

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

04/2010 VOL.53 NO.04

A Recipe for Efficiency

Some Principles of Power-Aware Computing

Toward Robotic Cars

Functional Logic Programming

CTOs on Malware Defense

Spies Among Us?

Data-Structure Canon

Chair@...
Siobhán Clarke

Papers@...
Kevin Sullivan

Essays@...
Daniel Steinberg

Workshops@...
Jonathan Edwards

Films@...
Bernd Brügge



Onward!

ACM Conference
on New Ideas
in Programming
and Reflections
on Software

October 17–21, 2010
Co-located with Splash
John Ascuaga's Nugget
...@onward-conference.org
SIGPLAN

Reno-Tahoe



Association for
Computing Machinery

Advancing Computing as a Science & Profession

membership application & digital library order form

Priority Code: ACACM10

You can join ACM in several easy ways:

Online http://www.acm.org/join	Phone +1-800-342-6626 (US & Canada) +1-212-626-0500 (Global)	Fax +1-212-944-1318
--	---	-------------------------------

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:

<http://www.acm.org/membership/L2-3/>

Special rates for members of sister societies:

<http://www.acm.org/membership/dues.html>

Please print clearly

Name _____

Address _____

City _____ State/Province _____ Postal code/Zip _____

Country _____ E-mail address _____

Area code & Daytime phone _____ Fax _____ Member number, if applicable _____

Purposes of ACM

ACM is dedicated to:

- 1) advancing the art, science, engineering, and application of information technology
- 2) fostering the open interchange of information to serve both professionals and the public
- 3) promoting the highest professional and ethics standards

I agree with the Purposes of ACM:

Signature _____

ACM Code of Ethics:

<http://www.acm.org/serving/ethics.html>

choose one membership option:

PROFESSIONAL MEMBERSHIP:

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

STUDENT MEMBERSHIP:

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

All new ACM members will receive an ACM membership card.
For more information, please visit us at www.acm.org

Professional membership dues include \$40 toward a subscription to *Communications of the ACM*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:

Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

- Visa/MasterCard American Express Check/money order
- Professional Member Dues (\$99 or \$198) \$ _____
- ACM Digital Library (\$99) \$ _____
- Student Member Dues (\$19, \$42, or \$62) \$ _____
- Total Amount Due** \$ _____

Card # _____

Expiration date _____

Signature _____

Departments

- 5 **ACM Awards Committee**
The Work of ACM's Awards Committee
By Calvin Gotlieb and James Horning
-
- 6 **Letters to the Editor**
Computing Paradigm Not a Branch of Science
-
- 8 **In the Virtual Extension**
-
- 10 **BLOG@CACM**
SQL Databases v. NoSQL Databases
Michael Stonebraker considers several performance arguments in favor of NoSQL databases—and finds them insufficient.
-
- 12 **CACM Online**
Going Mobile
By David Roman
-
- 31 **Calendar**
-
- 107 **Careers**

Last Byte

- 112 **Q&A**
Systematic Thinking
Andrew S. Tanenbaum talks about MINIX, microkernels, and electronic voting systems.
By Leah Hoffmann

News



- 13 **Data Streaming 2.0**
In today's real-time Web, data streaming applications no longer have the luxury of making multiple passes over a recorded data set.
By Alex Wright
-
- 15 **Robots Gear Up for Disaster Response**
After 15 years of research, robots for search and rescue may be nearing prime time.
By Gary Anthes
-
- 17 **Spies Among Us?**
Governments' practice of electronic surveillance—and the growing use of warrantless wiretapping—has observers deeply concerned.
By Samuel Greengard

Viewpoints

- 22 **Emerging Markets**
Development 2.0: The IT-Enabled Transformation of International Development
The fundamental assumptions of international development are changing, increasingly putting the tools for a digital economy into the hands of the world's poor.
By Richard Heeks
-
- 25 **Historical Reflections**
Be Careful What You Wish For
Reflections on the decline of mathematical tables.
By Martin Campbell-Kelly
-
- 27 **Technology Strategy and Management**
Cloud Computing and SaaS as New Computing Platforms
To become an industry platform, vendors must open their infrastructure technology to other product companies.
By Michael Cusumano
-
- 30 **Viewpoint**
When Network Neutrality Met Privacy
Incorporating the consideration of privacy into the ongoing debate concerning network neutrality.
By Paul Ohm
-
- 33 **Kode Vicious**
The Data-Structure Canon
Data structures are part of the foundation of computer science. It pays to revisit them from time to time.
By George V. Neville-Neil

