of the world. 

References

1. Abrazah, A. and Wee, M.C. Malaysia's computer science research productivity based on publications in the Web of Science, 2000–2010. *Malaysian Journal of Library & Information Science* 16, 1 (Apr. 2011), 109–124.
2. Bar-Ilan, J. Web of Science with the Conference Proceedings Citation Indexes: The case of computer science. *Scientometrics* 83, 3 (June 3, 2010), 809–824.
3. Cañas-Guerrero, I., Mazarrón, F.R., Calleja-Perucho, C., and Pou-Merina, A. Bibliometric analysis in the international context of the 'construction & building technology' category from the Web of Science database. *Construction and Building Materials* 53 (Feb. 2014), 13–25.
4. Cañas-Guerrero, I., Mazarrón, F.R., Pou-Merina, A., Calleja-Perucho, C., and Díaz-Rubio, G. Bibliometric analysis of research activity in the 'agronomy' category from the Web of Science, 1997–2011. *European Journal of Agronomy* 50 (Oct. 2013), 19–28.
5. Cañas-Guerrero, I., Mazarrón, F.R., Pou-Merina, A., Calleja-Perucho, C., and Suárez-Tejero, M.F. Analysis of research activity in the field 'Engineering, Civil' through bibliometric methods. *Engineering Structures* 56 (Nov. 2013), 2273–2286.
6. De Sutter, B. and Van den Oord, A. To be or not to be cited in computer science. *Commun. ACM* 55, 8 (Aug. 2012), 69–75.
7. Franceschet, M. Collaboration in computer science: A network science approach. *Journal of the American Society for Information Science and Technology* 62, 10 (Oct. 2011), 1992–2012.
8. Ibanez, A., Bielza, C., and Larranaga, P. Analysis of scientific activity in Spanish public universities in the area of computer science. *Revista Espanola De Documentacion Cientifica* 36, 1 (Mar. 2013).
9. Ma, R.M., Ni, C.Q., and Qiu, J.P. Scientific research competitiveness of world universities in computer science. *Scientometrics* 76, 2 (Aug. 2008), 245–260.
10. The Organisation for Economic Co-operation and Development (OECD); <http://stats.oecd.org/>
11. Rojas-Sola, J.I. and Jorda-Albinana, B. Bibliometric analysis of Venezuelan publications in the computer sciences category of the JCR Data Base (1997–2007). *Interciencia* 34, 10 (Oct. 2009), 689–695.
12. Xie, S.D., Zhang, J., and Ho, Y.S. Assessment of world aerosol research trends by bibliometric analysis. *Scientometrics* 77, 1 (Oct. 2008), 113–130.

Virender Singh (padocviren@gmail.com) is a Ph.D. candidate at the Polytechnic University of Madrid, Spain, and manager of software test engineering at Garmin, Wuerzburg, Germany.

Alicia Perdígones (alicia.perdigones@upm.es) is an associate professor in the Polytechnic University of Madrid, Spain, and member of the Education Innovation Group of Electric and Automatic Technologies for Rural Engineering, Madrid, Spain.

José Luis García (jose.luis.garcia@upm.es) is a full professor in the Polytechnic University of Madrid, Spain, and coordinator of the Education Innovation Group in Electric and Automatic Technologies for Rural Engineering, Madrid, Spain.

Ignacio Cañas-Guerrero (i.canas@upm.es) is a full professor in the Polytechnic University of Madrid, Spain, and coordinator of the PADOC Research Group, Madrid, Spain.

Fernando R. Mazarrón (f.ruiz@upm.es) is an assistant professor of the Polytechnic University of Madrid, Spain, member of the Education Innovation Group in Electric and Automatic Technologies for Rural Engineering, Madrid, Spain, and member of the PADOC Research Group, Madrid, Spain.

DOI:10.1145/2656334

The system should let users incrementally direct their search toward relevant, though not initially obvious, information.

BY TUUKKA RUOTSALO, GIULIO JACUCCI,
PETRI MYLLYMÄKI, AND SAMUEL KASKI

Interactive Intent Modeling: Information Discovery Beyond Search

COMBINING INTENT MODELING and visual user interfaces can help users discover novel information and dramatically improve their information-exploration performance.

Current-generation search engines serve billions of requests each day, returning responses to search queries in fractions of a second. They are great tools for checking facts and looking up information for

which users can easily create queries (such as “Find the closest restaurants” or “Find reviews of a book”). What search engines are not good at is supporting complex information-exploration and discovery tasks that go beyond simple keyword queries. In information exploration and discovery, often called “exploratory search,” users may have difficulty expressing their information needs, and new search intents may emerge and be discovered only as they learn by reflecting on the acquired information.^{8,9,18} This finding roots back to the “vocabulary mismatch problem”¹³ that was identified in the 1980s but has remained difficult to tackle in operational information retrieval (IR) systems (see the sidebar “Background”). In essence, the problem refers to human communication behavior in which the humans writing the documents to be retrieved and the humans searching for them are likely to use very different vocabularies to encode and decode their intended meaning.^{8,21}

Assisting users in the search process is increasingly important, as everyday search behavior ranges from simple look-ups to a spectrum of search tasks²³ in which search behavior is more exploratory and information needs and search intents uncertain and evolving over time.

We introduce interactive intent modeling, an approach promoting resourceful interaction between hu-

» key insights

- **Current search engines offer limited assistance in complex search tasks; users are distracted by having to focus their cognitive effort on finding navigation cues rather than on learning and selecting relevant information.**
- **Interactive intent modeling enhances human information exploration through computational modeling (visualized for interaction), helping users search and explore via user interfaces that are highly functional but not cluttered or distracting.**
- **Interactive intent modeling can improve task-level information-seeking performance by over 100%.**



mans and IR systems to enable information discovery that goes beyond search. It addresses the vocabulary mismatch problem by giving users potential intents to explore, visualizing them as directions in the information space around the user's present position, and allowing interaction to improve estimates of the user's search intents.

Interactive intent modeling is based on two scientific principles (see Figure 1): *Visualization*. Visualizing the current search intent and directions in the information space; and

Adaptation. Interactive adaptation of the intent model, balancing exploration of the information space and exploitation of user feedback; the intent model must be able to rigorously handle uncertainty due to limited, possibly suboptimal, user feedback.

By visualizing query and data elements (such as keywords), this approach enables the system to show its understanding of user search intent to the user and also provide a view of available search directions around the user's current position in the information space. The initial evidence concerning user search intent is often limited. The intent model is thus manageable for the user only if the system is able to predict a sufficient subset of the potentially relevant intents. Given the visualization of the intent model and its relation to the information space, the user is able to provide feedback for the intent model, allowing the system to improve intent estimates on subsequent iterations, retrieve and rank data, and update the visualization of directions in the information space.

Interactive Intent Modeling Example

The SciNet system for scientific literature search (<http://augmentedresearch.hiit.fi/>) is an example of the two principles in interactive intent modeling (see Figure 2).²⁰ The system currently indexes more than 50 million scientific articles and is designed to assist users exploring information related to a particular research topic through rapid feedback loops and in making sense of the available information around the initial query context.^{14,20}

In the Figure 2 scenario, a user is trying to learn about “3D gestures” and types in the corresponding query. The user is visualized with an estimate of his or her present search intents, as well as potential intents, and directions in the information space on a radar screen. The user then navigates by

directly manipulating the estimated intents on the display.

Figure 2a is the system’s response to the initial query on 3D gestures, offering directions to, say, “video games,” “user interfaces,” “gesture recognition,” and “virtual reality.” In Figure

2b, the user has selected “gesture recognition” and is offered further options to continue the exploration to more specific topics (such as “nearest neighbor approach” and “hidden Markov models”) but also to more general topics (such as “pointing gestures”

and “spatial interaction”) estimated to be relevant for the user’s interaction history. The modeling is based on a fast online regression model that estimates task-level search intents.²⁰ The model estimates relevance related to potential search intent and uncertainty related to these estimates based on user feedback. Search intents are visualized as keywords, and selection of which intents to visualize is determined through the exploration-exploitation paradigm. The trick is to present to the user not only choices estimated to be most relevant but those for which the upper confidence bound is the greatest. The user decides whether to explore or exploit, as both relevant and uncertain keywords are visualized; for example, if the user first selects “gesture recognition” and then “hidden Markov models,” the system would then suggest specific hidden Markov model applications in gesture recognition, which would be exploitative, as they are estimated to be most relevant and also allow the user to continue to explore more uncertain directions (such as other computational techniques in gesture recognition).

While the idea of providing the user uncertain interaction options may be counterintuitive from a conventional

Figure 1. Exploring information with interactive intent modeling is based on two principles: visualizing current search intent and direction; and balancing exploration and exploitation of user feedback. The user’s cognitive effort is thus reduced, as it is easier to recognize items instead of having to remember them when reformulating queries.

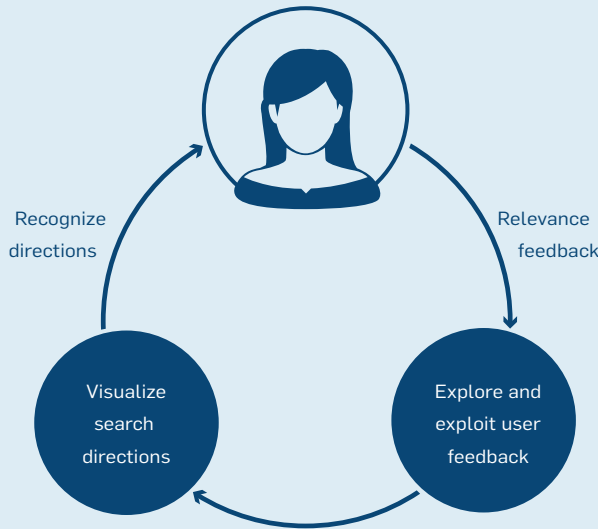


Figure 2. SciNet system search user interface.



(a) The user issues the query “3D gestures,” and the system visualizes an intent model on the radar screen consisting of potentially interesting intent as keywords and a ranked document list. The estimated intents, for which the results on the right side have been retrieved, are visualized for the user (inner darker-gray area). The angular distance corresponds to similarity of intent and the radial distance from the center to relevance. Predicted potential future intents, which help users orient themselves on the radar, is visualized in the outer (lighter-gray) area. The user provides positive feedback by dragging keywords closer to the center and negative feedback by dragging them further away. Multiple keywords can be dragged in each iteration. Online learning methods make it possible for the system to respond in less than one second.



(b) The user has increased the relevance of “gesture recognition” by dragging the corresponding keywords to the center of the radar screen. The system then visualizes new estimated relevant intents as a set of keywords (such as “pattern recognition,” “pointing gestures,” “recognition rates,” “nearest neighbor approaches,” and “hidden Markov models”).

IR perspective, which is based on the principle of maximizing relevance, our interactive intent modeling helps users overcome the vocabulary-mismatch problem, as the system provides them interaction resources to continuously direct the search and actively explore relevant, though not initially obvious, information. Experiments show user-task-level performance can be improved significantly.

To deliver such support, the interface provides a nonintrusive relevance-feedback mechanism in which the user pulls keywords closer to the center of the radar screen to increase their importance and pushes keywords away from the center of the radar screen to decrease their importance. The keywords can be enlarged with a fisheye lens that follows the mouse cursor anywhere on the radar screen. In response, the system updates intent visualization and search results. The radar screen's radial layout represents good balance between the amount of information shown and comprehensibility compared to alternative visualizations with lower or higher degrees of freedom that could make interaction with the visualization more difficult.

Interactive Visualization of Search Intent and Direction

The SciNet example demonstrates how visualization can be used to elicit feedback. Feedback can be targeted directly to the intent model (the inner circle of the radar screen in Figure 2) or to possible future directions (the outer rim of the radar screen in Figure 2). Due to the vocabulary-mismatch problem, users often have trouble expressing their needs as written queries and are likely to start their search with imprecise queries. Hence, interaction and feedback mechanisms that engage users to provide feedback on how to direct their search in the subsequent iterations are crucial. This is grounded in a well-known cognitive-science theory stating users find recognition easier than recall.³ It is usually easier for humans to recognize something they see than describe it from scratch.

However, increasing evidence from IR research supports the finding that while relevance feedback is useful in enabling systems to better serve user

Background

Recent behavioral studies show a large portion of user information-seeking activities are exploratory and characterized by the complex and evolving nature of user information needs.¹⁸ As a result, users face the problem of entering correct terms that describe their search intents so the desired information can be retrieved on subsequent iterations. This is one of the major findings of information-seeking research, as identified by Furnas et al.,¹³ Saracevic and Kantor,²¹ and Bates.^{7,8} All demonstrated human communication patterns are not likely well suited to creating written queries. For example, two study subjects in Furnas et al.¹³ favored the same search term with a probability smaller than 0.2, which was shown to lead to 80%–90% failure rates in many common search situations; Saracevic and Kantor²¹ and Zhao and Callan²⁵ later obtained similar results. These findings limit the success of various design methodologies for written-query-driven search interfaces, highlighting the importance of the user interface for modeling and discovering users' potential, yet unspecified, search intent.

The simplistic search interfaces of the current generation of search engines not only force users to focus their cognitive effort on the discovery of information potentially relevant for their search intents, in most cases, they deliver only the search result listing that supports this activity. A system's ability to discover potentially relevant search intents could be better supported by search engines in the first place. As user expressions of information needs are often suboptimal, reflecting only limited evidence of their real search intents, opportunities for IR software developers focus on figuring out ways to get users to negotiate with the search engines to better capture their needs. This goal of utilizing the human information-processing system more effectively in the search process has led researchers to combine work in human-computer interaction, information retrieval, and machine learning to overcome barriers between techniques that rely on only human information-processing capabilities and computational methods employed by search engines.

Other researchers have investigated adaptive and interactive search user interfaces and their effect on retrieval performance. For example, Hearst et al.¹⁶ developed a variety of search user interfaces that use filtering and visualization techniques ranging from hierarchical, faceted metadata²⁴ and search result clustering¹⁶ to visualization of the similarity between query term and result item.¹⁵ Marchionini¹⁸ proposed user interfaces to support exploratory search, including browsing and retrieving video content. While these techniques are all highly functional and have shown increased search effectiveness, they exploit only the information already found by the search engine in response to a user's query. They did not take into account that the initial expression and future refinement of information needs are often suboptimal,¹² especially when users are unfamiliar with the domain and its vocabulary.

Moreover, users are learning the search vocabulary as they make sense of the information space during the search process; for example, if a user searches for "search engines," techniques that exploit the results already found by the search engine may limit the user's options for exploring beyond the initially found results, as the user is not presented different words for the same highly relevant additional topics (such as "information retrieval" and "information seeking behavior"). Despite these findings, the objective of much current IR system theory is toward systems where the user has limited and often only reactive involvement in the search process.

search intent, in most cases users do not in fact use feedback mechanisms.¹⁸ This observation is related to two other cognitive science findings: users find it easier to recognize smaller units than more complex ones, and it is easier for them to make sense of information relative to a reference point than in isolation.⁶ Assessing the relevance of a full document may be an even more demanding task than formulating a new query.

Information visualization can turn laborious relevance assessment into a more fluent recognition task; for example, visualizing essential document

content can be faster for recognizing important directions toward finding relevant information than forcing users to read this information from an original document.^{1,17}

Recent visualizations applied in search tasks support sense making of bibliographic data by, for example, incrementally interactively exploring networks of data.¹⁰ While these systems show the importance of visualization for sense making, they are limited by not allowing users to negotiate the intent model with the system, allowing them only to explore information through direct links already present in

the network data.¹

By visualizing potential intents, an IR system can give users a spectrum of choices in a form suitable for the human visual system to process rapidly, even when the data changes dynamically as interaction occurs. Such an interface requires advanced data-driven visualizations that can be computed online. Moreover, a visualization should not contain only information already familiar to the user that would be good for recognition but lead to the intent model getting stuck in a “context bubble.” Instead, unseen parts of the information space must be offered to the user, facilitating sense making through the relation of these parts to already familiar information when possible.

Balancing Exploration and Exploitation

Given the evolutionary nature of search, as demonstrated in our example search scenario involving SciNet, it is important to not only exploit the feedback elicited from the user but balance it with exploration. Users must be able to focus on a specific location in the information space (exploit) and be able to broaden their search through more general areas (explore).

This insight is particularly important for users exploring information with which they are not familiar. Users often suffer from what psychologists call “anchoring,” or the tendency to make insufficient adjustments to initial values when judging under conditions of uncertainty.²² Users may thus tend to refrain from abandoning their initial expression of their information needs or from adjusting them very much, causing subsequent expression of information needs to be biased toward their current knowledge. This bias reduces the likelihood they will discover something novel.

This behavioral finding has consequences for machine-learning approaches to modeling search intent. A promising direction for predicting intent while still allowing users to be in control of the search process comes from machine-learning methods that learn online. Online learning methods are able to update models one observation at a time so future predictions can be made immediately when feedback is



Online learning methods are able to update models one observation at a time so future predictions can be made immediately when feedback is received.



received. The goal in online learning for search is to predict the relevance of content interactively, meaning that soon after prediction is made, the judgment of its usefulness is received from user feedback. This information can then be used by the IR system to refine the prediction hypothesis used by the method.

Standard machine learning for online prediction does not solve the problem of discovering what interaction options are most useful in allowing the learning method to improve its estimates and therefore create visualizations for the user. Straightforward exploitation by choosing the directions currently estimated to be most relevant could lead to converging to suboptimal goals and locking users in context bubbles predetermined by the user’s limited prior knowledge; for example, a user searching for “3D gestures” might never explore “pointing gestures,” as the initial query scope could already be too specific to allow such exploration. A promising solution for collecting feedback that also allows exploration is the “exploration-exploitation” paradigm of reinforcement learning.⁵ In it, the model and its environment (the user) form an online loop, and learning involves finding a balance between exploration (showing items from uncharted information space for feedback) and exploitation (showing items most likely to be relevant, given the current user intent model).

Users can thus be assisted to direct their searches under conditions of uncertainty by learning intent models online based on feedback they give about the models. Due to the limited and imperfect feedback available, the amount of uncertainty about user intent can be substantial. It is therefore important for an IR system to use models capable of handling uncertainty in a rigorous manner. Probabilistic online learning models can be used in the exploitation part of the exploration-exploitation dilemma; moreover, as the models are probabilistic, they quantify uncertainty related to the estimates, using it to determine the optimal alternative choices to be visualized for users.⁵

From Effectiveness to Task Performance

When studying IR systems designed to negotiate user search intent, it is important to realize the utility these

systems improve is not retrieval effectiveness at a query-response level but task-level performance. Within interactive IR systems, users are often required to do more work to complete their tasks, and, while some of it may be wasted, they may be more successful in correcting their initially sub-optimal actions. To get a complete picture of performance, two aspects of IR systems should be measured simultaneously: IR system effectiveness, given a complete description of an information need; and human task performance, given the system's interaction modes.

The SciNet system was recently studied in task-based experiments in which users were given 30 minutes to solve research tasks using IR systems operating on a database of more than 50 million scholarly articles. We compared a system setup with interactive intent modeling against a conventional IR system with a list-based visualization and interaction with typed queries. We quantified the quality of retrieved information, adoption of the visualization, and feedback mechanisms separately.^{14,20} We found interactive search intent modeling significantly improved users task performance. We also found the task outcomes graded more highly by experts, and the search user interface enhanced interaction without compromising task execution time. We attributed the improved task performance to the improved quality of retrieved information and to the improved visualizations and interaction modes offered by the system. In particular, interactive intent modeling increased recall of novel information without losing precision. This performance demonstrates the power of the interactive intent modeling technique in supporting exploration and discovery of information that can be difficult to find with systems that rely on conventional search user interfaces (see the table here).

Making Intent Modeling Ubiquitous
Engaging users to interact with IR systems is crucial for such systems to be able to offer better interaction modes and reduce uncertainty related to user expression of search intent. Despite significant improvement in user task performance on the example SciNet

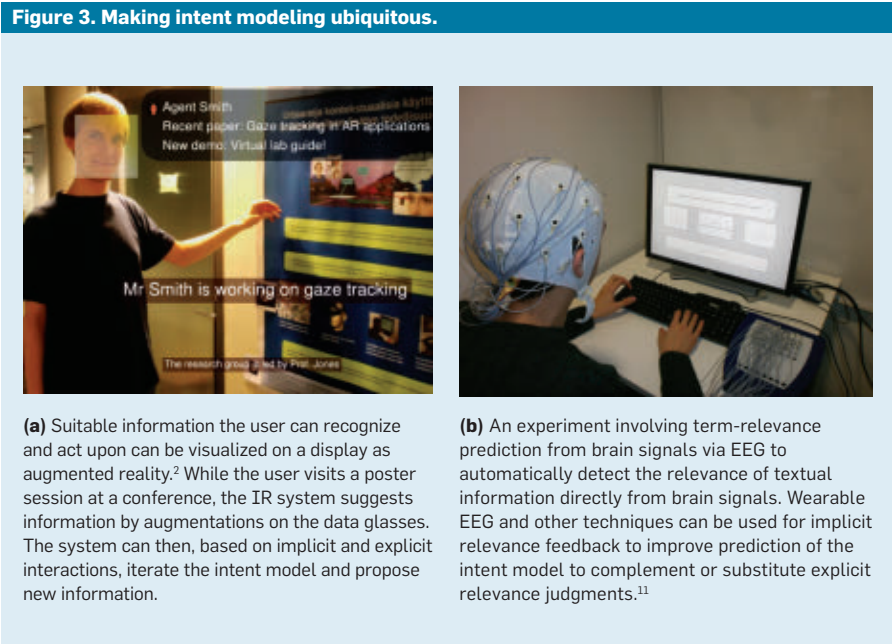
system discussed earlier, we are only scratching the surface of human-centered computing as part of the search activity. Intent-aware IR systems can benefit from ubiquitous computing in at least two ways, as discussed next.

Wearable User Interfaces and Augmented Reality
IR systems can be extended by augmenting a real scene with predictions of what the user might find useful, shown as augmented reality on head-mounted displays (HMDs). Users' implicit and explicit reactions to visualized content can reveal their intent and help improve the user intent model contextualized to the immediate setting. Figure 3a shows how suitable information (such as topics, research group, and publications) the user can recognize and act upon can be visual-

ized on a HMD superimposed on the real scene;² for example, augmenting a user's environment when visiting a poster session at a conference with visual cues and information can help the system collect information about the user's intent even when the user is not actively engaged with a search engine.

Implicit feedback from physiological computing. Recent advances in wearable computing have facilitated capturing users' affective and cognitive states (such as wearable electroencephalography, or EEG, systems). Moreover, other physiological sensors (such as galvanic skin response and heart-rate sensors) are being integrated into wrist-wearable products like smartwatches. Such physiological signals give researchers additional sources of feedback information not previously available.

Key benefits of interactive intent modeling.	
Improved task performance	Interactive intent modeling improves users' task performance compared to state-of-the-art retrieval methods and alternative search user-interface techniques. ²⁰
Quality of retrieved information	Interactive intent modeling helps users go beyond their initial query context, allowing them to increase recall significantly while preserving precision, particularly for novel information, leading to session-level improvement of 100%. ^{14,20}
Enhanced interaction	Interactive visualizations enhance interaction without compromising task execution time. Users in our experiments chose visualization as their main user-interface component for making sense of returned information and for expressing their search intent. ²⁰



It has been shown that affective state information can be used for relevance judgment prediction,⁴ and affective and psychophysiological signals are being employed in multimedia search systems with encouraging results.¹⁹ Figure 3b shows an example of an EEG sensor setup used to demonstrate term-relevance prediction from brain signals. The experiment shows it is possible to automatically detect the relevance of text information visualized for the user directly from brain signals by analyzing neural activity of participants while providing relevance judgments to text terms for a given topic.¹¹ Employing such physiology-based relevance detection for implicit relevance feedback on visualized information can be used by IR system developers to improve the prediction of the intent model to complement or substitute explicit user relevance ratings.

Conclusion

Recent work demonstrates there is significant room for improving the support provided to users involved in exploratory forms of search. Overall, researchers recognize the need for search and information-exploration systems that combine the information-processing capabilities of humans and computers.

Interactive intent modeling is a theoretically motivated, empirically proven way to support information exploration and discovery. It can increase users' capacity for information processing and discovery through computing technologies that assist users navigating complex information spaces.

Interactive intent modeling provides additional resources for users to better learn about the information space and give increased feedback for the system so it can efficiently adapt its understanding of user-search intent.

Engaging users to adopt interactive feedback mechanisms for information exploration and sense making requires user-interface techniques that go beyond search boxes and lists of links to enable them to better interact with the system and have control over their findings. Modeling user intent online as interaction occurs and even in situations where user feedback is noisy and suboptimal requires machine-learning models that learn online and are able

to explore, not just exploit. IR system design must ultimately integrate interactive visualizations, intent prediction, multimodal feedback, and a higher-level context of tasks and goals.

IR systems must be able to help users solve tasks, not just retrieve documents. Users need search engines and user interfaces that adapt to their capabilities and search behavior, rather than require them to adapt to them.

Acknowledgments

This work was partly supported by the Academy of Finland (278090, Multi-vire 255725 and the COIN Center of Excellence 251170), TEKES (D2I and Re:KnoW), and the European Commission through the FP7 Project MindSee 611570. Certain data included here is derived from the Web of Science prepared by Thomson Reuters, Inc., Philadelphia, PA. Data is also included from the Digital Libraries of the ACM, IEEE, and Springer.

References

- Ahn, J.-W. and Brusilovsky, P. Adaptive visualization for exploratory information retrieval. *Information Processing & Management* 49, 5 (Sept. 2013), 1139–1164.
- Ajanki, A., Billingham, M., Gamper, H., Järvenpää, T., Kandemir, M., Kaski, S., Koskela, M., Kurimo, M., Laaksonen, J., Puolamäki, K., Ruokolainen, T., and Tossavainen, T. An augmented reality interface to contextual information. *Virtual Reality* 15, 2-3 (June 2011), 161–173.
- Anderson, J.R. *Learning and Memory: An Integrated Approach, First Edition*. John Wiley & Sons, New York, 2000.
- Arapakis, I., Athanasakos, K., and Jose, J.M. A comparison of general vs. personalised affective models for the prediction of topical relevance. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Geneva, Switzerland, July 19–23). ACM Press, New York, 2010, 371–378.
- Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3 (Mar. 2003), 397–422.
- Avons, S., Wright, K., and Pammer, K. The word-length effect in probed and serial recall. *The Quarterly Journal of Experimental Psychology* 47, 1 (1994), 207–231.
- Bates, M.J. Subject access in online catalogs: A design model. *Journal of the American Society for Information Science* 37, 6 (Nov. 1986), 357–376.
- Bates, M.J. Indexing and access for digital libraries and the Internet: Human, database, and domain factors. *Journal of the American Society for Information Science* 49, 13 (1998), 1185–1205.
- Belkin, N.J., Oddy, R.N., and Brooks, H.M. Ask for information retrieval: Part I: Background and theory. In *Readings in Information Retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1997, 299–304.
- Chau, D.H., Kittur, A., Hong, J.I., and Faloutsos, C. Apollo: Making sense of large network data by combining rich user interaction and machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada, May 7–12). ACM Press, New York, 2011, 167–176.
- Eugster, M.J.A., Ruotsalo, T., Spapé, M.M., Kosunen, I., Barral, O., Ravaja, N., Jacucci, G., and Kaski, S. Predicting term-relevance from brain signals. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Gold Coast, Queensland,

- Australia, July 6–11). ACM Press, New York, 2014, 425–434.
- Fu, W.-T. and Gray, W.D. Resolving the paradox of the active user: Stable suboptimal performance in interactive tasks. *Cognitive Science* 28, 6 (Nov. 2004), 901–935.
- Furnas, G.W., Landauer, T.K., Gomez, L.M., and Dumais, S.T. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (Nov. 1987), 964–971.
- Glowacka, D., Ruotsalo, T., Konyushkova, K., Athukorala, K., Kaski, S., and Jacucci, G. Directing exploratory search: Reinforcement learning from user interactions with keywords. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces* (Santa Monica, CA, Mar. 19–22). ACM Press, New York, 2013, 117–128.
- Hearst, M.A. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, CO, May 7–11). ACM Press/Addison-Wesley Publishing Co., New York, 1995, 59–66.
- Hearst, M.A. and Pedersen, J.O. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Zurich, Switzerland, Aug. 18–22). ACM Press, New York, 1996, 76–84.
- Lehmann, S., Schwanecke, U., and Dörner, R. Interactive visualization for opportunistic exploration of large document collections. *Information Systems* 35, 2 (Apr. 2010), 260–269.
- Marchionini, G. Exploratory search: From finding to understanding. *Commun. ACM* 49, 4 (Apr. 2006), 41–46.
- Moshfeghi, Y. and Jose, J.M. An effective implicit relevance feedback technique using affective, physiological and behavioural features. In *Proceedings of the 36th International ACM Conference on Research and Development in Information Retrieval* (Dublin, Ireland, July 28–Aug. 1). ACM Press, New York, 2013, 133–142.
- Ruotsalo, T., Peltonen, J., Eugster, M., Glowacka, D., Konyushkova, K., Athukorala, K., Kosunen, I., Reijonen, A., Myllymäki, P., Jacucci, G., and Kaski, S. Directing exploratory search with interactive intent modeling. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management* (San Francisco, CA, Oct. 27–Nov. 1). ACM Press, New York, 2013, 1759–1764.
- Saracevic, T., Kantor, P., Chamis, A. Y., and Trivison, D. A study of information seeking and retrieving. I. Background and methodology. *Journal of the American Society for Information Science* 39, 3 (May 1988), 161–176.
- Strack, F. and Mussweiler, T. Explaining the enigmatic anchoring effect: Mechanisms of selective accessibility. *Journal of Personality and Social Psychology* 73, 3 (Sept. 1997), 437–446.
- Wilson, M.L., Kules, B., Schraefel, M.C., and Shneiderman, B. From keyword search to exploration: Designing future search interfaces for the Web. *Foundations and Trends in Web Science* 2, 1 (Jan. 2010), 1–97.
- Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, FL, Apr. 5–10). ACM Press, New York, 2003, 401–408.
- Zhao, L. and Callan, J. Term necessity prediction. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (Toronto, Canada, Oct. 26–30). ACM Press, New York, 2010, 259–268.

Tuukka Ruotsalo (tuukka.ruotsalo@aalto.fi) is a postdoctoral researcher in the Helsinki Institute for Information Technology HIIT at Aalto University, Helsinki, Finland.

Giulio Jacucci (giulio.jacucci@cs.helsinki.fi) is a professor in the Department of Computer Science, University of Helsinki, Finland, and a co-founder of MultiTaction Ltd.

Petri Myllymäki (petri.myllymaki@cs.helsinki.fi) is a professor in the Department of Computer Science at the University of Helsinki, Finland.

Samuel Kaski (samuel.kaski@hiit.fi) is director of the Helsinki Institute for Information Technology HIIT and a professor in the Department of Information and Computer Science at Aalto University, Helsinki, Finland.

Copyright held by Owners/Author(s)



ACM Books



MORGAN & CLAYPOOL
PUBLISHERS

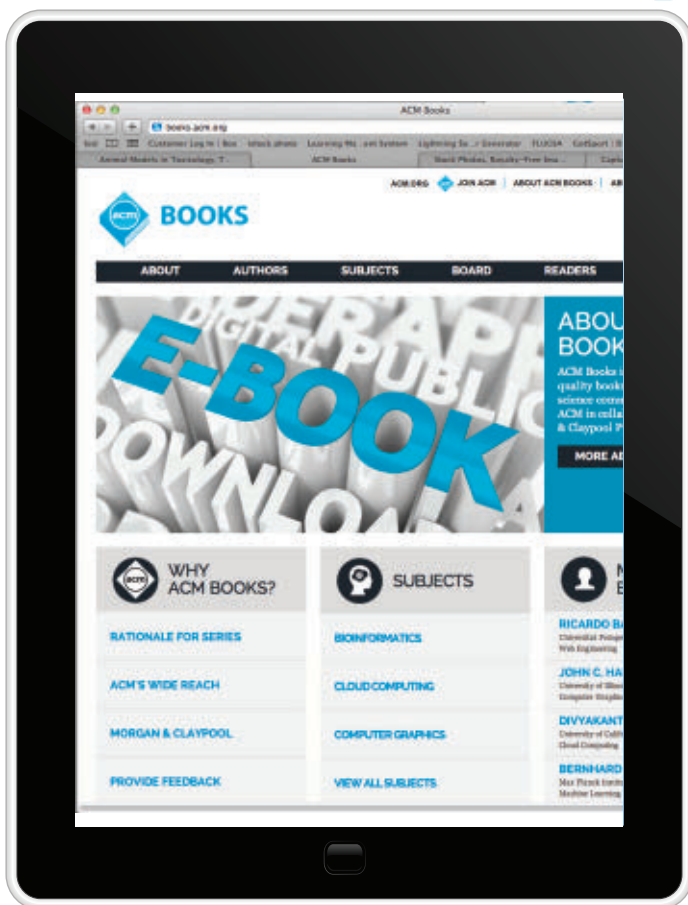
Publish your next book in the ACM Digital Library

ACM Books is a new series of advanced level books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers.

I'm pleased that ACM Books is directed by a volunteer organization headed by a dynamic, informed, energetic, visionary Editor-in-Chief (Tamer Özsu), working closely with a forward-looking publisher (Morgan and Claypool).

—Richard Snodgrass, University of Arizona

books.acm.org **ACM Books**



- ◆ will include books from across the entire spectrum of computer science subject matter and will appeal to computing practitioners, researchers, educators, and students.
- ◆ will publish graduate level texts; research monographs/overviews of established and emerging fields; practitioner-level professional books; and books devoted to the history and social impact of computing.
- ◆ will be quickly and attractively published as ebooks and print volumes at affordable prices, and widely distributed in both print and digital formats through booksellers and to libraries and individual ACM members via the ACM Digital Library platform.
- ◆ is led by EIC M. Tamer Özsu, University of Waterloo, and a distinguished editorial board representing most areas of CS.

Proposals and inquiries welcome!

Contact: **M. Tamer Özsu**, Editor in Chief
booksubmissions@acm.org



Association for
Computing Machinery

Advancing Computing as a Science & Profession

Exploring the similarities and differences between distributed computations in biological and computational systems.

BY SAKET NAVLAKHA AND ZIV BAR-JOSEPH

Distributed Information Processing in Biological and Computational Systems

BIOLOGICAL SYSTEMS, RANGING from the molecular to the cellular to the organism level, are distributed and in most cases operate without central control. Such systems must solve information processing problems that are often very similar to problems faced by computational systems, including coordinated decision making,²⁹ leader election,² routing and navigation,⁵² and more.⁴²

Over the last few years our ability to study and model biological systems has improved dramatically. Using advanced sequencing technologies we can now determine the composition of the genomes of hundreds of organisms. For specific cells and tissues



» key insights

- Biological and computational systems are often required to solve similar distributed information processing problems including coordinated decision making, leader election, routing, and navigation.
- The emergence of new computing technologies (wireless, sensor, and mobile computing), coupled with our ever-increasing ability to obtain large quantitative datasets describing biological systems at unprecedented details, opens the door to new joint studies of these two domains.
- Bidirectional studies in which researchers use ideas from one domain to study the other can concurrently lead to improved biologically inspired algorithms and novel computational understandings of how biological systems function.



we can query the set of active genes, their expression levels, and how their responses change in different conditions and over time. We can also determine which molecules within cells interact and how such interactions are wired to form the control diagram regulating cellular activity. Using this data we can build models of information processing within and between cells and answer questions regarding the methods biological systems use to achieve their goals that were out of reach even a few years ago. This data can also help us understand what computational problems are being solved by biological systems and how, which in turn can lead to specific algorithms that may also benefit computational systems.

Theoretical distributed computing has also been transformed by the recent adaptation and pervasiveness of wireless and mobile computing devices. These technologies have introduced new computational problems not typically faced by traditional, wired-based distributed systems.³³ Even in cases where algorithms initially designed for wired networks can be employed, new solutions are required to account for the dynamic nature and new constraints imposed by mobile devices (including energy conservation,⁴ limited transmission range,²⁸ reliance on broadcast communication,³³ and many more).

These two converging technological changes have led to several recent studies in which researchers use ideas from

one domain (either biology or computer science) to study the other. Such bidirectional studies can concurrently lead to biologically inspired algorithms and novel computational understandings of how biological systems function, which in turn can lead to new testable hypotheses (for example, Tero et al.⁵² and Afek et al.²). Several recent reviews have discussed these studies primarily focusing on the type of computational problem⁴² (networks-related, coordination, computer vision) or a specific biological system and their modeling (flocking birds, social insects).^{14,22}

Our goal in this article is to focus on the similarities and differences in the constraints, goals, and algorithms employed in both domains, especially

with regard to distributed information processing. We hope this perspective will allow researchers in both areas to focus on the most promising problems that can, and should, be studied bidirectionally. We first discuss several constraints that affect models of communication between entities (molecules, cells, mobile devices, and so on) in the two domains (Figure 1). Next, we argue that speed (or runtime), a key optimization goal for computational algorithms, is typically less important for biological systems, which focus more on robustness and adaptability. Lastly, we discuss similarities and differences in algorithmic strategies employed by both systems to achieve their goals under these different constraints.

The key for successful studies at the intersection of distributed computing and biology is to identify problems in which similar constraints and goals may apply to both systems. Networks provide one of many popular abstractions that have been immensely useful in understanding large, distributed systems. In biology, networks depict how molecules (metabolites, proteins), cells (bacteria, neurons), or organisms (ants) interact to jointly solve problems and coordinate responses. In computer sci-

ence, they depict how processors, machines, and devices communicate and process information. On the biological side, systems that involve dynamic networks and message passing (either within and between cells or between members of a population) are often well suited for ‘distributed thinking.’ From the computational point of view, mobile and sensor networks are ideal candidates that can benefit from new models and algorithms. In this article, we take a broad perspective of networks at all levels of life to demonstrate how distributed perspectives may apply and guide future investigations.

Communication Models

Most biological systems are distributed and must make decisions and respond to stimuli without a centralized coordinator and under severe constraints (energy conservation, limited communication range, limited messaging language, among others). While computer scientists have explored distributed computing algorithms for decades, the study of a class of severely limited communication models is more recent and has largely emerged to support mobile and sensor networks that are often required to operate for long durations

with limited resources and energy.⁴

Most distributed communication models are based on message passing. Even simple algorithms under such models use messages whose size is logarithmic in the number of participating nodes, which allows messages to include a unique identifier for both sender and receiver (for example, Luby’s famous algorithm under PRAM,³⁶ which can be adjusted to run under the message passing model). While such messages are very small compared to most traffic requirements in communication networks (for example, movie downloads), there are cases where even logarithmic message size may be problematic. For example, in crowded wireless networks, interference may cause larger messages to be dropped or missed,⁵¹ whereas in sensor networks, energy conservation also necessitates smaller messages. Information processing in biology is also often based on message passing. Cells secrete proteins to interact with other (neighboring or distant) cells in order to activate various signaling networks. While the number of proteins that can be secreted, and their levels, can vary greatly, there is recent evidence that most biological communication involves mes-

Figure 1. Similarities and differences in the properties of computational and biological systems.

Top row: Models, goals, and algorithmic strategies often used by conventional distributed systems but rarely in biological information processing. Bottom row: Shared features of dynamic distributed and biological systems. These and other common aspects are the basis for studies that model information processing in biology or develop a biologically motivated algorithm for a distributed computing problem.

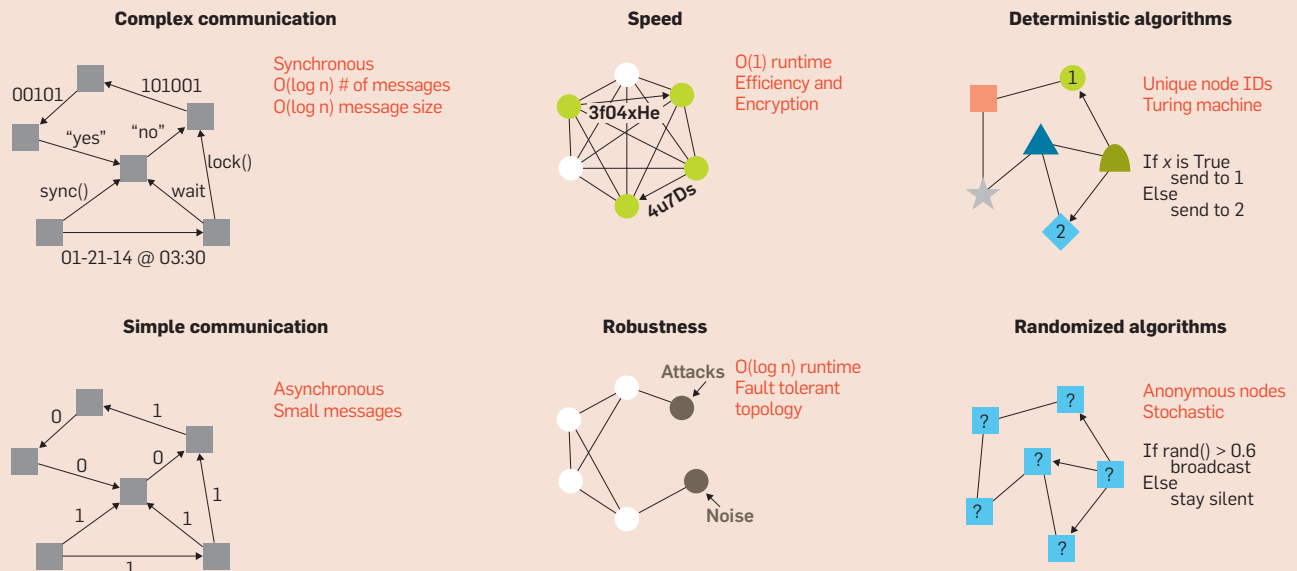
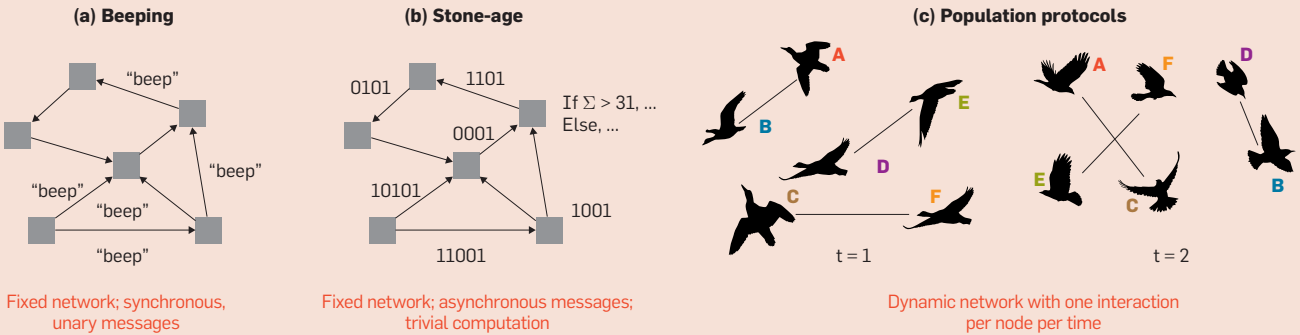


Figure 2. Distributed communication models.

The beeping model assumes an anonymous broadcast network with synchronous communication and allows only unary messages (beeps) to be sent. The stone-age model also assumes an anonymous broadcast network, but allows asynchronous communication; it also allows a richer messaging language than the beeping model but provides less computing power to each node. The population protocol assumes random, asynchronous pairwise interactions between agents also with limited memory and computational power.



sages whose effective information content is very small³¹ (on the order of one bit). For example, Cheong et al.¹⁵ recently demonstrated that cells cannot distinguish between varying levels of a certain protein input, leading to an effective binary message communicated by secreting this protein.

In addition to using a limited communication language (often of constant size), another goal shared by the two domains is reducing overall message complexity. Such reduction leads to more efficient use of available resources and energy (in biology, metabolites and for computational systems, electric power).

Motivated by these similar requirements, recent theoretical work in distributed computing and systems biology has analyzed the ability of extremely weak communication models to solve important computational problems and to explain the activity of key biological processes. In both cases the focus is on limiting message size and complexity, often at the expense of runtime. Here, we provide a few examples of the communication models that have been proposed for such systems and discuss both their use for solving fundamental distributed computing problems and their application to study the activity of complex biological processes.

Beeping: The beeping model¹⁷ (Figure 2a) assumes the only message that can be sent or received is a beep

(a unary signal). The model assumes an anonymous broadcast network in which nodes have no knowledge about the topology of the network or even an upper bound on its size. In each time slot a node can either beep or be silent. At a particular time slot, beeping nodes receive no feedback (they cannot determine if other nodes beeped as well), while silent nodes can only differentiate between two states: none of its neighbors beeping, or at least one neighbor beeping. Such a model is also appropriate for cellular signaling networks as discussed here.

Even with such limits on communication, several important distributed problems can be solved. The first problem solved under the beeping model was interval coloring, a variant of vertex coloring.¹⁷ Given a set of resources, the goal of interval coloring is to assign every node a large contiguous fraction of the resources, such that neighboring nodes have disjoint resources. Using beeping, the problem can be solved in $\mathcal{O}(\log n)$ time with high probability compared to $\mathcal{O}(\sqrt{\log n})$ when using unrestricted message sizes. More recently beeping was used to solve an even harder coordination problem: Maximal Independent Set (MIS). MIS attempts to find a subset of the nodes in the network such that: (1) Every node is either a MIS node or directly connected to one and (2) no two nodes in the set are connected to each other. MIS is a basic procedure in distributed computing and

serves as a building block for several algorithms including routing and clustering. Under mild additional assumptions (nodes can be woken up by neighbors' beeps), MIS can be solved in the beeping model in $\mathcal{O}(\log^2 n)$ time, as opposed to the classic $\mathcal{O}(\log n)$ solution when using larger message sizes (Luby's algorithm¹) or as opposed to relying on knowledge of the topology (Metivier's algorithm³⁹). The beeping model leads to an optimal communication load minimizing overall system requirements.² Follow-up work also showed that MIS could be solved under the beeping model in logarithmic time assuming sender collision detection.⁴⁹

Sensory Organ Precursor (SOP) selection. To illustrate the usefulness of the beeping model to study a biological system consider the SOP selection process, which is a key step in the development of the fruit fly brain. The process involves the selection of a subset of cells that later become sensory bristles on the fly's forehead. Similar to the MIS problem, such selection requires that no two neighboring cells become SOPs (known as lateral inhibition in biology) and that every cell is either a SOP or connected to a SOP. Cells communicate to determine which will become a SOP and while several mathematical models previously studied the two-way signaling involved in lateral inhibition (interactions between a SOP and one of its neighbors), only recently have biologists looked at the bigger picture: How

a subset of cells is selected from the overall population of cells. A variant of the beeping model can be used to explain such behavior (where the beeps in this case are a specific type of a protein called Delta).² This insight, coupled with new microscopy experiments following SOP selection in developing flies, led to the discovery of a novel stochastic feedback process used to determine cell fate and to a new distributed algorithm for MIS using the beeping model as discussed earlier.

Stone-age distributed computing. While beeping uses a very limited set of messages, it assumes nodes can access internal memory to perform computations that is logarithmic in the size of the network (that is, nodes can count up to $\mathcal{O}(\log n)$). Emek et al.²⁰ proposed a new communication model based on a network of finite state machines (nFSM) (Figure 2b). The nFSM model assumes a richer set of messages coming from a fixed-size language. These messages are asynchronously delivered to a dedicated channel in the receiving node (similar to receptors on interacting cells). However, unlike the beeping model, in nFSM nodes can only count up to a constant number. In other words, when transitioning to a new state, nodes evaluate the set of incoming messages (from all neighbors, though these neighbors are anonymous) according to the one-two-many principle: a node can only count up to some predetermined number and all values beyond this threshold are indistinguishable. As mentioned earlier, such a model corresponds to recent biological findings regarding the limited ability of cells to ‘count’ the levels of incoming proteins.¹⁵ Using the nFSM model, Emek et al. showed MIS can be solved in $\mathcal{O}(\log^2 n)$ time and it can also be used to 3-color an undirected tree in $\mathcal{O}(\log n)$ matching the optimal bound for this problem.