Practice

- 36 **Cooling the Data Center**
What can be done to make cooling systems in data centers more energy efficient?
By Andy Woods
-
- 43 **CTO Roundtable: Malware Defense**
The battle is bigger than most of us realize.
By Mache Creeger
-
- 50 **A View of Cloud Computing**
Clearing the clouds away from the true potential and obstacles posed by this computing capability.
By Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia

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

Review Articles

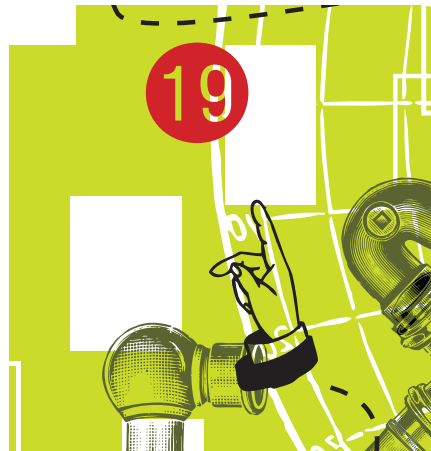
- 74 **Functional Logic Programming**
Combining the paradigm features of both logic and functional programming makes for some powerful implementations.
By Sergio Antoy and Michael Hanus



About the Cover: Power management, particularly in ways to improve energy efficiency, is fast becoming a top priority throughout the computing industry, as detailed by Parthasarathy Ranganathan beginning on page 60. Jean-Francois Podevin, a California-based illustrator who interlaces conceptual and literal illustrations using computer 3D-generation

techniques, captures a galaxy of power icons. For more on his work, see <http://www.podevin.com/>

Contributed Articles



- 60 **Recipe for Efficiency: Principles of Power-Aware Computing**
Prior work on power management reflects recurring themes that can be leveraged to make future systems more energy efficient.
By Parthasarathy Ranganathan
-
- 68 **Private Information Retrieval**
Cryptographic protocols safeguard the privacy of user queries to public databases.
By Sergey Yekhanin

Research Highlights

- 88 **Technical Perspective**
Creativity Helps Influence Prediction Precision
By Padhraic Smyth and Charles Elkan
-
- 89 **Collaborative Filtering with Temporal Dynamics**
By Yehuda Koren
-
- 98 **Technical Perspective**
New Bar Set for Intelligent Vehicles
By Leslie Pack Kaelbling
-
- 99 **Toward Robotic Cars**
By Sebastian Thrun

Virtual Extension

As with all magazines, page limitations often prevent the publication of articles that might otherwise be included in the print edition.

To ensure timely publication, ACM created *Communications'* Virtual Extension (VE).

VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on their merit. These articles are now available to ACM members in the Digital Library.

Choc'late: A Framework for Specification-based Testing

Pak-Lok Poon, Sau-Fun Tang, T.H. Tse, and T.Y. Chen

Designing for Collective Intelligence

Dawn Gregg

Data Mining and Revenue Management Methodologies in College Admissions

Surje Rebbapragada, Amit Basu, and John Semple

WWW Recycling for a Better World

Stefano Ferretti, Marco Furini, Claudio E. Palazzi, Marco Rocchetti, and Paola Salomoni

Individual Resistance to IT Innovations

Rhoda C. Joseph

A Tale of Two Internet Service Providers

Robert J. Aalberts, Percy S. Poon, and Paul D. Thistle

Capstone Programming Courses Considered Harmful

M. Keith Wright



Association for Computing Machinery
Advancing Computing as a Science & Profession