Probabilistic inference by neurons. One example of the potential usage of the nFSM model comes from networks of spiking neurons in the brain. Neuroscientists have experimentally shown that neurons are often “unreliable,” that is, there is significant trial-to-trial variability in neural output under the same input conditions.¹⁹ This suggests neural populations encode information by sampling from underlying

probability distributions. These distributions represent internal models of the external world that integrate new sensory stimuli with prior knowledge and memories. Recent computational work has remarkably shown that networks of stochastically firing neurons can carry forth probabilistic inference in a manner similar to Markov chain Monte Carlo sampling in distributed systems,¹¹ and such work has also led to several experimentally testable predictions about the firing dynamics of collections of neurons.⁴⁵ Neurons also use a one-two-many-like principle in the sense they count inputs within a time window up to a certain threshold before firing, and firing thresholds can vary depending on the type of neuron. Experimental work has also shown such internal representations are often sparse (only a few neurons are engaged per stimulus), indicating such energy-efficient representations may also be applicable to information processing problems in wireless sensor networks.²⁷

Population protocols. Unlike the previous two models that rely on broadcast, population protocols are based on direct, asynchronous physical interactions between a pair of agents⁶ (Figure 2c). Such interaction models are very common in nature and indeed, the development of population protocols has been motivated by a study of sensor networks attached to a flock of birds and was also recently applied to study distributed sensor networks on zebras.⁹ The model assumes that in each time slot, interactions occur between a pair of agents, which allows them to directly exchange messages and update their states. Unlike standard networks though, these interactions are either random or scheduled by an adversary, subject to a fairness constraint, which provide weak guarantees about the ability of every pair to interact eventually. Several types of problems can be solved distributively under this model including OR computations, majority, summation, and under some additional assumptions regarding the inputs, leader election, and consensus.⁶

Ant foraging. While some ants communicate by leaving pheromone trails, several harvester ant species only interact by direct physical contact. However, these ants are still able to find food

sources, while also recruiting other ants in the search and determining the amount of food available in an environment. A recent study demonstrated that with limited communication, ants solve the foraging problem by implementing a version of the Transmission Control Protocol (TCP), which is used on the Internet to determine available bandwidth when routing packets.⁴⁶ If packet acknowledgments (ACKs) are received quickly, the sender assumes bandwidth is available and boosts transmission; but if ACKs are returned slowly, the sender assumes the network is congested and throttles down transmission. Similarly, the important factor for the ants is the rate of antennal contacts (a binary indicator) between ants currently in the nest and successful ants (with food) returning to the nest. If the rate of contact is high, it implies food in the environment is plentiful, and thus outgoing ants also leave the nest at a faster rate.

Shared memory models. The previous models assume nodes communicate by exchanging messages. Another popular distributed communication method is the use of shared memory.³⁷ In such models, readers read from the shared memory, writers write to the shared memory, and erasers (which may or may not be the writers), remove data from this shared memory. Several papers have discussed the relationships between message passing and shared memory models. A classic result in this area is that any message-passing algorithm can also be solved in the shared memory framework,⁷ and vice versa, though runtime may drastically increase.

Chromatin computation. While most communication in biology is through message passing, in recent years shared memory has also emerged as a novel type of communication between proteins inside cells. Specifically, proteins were shown to modify DNA (by leaving, erasing, or reading specific marks on a set of proteins called histones over which DNA is wrapped). These marks play an important role in regulating the expression and activity of genes. There are many proteins (processors) that interact with these histones, which can lead to downstream effects on expression levels of genes next to marked sites. While the

large-scale study of such modifications is still in its infancy, issues that have been addressed by distributed algorithms—including competition, ordering of shared memory access, and the set of proteins that are able to access a specific site (memory location)—are of great current interest in molecular biology.³⁰ Recent studies have proposed computational models for such processes,¹⁰ and it is widely believed that histone modification and other epigenetic events play vital roles in development.

In addition to the specific communication type (broadcast or physical), message size (unary, fixed, or logarithmic) and buffer size, several other issues can be studied in the context of limited communication protocols for both biology and computational systems. These include various collision detection models, asynchronous vs. synchronous (or weakly synchronous) models,¹ bounded synchrony models,⁴⁴ and wakeup protocols.⁸ In all cases, parallels between biological and computational systems are leading to new insights that benefit both fields.

Speed vs. Robustness

Algorithms using restricted communication models often require longer runtimes compared to methods that utilize larger messages. This is no coincidence. Biological algorithms often need to trade off among speed, accuracy, and robustness, especially in noisy environments.¹⁶ Robustness can be improved by using extremely weak communication models that require few assumptions, as mentioned here. Here, we discuss another issue that affects robustness: network topology.

In this context, robustness denotes the ability of an algorithm to tolerate faults. Many models have been proposed in the distributed computing literature based on common types of failures, including link failures (where a single edge in the network is lost), node failures, and Byzantine failures (where a node is compromised and can participate in adversarial attacks). Many algorithms have been proposed to solve key distributed computing problems under each of these failure models, though they often assume fixed topologies, such as rings or cliques.³⁷ In biological systems, link and node



The key for successful studies at the intersection of distributed computing and biology is to identify problems in which similar constraints and goals may be applied to both systems.

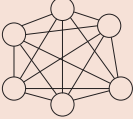
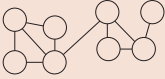
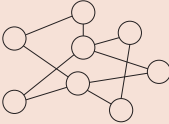


failures are also common; for example, mutation or protein misfolding can result in loss of specific interaction,⁵⁷ and genetic mutations can also result in a complete loss of a gene. Targeted attacks on specific nodes are also common, for example, in host-pathogen interactions, where virus proteins attack host proteins in an attempt to infect host cells.³² Overcoming these types of failures is a key requirement for any self-sustaining biological system.

While both computational and biological systems need to address these similar types of failures, the methods they use to do so differs. In distributed computing, failures have primarily been handled by majority voting methods,³⁷ by using dedicated failure detectors,²⁵ or via cryptography.⁴¹ In contrast, most biological systems rely on various network topological features to handle failures. Consider for example the use of failure detectors. In distributed computing, these are either implemented in hardware or in dedicated additional software. In contrast, biology implements implicit failure detector mechanisms by relying on backup nodes²⁴ or alternative pathways. Several proteins have paralogs, that is, structurally similar proteins that in most cases originated from the same ancestral protein (roughly 40% of yeast and human proteins have at least one paralog²⁶). In several cases, when one protein fails or is altered, its paralog can automatically take its place²⁴ or protect the cell against the mutation.²⁶ Thus, by preserving backup functionality in the protein interaction network, cells can overcome node failures without explicit use of failure detection mechanisms. While node failures occur within cells, a much more common scenario is the need to handle noisy, unreliable, inputs. For example, fluctuating environmental conditions can make it difficult for a bacteria to decide whether to sporulate or germinate.³⁸ In molecular networks, environmental noise can make it difficult to determine what type of regulatory response is needed and how quickly. Noise may also spread through the network and infect communication partners in a similar manner to epidemiological virus propagation models. To handle such Byzantine-like failures biological systems have optimized the topology

Figure 3. The effect of topology on speed and robustness.

Cliques are highly efficient in terms of routing distance and can tolerate any single node failure, but are quickly overcome by cascading failures or noise. Weakly connected networks are slightly less efficient and cannot overcome targeted attacks on bottlenecks, but they typically can isolate cascading failures to localized modules. Sparsely connected networks have longer routing time but can better overcome both types of failures.

		Speed	Robust (single-point attack)	Robust (cascading failure)
Densely connected (cliques)		✓	✓	✗
Weakly-connected modules		✓	✗ (bottlenecks)	✓
Sparsely connected		✗	✓	✓

of the networks they utilize (Figure 3).⁴³ For example, dense topologies with clique-like structures are often used in instances where little-to-no noise is expected, whereas sparser topologies are preferred when networks are expected to face more noise.⁴⁰ Of course, sparser topologies are also less efficient (in terms of routing distance, for example) which means execution times will be longer for such topologies. Weakly linked modules, on the other hand, can isolate occasional noise into nearly independent modules that each perform efficiently.⁵⁶

Along with robustness, biological algorithms are also designed to be adaptive and this is reflected in their underlying network topology. For example, activity-dependent plasticity of synapses is a well-known phenomenon by which neural networks are shaped by environmental stimuli. These signals are processed in a streaming fashion, and input patterns can change the topology of the networks designed to process them. Foraging slime molds have also been shown to adaptively adjust their networks of tubular junctions based on the distribution and availability of food sources in the area, which is typically unknown a priori.⁵² Such adaptive behavior often comes at the expense of runtime and resources. Indeed, while most body organs are

well-formed in young babies (though they still grow in size, their functionality does not change), the human brain overproduces synapses by 50%–60% during development and only converges to a more stable neural circuit in late adolescence. Slime molds also forage using breadth-first search using real cellular material, which is later pruned when optimal paths are found. In both cases, these systems sacrifice speed and resources for robustness of the resulting networks. These issues (of dynamic network structure and participation) are also key in the design of mobile ad hoc networks³³ and ideas from the biological systems mentioned here may prove useful for problems that communication networks face.

Strategies Used By Distributed Algorithms

We have described the operating constraints (communication models) and the goals (speed and robustness) applicable to biological and computational systems. In this section, we focus on differences in the algorithmic strategies used to achieve these goals and how such strategies are utilized in both domains.

One of the most widely used strategy in biological systems is stochastic decision-making. Randomness is used by biological processes to break

symmetry,² to overcome noise, and to ensure the survival of a population under changing environmental conditions.³⁴ Given the similarities mentioned earlier, it is not surprising that stochasticity has also long been employed within distributed algorithms for very similar purposes. For example, for leader election it was shown that if all processes are initially identical it is impossible to elect a leader without using randomized algorithms (in which case, it can be solved only with high probability⁵). Similarly, no deterministic algorithm can solve consensus even if only one node fails, while randomized algorithms can handle such failures and foil adversaries.³⁷ Another widely used strategy is feedback, both positive and negative. The use of feedback requires networks that contain several (often quite short) loops. For example, feedback inhibition is an important mechanism used to control the amount or concentration of a substance produced by a biochemical pathway to the appropriate level regardless of its current state or environmental availability. Homeostatic plasticity is another feedback mechanism that regulates activity levels of neurons to a certain range as a means to control circuit activity and prevent runaway excitation (seizures). Feedback is also central to several computational routing and initialization algorithms (using various types of backward acknowledgments). While useful, feedback can also amplify errors and noise in the system. This occurs in many message-passing schemes when incorrect or malicious information is distributed and amplified through the network over time. Biological systems can deal with this by adjusting network topology as mentioned earlier.

One common difference is in using unique identifiers for nodes. As mentioned in the communication models section, message sizes are usually one bit or of constant size indicating that unlike many traditional distributed algorithms, biological processes do not use such an identifier to label the sender and receiver. While targeted interactions do exist in biology (for example, synapses between two neural cells or domains mediating protein interactions), a node receiving two

messages usually cannot tell if these messages came from the same node or two different nodes.

Finally, while several distributed computing algorithms use majority voting³⁷ to solve coordination problems, biological systems often employ weighted voting schemes. This allows some nodes to have a greater influence on a population based on their own belief. For example, in bacterial swarms, a subgroup may find an undesirable path when foraging and lead the entire population astray. To overcome this, it was found that bacterium can dynamically adjust their decisions based on their own confidence and messages received from other cells.⁵⁰ While many of the rules used by these and other weighted-voting systems are yet to be worked out, they will likely be applicable in similar computing scenarios, for example, when programming distributed robot swarms for search-and-rescue operations.

Discussion

A resurgence of interest in studying how distributed biological systems process information and solve computational problems has occurred during the last few years. This revival has largely been triggered by two phenomena: first, our ability to experimentally probe the inner workings of molecular and cellular systems using a variety of new technological devices; and second, the emergence of new distributed computing models inspired by the pervasiveness of mobile, wireless, and sensor devices. Together, the new data and communication

models present a unique opportunity to jointly model, analyze, and learn from biological systems.

To effectively perform such bi-directional studies it is important that researchers realize both the similarities and differences between the models, goals, and algorithms used by the two domains. Most distributed computing models allow for large messages and high communication loads. Algorithms to address problems under such models often focus on speed (rounds), assume fully connected networks, and in many cases are deterministic, breaking symmetry by relying on unique identifiers. In contrast, information processing in biological systems uses much smaller messages (binary or constant), often emphasizes robustness over speed, uses incomplete (sometimes even sparse) networks, and is often stochastic. These constraints, goals, and methods are appropriate to some, but not all (or even most) computational problems. The accompanying table presents several relevant problems that have been addressed using insights from biological systems. In some cases, additional experiments were also required to understand how the problems were solved biologically and to derive algorithms for the corresponding computational problems.

While we discussed some reoccurring algorithmic strategies used within both types of systems (for example, stochasticity and feedback), there is much more to learn in this regard. From the distributed computing side, new models are needed to address the dy-

namic aspects of communication (for example, nodes joining and leaving the network, and edges added and being subtracted), which are also relevant in mobile computing scenarios. Further, while the biological systems we discussed all operate without a single centralized controller, there is in fact a continuum in the term “distributed.” For example, hierarchical distributed models, where higher layers “control” lower layers with possible feedback, represent a more structured type of control system than traditional distributed systems without such a hierarchy. Gene regulatory networks⁵⁵ and neuronal networks (layered columns) both share such a hierarchical structure, and this structure has been well-conserved across many different species, suggesting their importance to computation. Such models, however, have received less attention in the distributed computing literature.³ There are also many abstractions used for analyzing biological systems, ranging from discrete mathematics and graph theory to nonlinear dynamics and differential equations,⁵³ and such distributed computing models can likely benefit both modes of analysis.

Finally, as a cautionary note, biological algorithms designed by millions of years of natural selection do not guarantee optimality⁴⁷ and thus should not be couched in unrealistic terms. Indeed, some results discussed in this review (for example, network motifs,⁴⁰ and gene backup relationships²⁴) are based on high-throughput biological datasets that may be noisy and sometimes even contradictory.¹²

Examples of biological systems and their computational analogs.

Biological System	Computational Problem	Communication	Topology	Stochastic?	Alg.?	Refs.
Slime mold	Network routing	Stone-Age	Incomplete	Yes	Yes	35,48,52 48
Fly brain	Max. independent set	Beeping	Sparse	Yes	Yes	1,2
Harvester ants	TCP congestion control	Population	Random	No	Yes	46
Ants, swarms	Distributed search	Population	Random	Yes	Yes	23,50
Plants	Consensus	Stone-age	Incomplete	No	Yes	21
Fish schools	Consensus	Population	Random	Yes	No	29
Cell cycle switch	Approximate majority	Population	Random	Yes	Yes	13
Spiking neurons	Probabilistic inference	Stone-Age	Incomplete	No	Yes	11,45
Dendritic branching	Distributed MSTs	Stone-Age	Incomplete	No	Yes	18
Gannet colonies	Space partitioning	Population	Random	No	Yes	54
Protein interactions	Network design	Population	Random	Yes	Yes	43

Thus, care should be taken by analyzing multiple datasets to determine whether generalizations hold. There is, however, no doubt that evolution fine-tunes biological processes to optimize function. The emergent systems share many properties with those coveted in man-made distributed systems, and they also make interesting trade-offs between common optimization criteria for example, between efficiency and robustness, or runtime and message complexity) that can be learned from, especially to improve fault tolerance and adaptability. On the biological side, as technology continues to improve and sheds light on molecular and cellular decision-making, we believe computational perspectives will be essential to understand how local, distributed rules give rise to robust, global systems.

Acknowledgments

This work was supported in part by the National Institutes of Health award no. F32-MH099784 to S.N.; and by grants from the McDonnell Foundation program on Studying Complex Systems and from the U.S. National Science Foundation award nos. DBI-0965316 and DBI-1356505 to Z.B.-J. 

References

- Afek, Y. et al. Beeping a maximal independent set. *Distributed Computing; Lecture Notes in Computer Science*. D. Peleg, Ed. (2011). Springer-Verlag, Berlin Heidelberg, 32–50.
- Afek, Y. et al. A biological solution to a fundamental distributed computing problem. *Science* 331, 6014 (2011), 183–185.
- Aida, K., Natsume, W. and Futakata, Y. Distributed computing with hierarchical master-worker paradigm for parallel branch and bound algorithm. In *Proceedings of the 31st International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, Washington, DC, 2003.
- Anastasi, G., Conti, M., Di Francesco, M. and Passarella, A. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Netw.* 7, 3 (May 2009), 537–568.
- Angluin, D. Local and global properties in networks of processors (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*. ACM, New York, NY, 1980, 82–93.
- Aspnes, J. and Ruppert, E. An introduction to population protocols. *Middleware for Network Eccentric and Mobile Applications*. B. Garbinato, H. Miranda, and L. Rodrigues, Eds. Springer-Verlag, Berlin, Heidelberg, 2009, 97–120.
- Attiya, H., Bar-Noy, A. and Dolev, D. Sharing memory robustly in message-passing systems. *JACM* 42, 1 (Jan. 1995), 124–142.
- Babaoglu, O., Binci, T., Jelasity, M. and Montresor, A. Firefly-inspired heartbeat synchronization in overlay networks. In *Proceeding of the First International Conference Self-Adaptive and Self-Organizing Systems* (2007), 77–86.
- Beauquier, J., Blanchard, P., Burman, J. and Delaët, S. Tight complexity analysis of population protocols with cover times—the ZebraNet example. *Theor. Comput. Sci.* 512 (Nov. 2013), 15–27.
- Bryant, B. Chromatin computation. *PLoS ONE* 7, 5 (2012), e35703.
- Buesing, L., Bill, J., Nessler, B. and Maass, W. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput. Biol.* 7, 11 (Nov. 2011), e1002211.
- Bushman, F. et al. Host cell factors in HIV replication: meta-analysis of genome-wide studies. *PLoS Pathog.* 5, 5 (May 2009), e1000437.
- Cardelli, L. and Csikasz-Nagy, A. The cell cycle switch computes approximate majority. *Sci Rep.* 2:656 (2012).
- Chazelle, B. Natural algorithms and influence systems. *Commun. ACM* 55, 12 (Dec. 2012), 101–110.
- Cheong, R., Rhee, A., Wang, C.J., Nemenman, I. and Levchenko, A. Information transduction capacity of noisy biochemical signaling networks. *Science* 334, 6054 (Oct. 2011), 354–358.
- Chittka, L., Skorupski, P., and Raine, N.E. Speed-accuracy tradeoffs in animal decision-making. *Trends Ecol. Evol.* 24, 7 (July 2009), 400–407.
- Cornejo, A. and Kuhn, F. Deploying wireless networks with beeps. In *Proceedings of the 24th International Conference on Distributed Computing* (2010). Springer-Verlag, Berlin, Heidelberg, 148–162.
- Cuntz, H., Forstner, F., Borst, A. and Hausser, M. One rule to grow them all: A general theory of neuronal branching and its practical application. *PLoS Comput. Biol.* 6, 8 (2010).
- Destexhe, A. and Contreras, D. Neuronal computations with stochastic network states. *Science* 314, 5796 (Oct. 2006), 85–90.
- Emek, Y. and Wattenhofer, R. Stone age distributed computing. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing* (2013). ACM, New York, NY, 137–146.
- Falik, O., Mordoch, Y., Quansah, L., Fait, A. and Novoplansky, A. Rumor has it ... : Relay communication of stress cues in plants. *PLoS ONE* 6, 11 (2011), e23625.
- Feinerman, O. and Korman, A. Theoretical distributed computing meets biology: A review. *Distributed Computing and Internet Technology*. C. Hota and P. Srimani, Eds. *Lecture Notes in Computer Science* 7753 (2013), 1–18. Springer-Verlag, Berlin Heidelberg, 1–18.
- Feinerman, O., Korman, A., Lotker, Z. and Sereni, J.-S. Collaborative search on the plane without communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing* (2012). ACM, New York, NY, 77–86.
- Gitter, A. et al. Backup in gene regulatory networks explains differences between binding and knockout results. *Mol. Syst. Biol.* 5, 276 (2009).
- Gupta, I., Chandra, T.D., and Goldszmidt, G.S. On scalable and efficient distributed failure detectors. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing* (2001). ACM, New York, NY, 170–179.
- Hsiao, T.L. and Vitkup, D. Role of duplicate genes in robustness against deleterious human mutations. *PLoS Genet.* 4, 3 (Mar. 2008), e1000014.
- Hu, T., Genkin, A. and Chklovskii, D.B. A network of spiking neurons for computing sparse representations in an energy-efficient way. *Neural Comput.* 24, 11 (Nov. 2012), 2852–2872.
- Hu, Z. and Li, B. Fundamental performance limits of wireless sensor networks. *Ad Hoc and Sensor Networks* (2004), 81–101.
- Toannou, C.C., Guttal, V. and Couzin, I.D. Predatory fish select for coordinated collective motion in virtual prey. *Science* 337, 6099 (Sept. 2012), 1212–1215.
- Jakovcsevski, M. and Akbarian, S. Epigenetic mechanisms in neurological disease. *Nat. Med.* 18, 8 (Aug. 2012), 1194–1204.
- Jongeneel, C.V. et al. An atlas of human gene expression from massively parallel signature sequencing (MPSS). *Genome Res* 15, 7 (July 2005), 1007–1014.
- Kitano, H. and Oda, K. Robustness trade-offs and host-microbial symbiosis in the immune system. *Mol. Syst. Biol.* 2:2006.0022 (2006).
- Kuhn, F., Lynch, N., and Oshman, R. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing* (2010). ACM, New York, NY, 513–522.
- Levy, S., Kafri, M., Carmi, M. and Barkai, N. The competitive advantage of a dual-transporter system. *Science* 334, 6061 (Dec. 2011), 1408–1412.
- Li, K., Thomas, K., Torres, C., Rossi, L. and Shen, C.-C. Slime mold inspired path formation protocol for wireless sensor networks. In *Proceedings of the 7th International Conference on Swarm Intelligence* (2010). Springer-Verlag, Berlin, Heidelberg, 299–311.
- Luby, M. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (1985). ACM, New York, NY, 1–10.
- Lynch, N.A. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, CA, 1996.
- Mehta, P. and Schwab, D.J. Energetic costs of cellular computation. In *Proceedings of the Natl. Acad. Sci.* 109, 44 (Oct. 2012), 17978–17982.
- Métivier, Y., Robson, J., Saheb-Djahromi, N. and Zemmari, A. An optimal bit complexity randomized distributed mis algorithm. *Distributed Computing* 23, 5–6 (2011), 331–340.
- Milo, R. et al. Network motifs: Simple building blocks of complex networks. *Science* 298, 5594 (Oct. 2002), 824–827.
- Mittal, P. and Borisov, N. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the Conf. on Computer and Communications Security* (2008). ACM, New York, NY, 267–278.
- Navlakha, S. and Bar-Joseph, Z. Algorithms in nature: The convergence of systems biology and computational thinking. *Mol. Syst. Biol.* 7:546 (2011).
- Navlakha, S., He, X., Faloutsos, C. and Bar-Joseph, Z. Topological properties of robust biological and computational networks. *J.R. Soc. Interface* 11, 96 (2014).
- Nusser-Stein, S. et al. Cell-cycle regulation of NOTCH signaling during C. elegans vulval development. *Mol. Syst. Biol.* 8:618 (2012).
- Peceviski, D., Buesing, L. and Maass, W. Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Comput. Biol.* 7, 12 (Dec. 2011), e1002294.
- Prabhakar, B., Dektar, K.N., and Gordon, D.M. The regulation of ant colony foraging activity without spatial information. *PLoS Comput. Biol.* 8, 8 (2012), e1002670.
- Price, M.N. et al. Indirect and suboptimal control of gene expression is widespread in bacteria. *Mol. Syst. Biol.* 9:600, (2013).
- Reid, C.R., Latty, T., Dussutour, A., and Beekman, M. Slime mold uses an externalized spatial “memory” to navigate in complex environments. In *Proceedings of the Natl. Acad. Sci.* 109, 43 (Oct. 2012), 17490–17494.
- Scott, A., Jeavons, P., and Xu, L. Feedback from nature: An optimal distributed algorithm for maximal independent set selection. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*. ACM, New York, NY, 147–156.
- Shklarsh, A., Ariel, G., Schneidman, E. and Ben-Jacob, E. Smart swarms of bacteria-inspired agents with performance adaptable interactions. *PLoS Comput. Biol.* 7, 9 (Sept. 2011), e1002177.
- Tang, J., Xue, G., Chandler, C., and Zhang, W. Interference-aware routing in multihop wireless networks using directional antennas. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies* (2005). IEEE, 751–760.
- Tero, A. et al. Rules for biologically inspired adaptive network design. *Science* 327, 5964 (2010), 439–442.
- Tyson, J.J., Chen, K.C., and Novak, B. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Curr. Opin. Cell Biol.* 15, 2 (Apr. 2003), 221–231.
- Wakefield, E.D. et al. Space partitioning without territoriality in gannets. *Science* 341, 6141 (July 2013), 68–70.
- Yu, H. and Gerstein, M. Genomic analysis of the hierarchical structure of regulatory networks. In *Proceedings of the Natl. Acad. Sci.* 103, 40 (Oct. 2006), 14724–14731.
- Yu, H., Kim, P.M., Sprecher, E., Trifonov, V. and Gerstein, M. The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics. *PLoS Comput. Biol.* 3, 4 (Apr. 2007), e59.
- Zhong, Q. et al. Edgetic perturbation models of human inherited disorders. *Mol. Syst. Biol.* 5, 321 (2009).

Saket Navlakha (navlakha@salk.edu) is an assistant professor in the Center for Integrative Biology at the Salk Institute for Biological Studies, La Jolla, CA. This work occurred when he was a postdoc at Carnegie Mellon University, Pittsburgh, PA.

Ziv Bar-Joseph (zivbj@cs.cmu.edu) is an associate professor in the Machine Learning Department & Lane Center for Computational Biology, School of Computer Science, at Carnegie Mellon University, Pittsburgh, PA.

Copyright held by owners/authors. Publication rights licensed to ACM. \$15.00.

research highlights

P. 104

**Technical
Perspective
Big Data Needs
Approximate
Computing**

By Ravi Nair

P. 105

**Neural Acceleration
for General-Purpose
Approximate Programs**

By Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger

Technical Perspective

Big Data Needs Approximate Computing

By Ravi Nair

THE PHENOMENAL GROWTH of digital data produced by all kinds of systems and devices has been well documented. Studies indicate that until the year 2000, the amount of digital data in the world was dwarfed by the amount of analog data. Today, virtually all data being generated, transmitted, and consumed is in a digital form.

Commercial computers evolved when a large fraction of data in the digital universe was of the transactional kind. Transactional data continues to grow today, but it is a much smaller part of a digital universe dominated by consumer images, surveillance footage, entertainment feeds, social media, and sensor data. Yet, most systems today, including personal computers, use fundamentally the same type of processors, the same storage hierarchies, and the same computing paradigms pioneered by the transactional systems.

Systems built to support processing of transactions, for example, financial transactions, have special requirements. Much of the complexity in processors for these systems is incurred in satisfying high reliability, high precision, and a well-defined ordering of operations. These properties help in reasoning about and debugging a program but impose a cost on even those operations and phases of execution of a program where they could have been relaxed.

What would systems look like if we had to deal with only this new body of nontransactional data? Typical applications that mine and process this data can often tolerate lower precision, imprecise ordering, and even some unreliability in the operation of the system. Thus, an occasional stale or approximately correct piece of information delivered promptly is often more useful than up-to-date and precise information delivered later or at greater cost.

It may be argued that approximate computing makes it difficult to reason

logically about the results produced by a program. However, such reasoning is often difficult even in traditional computing—real numbers cannot always be represented precisely, the order of access to shared variables by multiple processors is often unpredictable and may lead to non-deterministic results, and it is virtually impossible to eliminate all potential sources of errors, hardware and software, from any system. Unlike traditional computing, approximate computing grants that errors will occur and transfers the responsibility for tolerating errors to the runtime, compiler, or even the application itself.

The following paper demonstrates the significant advantages in cost, power, and latency through approximate computing. The programming model in the paper allows a region of a program to be deemed approximable by the programmer. The compiler produces two versions of the approximable code, one that runs on the host and another on a separate accelerator that potentially performs the computation faster and with lower consumption of energy, but with results that may deviate from the exact results that would have been produced by the host.

The authors postulate that in many approximable programs, the solution space is such that knowledge of the past behavior of the program on a range of inputs is a good predictor of the behavior of the program on some new input. As an accelerator for the approximable region they use a learning engine to deliver an approximately correct result. The learning engine chosen by the authors is a digital neural network. The host version of the approximable section is executed on either a set of typical inputs or a set of random inputs. The outputs from this execution are used to train the neural network accelerator. On a variety of programs that tolerate approximation the authors show impressive per-

formance gains and power reduction for a tolerable loss of quality. It is also impressive that these gains and power reductions did not require different accelerator implementations for different programs—they were obtained on the same approximate accelerator working on all types of approximable code regions across all programs.

Beyond what is described in the paper, there is more energy savings to be harnessed by operating such approximate computing engines at considerably lower voltages. The real engine of growth in the industry—Dennard scaling—has allowed transistors to shrink in size, and simultaneously consume less power and improve performance. But Dennard scaling is ending. Device sizes have shrunk to atomic levels and voltages cannot be scaled down much without compromising the reliability of devices. New technologies promise good power-performance, but their advantage is offset by the redundancy needed to compensate for their unreliability. Thus, technologies of the future are unlikely to provide much better power-performance characteristics for traditional transactional systems. However, for systems that produce approximate results such as the one described by the authors, the redundancy needed is likely to be different and considerably simpler, making low-voltage operation practical for these types of systems.

Approximate computing appears to be our best hope to put massive amounts of computation to work on the massive amounts of data that will be produced in the world using future scaled CMOS and post-silicon technologies. The following paper is a good example of the type of work that needs to be done to fulfill this promise. 

Ravi Nair (nair@us.ibm.com) is a Distinguished Research Staff Member at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

Copyright held by author.

Neural Acceleration for General-Purpose Approximate Programs

By Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger

Abstract

As improvements in per-transistor speed and energy efficiency diminish, radical departures from conventional approaches are needed to continue improvements in the performance and energy efficiency of general-purpose processors. One such departure is approximate computing, where error in computation is acceptable and the traditional robust digital abstraction of near-perfect accuracy is relaxed. Conventional techniques in energy-efficient computing navigate a design space defined by the two dimensions of performance and energy, and traditionally trade one for the other. General-purpose approximate computing explores a third dimension—error—and trades the accuracy of computation for gains in both energy and performance. Techniques to harvest large savings from small errors have proven elusive. This paper describes a new approach that uses machine learning-based transformations to accelerate approximation-tolerant programs. The core idea is to train a learning model how an approximable *region* of code—code that can produce imprecise but acceptable results—behaves and replace the original code region with an efficient computation of the learned model. We use neural networks to learn *code* behavior and approximate it. We describe the *Parrot algorithmic transformation*, which leverages a simple programmer annotation (“approximable”) to transform a code region from a von Neumann model to a neural model. After the learning phase, the compiler replaces the original code with an invocation of a low-power accelerator called a *neural processing unit* (NPU). The NPU is tightly coupled to the processor pipeline to permit profitable acceleration even when small regions of code are transformed. Offloading approximable code regions to NPUs is faster and more energy efficient than executing the original code. For a set of diverse applications, NPU acceleration provides whole-application speedup of 2.3× and energy savings of 3.0× on average with average quality loss of at most 9.6%. NPUs form a new class of accelerators and show that significant gains in both performance and efficiency are achievable when the traditional abstraction of near-perfect accuracy is relaxed in general-purpose computing.

1. INTRODUCTION

It is widely understood that energy efficiency now fundamentally limits microprocessor performance gains. CMOS scaling is no longer providing gains in efficiency commensurate with transistor density increases.^{7,15} As a result, both the semiconductor industry and the research community are increasingly focusing on specialized accelerators, which can provide large gains in efficiency and performance by

restricting the workloads that benefit. Recent work has quantified three orders of magnitude of difference in efficiency between general-purpose processors and ASICs.¹⁴ The community is facing an “iron triangle” in this respect; we can choose any two of performance, energy efficiency, and generality at the expense of the third. Before the traditional trend of transistor scaling—Dennard scaling⁵—broke down, we were able to improve all three on a consistent basis for decades. In this post Dennard scaling era, solutions that improve performance and efficiency while retaining as much generality as possible are highly desirable; hence the exploding interest in GPGPUs and FPGAs. Such programmable accelerators exploit some characteristic of an application domain to achieve efficiency gains at the cost of generality. FPGAs, for example, exploit copious, fine-grained, and irregular parallelism while GPUs exploit many threads and data-level SIMD-style parallelism. Whether an application can use an accelerator effectively depends on the degree to which it exhibits the accelerator’s required characteristics.

Tolerance to approximation is one such program characteristic. A growing body of recent work^{2, 4, 8, 19, 26, 27} has focused on *approximation* as a strategy for improving efficiency. Large classes of applications can tolerate small errors in their outputs with no discernible loss in their *quality of result* (QoR). These applications are common in mobile, embedded, and server systems and fall into four broad categories:

1. *Applications with analog inputs.* This category includes image processing, sensor data processing, voice recognition, etc., that operate on noisy real-world data. They are inherently resilient to some noise and can handle an “extra noise” resulting from approximation.
2. *Applications with analog output.* These applications comprise multimedia, image rendering, sound synthesis, etc. Their output is intended for human perception and can inherently tolerate errors imperceptible to users.
3. *Applications with no unique answer.* This class of applications includes web search, machine learning, autonomous agents, etc., which do not offer a unique answer and multiple possible answers are acceptable.

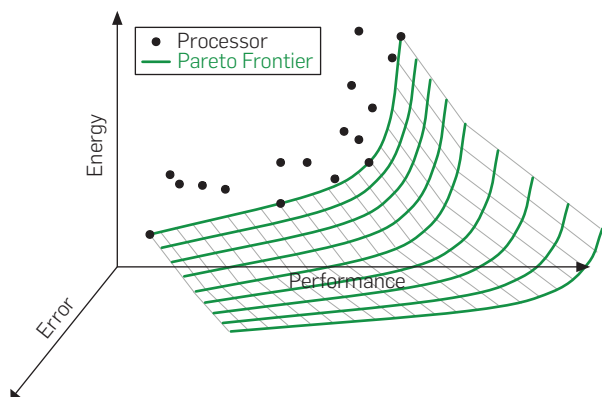
A previous version of this article appeared in *Proceedings of the 45th Symposium on Microarchitecture* (December 2012). Parts of this article appeared in *IEEE Micro Top Picks from the Computer Architecture Conferences of 2012* (May/June 2013).

4. *Iterative and convergent applications.* This category includes applications such as data analytics and numerical computations that iteratively process large amounts of data. They often sample data, stop the convergence procedure early, or apply approximate heuristics. Thus, these applications can naturally benefit from approximation techniques.

For these classes of applications, trading off computation accuracy can potentially lead to larger gains in performance and efficiency. One may visualize these trade-offs as finding the Pareto-optimal points in the processor design space, as shown in Figure 1. Traditionally, for any set of workloads, the set of possible processor implementations may be plotted, with energy efficiency on one axis and performance on the other, and the best implementations residing on the two-dimensional frontier. When approximation is supported, the degree of permissible error represents a third axis. The Pareto surface in this three-dimensional space represents the best points of performance, efficiency, and error. However, this surface is not yet well understood.

This paper defines some new points on this Pareto surface by developing a new class of programmable accelerators that exploit approximation for better performance and energy efficiency. The core idea is to *learn* how a region of approximable code behaves and *automatically* replace the original code with an efficient computation of the learned model. This approach contrasts with previous work on approximate computation that extends conventional microarchitectures to support selective approximate execution, incurring instruction bookkeeping overheads^{4, 8, 19} or requires vastly different programming paradigms.^{2, 21} Like emerging flexible accelerators,^{11, 12, 29} our technique automatically offloads code segments from programs written in mainstream languages. However, unlike prior work, it leverages changes in the semantics of the offloaded code and the nature of computation. Such changes are possible because the transformed code region is approximable and can tolerate small errors.

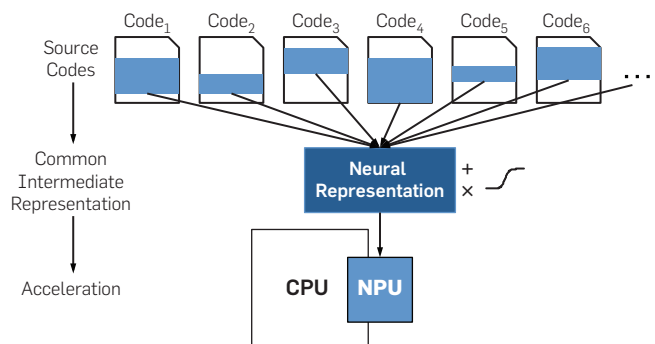
Figure 1. Adding the dimension of error to the design space of general-purpose processors changes the problem of finding the Pareto frontier to finding the Pareto surface. Navigating this three-dimensional space, finding, and understanding this Pareto surface is a fascinating research direction.



NPU-enabled systems rely on our learning-based algorithmic transformation that we refer to as the Parrot transformation^a. This transformation converts regions of approximable general-purpose code into a neural representation—specifically, multilayer perceptrons—at compile time. At run time, while the processor executes the program, it invokes the NPU instead of running the original region of code. NPUs result in large performance and efficiency gains, since they subsume a region of code, eliminating nearly all of the instruction fetch, decode, etc., overheads that would have been incurred if that region was executed on the processor. These overheads are replaced by a single, efficient invocation of neural hardware that exploits hard-wired control for additional efficiency. As illustrated in Figure 2, NPU acceleration provides improved generality over task-specific accelerators, as the Parrot transformation converts many distinct code patterns into a common representation that can be run on a single physical accelerator. In fact, our Parrot algorithmic transformation replaces unstructured serial code with a neural network that has structured fine grain parallelism that is executed on an efficient statically scheduled hardware structure, the NPU. Therefore, converting diverse regions of code to the common neural representation can lead to significant performance and efficiency gains because neural networks consist of simple, regular, and parallel operations.

We show that using neural networks to replace regions of imperative code is both feasible and profitable by accelerating a diverse range of applications, including FFT, gaming, clustering, and vision algorithms (Section 6). These applications do not belong to the class of modeling and prediction tasks that typically use neural networks. For each application, we apply the transformation on a single approximable function that dominates the program's execution time. NPU acceleration provides 2.3× average whole-application speedup and 3.0× average energy savings for these benchmarks with average accuracy greater than 90% in all cases.

Figure 2. The Parrot algorithmic transformation converts different regions of code to a common neural intermediate representation. Neural networks as a common representation enable acceleration of diverse applications using a single physical NPU.



^a We named our algorithmic transformation, the Parrot transformation because its output is a learning model that mimics the original region of code.

The Parrot algorithmic transformation and the NPU acceleration bridge von Neumann and neural models of computing. These techniques make neural hardware programmable via conventional programming languages and extend their use beyond prediction and modeling to accelerating general-purpose code. The results from this paper show that when the traditional abstraction of near-perfect accuracy is relaxed, different models of computing can be merged to obtain large gains in performance and efficiency.

2. OVERVIEW

As depicted in Figure 3, the *Parrot transformation* is an algorithmic transformation that converts regions of imperative code to neural networks. Because neural networks expose considerable parallelism and consist of simple operations, they can be efficiently accelerated using dedicated hardware. Therefore, the Parrot transformation can yield significant performance and energy improvements. The transformation uses a training-based approach to produce a neural network that approximates the behavior of candidate code. A transformed program runs primarily on the main core and invokes an auxiliary hardware structure, the NPU, to perform neural evaluation instead of executing the replaced code. Figure 3 shows an overview of the Parrot algorithmic transformation, which has three key phases: programming, in which the programmer marks code regions to be transformed; compilation, in which the compiler selects and trains a suitable neural network and replaces the original code with a neural network invocation; and execution.

Programming. The Parrot transformation starts with the programmer identifying candidate code regions as approximable. Because tolerance of approximation is a semantic property, it is the programmer's responsibility to select code whose approximate execution would not compromise the overall reliability of the application. This requirement is a common practice in the approximate computing literature.^{4, 8, 26}

Compilation. Once the source code is annotated, as shown in Figure 3, the compiler applies the Parrot transformation in three steps: (1) code observation, (2) neural network selection and training, and (3) binary generation.

Training neural networks for any task requires a collection of input-output pairs that capture the task's

function. Therefore, in the code observation step, the compiler observes the behavior of the candidate code region by logging its inputs and outputs. This step is similar to profiling. The compiler instruments the program with probes on the inputs and outputs of the candidate functions. Then, the instrumented program is run using representative input sets such as those from a test suite. The probes log the inputs and outputs of the candidate functions. The logged input-output pairs constitute the training and validation data for the next step.

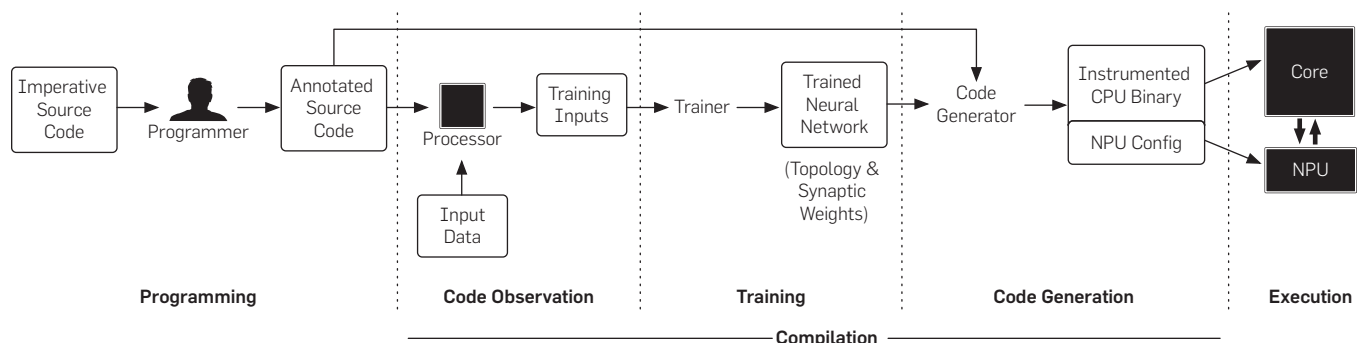
The compiler uses the collected input-output data to configure and train a neural network that mimics the candidate region. The compiler must find the simplest topology of the neural network that provides acceptable error, for which more complex networks would provide diminishing returns in QoR. The compiler also needs to find the synaptic weights of the network. It uses the backpropagation algorithm²⁵ coupled with a topology search to configure and train the neural network.

The final step of the Parrot transformation is code generation. The compiler first generates a configuration for the NPU that implements the trained neural network. Then, the compiler replaces each call to the original function with a series of special instructions that invoke the NPU, sending the inputs and receiving the computed outputs. The NPU configuration and invocation is performed through ISA extensions that are added to the core.

Execution. During deployment, the transformed program begins execution on the main core and configures the NPU. Throughout execution, the NPU is invoked to perform a neural network evaluation in lieu of executing the original code region. The NPU is integrated as a tightly coupled accelerator in the processor pipeline. Invoking the NPU is faster and more energy efficient than executing the original code region, so the program as a whole is accelerated.

As Figure 4 shows, many NPU implementations are feasible, from all-software execution to specialized analog circuits. Because the Parrot transformation's effectiveness rests on the efficiency of neural network evaluation, it is essential that invoking the NPU be fast and low power. Therefore, we describe a high-performance hardware NPU design based on a digital neural network ASIC and architecture support to facilitate low-latency NPU invocations.

Figure 3. The Parrot transformation at a glance: from annotated code to accelerated execution on an NPU-augmented core.



3. PROGRAMMING MODEL

The Parrot transformation starts with the programmer identifying candidate code regions. This section discusses these criteria as well as the concrete language interface exposed to the programmer. After the candidate regions are identified, the Parrot transformation is fully automated.

3.1. Code region criteria

Candidate code for the Parrot transformation must satisfy three criteria: it must be frequently executed (i.e., a “hot” function); it must tolerate imprecision in its computation; and it must have well-defined inputs and outputs.

Hot code. Like any acceleration technique, the Parrot transformation should replace hot code. The Parrot transformation can be applied to a wide range of code from small functions to entire algorithms. The code region can contain function calls, loops, and complex control flow whose cost

Figure 4. Design space of NPU implementations. This work focuses on a precise digital ASIC design.

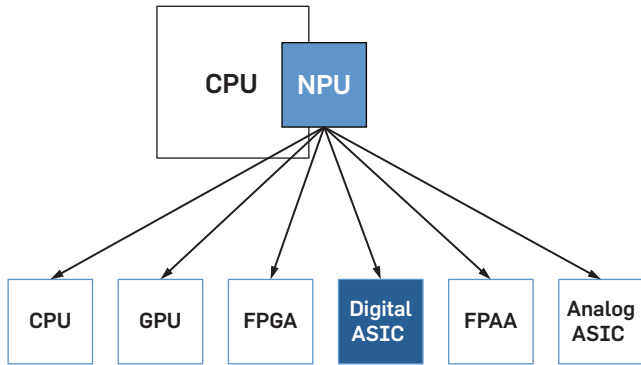
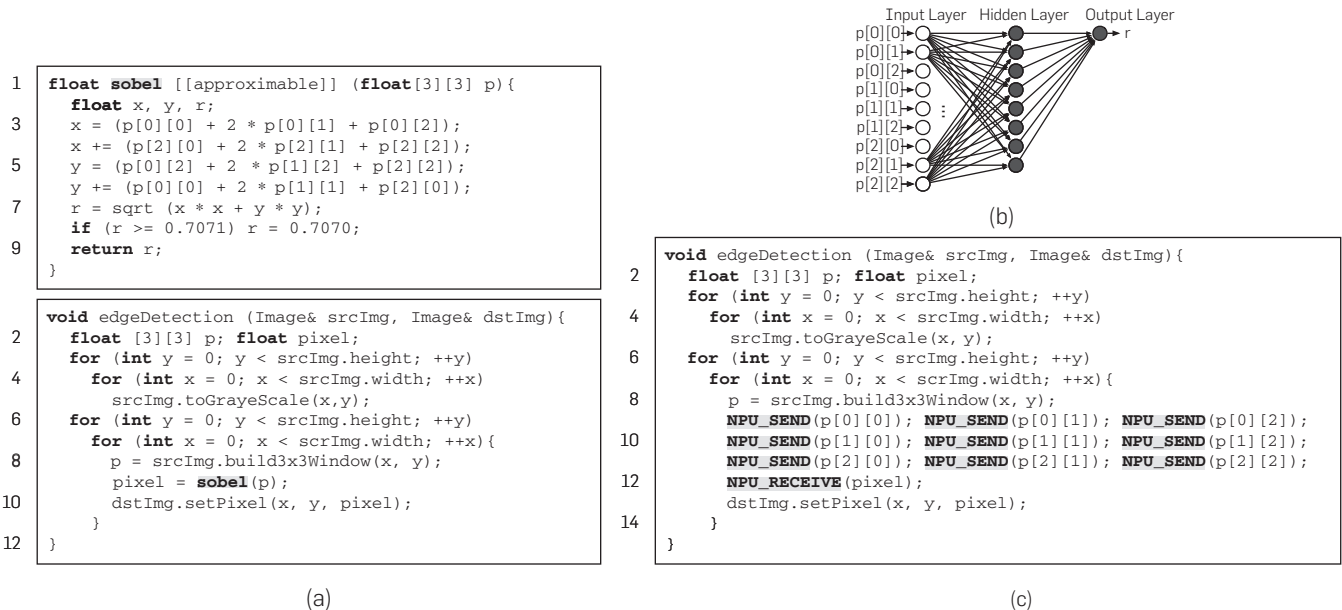


Figure 5. Three stages in the transformation of an edge detection algorithm using the Sobel filter: (a) original implementation of the Sobel filter, (b) sobel function transformed to a $9 \rightarrow 8 \rightarrow 1$ NN, and (c) Parrot transformed code; NPU invocation replaces the function call.



can be elided by the Parrot transformation. When applied to smaller regions of code, the overhead of NPU invocation needs to be low to make the transformation profitable. A traditional performance profiler can reveal hot code.