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
Russell Harris

Director, Office of Membership
Lillian Israel

Director, Office of SIG Services
Donna Cappel

Director, Office of Publications
Bernard Rous

Director, Office of Group Publishing
Scott Delman

ACM COUNCIL

President

Wendy Hall

Vice-President

Alain Chesnais

Secretary/Treasurer

Barbara Ryder

Past President

Stuart I. Feldman

Chair, SGB Board

Alexander Wolf

Co-Chairs, Publications Board

Ronald Boisvert, Holly Rushmeier

Members-at-Large

Carlo Ghezzi;

Anthony Joseph;

Mathai Joseph;

Kelly Lyons;

Bruce Maggs;

Mary Lou Soffa;

Fei-Yue Wang

SGB Council Representatives

Joseph A. Konstan;

Robert A. Walker;

Jack Davidson

PUBLICATIONS BOARD

Co-Chairs

Ronald F. Boisvert and Holly Rushmeier

Board Members

Jack Davidson; Nikil Dutt; Carol Hutchins;

Ee-Peng Lim; Catherine McGeoch;

M. Tamer Ozsu; Vincent Shen;

Mary Lou Soffa; Ricardo Baeza-Yates

ACM U.S. Public Policy Office

Cameron Wilson, Director

1100 Seventeenth St., NW, Suite 50

Washington, DC 20036 USA

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

Computer Science Teachers Association

Chris Stephenson

Executive Director

2 Penn Plaza, Suite 701

New York, NY 10121-0701 USA

T (800) 401-1799; F (541) 687-1840

Association for Computing Machinery (ACM)

2 Penn Plaza, Suite 701

New York, NY 10121-0701 USA

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

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
publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Jack Rosenberger

Web Editor

David Roman

Editorial Assistant

Zarina Strakhan

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Alicia Kubista

Assistant Art Director

Mia Angelica Balaquiot

Production Manager

Lynn D'Addesio

Director of Media Sales

Jennifer Ruzicka

Marketing & Communications Manager

Brian Hebert

Public Relations Coordinator

Virginia Gold

Publications Assistant

Emily Eng

Columnists

Alok Aggarwal; Phillip G. Armour;

Martin Campbell-Kelly;

Michael Cusumano; Peter J. Denning;

Shane Greenstein; Mark Guzdial;

Peter Harsha; Leah Hoffmann;

Mari Sako; Pamela Samuelson;

Gene Spafford; Cameron Wilson

CONTACT POINTS

Copyright permission

permissions@cacm.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmcoa@cacm.acm.org

Letters to the Editor

letters@cacm.acm.org

WEB SITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/guidelines

ADVERTISING

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY

10121-0701

T (212) 869-7440

F (212) 869-0481

Director of Media Sales

Jennifer Ruzicka

jen.ruzicka@hq.acm.org

Media Kit acmm mediasales@acm.org

EDITORIAL BOARD

EDITOR-IN-CHIEF

Moshe Y. Vardi
eic@cacm.acm.org

NEWS

Co-chairs

Marc Najork and Prabhakar Raghavan

Board Members

Brian Bershad; Hsiao-Wuen Hon;

Mei Kobayashi; Rajeev Rastogi;

Jeannette Wing

VIEWPOINTS

Co-chairs

Susanne E. Hambrusch; John Leslie King;

J Strother Moore

Board Members

P. Anandan; William Aspray;

Stefan Bechtold; Judith Bishop;

Stuart I. Feldman; Peter Freeman;

Seymour Goodman; Shane Greenstein;

Mark Guzdial; Richard Heeks;

Rachelle Hollander; Richard Ladner;

Susan Landau; Carlos Jose Pereira de Lucena;

Beng Chin Ooi; Loren Terveen

Q PRACTICE

Chair

Stephen Bourne

Board Members

Eric Allman; Charles Beeler; David J. Brown;

Bryan Cantrill; Terry Coatta; Mark Compton;

Stuart Feldman; Benjamin Fried;

Pat Hanrahan; Marshall Kirk McKusick;

George Neville-Neil; Theo Schlossnagle;

Jim Waldo

The Practice section of the CACM

Editorial Board also serves as

the Editorial Board of **ACM QUEUE**.