For example, edge detection is a widely applicable image processing computation. Many implementations of edge detection use the Sobel filter, a 3×3 matrix convolution that approximates the image’s intensity gradient. As the bottom box in Figure 5a shows, the local Sobel filter computation (the `sobel` function) is executed many times during edge detection, so the convolution is a hot function in the overall algorithm and a good candidate for the Parrot transformation.

Approximability. Code regions identified for the Parrot transformation will behave approximately during execution. Therefore, programs must incorporate application-level tolerance of imprecision. This requires the programmer to ensure that imprecise results from candidate regions will not cause catastrophic failures. As prior work on approximate programming^{1, 4, 19, 26, 27} has shown, it is not difficult to deem regions approximable.

Beyond determining that a code region may safely produce imprecise results, the programmer need not reason about the mapping between the code and a neural network. While neural networks are more precise for some functions than they are for others, we find that they can accurately mimic many functions from real programs (see Section 6). Intuitively, however, they are less likely to effectively approximate chaotic functions, in which even large training sets can fail to capture enough of the function’s behavior to generalize to new inputs. However, the efficacy of neural network approximation can be assessed empirically. The programmer should annotate all approximate code; the compiler can then assess the accuracy of a trained neural network in

replacing each function and select only those functions for which neural networks are a good match.

In the Sobel filter example, parts of the code that process the pixels can be approximated. The code region that computes pixel addresses and builds the window for the sobel function (line 8 in the bottom box of Figure 5a) needs to be precise to avoid memory access violations. However, the sobel function, which estimates the intensity gradient of a pixel, is fundamentally approximate. Thus, approximate execution of this function will not result in catastrophic failure and, moreover, is unlikely to cause major degradation of the overall edge detection quality. These properties make the sobel function a suitable candidate region for approximate execution.

Well-defined inputs and outputs. The Parrot transformation replaces a region of code with a neural network that has a fixed number of inputs and outputs. Therefore, it imposes two restrictions on the code regions that can feasibly be replaced. First, the inputs to and outputs from the candidate region must be of a fixed size known at compile time. For example, the code may not dynamically write an unbounded amount of data to a variable-length array. Second, the code must not cause side effects via system calls. These two criteria can be checked statically.

The sobel function in Figure 5a complies with these requirements. It takes nine statically identifiable floating-point numbers as input, produces a single output, and has no side effects.

3.2. Annotation

In this work, we apply the Parrot transformation to entire functions. To identify candidate functions, the programmer marks them with an annotation (e.g., using C++11 `[[approximable]]` syntax) as shown in Figure 5a. The programmer is responsible for ensuring that the function has well-defined inputs and outputs. All the inputs are in the argument list and all the outputs are part of the return value. Each argument type and the return type must have a fixed size; however, they may have multiple elements, for example, fixed-size array or a record. If any of these types is a pointer type, it must point to a fixed-size value; this referenced value is then considered the neural network input or output rather than the pointer itself. If the function needs to return multiple values, it can return a fixed-size array or a C struct. After the programmer annotates the candidate functions, the Parrot transformation is completely automatic and transparent: no further programmer intervention is necessary.

Like prior work on approximate computing, we acknowledge that some programmer guidance is essential when identifying error-tolerant code.^{1, 4, 8, 19, 26} Tolerance to approximation is an inherently application-specific property. While we find that the simple explicit function annotations are straightforward to apply (see Section 6), static analysis techniques could be used to further simplify the annotation process and significantly automate it.

4. COMPILATION WORKFLOW

Once the program has been annotated, the compilation workflow implements the Parrot transformation in three steps: observation, training, and instrumented binary generation.

4.1. Code observation

Training neural networks for any task requires a collection of input–output pairs that capture the task’s function. Therefore, in the first phase, the compiler collects input–output pairs for the target code that reflect real program executions. This in-context observation allows the compiler to train the neural network on a realistic data set. The compiler produces an instrumented binary for the source program that includes probes on the input and output of the annotated function. Each time the candidate function executes, the probes record its inputs and outputs. The program is run repeatedly using test inputs. The output of this phase is a training data set: each input–output pair represents a sample for the training algorithm.

The observation phase resembles the profiling runs used in profile-guided compilation. Specifically, it requires representative test inputs for the application. The inputs may be part of an existing test suite or randomly generated. In many cases, a small number of application test inputs are sufficient to train a neural network because the candidate function is executed many times in a single application run.

4.2. Training

The compiler uses the training data to produce a neural network that replaces the original function. There are a variety of types of artificial neural networks in the literature, but we narrow the search space to multilayer perceptrons (MLPs) due to their broad applicability.

The compiler uses the backpropagation algorithm²⁵ to train the neural network. Backpropagation is a gradient descent algorithm that iteratively adjusts the weights of the neural network according to each input–output pair.

Neural network topology selection. In addition to running backpropagation, this phase selects a network topology that balances accuracy and efficiency. An MLP consists of a fully connected set of neurons organized into layers: the input layer, any number of “hidden” layers, and the output layer. A larger, more complex network offers better accuracy potential but is likely to be slower and less power efficient than a small, simple neural network. The objective is to find the smallest neural network that achieves acceptable accuracy.

To choose the topology, we use a simple search algorithm guided by the mean squared error of the neural network when tested on an unseen subset of the observed data. The error evaluation uses a typical cross-validation approach: the compiler partitions the data collected during observation into a *training set*, 70% of the observed data, and a *test set*, the remaining 30%. The topology search algorithm trains many different neural network topologies using the training set and chooses the one with the highest accuracy on the test set and the lowest latency on the NPU (prioritizing accuracy).

The output from this phase consists of a neural network topology—specifying the number of layers and the number of neurons in each layer—along with the weight for each neuron and the normalization range for each input and output.

Online training. The current system performs observation and training prior to deployment; an alternative

design could train the neural network concurrently with in vivo operation. Online training could improve accuracy but would result in runtime overheads. To address these overheads, an online training system could offload neural network training and configuration to a remote server. With off-site training, multiple deployed application instances could centralize their training to increase input space coverage.

4.3. Code generation

After the training phase, the compiler generates an instrumented binary that runs on the core and invokes the NPU instead of calling the original function. The program configures the NPU when it is first loaded by sending the topology parameters and synaptic weights to the NPU. The compiler replaces the calls to the original function with special instructions that send the inputs to the NPU and collect the outputs from it.

5. ARCHITECTURE DESIGN FOR NPU ACCELERATION

Since candidate regions for the Parrot transformation can be fine grained, NPU invocation must be low-overhead to be beneficial. Ideally, the NPU should integrate tightly with the processor pipeline. The processor ISA also needs to be extended to allow programs to configure and invoke the NPU during execution.

5.1. ISA support for NPU acceleration

The NPU is a variable-delay, tightly coupled accelerator that communicates with the rest of the core via FIFO queues. The CPU–NPU interface consists of three queues: one for sending and retrieving the configuration, one for sending the inputs, and one for retrieving the neural network's outputs. The ISA is extended with four instructions to access the queues. These instructions assume that the processor is equipped with a single NPU; if the architecture supports multiple NPUs or multiple stored configurations per NPU, the instructions may be parameterized with an operand that identifies the target NPU.

- `enq.c %r`: enqueues the value of the register `r` into the config FIFO.
- `deq.c %r`: dequeues a configuration value from the config FIFO to the register `r`.
- `enq.d %r`: enqueues the value of the register `r` into the input FIFO.
- `deq.d %r`: dequeues the head of the output FIFO to the register `r`.

To set up the NPU, the program executes a series of `enq.c` instructions to send configuration parameters—number of inputs and outputs, network topology, and synaptic weights—to the NPU. The operating system uses `deq.c` instructions to save the NPU configuration during context switches. To invoke the NPU, the program executes `enq.d` repeatedly to send inputs to the configured neural network. As soon as all of the inputs of the neural network are enqueued, the NPU starts computation and puts the results in its output FIFO. The program executes `deq.d` repeatedly to retrieve the output values.

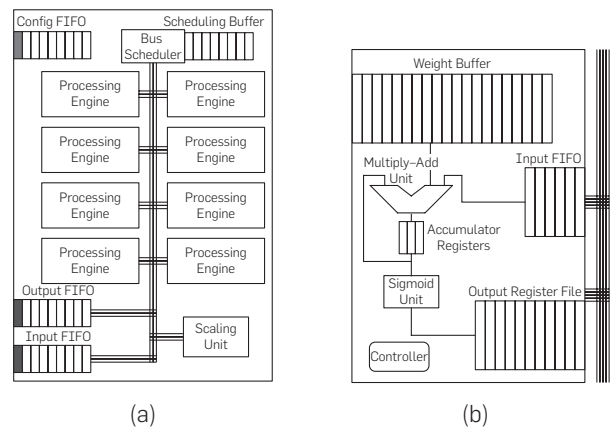
5.2. Neural processing unit

As Figure 4 illustrates, there are many possibilities for the implementation of the NPU itself. Neural networks have previously been implemented in software on the CPU or GPU,¹³ on FPGAs,³⁰ in digital ASICs,⁶ and even in analog circuitry or FPAAs.¹⁷ As shown in Figure 6, we designed a reconfigurable digital NPU circuit that operates at the same voltage and frequency as the main core. More details on our NPU design can be found in Esmaeilzadeh et al.⁹ This implementation represents a reasonable trade-off between efficiency and complexity. However, we believe that analog NPUs have significant potential and we plan to explore them in future work.

The Parrot transformation produces different neural network topologies for different code regions. Thus, our reconfigurable NPU accelerates the evaluation of a range of neural topologies. As shown in Figure 6, the NPU contains eight identical processing engines (PEs) and one scaling unit. The scaling unit scales the neural network's inputs and outputs if necessary using scaling factors defined in the NPU configuration process.

The PEs in the NPU are statically scheduled. The scheduling information is part of the configuration information for the NPU, which is based on the neural network topology derived during the training process. In the NPU's schedule, each neuron in the neural network is assigned to one of the eight PEs. The neural network's topology determines a static schedule for the timing of the PE computations, bus accesses, and queue accesses.

Figure 6. Reconfigurable 8-PE NPU: (a) 8-PE NPU, (b) single processing engine (PE), and (c) design parameters of the NPU.



Parameter	Configuration
Number of PEs	8
Bus Schedule FIFO	512 × 20-bit
Input FIFO	128 × 32-bit
Output FIFO	128 × 32-bit
Config FIFO	8 × 32-bit
PE Weight Cache	512 × 33-bit
PE Input FIFO	8 × 32-bit
PE Output Register File	8 × 32-bit
Sigmoid Unit LUT	128 × 32-bit
Multiply-Add Unit	32-bit Single Precision FP

(c)

The NPU stores the bus scheduling information in its circular scheduling buffer (shown in Figure 6). Figure 6b shows the internal structure of a single PE. Each PE performs the computation for all of its assigned neurons. Namely, because the NPU implements a sigmoid-activation MLP, each neuron computes its output as $y = \text{sigmoid}(\sum_i(x_i \times w_i))$, where x_i is an input to the neuron and w_i is its corresponding weight. The weight buffer, a circular buffer, stores the weights. When a PE receives an input from the bus, it stores the value in its input FIFO. When the neuron weights for each PE are configured, they are placed into the weight buffer; the compiler-directed schedule ensures that the inputs arrive in the same order that their corresponding weights appear in the buffer. This way, the PE can perform multiply-and-add operations in the order the inputs enter the PE's input FIFO.

Since the computation of the neural network requires only simple operations that are limited to multiply-add and sigmoid lookup, the structure of the PEs is fairly simple. Furthermore, neural network computation exposes fine grain regular parallelism that we exploited in our NPU design by including multiple identical PEs. In fact, our Parrot algorithmic transformation replaces unstructured serial code with a neural network that has structured fine grain parallelism that is executed on an efficient statically scheduled hardware structure, NPU.

6. EVALUATION

To evaluate the effectiveness of the Parrot transformation, we apply it to several benchmarks from diverse application domains. For each benchmark, we identify a region of code that is amenable to the Parrot transformation. We evaluate whole-application speedup and energy savings using cycle-accurate simulation and a power model. We also examine the resulting trade-off in computation accuracy. We refer the reader to the original paper for more details on the evaluation.⁹

6.1. Benchmarks

Table 1 lists the approximation-tolerant applications from diverse domains that are used to evaluate the broad applicability of our technique. These benchmarks are all written in C. The application domains—signal processing, robotics, gaming, compression, machine learning, and image processing—are selected for their usefulness to general applications and tolerance to imprecision. The domains are commensurate with evaluations of previous work on approximate computing.^{8, 19, 26, 27} To be able to assess the effect of the Parrot transformation perceptually, we selected a number of benchmarks that generate image outputs and compared the output image with and without the transformation. We did not reject any of the applications based on performance, energy, or accuracy shortfalls.

Table 1 also lists the input sets used for performance, energy, and accuracy assessment. These input sets are different from the ones used during the training phase of

Table 1. The benchmarks evaluated, details for each transformed function, input data, and the result of the Parrot transformation.

	Description	Type	Evaluation input set	No. of function calls	No. of loops	No. of ifs/elses	No. of x86-64 instructions	Training input set	Neural network topology	NN MSE	Error metric	Error
fft	Radix-2 Cooley-Tukey fast Fourier	Signal processing	2048 random floating-point numbers	2	0	0	34	32,768 random floating-point numbers	1 → 4 → 4 → 2	0.00002	Average relative error	7.22%
inversek2j	Inverse kinematics for two-joint arm	Robotics	10,000 (x, y) random coordinates	4	0	0	100	10,000 (x, y) random coordinates	2 → 8 → 2	0.00563	Average relative error	7.50%
jmeint	Triangle intersection detection	3D gaming	10,000 random pairs of 3D triangle coordinates	32	0	23	1079	100,000 random pairs of 3D triangle coordinates	18 → 32 → 8 → 2	0.00530	Miss rate	7.32%
jpeg	JPEG encoding	Compression	220 × 200-pixel color image	3	4	0	1257	Three 512 × 512-pixel color images	64 → 16 → 64	0.00890	Image difference	9.56%
kmeans	K-means clustering	Machine learning	220 × 200-pixel color image	1	0	0	26	50,000 pairs of random (r, g, b) values	6 → 8 → 4 → 1	0.00169	Image difference	6.18%
sobel	Sobel edge detector	Image processing	220 × 200-pixel color image	3	2	1	88	One 512 × 512-pixel color image	9 → 8 → 1	0.00234	Image difference	3.44%

the Parrot transformation. For applications with random inputs, we use a different random input set. For applications with image input, we use a different image.

Code annotation. The C source code for each benchmark was annotated as described in Section 2: we identified a single pure function with fixed-size inputs and outputs. No algorithmic changes were made to the benchmarks to accommodate the Parrot transformation. There are many choices for the selection of target code and, for some programs, multiple NPUs may even have been beneficial. For the purposes of this evaluation, however, we selected a single target region per benchmark that was easy to identify, frequently executed as to allow for efficiency gains, and amenable to learning by a neural network. Qualitatively, we found it straightforward to identify a reasonable candidate function in each benchmark.

As shown in Table 1, in most of the benchmarks we examined, the target code contains complex control flow including conditionals, loops, and method calls. In *jmeint*, the target code contains the bulk of the algorithm, including many nested method calls and numerous conditionals. In *jpeg*, the transformation subsumes the discrete cosine transform and quantization phases, which contain function calls and loops. In *fft*, *inversek2j*, and *sobel*, the target code consists mainly of arithmetic operations and simpler control flow. In *kmeans*, the target code is the Euclidean distance calculation, which is simple and fine grained yet frequently executed. In each case, the target code is side-effect-free and the number of inputs/outputs is statically identifiable.

Training data. To train the NPU for each application, we have used either (1) typical program inputs (e.g., sample images) or (2) a limited number of random inputs. For the benchmarks that use random inputs, we determined the permissible range of parameters in the code and generated uniform random inputs in that range. For the image-based benchmarks, we used three standard images that are used to evaluate image processing algorithms. For *kmeans*, we supplied random inputs to the code region to avoid overtraining on a particular test image.

Output quality. We use an application-specific error metric to assess the QoR for each benchmark. In all cases, we compare the output of the original untransformed application to the output of the transformed application. For *fft* and *inversek2j*, which generate numeric outputs, we measure the average relative error. *jmeint* calculates whether two three-dimensional triangles intersect; we report the misclassification rate. For *jpeg*, *kmeans*, and *sobel*, which produce image outputs, we use the average root-mean-square image difference.

As the last column of Table 1 shows, application average error rates range from 3% to 10%. This QoR loss is commensurate with other work on quality trade-offs.^{1, 8, 26}

^b **Processor:** Fetch/Issue Width: 4/6, INT ALUs/FPU: 3/2, Load/Store FUs: 2/2, ROB Entries: 96, Issue Queue Entries: 32, INT/FP Physical Registers: 256/256, Branch Predictor: Tournament 48KB, BTB Sets/Ways: 1024/4, RAS Entries: 64, Load/Store Queue Entries: 48/48, Dependence Predictor: 4096-entry Bloom Filter, ITLB/DTLB Entries: 128/256, L1: 32KB Instruction, 32KB Data, Line Width: 64 bytes, 8-Way, Latency: 3 cycles, L2: 2MB, Line Width 64 bytes, 8-Way, Latency: 12 cycles, **Memory Latency:** 50 ns (104 cycles).

6.2. Experimental setup

Simulation. We use the MARSSx86 cycle-accurate x86-64 simulator²² to evaluate the performance effect of the Parrot transformation and NPU acceleration. We configure the simulator to resemble Intel's Penryn microarchitecture^b, which is an aggressive out-of-order design. We augment MARSSx86 with a cycle-accurate NPU simulator and add support for NPU queue instructions through unused x86 opcodes. We use C assembly inlining to add the NPU invocation code. We compile the benchmarks using GCC version 4.4.6 with the `-O3` flag to enable aggressive compiler optimizations. The baseline in all of the reported results is the execution of the entire benchmark on the core without the Parrot transformation.

Energy modeling. MARSSx86 generates an event log during the cycle-accurate simulation of the program. The resulting statistics are sent to a modified version of McPAT¹⁸ to estimate the energy consumption of each execution. We model the energy consumption of an 8-PE NPU using the results from McPAT and CACTI 6.5²⁰ for memory arrays, buses, and steering logic. We use the results from Galal and Horowitz¹⁰ to estimate the energy of multiply-and-add operations. We model the NPU and the core at the 45 nm technology node. The NPU operates at the same frequency and voltage as the main core. We use the 2080 MHz frequency and $V_{dd} = 0.9$ V settings because the energy results in Galal and Horowitz¹⁰ use this frequency and voltage setting.

6.3. Experimental results

Figure 7a shows the application speedup when an 8-PE NPU is used to replace each benchmark's target function. The rest of the code runs on the core. The baseline is executing the entire, untransformed benchmark on the CPU. The plots also show the potential available speedup: the hypothetical speedup if the NPU takes zero cycles for computation. Among the benchmarks, *inversek2j* sees the highest speedup (11.1×) since the Parrot transformation substitutes the bulk of the application with a relatively small NN ($2 \rightarrow 8 \rightarrow 2$). On the other hand, *kmeans* sees a 24% slowdown even though it shows a potential speedup of 20% in the limit. The transformed region of code in *kmeans* consists of 26 mostly arithmetic instructions that can efficiently run on the core while the NN ($6 \rightarrow 8 \rightarrow 4 \rightarrow 1$) for this benchmark is comparatively complex and involves more computation (84 multiply-adds and 12 sigmoids) than the original code. On average, the benchmarks see a speedup of 2.3× through NPU acceleration.

Figure 7b shows the energy reduction for each benchmark. The baseline is the energy consumed by running the entire benchmark on the unmodified CPU and the ideal energy savings for a hypothetical zero-energy NPU. The Parrot transformation elides the execution of significant portion of dynamic instructions that otherwise would go through power-hungry stages of the OoO pipeline. The reduction in the number of dynamic instructions and the energy-efficient design of the NPU yield a 3.0× average application energy reduction.

Figure 7. Performance and energy improvements: (a) total application speedup with 8-PE NPU and (b) total application energy saving with 8-PE NPU.

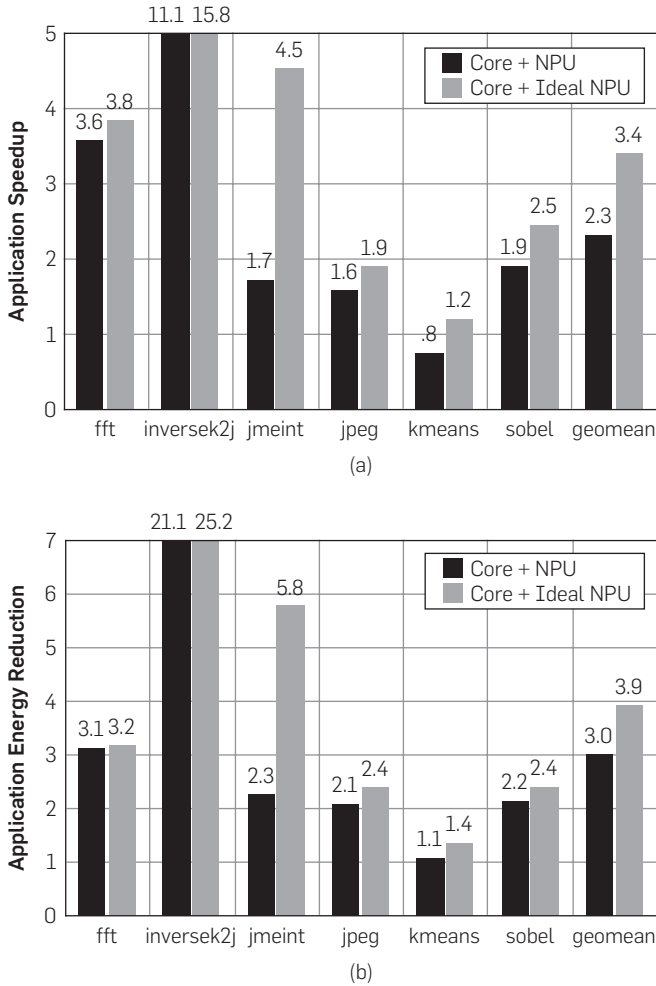
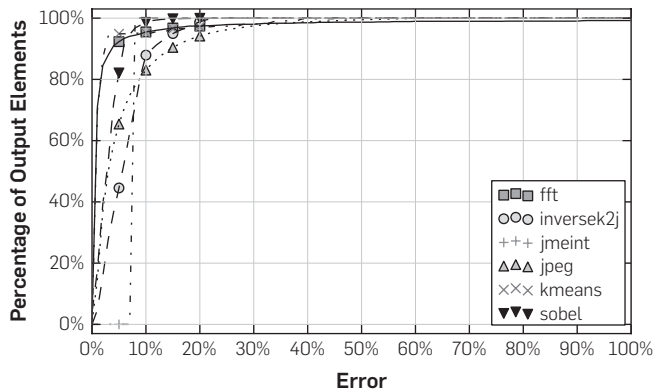


Figure 8. Cumulative distribution function (CDF) plot of the applications' output error. A point (x, y) indicates that y fraction of the output elements see error less than or equal to x.



To study the application-level quality loss in more detail, Figure 8 depicts the CDF (cumulative distribution function) plot of final error for each element of application's output. The output of each benchmark consists of

a collection of elements—an image consists of pixels, a vector consists of scalars, etc. The error CDF reveals the distribution of output errors among an application's output elements and shows that very few output elements see large quality loss. The majority (80–100%) of each transformed application's output elements have error less than 10%.

7. RELATED WORK

This work represents a convergence of three main bodies of research: approximate computing, general-purpose configurable acceleration, and hardware neural networks. Fundamentally, the Parrot transformation leverages hardware neural networks to create a new class of configurable accelerators for approximate programs.

Approximate computing. Prior work has explored relaxed hardware semantics and their impact on applications with “soft” output requirements, both as (1) extensions to traditional architectures^{4, 8, 19} and (2) in the form of fully approximate processing units.^{2, 21} In contrast, NPUs accelerate coarse-grained blocks of code in larger applications. No special code must be written to take advantage of the approximate unit; only lightweight annotation is required. Some work has also exposed relaxed semantics in the programming language to give programmers control over the precision of software.^{1, 4, 26} As an implementation of approximate semantics, the Parrot transformation dovetails with these programming models.

General-purpose configurable acceleration. The Parrot transformation extends prior work on configurable computing, synthesis, specialization, and acceleration that focuses on compiling traditional, imperative code for efficient hardware structures. One research direction seeks to synthesize efficient circuits or configure FPGAs to accelerate general-purpose code.^{23, 24} Similarly, static specialization has shown significant efficiency gains for irregular and legacy code.²⁹ More recently, configurable accelerators have been proposed that allow the main CPU to offload certain code to a small, efficient structure.^{11, 12} This work differs in its focus on accelerating *approximate* code. NPUs represent an opportunity to go beyond the efficiency gains that are possible when strict correctness is not required.

Neural networks. There is an extensive body of work on hardware implementation of neural networks (neural hardware) both digital⁶ and analog.¹⁷ Other work has examined the fault tolerance of hardware neural networks.^{16, 28} A recent study³ showed that 5 of 13 applications from the PARSEC suite can be manually reimplemented to make use of various kinds of neural networks.

8. LIMITATIONS AND FUTURE DIRECTIONS

Our results suggest that the Parrot transformation and NPU acceleration can provide significant performance and energy benefits. However, further research must address three limitations to the Parrot transformation as described in this work: (1) applicability, (2) programmer effort, and (3) quality and error control.

Applicability. Since neural networks inherently produce approximate results, not all code regions can undergo the

Parrot transformation. As enumerated in Section 3.1, a target code region must satisfy the following conditions:

- The region must be approximable. That is, the program must incorporate application-level tolerance of imprecision in the results of the candidate region.
- The region must have a bounded number of statically identifiable inputs and outputs.
- The region must be hot to benefit from acceleration.

Although these criteria form a basis for programmers or compilers to identify nominees for the Parrot transformation, they do not guarantee that the resulting neural network will accurately approximate the code region. There is no simple criterion that makes a certain task (here a candidate region) suited for learning by a neural network. However, our experience and results suggest that empirical assessment is effective to classify a wide variety of approximate functions as NPU-suitable. Follow-on work can improve on empirical assessment by identifying static code features that tend to indicate suitability for learning-based acceleration.

Programmer effort. In this paper, the Parrot transformation requires programmers to (1) identify approximable code regions and (2) provide application inputs to be used for training data collection.

As with the other approaches that ensure the safety of approximate computation and avoid catastrophic failures,²⁶ the programmer must explicitly provide information for the compiler to determine which code regions are safe to approximate. As Section 3.2 outlines, future work should explore allowing the compiler to automatically infer which blocks are amenable to approximation.

Because NPU acceleration depends on representative test cases, it resembles a large body of other techniques that use programmer-provided test inputs, including quality assurance (e.g., unit and integration testing) and profile-driven compilers. Future work should apply traditional coverage measurement and improvement techniques, such as test generation, to the Parrot transformation. In general, however, we found that it was straightforward to provide sufficient inputs for the programs we examined. This is in part because the candidate function is executed many times in a single application run, so a small number of inputs can suffice. Furthermore, as Section 4.2 mentions, an online version of the Parrot transformation workflow could use samples of postdeployment inputs if representative tests are not available predeployment.

Quality and error control. The results in this paper suggest that NPU acceleration can effectively approximate code with accuracy that is commensurate with state-of-the-art approximate computing techniques. However, there is always a possibility that, for some inputs, the NPU computes a significantly lower-quality result than the average case. In other words, without exhaustively exploring the NPU's input space, it is impossible to provide formal guarantees about its worst-case accuracy.

This unpredictability is common to other approximation techniques.^{8,26} As long as the frequency of low-quality

results is low and the application can tolerate these infrequent large errors, approximation techniques like NPUs can be effective. For this reason, future research should explore mechanisms to mitigate the frequency of such low-quality results. One such mechanism is to predict whether the NPU execution of the candidate region will be acceptable. For example, one embodiment would check whether an input falls in the range of inputs seen previously during training. If the prediction is negative, the original code can be invoked instead of the NPU. Alternatively, the runtime system could occasionally measure the error by comparing the NPU output to the original function's output. In case the sampled error is greater than a threshold, the neural network can be retrained.

9. CONCLUSION

Traditionally, hardware implementations of neural networks have been confined to specific classes of learning applications due to lack of traditional programming models. In this paper, we showed that the potential exists to use them to mimic and accelerate general-purpose programs that can tolerate small errors. Our learning transformation provides the bridge between neural and von Neumann models of computing and enables a general-purpose use case for neural hardware. The acceleration capability of NPUs aligns with both transistor and application trends, as transistors become less reliable and as imprecise applications grow in importance. In fact, our work demonstrates that neural accelerators can successfully mimic diverse regions of approximable imperative code. The Parrot algorithmic transformation converts different regions of code to a common neural network representation. Using neural networks as the common representation enables a new class of accelerators, NPUs, that yield significant application-level energy and performance savings. The levels of error introduced are comparable to those seen in previous approximate computing techniques. Besides introducing the Parrot algorithmic transformation and a new class of accelerators, this work leads to the following two additional key insights. First, the program transformation must consider a range of neural network topologies; a single topology is ineffective across diverse applications. Second, the accelerator must be tightly coupled with a processor's pipeline to enable acceleration even when fine-grained regions of code are transformed. By providing an end-to-end solution to meet these key requirements, the evaluated application suite ran 2.3× faster on average while using 3.0× less energy and maintaining accuracy greater than 90% in all cases. NPUs form a new class of trainable accelerators with potential implementations in both the digital and analog domains.

Acknowledgments

We thank Brandon Lucia, Jacob Nelson, Ardavan Pedram, Renée St. Amant, Karin Strauss, Xi Yang, Amir Yazdanbakhsh, and the members of the Sampa group for their feedback. This work was supported in part by NSF grant CCF-1016495 and research support from Microsoft. ■

References

- Baek, W. and Chilimbi, T.M. Green: A framework for supporting energy-conscious programming using controlled approximation. In *PLDI* (2010).
- Chakrapani, L.N., Akgul, B.E.S., Cheemalavagu, S., Korkmaz, P., Palem, K.V., and Seshasayee, B. Ultra-efficient (embedded) SOC architectures based on probabilistic CMOS (PCMOS) technology. In *DATE* (2006).
- Chen, T., Chen, Y., Duranton, M., Guo, Q., Hashmi, A., Lipasti, M., Nere, A., Qiu, S., Sebag, M., Temam, O., and Bench, N.N. On the broad potential application scope of hardware neural network accelerators. In *IISWC* (2012).
- de Kruijf, M., Nomura, S., and Sankaralingam, K. Relax: An architectural framework for software recovery of hardware faults. In *ISCA* (2010).
- Dennard, R.H., Gaensslen, F.H., Rideout, V.L., Bassous, E., and LeBlanc, A.R. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE J. Solid-State Circ.* 9 (Oct. 1974), 256–268.
- Esmailzadeh, H., Saeedi, P., Araabi, B.N., Lucas, C., and Fakhraie, S.M. Neural network stream processing core (NnSP) for embedded systems. In *ISCAS* (2006).
- Esmailzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., and Burger, D. Power challenges may end the multicore era. *Commun. ACM* 56, 2 (Feb. 2013), 93–102.
- Esmailzadeh, H. et al. Architecture support for disciplined approximate programming. In *ASPLOS* (2012).
- Esmailzadeh, H., Sampson, A., Ceze, L., and Burger, D. Neural acceleration for general-purpose approximate programs. In *MICRO* (2012).
- Galal, S. and Horowitz, M. Energy-efficient floating-point unit design. *IEEE Trans. Comput.* 60, 7 (2011) 913–922.
- Govindaraju, V., Ho, C.H., and Sankaralingam, K. Dynamically specialized datapaths for energy efficient computing. In *HPCA* (2011).
- Gupta, S., Feng, S., Ansari, A., Mahke, S., and August, D. Bundled execution of recurring traces for energy-efficient general purpose processing. In *MICRO* (2011).
- Guzhva, A., Dolenko, S., and Persiantsev, I. Multifold acceleration of neural network computations using GPU. In *ICANN* (2009).
- Hameed, R., Qadeer, W., Wachs, M., Azizi, O., Solomatnikov, A., Lee, B.C., Richardson, S., Kozyrak, C., and Horowitz, M. Understanding sources of inefficiency in general-purpose chips. In *ISCA* (2010).
- Hardavellas, N., Ferdman, M., Falsafi, B., and Ailamaki, A. Toward dark silicon in servers. *IEEE Micro* 31, 4 (July–Aug. 2011), 6–15.
- Hashmi, A., Berry, H., Temam, O., and Lipasti, M. Automatic abstraction and fault tolerance in cortical microarchitectures. In *ISCA* (2011).
- Joubert, A., Belhadji, B., Temam, O., and Hélot, R. Hardware spiking neurons design: Analog or digital? In *IJCNN* (2012).
- Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M., and Jouppi, N.P. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO* (2009).
- Liu, S., Pattabiraman, K., Moscibroda, T., and Zorn, B.G. Flicker: Saving refresh-power in mobile devices through critical data partitioning. In *ASPLOS* (2011).
- Muralimanoohar, N., Balasubramanian, R., and Jouppi, N. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *MICRO* (2007).
- Narayanan, S., Sartori, J., Kumar, R., and Jones, D.L. Scalable stochastic processors. In *DATE* (2010).
- Patel, A., Afram, F., Chen, S., and Ghose, K. MARSx86: A full system simulator for x86 CPUs. In *DAC* (2011).
- Putnam, A., Bennett, D., Dellinger, E., Mason, J., Sundararajan, P., and Eggers, S. CHIMPS: A high-level compilation flow for hybrid CPU-FPGA architectures. In *FPGA* (2008).
- Razdan, R. and Smith, M.D. A high-performance microarchitecture with hardware-programmable functional units. In *MICRO* (1994).
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. D.E. Rumelhart, J.L. McClelland, and PDP Research Group, eds. Volume 1. MIT Press, 1986, 318–362.
- Sampson, A., Dietl, W., Fortuna, E., Gnanapragasam, D., Ceze, L., and Grossman, D. EnerJ: Approximate data types for safe and general low-power computation. In *PLDI* (2011).
- Sidiropoulos-Douskos, S., Misailovic, S., Hoffmann, H., and Rinard, M. Managing performance vs. accuracy trade-offs with loop perforation. In *FSE* (2011).
- Temam, O. A defect-tolerant accelerator for emerging high-performance applications. In *ISCA* (2012).
- Venkatesh, G., Sampson, J., Goulding, N., Garcia, S., Bryksin, V., Lugo-Martinez, J., Swanson, S., and Taylor, M.B. Conservation cores: Reducing the energy of mature computations. In *ASPLOS* (2010).
- Zhu, J. and Sutton, P. FPGA implementations of neural networks: A survey of a decade of progress. In *FPL* (2003).

Hadi Esmailzadeh (hadi@cc.gatech.edu), Georgia Institute of Technology, Atlanta, GA.

Doug Burger (dburger@microsoft.com), Microsoft Research, Redmond, WA.

Adrian Sampson and Luis Ceze ([asampson, luisceze]@cs.washington.edu), University of Washington, Seattle, WA.

© 2015 ACM 0001-0782/15/01 \$15.00

World-Renowned Journals from ACM

ACM publishes over 50 magazines and journals that cover an array of established as well as emerging areas of the computing field. IT professionals worldwide depend on ACM's publications to keep them abreast of the latest technological developments and industry news in a timely, comprehensive manner of the highest quality and integrity. For a complete listing of ACM's leading magazines & journals, including our renowned Transaction Series, please visit the ACM publications homepage: www.acm.org/pubs.

ACM Transactions on Interactive Intelligent Systems



ACM Transactions on Interactive Intelligent Systems (TIIS). This quarterly journal publishes papers on research encompassing the design, realization, or evaluation of interactive systems incorporating some form of machine intelligence.

ACM Transactions on Computation Theory



ACM Transactions on Computation Theory (ToCT). This quarterly peer-reviewed journal has an emphasis on computational complexity, foundations of cryptography and other computation-based topics in theoretical computer science.

PLEASE CONTACT ACM MEMBER SERVICES TO PLACE AN ORDER
 Phone: 1.800.342.6626 (U.S. and Canada)
 +1.212.626.0500 (Global)
 Fax: +1.212.944.1318
 (Hours: 8:30am–4:30pm, Eastern Time)
 Email: acmhelp@acm.org
 Mail: ACM Member Services
 General Post Office
 PO Box 30777
 New York, NY 10087-0777 USA



Association for
Computing Machinery

Advancing Computing as a Science & Profession

www.acm.org/pubs

CAREERS

Albright College **Assistant Professor, Computer Science**

Albright College seeks applicants for a full-time tenure-track position in the Computer Science Department beginning in August 2015. Qualified candidates should have a Ph.D. (in hand or expected by August 2015) in Computer Science (or closely related discipline) and show evidence of a strong commitment to teaching at a liberal arts institution where students expect close contact with their professors and welcome opportunities to engage in collaborative research. A more detailed description can be found online: albright.edu/hr. Applicants should send a letter of application, vita, evidence of teaching effectiveness and scholarly activity, and three letters of reference (at least one about teaching) to hr@alb.edu. Review of applications will continue until the position is filled. Albright College is an AA/EOE and is strongly committed to diversity within its community. In pursuit of that, we actively encourage diversity among applicants for this position.

Arizona State University **Associate Professor in Social Computing** **School of Arts, Media and Engineering** **Herberger Institute for Design and the Arts**

The School of Arts, Media and Engineering (ame.asu.edu) in the Herberger Institute for Design and the Arts at Arizona State University invites applicants for a full-time tenure-track appointment in social computing at the Associate Professor level beginning Fall 2015. The successful candidate will take a leadership role in the Human Computer Interaction group, contribute significantly to the development of the education and research mission of the school in new areas such as culture and computing, and develop areas of computing with high potential social impact, such as cultural informatics or social networks.

Application Deadline: The application deadline is December 15, 2014; if not filled, reviews will occur monthly thereafter until the search is closed. For complete qualifications and application information, see http://ame.asu.edu/about/employment_faculty.php

Arizona State University is a VEVRAA Federal Contractor and an Equal Opportunity/Affirmative Action Employer. All qualified applicants will be considered without regard to race, color, sex, religion, national origin, disability, protected veteran status, or any other basis protected by law. See ASU's complete nondiscrimination statement at <https://asu.edu/aad/manuals/acd/acd401.html>. See ASU's Title IX policy at <https://www.asu.edu/titleIX/>

Arizona State University **Assistant Professor in Active Materials Design** **School of Arts, Media and Engineering** **Herberger Institute for Design and the Arts**

The School of Arts, Media and Engineering (ame.asu.edu) in the Herberger Institute for Design

and the Arts at Arizona State University invites applicants for a full-time tenure-track appointment in active materials design at the associate/assistant professor level beginning Fall 2015. The successful candidate will take a leadership role in the design, and the critical, cultural or conceptual studies of active materials and analog-computational media, and contribute significantly to the development of the education and research mission of the school.

Application Deadline: The application deadline is December 15, 2014; if not filled, reviews will occur monthly thereafter until the search is closed. For complete qualifications and application information, see http://ame.asu.edu/about/employment_faculty.php

Arizona State University is a VEVRAA Federal Contractor and an Equal Opportunity/Affirmative Action Employer. All qualified applicants will be considered without regard to race, color, sex, religion, national origin, disability, protected veteran status, or any other basis protected by law. See ASU's complete nondiscrimination statement at <https://asu.edu/aad/manuals/acd/acd401.html>. See ASU's Title IX policy at <https://www.asu.edu/titleIX/>

Arizona State University **Assistant Professor of Applied Computing** **Cyber Forensics**

Assistant Professor of Applied Computing, School of Mathematical and Natural Sciences, Arizona State University. Full-time tenure track position beginning August 2015. Seeking applicants to conduct research in Computer Science, to teach undergraduate courses, and to forge interdisciplinary collaborations in research and teaching. Ph.D. in Computer Science (or closely related field) and expertise in Cyber Forensics and/or Information Security is required. For complete qualification/application information, see <http://newcollege.asu.edu/jobs>. Application deadline is January 2, 2015. EO/AA.

Arizona State University **Assistant Professor in Philosophy of** **Technology and Science Technology Studies** **School of Arts, Media and Engineering** **Herberger Institute for Design and the Arts**

The School of Arts, Media and Engineering (ame.asu.edu) in the Herberger Institute for Design and the Arts at Arizona State University invites applicants for a full-time tenure-track appointment in philosophy of technology and science technology studies assistant professor level beginning fall 2014. The successful candidate will take a leadership role in critical, conceptual and historical studies of recent or emerging technology.

Application Deadline: The application deadline is December 15, 2014; if not filled, reviews will occur monthly thereafter until the search is closed. For complete qualifications and application information, see http://ame.asu.edu/about/employment_faculty.php

Arizona State University is a VEVRAA Federal Contractor and an Equal Opportunity/Affirmative Action Employer. All qualified applicants will be considered without regard to race, color, sex, religion, national origin, disability, protected veteran status, or any other basis protected by law. See ASU's complete non-discrimination statement at <https://asu.edu/aad/manuals/acd/acd401.html>. See ASU's Title IX policy at <https://www.asu.edu/titleIX/>

Arizona State University **Assistant Professor in Sound Studies /** **Sound Design** **School of Arts, Media and Engineering** **Herberger Institute for Design and the Arts**

The School of Arts, Media and Engineering (ame.asu.edu) in the Herberger Institute for Design and the Arts at Arizona State University invites applicants for a full-time tenure-track appointment in sound studies/sound design at the assistant professor level beginning Fall 2015. The successful candidate will take a leadership role in sound studies, sound art and design, and the aural or acoustic dimension of experience in contexts ranging from architecture, interior design, health, games, and performance to culture and history and contribute significantly to the development of the education and research mission of the School.

Application Deadline: The application deadline is December 15, 2014; if not filled, reviews will occur monthly thereafter until the search is closed. For complete qualifications and application information, see http://ame.asu.edu/about/employment_faculty.php

Arizona State University is a VEVRAA Federal Contractor and an Equal Opportunity/Affirmative Action Employer. All qualified applicants will be considered without regard to race, color, sex, religion, national origin, disability, protected veteran status, or any other basis protected by law. See ASU's complete nondiscrimination statement at <https://asu.edu/aad/manuals/acd/acd401.html>. See ASU's Title IX policy at <https://www.asu.edu/titleIX/>

California State University, Fullerton **Department of Computer Science** **Assistant Professor**

The Department of Computer Science invites applications for **several tenure-track positions at the Assistant Professor level** starting January or August 2015. For a complete description of the department, the position, desired specialization and other qualifications, please visit <http://hr.fullerton.edu/diversity/job-openings/>

Dartmouth College **Assistant Professor of Computer Science**

The Dartmouth College Department of Computer Science invites applications for a tenure-track faculty position at the level of assistant professor. We

seek candidates who will be excellent researchers and teachers in the areas of systems-level human-computer interaction, security, machine learning, and robotics, although outstanding candidates in any area will be considered. We particularly seek candidates who will help lead, initiate, and participate in collaborative research projects both within Computer Science and involving other Dartmouth researchers, including those in other Arts & Sciences departments, Dartmouth's Geisel School of Medicine, Thayer School of Engineering, and Tuck School of Business.

The department is home to 18 tenured and tenure-track faculty members and two research faculty members. Research areas of the department encompass the areas of security, computational biology, machine learning, robotics, systems, algorithms, theory, digital arts, vision, and graphics. The Computer Science department is in the School of Arts & Sciences, and it has strong Ph.D. and M.S. programs and outstanding undergraduate majors. The department is affiliated with Dartmouth's M.D.-Ph.D. program and has strong collaborations with Dartmouth's other schools.

Dartmouth College, a member of the Ivy League, is located in Hanover, New Hampshire (on the Vermont border). Dartmouth has a beautiful, historic campus, located in a scenic area on the Connecticut River. Recreational opportunities abound in all four seasons. With an even distribution of male and female students and over one third of the undergraduate student population members of minority groups, Dartmouth is committed to diversity and encourages applica-

tions from women and minorities.

To create an atmosphere supportive of research, Dartmouth offers new faculty members grants for research-related expenses, a quarter of sabbatical leave for each three academic years in residence, and flexible scheduling of teaching responsibilities.

Application Instructions

Applicants are invited to submit application materials via Interfolio at apply.interfolio.com/25574. Upload a CV, research statement, and teaching statement, and request at least four references to upload letters of recommendation, at least one of which should comment on teaching. Email facsearch15@cs.dartmouth.edu with any questions.

Application review will begin November 1, 2014, and continue until the position is filled.

Recommendations: After you have entered the contact information for your references and sent them reference requests, you may submit your application, even before they upload the letters. Once the letters are uploaded, there's an extra step: you must associate the letters with your application for this particular position.

Equal Employment Opportunity Statement

Dartmouth is an equal opportunity/ affirmative action employer with a strong commitment to diversity. In that spirit, we are particularly interested in receiving applications from a broad spectrum of people, including women, persons of color, persons with disabilities, veterans or any other legally protected group.

Florida State University Department of Computer Science Tenure-Track Assistant Professor

The Department of Computer Science at the Florida State University invites applications for one tenure-track Assistant Professor position to begin August 2015. The position is 9-mo, full-time, tenure-track, and benefits eligible. We are seeking outstanding applicants with strengths in the areas of Big Data and Digital Forensics. Outstanding applicants specializing in other research areas will also be considered. Applicants should hold a PhD in Computer Science or closely related field, and have excellent research and teaching accomplishments or potential. The department offers degrees at the BS, MS, and PhD levels. The department is an NSA Center of Academic Excellence in Information Assurance Education (CAE/IAE) and Research (CAE-R).

FSU is classified as a Carnegie Research I university. Its primary role is to serve as a center for advanced graduate and professional studies while emphasizing research and providing excellence in undergraduate education. Further information can be found at <http://www.cs.fsu.edu>

Screening will begin January 1, 2015 and will continue until the position is filled. Please apply online with curriculum vitae, statements of teaching and research philosophy, and the names of five references, at <http://www.cs.fsu.edu/positions/apply.html>

Questions can be e-mailed to Prof. Xiuwen Liu, Faculty Search Committee Chair, recruitment@cs.fsu.edu.



Florida International University is recognized as a Carnegie engaged university. It is a public research university with colleges and schools that offers more than 180 bachelor's, master's and doctoral programs in fields such as engineering, international relations, architecture, law and medicine. As one of South Florida's anchor institutions, FIU contributes \$9.8 billion each year to the local economy. FIU is Worlds Ahead in finding solutions to the most challenging problems of our time. FIU emphasizes research as a major component of its mission. FIU has awarded over 200,000 degrees and enrolls more than 54,000 students in two campuses and three centers including FIU Downtown on Brickell, FIU@I-75, and the Miami Beach Urban Studios. FIU also supports artistic and cultural engagement through its three museums: the Patricia & Phillip Frost Art Museum, the Wolfsonian-FIU, and the Jewish Museum of Florida-FIU. FIU is a member of Conference USA and has over 400 student-athletes participating in 18 sports. For more information about FIU, visit <http://www.fiu.edu/>.

The School of Computing and Information Sciences (SCIS) seeks exceptionally qualified candidates for tenure-track and tenured faculty positions at all levels as well as non-tenure track faculty positions at the level of Instructor, including visiting instructor appointments. SCIS is a rapidly growing program of excellence at the University, with 30 tenure-track faculty members and over 1,800 students, including over 80 Ph.D. students. SCIS offers B.S., M.S., and Ph.D. degrees in Computer Science, an M.S. degree in Telecommunications and Networking, and B.S., B.A., and M.S. degrees in Information Technology. SCIS has received over \$22M in the last four years in external research funding, has six research centers/clusters with first-class computing and support infrastructure, and enjoys broad and dynamic industry and international partnerships.

Open-Rank Tenure Track/Tenured Positions (Job ID# 508676)

SCIS seeks exceptionally qualified candidates for tenure-track and tenured faculty positions at all levels. We seek well-qualified candidates in all areas; researchers in the areas of cybersecurity, cognitive modeling, compilers and programming languages, computer architecture, databases, information retrieval and big data, and biomedical systems are particularly encouraged to apply. Preference will be given to candidates who will enhance or complement our existing research strengths.

Ideal candidates for junior positions should have a record of exceptional research in their early careers. Candidates for senior positions must have an active and proven record of excellence in funded research, publications, and professional service, as well as a demonstrated ability to develop and lead collaborative research projects. In addition to developing or expanding a high-quality research program, all successful applicants must be committed to excellence in teaching at both the graduate and undergraduate levels. An earned Ph.D. in Computer Science or related disciplines is required.

Non-tenure track instructor positions (Job Opening 507474)

We seek well-qualified candidates in all areas of Computer Science and Information Technology. Ideal candidates must be committed to excellence in teaching a variety of courses at the undergraduate level. A graduate degree in Computer Science or related disciplines is required; significant prior teaching and industry experience and/or a Ph.D. in Computer Science is preferred.

HOW TO APPLY: Qualified candidates for open-rank faculty positions are encouraged to apply to (Job Opening ID #508676); and candidates for instructor positions are encouraged to apply to (Job Opening ID # 507474). Submit applications at facultycareers.fiu.edu and attach cover letter, curriculum vitae, statement of teaching philosophy, research statement, etc as individual attachments. Candidates will be required to provide names and contact information for at least three references who will be contacted as determined by the search committee. To receive full consideration, applications and required materials should be received by December 31st, 2014. Review will continue until position is filled.

If you are interested in a visiting appointment please contact the department directly by emailing Dr. Mark Weiss at Weiss@cis.fiu.edu. All other applicants should apply by going to facultycareers.fiu.edu.

FIU is a member of the State University System of Florida and an Equal Opportunity, Equal Access Affirmative Action Employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, national origin, disability status, protected veteran status, or any other characteristic protected by law.

Faculty Positions in Visualization



The Computer, Electrical, and Mathematical Sciences and Engineering (CEMSE) Division at King Abdullah University of Science and Technology (KAUST) invites applications for faculty positions at all levels (Full, Associate, and Assistant Professor) in the area of visualization.

KAUST is an international, graduate-level research university dedicated to advancing science and technology through interdisciplinary research, education, and innovation. Located on the shores of the Red Sea in Saudi Arabia, KAUST offers superb research facilities, generous assured research funding, and internationally competitive salaries, attracting top international faculty, scientists, engineers, and students to conduct curiosity-driven and goal-oriented research to address the world's pressing scientific and technological challenges in the areas of energy, food, water, and the environment.

KAUST is seeking candidates with an established track record of research in scientific visualization, information visualization, or visual analytics. Successful candidates will have a PhD in Computer Science or equivalent, and a strong publication record in top-tier conferences and journals. Senior candidates must have demonstrated strong leadership in the field. Successful candidates will be appointed within the Computer Science program, and are expected to engage with the KAUST Visual Computing Center (VCC).

The VCC is KAUST's hub for research activities spanning all areas of visual computing, ranging including imaging, computer vision, computer graphics, and visualization, as well as interdisciplinary applications in visual computing. The VCC offers a unique combination of an intellectually stimulating environment and access to superb facilities, including large-scale virtual reality installations in the KAUST Visualization Core Lab (KVL) and KAUST's 5 petaflop/s Shaheen-2 supercomputer.

Candidates should apply on the <http://apptrkr.com/537705> employment site. Candidates should include the names of three references for Assistant Professor positions and at least six for senior positions. Applications will be considered until the positions are filled. The application cycle will close on April 15, 2015. Prospective candidates are advised to apply as soon as possible.



www.kaust.edu.sa

Equal Employment Opportunity

An Equal Opportunity/Access/Affirmative Action/Pro Disabled & Veteran Employer committed to enhancing the diversity of its faculty and students. Individuals from traditionally underrepresented groups are encouraged to apply.

FSU's Equal Opportunity Statement can be viewed at: http://www.hr.fsu.edu/PDF/Publications/diversity/EEO_Statement.pdf

George Mason University Computer Science Department Volgenau School of Engineering Tenure Track Assistant Professor

The Department of Computer Science in the Volgenau School of Engineering at George Mason University invites applications for a tenure-track faculty position at the rank of Assistant Professor beginning Fall 2015. While applicants in all areas of computer science will be considered, we are particularly interested in candidates in the areas of security and privacy.

Minimum qualifications for the position include a Ph.D. in Computer Science or a related field, demonstrated potential for excellence and productivity in research, and a commitment to high quality teaching. The department has over 40 faculty members with wide-ranging research interests including artificial intelligence, algorithms, computational biology, computer graphics, computer vision, databases, data mining, parallel and distributed systems, real-time systems, robotics, security, software engineering, and wireless and

mobile computing. The CS department has over \$6 Million in annual research funding and has 10 recipients of NSF's prestigious CAREER awards.

In addition to BS, MS and PhD programs in Computer Science, the department offers a MS in Information Security and Assurance. The department also participates in an inter-disciplinary BS in Cyber-Security Engineering offered by the Volgenau School of Engineering. For more information on the department, visit our Web site: <http://cs.gmu.edu/>.

For full consideration please submit application and application materials on-line at <http://jobs.gmu.edu> (position number F9866Z). To apply, you will need a statement of professional goals including your perspective on teaching and research, a complete C.V. with publications, and the names of three references. The review of applications will begin on January 16, 2015 and continue until the position is filled.

George Mason University is an equal opportunity employer encouraging diversity.

Georgetown University Department of Computer Science Assistant Professor

The Department of Computer Science at Georgetown University invites applications for a tenure-track position in Computer Science at the Assistant Professor level. We are particularly interested in applicants with expertise in large-scale data analytics, which includes web analytics, data science, database, data mining, machine learning, applied algorithms, data and visual analytic

systems, or related fields. A successful candidate is one who has demonstrated ability to conduct high-quality research in one or more of the areas of interest and has the potential to excel in teaching at the undergraduate and graduate levels. The proposed start date is August 2015. Required Qualifications: Applicants must have a Ph.D. in Computer Science or a closely related field. The degree must be completed before August 2015.

Candidates should visit <http://www.cs.georgetown.edu/positions> for instructions on submitting applications. Applications received by January 15, 2015, will receive full consideration for first-round interviews though later applications are welcome.

Georgetown University is an Equal Opportunity, Affirmative Action Employer fully dedicated to achieving a diverse faculty and staff. All qualified candidates are encouraged to apply and will receive consideration for employment without regard to race, sex, sexual orientation, age, religion, national origin, marital status, veteran status, disability or other categories protected by law.

Max Planck Institute for Software Systems

Tenure Track Openings

Applications are invited for tenure-track faculty positions in all areas related to the theory and practice of software systems. These areas include, but are not limited to, security and privacy, embedded and mobile systems, social computing, data science, programming languages and systems, software verification and analysis, parallel, distributed, and networked systems.

A doctoral degree in computer science or related areas and an outstanding research record are required. Successful candidates are expected to build a team and pursue a highly visible research agenda, both independently and in collaboration with other groups.

MPI-SWS, founded in 2005, is part of a network of over 80 Max Planck Institutes, Germany's premier basic research facilities. MPIs have an established record of world-class, foundational research in the fields of medicine, biology, chemistry, physics, technology and humanities. Since 1948, MPI researchers have won 17 Nobel prizes. MPI-SWS aspires to meet the highest standards of excellence and international recognition with its research in software systems.

To this end, the institute offers a unique environment that combines the best aspects of a university department and a research laboratory:

- Faculty enjoy academic freedom, receive institutional funding and attract additional third-party funds to build and lead a team of graduate students and post-docs.
- Faculty supervise doctoral theses, and have the opportunity to teach graduate and undergraduate courses.
- Faculty enjoy outstanding technical and administrative support, as well as internationally competitive compensation.

MPI-SWS currently has 10 tenured and tenure-track faculty and about 50 doctoral and post-doctoral researchers. The institute will grow to at least 17 faculty over the coming years. We maintain an open, international and diverse work environment and seek applications from outstanding researchers regardless of national origin or citizenship. The working language is English; knowl-



THE CHINESE UNIVERSITY OF HONG KONG

Applications are invited for:-

Department of Computer Science and Engineering Professors / Associate Professors / Assistant Professors

(Ref. 1415/078(255)/2)

The Department invites applications for Professorships / Associate Professorships / Assistant Professorships in computer engineering to pursue new strategic research initiatives, to fill faculty openings within current strengths and to teach in the new curriculum. The Department is looking for a leader and a couple of young and aspiring professors for added momentum to its Computer Engineering Programme, and in particular, talents in the following areas:

- advanced architectural and 3D chip design for energy efficient computing; and
- hardware security for cloud computing.

Applicants should have (i) a PhD degree; and (ii) a good scholarly record demonstrating potential for teaching and research excellence. The appointees will (a) teach both undergraduate and postgraduate courses; (b) develop a significant independent research programme with external funding; and (c) supervise postgraduate students. Appointments will normally be made on contract basis for two to three years initially commencing August 2015, which, subject to performance and mutual agreement, may lead to longer-term appointment or substantiation later. Applications will be accepted until the posts are filled.

Salary and Fringe Benefits

Salary will be highly competitive, commensurate with qualifications and experience. The University offers a comprehensive fringe benefit package, including medical care, plus a contract-end gratuity for appointments of two years or longer, and housing benefits for eligible appointees. Further information about the University and the general terms of service for appointments is available at <http://www.per.cuhk.edu.hk>. The terms mentioned herein are for reference only and are subject to revision by the University.

Application Procedure

Please send full resume, copies of academic credentials, publication list with abstracts of selected published papers, details of courses taught and evaluation results (if any), a research plan and a teaching statement, together with names of three to five referees to the Dean of Engineering by e-mail to recruit@erg.cuhk.edu.hk. Applicants should mark clearly the area(s) of their interests. The Personal Information Collection Statement will be provided upon request. Please quote the reference number and mark 'Application - Confidential' on cover.

Faculty Positions: COMPUTATIONAL SIMULATION AND DATA ANALYTICS AT EXTREME SCALE

The Computer, Electrical, and Mathematical Sciences and Engineering (CEMSE) Division at King Abdullah University of Science and Technology (KAUST) invites applications for faculty positions at all levels (Full, Associate, and Assistant Professor) in computational simulation and data analytics at an extreme scale for Fall 2015.

KAUST is an international graduate-level research university dedicated to advancing science and technology through interdisciplinary research, education, and innovation. Located on the shores of the Red Sea in Saudi Arabia, KAUST offers superb research facilities, generous assured research funding, and internationally competitive salaries, attracting top international faculty, scientists, engineers, and students to conduct curiosity-driven and goal-oriented research to address the world's pressing scientific and technological challenges in the areas of energy, food, water, and the environment.

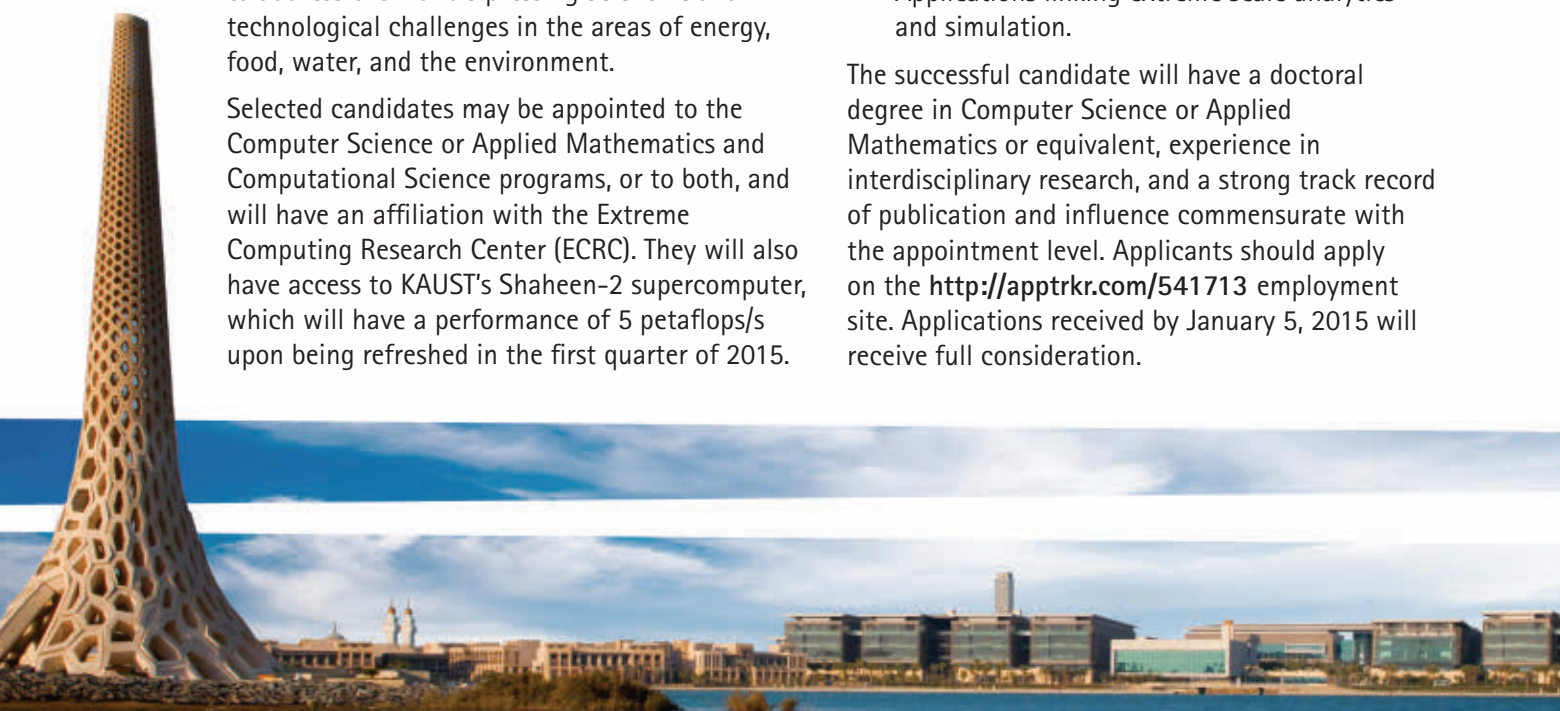
Selected candidates may be appointed to the Computer Science or Applied Mathematics and Computational Science programs, or to both, and will have an affiliation with the Extreme Computing Research Center (ECRC). They will also have access to KAUST's Shaheen-2 supercomputer, which will have a performance of 5 petaflops/s upon being refreshed in the first quarter of 2015.

The primary focus of the ECRC is numerical algorithms for emerging architectures with a premium on power efficiency at scale through reduction in communication, synchronization, and storage, and algorithm-based fault tolerance. Experience with applications relevant to KAUST's technology missions (bioinformatics, catalysis, climate, combustion, molecular dynamics, reservoir modeling, seismic inversion, etc.) is a plus.

Priority areas for 2014-2015 in the ECRC are:

- High performance computing paradigms, techniques, and software tools
- High performance computing systems and performance analysis
- Algorithms for extreme scale analytics
- Algorithms for extreme scale simulation
- Applications linking extreme scale analytics and simulation.

The successful candidate will have a doctoral degree in Computer Science or Applied Mathematics or equivalent, experience in interdisciplinary research, and a strong track record of publication and influence commensurate with the appointment level. Applicants should apply on the <http://apptrkr.com/541713> employment site. Applications received by January 5, 2015 will receive full consideration.



edge of the German language is not required for a successful career at the institute.

The institute is located in Kaiserslautern and Saarbrücken, in the tri-border area of Germany, France and Luxembourg. In immediate proximity are the MPI for Informatics, Saarland University, the Technical University of Kaiserslautern, the German Center for Artificial Intelligence (DFKI), and the Fraunhofer Institutes for Experimental Software Engineering and for Industrial Mathematics.

Qualified candidates should apply online via the secure application form. The review of applications will begin on December 1, 2014, and applicants are strongly encouraged to apply by that date; however, applications will continue to be accepted through December 2014. Apply URL: <http://www.mpi-sws.org/index.php?n=careers/tenure-track>

The institute is committed to increasing the representation of minorities, women and individuals with physical disabilities in Computer Science. We particularly encourage such individuals to apply.

The initial tenure-track appointment is for five years; it can be extended to seven years based on a midterm evaluation in the fourth year. A permanent contract can be awarded after a successful tenure evaluation in the sixth year.

National Taiwan University **Professor-Associate Professor-Assistant Professor**

The Department of Computer Science and Information Engineering, and the Graduate Institute of Networking and Multimedia at National Taiwan

Univ. have faculty openings at all ranks beginning in August 2015. Highly qualified candidates in all areas of computer science are invited to apply. A Ph.D. or its equivalent is required. Applicants are expected to conduct outstanding research and be committed to teaching. Candidates should check http://www.csie.ntu.edu.tw/faculty_recruiting/ for submitting applications. The deadline is February 28, 2015. Contact Prof. Chia-Lin Yang at faculty_search@csie.ntu.edu.tw for any questions. An early submission is strongly encouraged.

Northern Illinois University **Computer Science Department** **Assistant Professor**

The **Computer Science Department** at **Northern Illinois University** invites applicants to the tenure-track appointments it anticipates making at the rank of **Assistant Professor** to start in August 2015. Responsibilities include teaching, scholarship, grantsmanship, and service.

Candidates must have or expect to complete a Ph.D. or equivalent degree in computer science by August 16, 2015. Candidates from all research areas are welcome, but preference will be given to candidates with expertise in Data Science and Applied Algorithms.

Candidates must have expertise or evident potential for quality teaching at both the undergraduate and graduate levels. Further, candidates must show evidence of, or potential for, publishing in premiere peer-reviewed journals, developing an independent line of research, and securing

external funding. Finally, candidates must have effective interpersonal communication skills and a commitment to teamwork and collegiality.

Northern Illinois University is an Affirmative Action/Equal Opportunity Employer.

A pre-employment criminal background investigation is required.

Qualified individuals must submit a cover letter, current curriculum vitae, a statement of research vision, a statement of teaching interests, and a list of three references. All materials must be submitted as a single PDF file to applicants@cs.niu.edu by January 31, 2015.

Ohio State University **Computer Science and Engineering** **Department** **Multiple Tenure-Track Positions**

The Computer Science and Engineering Department at the **Ohio State University** expects to fill multiple tenure-track positions and seeks applicants in all areas of computer science.

Particular emphasis will be placed on filling two open rank positions in conjunction with a broad university-wide research initiative on Data Analytics (<https://discovery.osu.edu/focus-areas/data-analytics/collaborative.html>), and a recently announced undergraduate program in Data Analytics (<https://data-analytics.osu.edu>). Areas of interest for these positions include (but are not limited to): data mining, big data management, cloud computing systems, data analytics; application of machine learning or data mining or data visual-



Your next job is waiting.

Many of today's senior tech jobs go unfilled as candidates lack the advanced skills and training needed. Demonstrate your commitment to get ahead and earn CEUs by completing non-credit courses from **Georgia Tech's Master of Science in Computer Science** online program.

Each course is just \$399.

For more information or to register, visit pe.gatech.edu/omscs-acm

Georgia Tech Professional Education

ization to problems in network science (including social networks), health, and climate science.

The department is committed to enhancing faculty diversity; women, minorities, and individuals with disabilities are especially encouraged to apply.

Applicants should hold or be completing a Ph.D. in CSE or a closely related field, have a commitment to and demonstrated record of excellence in research, and a commitment to excellence in teaching.

To apply, please submit your application via the online database. The link can be found at:

<https://web.cse.ohio-state.edu/cgi-bin/portal/fsearch/apply.cgi>.

Review of applications will begin in November 2014 and will continue until the positions are filled.

The Ohio State University is an Equal Opportunity/Affirmative Action Employer.

Oregon State University

School of Electrical Engineering and Computer Science

Assistant/Associate/Full Professor tenure-track Position in Computer Science

The School of Electrical Engineering and Computer Science at Oregon State University invites applications for two full time nine-month tenure-track faculty positions at the Assistant Professor level in Computer Science. (Appointments may also be made at the Associate or Full Professor level, see the OSU website for details). We seek candidates

with demonstrated research strengths in the areas of computer vision, databases and data management. A Ph.D. in Computer Science or related field is required by the start of employment. Candidates should have a strong commitment to high quality teaching and development of a vibrant research program. Applicants should demonstrate enthusiasm for collaborating with other research groups in the School of EECS, with other departments at Oregon State University, and with other universities. Duties include teaching undergraduate and graduate courses, conducting research in the area of interest, and securing funding.

The School of EECS is housed in the Kelley Engineering Center, a green building designed to support collaboration among faculty and students across campus. The School emphasizes a culture of collegiality and excellence in both education and research. With 52 tenure/tenure-track faculty, we enroll 200 PhD, 195 MS and 2,200 undergraduate students. Oregon State University is located in Corvallis, a college town renowned for its high quality of life. OSU is an AA/EOE/Vets/Disabled. For more information, including full position announcement and instructions for application, visit: <http://jobs.oregon-state.edu/applicants/Central?quickFind=64999>

Princeton University

Computer Science

Assistant Professor - Tenure Track

The Department of Computer Science at Princeton University invites applications for faculty positions at the **Assistant Professor** level. We are accepting

applications in all areas of Computer Science. Applicants must demonstrate superior research and scholarship potential as well as teaching ability. A PhD in Computer Science or a related area is required. Candidates should expect to receive their PhD before Fall, 2015. More senior appointments may be considered for extraordinary candidates. Successful candidates are expected to pursue an active research program and to contribute significantly to the teaching programs of the department. Applicants should include a CV and contact information for at least three people who can comment on the applicant's professional qualifications. There is no deadline, but review of applications will be underway by December 2014.

Princeton University is an equal opportunity employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, national origin, disability status, protected veteran status, or any other characteristic protected by law. This position is subject to the University's background check policy.

You may apply online at: <http://jobs.cs.princeton.edu/>

Purdue University

Assistant/Associate/Full Professor

Purdue University has identified **Big Data** as a major thrust for inter-disciplinary research and education. The departments of Computer Science, and Electrical and Computer Engineering will consider outstanding candidates with an interest in all data-related areas of physics with equal priority given to all areas.



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Rates: \$325.00 for six lines of text, 40 characters per line. \$32.50 for each additional line after the first six. The MINIMUM is six lines.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact: acmm mediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

Ads are listed for a period of 30 days.

For More Information Contact:

**ACM Media Sales
at 212-626-0686 or
acmm mediasales@acm.org**



Northeastern University College of Engineering

The **College of Engineering (COE)** seeks several outstanding engineering experts to join their faculty to teach in their professional MS programs. Recently, the college has seen significant growth with research initiatives (8 federally-funded research centers) and with enrollment in graduate programs. The COE seeks to hire a teaching faculty to support six different Master of Science programs that are also described as Professional Master of Science Programs (PMP). Successful candidates will join a group committed to the continued growth and expansion of graduate level programs that result in continued prominence in the field of Engineering education and professional preparation.

The main responsibilities for this position include developing course content, preparing highly engaging academic activities, advising students; collaborating to identify and hire part-time instructors, and serving at the department, university, and profession levels. Expertise of successful hires will focus on one of the Professional MS Program area and will have the ability to teach courses that support the other programs. This position requires active scholarship, with particular interest in educational, instructional, and industry-inspired research.

This position is highly collaborative with other faculty in the growing Professional MS group and with the cooperative education faculty.

Interdisciplinary Professional Master of Science Programs:

- Computer Systems Engineering
- Energy Systems
- Engineering Management
- Information Systems
- Sustainable Building Systems
- Telecommunications Systems Management

For more information and to apply, please visit:
www.coe.neu.edu/PMP-Faculty

Several faculty positions are available in enabling technologies as well as in domain sciences that deal with the use and management of digital data. The new hires will join a strong faculty group engaged in this thrust area and will have a unique opportunity to help shape Purdue's vision and research/education agenda in Big Data. Candidates for tenure and tenure-track positions at the Assistant/Associate professor level will be considered. Strong interest in inter-disciplinary activities is required. Outstanding candidates with an interest in all data-related areas of computer science and engineering, physics will be considered. Examples of specific fields of interest are data infrastructure (storage and networking), system software infrastructure (OS, programming models), scalable analytics (data mining and machine learning at scale), analytics presentation (information visualization) and data-enabled methods (data curation and preservation) and applications in domain sciences. A detailed description of open positions is available at <http://www.purdue.edu/bigdata/hiring>.

Candidates must hold a PhD degree in one of the above-mentioned areas or in a closely related discipline, have demonstrated potential for excellence in research, and be committed to excellence in teaching. Successful candidates will be expected to teach undergraduate and graduate courses, conduct research in their field of expertise, and participate in interdisciplinary Big-Data activities.

Salary and benefits are highly competitive, and Purdue is a dual career friendly employer. Submit applications online at <http://www.purdue.edu/bigdata/hiring>. The application should include a cover letter, a complete and detailed vitae, and statements of research and teaching interests. Also, please include names, addresses, telephone numbers, and email addresses for three or more references. Hard copy applications can be sent to: Faculty Search Chair, Big Data Search Committee,

305 N. University Street, Purdue University, West Lafayette, IN 47907. For questions regarding the application process, please contact Nicole Piegza (piegza@purdue.edu). Applications for positions to begin in Fall 2015 are being considered as of October 9, 2014; the review will continue until the positions are filled. A background check will be required for employment in these positions. Purdue University is an EEO/AA employer fully committed to achieving a diverse workforce. All individuals, including minorities, women, individuals with disabilities, and protected veterans are encouraged to apply. Apply URL: <https://hiring.science.purdue.edu>

Purdue University Computer Science Visiting Assistant/Associate/Full Professor

The Department of Computer Science at Purdue University is soliciting applications for visiting positions at the Assistant, Associate and Full Professor level. Visiting appointments are expected to be for a 2-year period with an expected teaching load of 3 courses per year. Candidates in all areas of computer science will be considered. Applicants should hold a PhD in Computer Science, or related discipline, have prior teaching experience, have a commitment to excellence in teaching and be able to teach a wide range of undergraduates and graduate courses. A demonstrated research record in an area represented in the department is expected. Review of applications and candidate interviews will begin early in December 2014, and will continue until the positions are filled.

The Department of Computer Science offers a stimulating and nurturing academic and educational environment with thriving undergraduates and graduate programs and active research programs in most areas of computer science. Information about the department and a description of open positions are available at <http://www.cs.purdue.edu>. Salary and benefits are competitive.

Applicants should hold a PhD in Computer Science, or related discipline, have prior teaching experience, have a commitment to excellence in teaching and be able to teach a wide range of undergraduates and graduate courses. A demonstrated research record in an area represented in the department is expected.

Applicants are strongly encouraged to apply online at <https://hiring.science.purdue.edu>. Alternatively, hardcopy applications can be sent to: Faculty Search Chair, Department of Computer Science, 305 N. University Street, Purdue University, West Lafayette, IN 47907. A background check will be required for employment. Apply URL: <https://www.cs.purdue.edu/hiring/visiting2014.html> Purdue University is an EEO/AA employer fully committed to achieving a diverse workforce. All individuals, including minorities, women, individuals with disabilities, and protected veterans are encouraged to apply.

State University of New York at Binghamton Department of Computer Science Six Tenure-Track Positions

The Computer Science Department at Binghamton University has **six tenure-track positions** beginning Fall 2015. Applicants should have a

Ph.D. in Computer Science or related discipline, a strong research record, and a commitment to teaching. Qualified applications are invited from candidates with specializations in any of these four areas: (1) portable device/system design & energy-aware systems design (two positions), (2) healthcare information systems & data analytics for healthcare (two positions), (3) cybersecurity (one position) and, (4) computer networks (Associate level will also be considered).

The Department has established graduate and undergraduate programs, including 60 full-time PhD students and 27 Faculty members. Junior faculty have a significantly reduced teaching load for at least the first three years. Further details and application information are available at:

<http://binghamton.interviewexchange.com/jobofferdetails.jsp?JOBID=53958>

For the faculty positions in areas (1) and (2) above, the Department seeks a research scholar with research that will affiliate with the Binghamton University Transdisciplinary Areas of Excellence Initiative in Smart Energy (<http://www.binghamton.edu/tae/smart-energy/>) and Healthcare systems (<http://www.binghamton.edu/tae/health-sciences/index.html>) respectively. Applications will be reviewed until positions are filled. First consideration will be given to applications received by February 20, 2015.

We are an EE/AA employer.

University of Arizona Assistant Professor

The Department of Computer Science at The University of Arizona invites applications for **two tenure-track faculty positions at the Assistant Professor level**:

1. Machine Learning and related areas. The successful candidate will conduct core computer science research in machine learning and related areas and also participate in the University of Arizona's Statistics Graduate Interdisciplinary Program.

2. Large-scale data-centric computing ("Big Data"). The research areas of distributed systems, storage and database systems, information visualization, security and privacy, and scalable storage are especially desirable. Outstanding candidates in related areas will also be considered.

The Department of Computer Science at the University of Arizona has a long history of research accomplishment, influential software distribution, and substantial external funding. Major funding has included four NSF infrastructure grants. Current research areas include algorithms, architecture, bioinformatics, compilers, computational geometry, databases, distributed systems, embedded systems, networks, operating systems, security, sensor networks, vision, and visualization. More information about the University of Arizona and its environs is available at www.whya.com.

The minimum requirement for the position is a PhD in Computer Science or closely related discipline. Review of applications will begin January 2, 2015 and continue until the position is filled. Please email search@cs.arizona.edu if you have any questions or need assistance.

The University of Arizona is a committed

ACM Inroads

Paving the way toward excellence in computing education

The magazine for computing educators worldwide

<http://inroads.acm.org>

acm Inroads

Association for Computing Machinery

Advancing Computing as a Science & Profession

Equal Opportunity/Affirmative Action Institution. Women, minorities, veterans and individuals with disabilities are encouraged to apply.

Application Instructions

Current information regarding this position and instructions for applying are available at the UA Human Resources web site at the following URLs:

- Big Data position:
<http://www.uacareers.com/56718>
- Machine Learning position:
<http://www.uacareers.com/56720>

Be sure to upload (a) your curriculum vitae, (b) a statement of your research and teaching interests, (c) the names of at least three references, and (d) your letter of interest.

Inquiries/Contact: Saumya Debray; Email: debray@email.arizona.edu

University of California, Irvine Department of Computer Science Three Faculty Positions

The Department of Computer Science (CS) at the University of California, Irvine (UC Irvine) invites applications for **three tenure-track Assistant Professor positions**. One position targets Artificial Intelligence and Machine Learning, the second position is in Computer Systems, and the third is CS-wide. Exceptionally qualified more advanced candidates may also be considered for a tenured position.

The department has 44 faculty members and 285 graduate students. Faculty research spans a broad spectrum of areas in CS. Prospective applicants are invited to visit our webpages at <http://www.cs.uci.edu>. Applicants must have an earned Ph.D. or equivalent degree. Screening will begin immediately upon receipt of a completed application. Applications will be accepted until the position is filled, although maximum consideration will be given to applications received by February 1, 2015. Each application must contain: a cover letter, CV, up to 3 key publications, a statement of research and teaching interests, and 3-5 letters of recommendation. A separate statement that addresses past and/or potential contributions to diversity, equity and inclusion should also be included in the application materials. **All materials must be uploaded at recruit.ap.uci.edu/apply using one of the links below.**

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING applicants:

Tenure-Track Assistant Professors may apply at: <https://recruit.ap.uci.edu/apply/JPF02666>
Tenured Associate Professor may apply at: <https://recruit.ap.uci.edu/apply/JPF02668>

COMPUTER SYSTEMS applicants:

Tenure-Track Assistant Professors may apply at: <https://recruit.ap.uci.edu/apply/JPF02669>

COMPUTER SCIENCE applicants:

Tenure-Track Assistant Professor may apply at: <https://recruit.ap.uci.edu/apply/JPF02670>
Tenured Associate Professor may apply at: <https://recruit.ap.uci.edu/apply/JPF02671>

The University of California, Irvine is an Equal Opportunity/Affirmative Action Employer advancing inclusive excellence. All qualified ap-

plicants will receive consideration for employment without regard to race, color, religion, sex, national origin, disability, age, protected veteran status, or other protected categories covered by the UC nondiscrimination policy. A recipient of an NSF ADVANCE award for gender equity, UCI is responsive to the needs of dual career couples, supports work-life balance through an array of family-friendly policies, and is dedicated to broadening participation in higher education.

University of Central Florida Assistant or Associate Professor

The University of Central Florida (UCF) is strategically investing in interdisciplinary **Data Analytics**.

The Computer Science Division in the College of Engineering and Computer Science at UCF invites candidates for a tenure/tenure-track position at the **Assistant, Associate, or Full Professor level** starting Fall 2015.

Successful candidates will have a record of high-quality publications and be recognized for their expertise and the impact of their research, will join a strong group engaged in the Data Analytics cluster composed of faculty from Computer Science, Statistics, Mathematics and Business, and will have a unique opportunity to shape computer science and interdisciplinary research and education in Big Data.

The position requires a Ph.D. from an accredited institution in an area appropriate to Computer Science and a strong commitment to the academic process, including teaching, scholarly publications, and sponsored research.

Computer Science at UCF has a rapidly-growing educational and research program with nearly \$4.6 million in research contracts and expenditures annually and over 215 graduate students. Computer Science has strong areas of research in Computer Vision, Machine Learning, Virtual and Mixed Reality, and HCI. Research sponsors include NSF, NIH, NASA, DOT, DARPA, ONR, and other agencies of the DOD. Industry sponsors include AMD, Boeing, Canon, Electronic Arts, General Dynamics, Harris, Hitachi, Intel, Lockheed Martin, Oracle, SAIC, Symantec, Toyota USA, and Walt Disney World, as well as local startups. More information about the Computer Science Division can be found at <http://www.cs.ucf.edu/>.

UCF has about 60,000 students and is the nation's second largest university. Located in Orlando, UCF is at the center of the I-4 High Tech Corridor. The corridor has an excellent industrial base that includes: software, defense, space, simulation and training, and a world-renowned entertainment industry. Adjacent to UCF is a thriving research park that hosts more than 100 high-technology companies and the Institute for Simulation and Training. The Central Florida area is designated by the State of Florida as the Center of Excellence in Modeling and Simulation. UCF also has an accredited medical school, which opened in 2009. Great weather, easy access to the seashore, one of the largest convention centers in the nation, and one of the world's best airports are just a few features that make Orlando an ideal location.

Candidates must complete an online job application at <https://www.jobswithucf.com/postings/40054>. The application should include a cover letter, curriculum vitae, statements of

research and teaching philosophy, and at least three professional references complete with address, phone number, and email address.

University of Central Florida Lecturer, Assistant, Associate, or Full Professor

The Computer Science Division of the College of Engineering and Computer Science is looking for a faculty member with specialization in **Digital Forensics and related areas of cybersecurity**. This is a nine-month position which will be non-tenure earning (at the lecturer level) or tenure-track (at the assistant, associate or full professor rank), depending on the selected candidate's qualifications and experience.

Candidates must have a Ph.D. from an accredited institution by the time of the appointment in an area appropriate to Computer Science, Digital Forensics, or Cyber Security. Candidates for a tenure-track position must have a record of high-quality research and publications. Candidates must have a strong commitment to the academic process, including teaching, scholarly work, advising, and service.

UCF confers a Master of Science degree in Digital Forensics (MSDF), which has been offered by the Computer Science Division of the EECS Department since 2008. The MSDF degree is a collaborative effort between Computer Science, Forensic Science, Criminal Justice, and Legal Studies. The degree program consists of 30 credit hours and prepares students, including working professionals who pursue the degree on a part-time or full-time basis, to gain the knowledge and skills required to work as an examiner in the field. More details about UCF's MSDF program including the curriculum, faculty, and recent classes can be found at the program's website: <http://msdf.ucf.edu>.

UCF is also home to a Collegiate Cyber Defense Competition (CCDC) team that won the Alamo Cup in the 2014 national competition. There is a large group of undergraduate students who are very enthusiastic about cyber security. The MSDF program works closely with many organizations to support student internships, and has an excellent reputation among Florida law enforcement officials and departments.

UCF has about 60,000 students and is the nation's second largest university. Located in Orlando, UCF is at the center of the I-4 High Tech Corridor. The corridor has an excellent industrial base that includes: software, defense, space, simulation and training, and a world-renowned entertainment industry. Adjacent to UCF is a thriving research park that hosts more than 100 high-technology companies and the Institute for Simulation and Training. Great weather, easy access to the seashore, one of the largest convention centers in the nation, and one of the world's best airports are just a few features that make Orlando an ideal location.

Candidates must complete an online application at <https://www.jobswithucf.com/postings/40053>. In addition, candidates must also submit a signed cover letter, complete curriculum vitae, maximum two page statement outlining research vision and teaching interests, and a list of at least three professional references complete with address, phone number, and email address.

University of Central Florida Assistant, Associate, or Full Professor

UCF Computer Science is seeking a faculty member with specialization in **Information Technology**. This is a tenure-track (or tenured) position (at any level). We offer a competitive salary and start up package with generous benefits.

As minimum qualifications all applicants must have a Ph.D. degree from an accredited institution in a field related to Information Technology, Computer Science, Computer Engineering, or Cyber Security. Candidates must have a strong commitment to the academic process, including scholarly work, advising, service, and teaching. Preference will be given to candidates with a record of high-quality research, publications, funding, and extensive industry experience.

UCF confers a bachelor's degree in Information Technology, which has been offered by the Computer Science Division of the EECS Department since 2000. We have an excellent networking lab where students can gain hands-on experience with networking technology. Details regarding the IT program can be found at <http://www.cs.ucf.edu/academics/academics.php>.

UCF is home to a Collegiate Cyber Defense Competition (CCDC) team that won the Alamo Cup in the 2014 national competition; this team and the Collegiate Cyber Defense Club have a high percentage of IT students. UCF has about 60,000 students and is the nation's second largest university. Located in Orlando, UCF is at the center of the I-4 High Tech Corridor. The corridor has an excellent industrial base that includes: software, defense, space, simulation and training, and a world-renowned entertainment industry. Adjacent to UCF is a thriving research park that hosts more than 100 high-technology companies and the Institute for Simulation and Training. Great weather, easy access to the seashore, one of the largest convention centers in the nation, and one of the world's best airports are just a few features that make Orlando an ideal location. UCF is an equal opportunity/affirmative action employer. All qualified applicants are encouraged to apply, including minorities, women, veterans and individuals with disabilities. As a Florida public university, UCF makes all application materials and selection procedures available to the public upon request.

Candidates must complete an online application at <https://www.jobswithucf.com/postings/40049>. In addition, candidates must also submit a signed cover letter, complete curriculum vitae, maximum two page statement outlining research vision and teaching interests, and a list of at least three professional references complete with address, phone number, and email address.

University of Central Florida Assistant or Associate Professor in Human-Computer Interaction

The Computer Science Division in the College of Engineering and Computer Science at UCF invites candidates for a tenure-track/tenure position at the Assistant or Associate Professor level, starting Fall 2015, in the area of **Human-Computer Interaction (HCI)**.

Successful candidates will have a record of high-quality publications and be recognized for their expertise and the impact of their research, and will have the opportunity to join the newly established Interactive Systems and User Experience Research Cluster of Excellence in the college.

The position requires a Ph.D. from an accredited institution in an area appropriate to Computer Science and a strong commitment to the academic process, including teaching, scholarly publications, and sponsored research. Candidates who conduct research in any area of HCI that are eager to establish multidisciplinary collaborations will be considered. Candidates with a distinguished record of scholarship, teaching, research funding and professional visibility may be considered for appointment as a tenured associate professor.

Computer Science at UCF has a rapidly-growing educational and research program with nearly \$4.6 million in research contracts and expenditures annually and over 215 graduate students. Computer Science has strong areas of research in HCI, Virtual and Mixed Reality, Computer Vision, and Machine Learning. Research sponsors include NSF, NIH, NASA, DOT, DARPA, ONR, and other agencies of the DOD. Industry sponsors include AMD, Boeing, Canon, Electronic Arts, General Dynamics, Harris, Hitachi, Intel, Lockheed Martin, Oracle, SAIC, Symantec, Toyota USA, and Walt Disney World, as well as local startups. More information about the Computer Science Division can be found at <http://www.cs.ucf.edu/>.

UCF has about 60,000 students and is the nation's second largest university. Located in Orlando, UCF is at the center of the I-4 High Tech Corridor. The corridor has an excellent industrial base that includes: software, defense, space, simulation and training, and a world-renowned entertainment industry. Adjacent to UCF is a thriving research park that hosts more than 100 high-technology companies and the Institute for Simulation and Training. The Central Florida area is designated by the State of Florida as the Center of Excellence in Modeling and Simulation. UCF also has an accredited medical school, which opened in 2009. Great weather, easy access to the seashore, one of the largest convention centers in the nation, and one of the world's best airports are just a few features that make Orlando an ideal location.

Candidates must complete an online job application at <https://www.jobswithucf.com/postings/40189>. The application should include a cover letter, curriculum vitae, statements of research and teaching philosophy, and at least three professional references complete with address, phone number, and email address.

University of Miami Assistant/Associate Rank Professor

The Department of Computer Science at the University of Miami invites applications for two faculty positions (Assistant/Associate rank) starting August 2015. Candidates must possess a Ph.D. in Computer Science or in a closely-related discipline. One position requires a strong research expertise in areas related to Human Computer Interaction (HCI): Intelligent Interfaces, Data and Information Visualization, or Integrated HCI

Systems. The second position requires strong research expertise in areas related to Knowledge Representation (KR): Technologies that support practical systems for the knowledge worker, or the development of systems that understand and extract knowledge in large real-world data sets.

The department encourages innovative interdisciplinary work with other units of the university, such as the science and engineering departments in the College of Arts and Sciences and the College of Engineering, the Miller School of Medicine, and the Center for Computational Sciences. The HCI position specifically entails work within the Visualization Program of the Center for Computational Sciences, to form collaborations across the University. The successful candidate will be expected to teach at both undergraduate and graduate levels and to develop and maintain an internationally recognized research program. To be considered at the Associate Professor level the candidate must have a proven record of successful independent teaching and research.

Applicants should submit a cover letter, CV, research plan, statement of teaching philosophy, sample preprints or reprints, and the names of at least three references online at <http://www.cs.miami.edu/search/>.

Review of applications will begin November 15, 2014 and continue until the position is filled. Information about the College can be found at <http://www.as.miami.edu/>.

The University of Miami offers competitive salaries and a comprehensive benefits package including medical and dental benefits, tuition remission, vacation, paid holidays and much more. The University of Miami is an Equal Opportunity Employer - Females/Minorities/Protected Veterans/Individuals with Disabilities are encouraged to apply. Applicants and employees are protected from discrimination based on certain categories protected by Federal law.

The University of Toledo Electrical Engineering and Computer Science Department

EECS Department seeks applications for two full-time tenure-track faculty positions at the level of **Assistant Professor**. Position 1: Research expertise in power electronics; power and energy systems, and renewable energy systems. Teaching expertise in Energy Conversion, Power Electronics, and Power Systems. Position 2: Research expertise in cyber security, human machine interaction modeling, virtual reality, and related areas. Teaching expertise in Computer Programming, Data Structures, and Gaming. The positions require a PhD in electrical or computer engineering or computer science. Expected start date is August 2015.

Application review will begin in January 2015. For full consideration, submit applications by January 31, 2015. Application must include (i) Cover letter, (2) Curriculum vitae, (3) Two page research plan, (4) Two page teaching philosophy including list of EECS courses that the applicant can teach effectively, (5) Contact info of four professional references. Email applications to mansoor.alam2@utoledo.edu as a single PDF file only. **The University of Toledo** is an equal access, equal opportunity, affirmative action employer and educator.

Are you looking for your next IT job?

Do you need Career Advice?

The **ACM Career & Job Center** offers ACM members a host of career-enhancing benefits:

- A **highly targeted focus** on job opportunities in the computing industry
- **Access to hundreds** of industry job postings
- Resume posting **keeping you connected** to the employment market while letting you maintain full control over your confidential information
- **Job Alert system** that notifies you of new opportunities matching your criteria
- **Career coaching** and guidance available from trained experts dedicated to your success
- **Free access** to a content library of the best career articles compiled from hundreds of sources, and much more!



Visit **ACM's Career & Job Center** at:
<http://jobs.acm.org>



Association for
Computing Machinery

Advancing Computing as a Science & Profession

The **ACM Career & Job Center** is the perfect place to begin searching for your next employment opportunity!

Visit today at **<http://jobs.acm.org>**

JANUARY 2015 | VOL. 58 | NO. 1 | COMMUNICATIONS OF THE ACM 127

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/2688485

William Sims Bainbridge

Future Tense Wow!

Human life is filled with illusions, so virtual worlds are not especially unreal.

APPROACHING MIDDLE AGE, Larry Jenkins had begun again to play his favorite massively multiplayer online game from his adolescence—*Wilds of Wonder*—ignorant that the result would turn into the leading psychotherapy craze of the mid-21st century, and that he would become its prime entrepreneur.

He thought he could brood better in an emotionally evocative electronic environment, recalling that one set of quests had previously sent him deep into Gagtooth jungle to find a character named Kurtz who represented psychotic rebellion. Before *Wilds of Wonder*, Kurtz had been the mysterious psychotic in Joseph Conrad's novel *Heart of Darkness* about colonial Africa and again in Francis Ford Coppola's movie *Apocalypse Now* about the Vietnam War. Jenkins feared he was sliding into his own darkness, hoping Kurtz would provide guidance. However, *Wilds of Wonder* had killed off its own Kurtz years before, and all Jenkins could find was a desiccated corpse that refused to answer his questions.

Descending into personal gloom, Jenkins wandered northwest through the game's vast digital landscape, vaguely planning to visit the shining metropolis of his youth, Blizzardsleet City. He first entered Sherwood Forest, recalling his early quests there decades before. Then, suddenly, he saw a familiar face—Logger, one of the quarrelsome hyena-like gnolls, who always looked to him like a humanoid pig—whom he had repeatedly killed years before, though now



An avatar seeking meaning in the wilderness.

it seemed more friendly than fierce. “Welcome home, Jenkins,” said Logger, “I’m so glad you’ve resubscribed. Most players these days are adults not unlike yourself, and I’m sure you’ll have a great time.” Logger had taken responsibility for recommending non-player character questgivers players should visit within *Wilds of Wonder* and gave Jenkins the names of three, each offering a different route to sanity.

Jenkins first visited Sickman Fraud, founder of sicko-analysis, in the Mage District of Blizzardsleet. Fraud’s specialty was psychodrama, so he assigned Jenkins several missions that involved disguises. Following debriefing sessions on Fraud’s

couch, Jenkins felt no better, so Fraud invoked his most powerful spell, the bipolar switch. Jenkins went instantly from depressed to manic, drew his longsword, and skewered the mage analyst before even the first word of a defensive spell had passed his virtual lips. When the spell faded, Jenkins was swept by a new emotion—guilt. He knew Fraud would rez back to simulated life in a minute or two, but murdering this therapist, even briefly, would be rather shameful.

Seeking remission for his sin, Jenkins begged guidance from a priest in the Cathedral of Blizzardsleet who explained that a cure would require spiritual transmutation. Jenkins would [CONTINUED ON P. 127]

Computing Reviews

Connect with our Community of Reviewers

“I like CR because it covers the full spectrum of computing research, beyond the comfort zone of one’s specialty. I always look forward to the next Editor’s Pick to get a new perspective.”

- Alessandro Berni



Association for
Computing Machinery

ThinkLoud

www.computingreviews.com



CHI 2015

CROSSINGS

SEOUL • KOREA

CALL FOR PARTICIPATION

Registration opens late November.

Several submission tracks are still open: Works-in-Progress, Panels, Student Design Competition, Student Game Competition, Student Research Competition, Video Showcase, SIG Meetings and alt.chi.
5 January 2015

See website for full details:
<http://chi2015.acm.org/>

Come to CHI2015, ACM's premier conference on human factors in computing systems! Join us for the first CHI conference in Asia in Seoul, Korea, a world-class center of emerging trends in culture, technology, and design.

The CHI2015 theme is Crossings: crossing boundaries, disciplines, and nations. You will see insights at the intersections of people and technology; scientists and practitioners; and academic and business interests. CHI showcases the latest research and design of the world's most innovative technologies across multiple disciplines including computer science, cognitive psychology, design, social science, human factors, AI, graphics, visualization, multimedia design and more.

Join us in Seoul, Korea

18-23 April 2015