CONTRIBUTED ARTICLES

Co-chairs

Al Aho and Georg Gottlob

Board Members

Yannis Bakos; Gilles Brassard; Alan Bundy;

Peter Buneman; Ghezzi Carlo;

Andrew Chien; Anja Feldmann;

Blake Ives; James Larus; Igor Markov;

Gail C. Murphy; Shree Nayar; Lionel M. Ni;

Sriram Rajamani; Jennifer Rexford;

Marie-Christine Rousset; Avi Rubin;

Abigail Sellen; Ron Shamir; Marc Snir;

Larry Snyder; Veda Storey;

Manuela Veloso; Michael Vitale;

Wolfgang Wahlster; Andy Chi-Chih Yao;

Willy Zwaenepoel

RESEARCH HIGHLIGHTS

Co-chairs

David A. Patterson and Stuart J. Russell

Board Members

Martin Abadi; Stuart K. Card; Deborah Estrin;

Shafi Goldwasser; Monika Henzinger;

Maurice Herlihy; Norm Jouppi;

Andrew B. Kahng; Gregory Morrisett;

Michael Reiter; Mendel Rosenblum;

Ronitt Rubinfeld; David Salesin;

Lawrence K. Saul; Guy Steele, Jr.;

Gerhard Weikum; Alexander L. Wolf;

Margaret H. Wright

WEB

Co-chairs

Marti Hearst and James Landay

Board Members

Jason I. Hong; Jeff Johnson;

Greg Linden; Wendy E. MacKay



ACM Copyright Notice

Copyright © 2010 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) 626-0654.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhelp@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



Association for Computing Machinery



Printed in the U.S.A.

The Work of ACM's Awards Committee

ACM established its first award—the A.M. Turing Award—in 1966. This honor is given annually to an individual whose contributions to the computing community are deemed

of major and lasting technical importance. Today ACM's prestigious Turing Award is widely considered throughout the industry as the “Nobel Prize of computing,” and carries a prize of \$250,000 funded by Intel and Google.

ACM currently sponsors over 100 awards for technical and professional excellence, ranging from Distinguished Service to Outstanding Educator; from Doctoral Dissertation to Theory and Practice. These awards serve many purposes within the computing community: they recognize and reward outstanding accomplishment; they attract media attention to the profession and its practitioners; they promote support from industry, academia, and society; and they spotlight role models for members and future computer scientists.

A recent addition to ACM's award roster is the ACM-Infosys Foundation Award, recognizing a personal contribution by a young scientist or system developer to a contemporary innovation that, through its depth, fundamental impact, and broad implications, exemplifies the greatest achievements in the discipline. The award carries a prize of \$150,000, endowed by the Infosys Foundation. Indeed, many other ACM Awards carry cash prizes, ranging up to \$35,000 each for the Grace Murray Hopper Award funded by Google, and the Software System Award funded by IBM. For more information about ACM's awards, visit <http://awards.acm.org.html/awards.cfm> for a complete list, including details regarding qualifications, associated prizes, previous winners, nomination procedures and deadlines, and the chairs and members of the selection committees.

Award recipients are honored each

year at ACM's Annual Awards Banquet,^a long recognized as the premier event for ACM members and leadership to celebrate achievements by the worldwide IT community.

The Award Process

The credibility of ACM awards depends on several conditions. Nominee selections must be clearly based on merit. Moreover, the overall composition of the selection committees must reflect the diversity of ACM's membership. There must be a careful balance between transparency and privacy. For example, while the names of the committee members are made public, the names of nominees, nominators, and endorsers are kept strictly confidential.

The ACM Awards Committee, which reports directly to ACM Council, is responsible for overseeing the process. The committee consists of six ex officio members (committee co-chairs, ACM's President, CEO, and COO, and a SIG Board representative) plus the chair of each selection committee. The co-chairs are appointed by the ACM President.

The recipient of each award—including the three advanced membership grades (Fellow, Distinguished Member, and Senior Member)—is selected by a separate committee. In addition, Special Interest Group (SIG) award committees are selected by the SIGs themselves, under procedures approved by the Awards Committee.

Most selection committees have three to five members who serve for three to five years, some of whom chair the committee toward the end of their term. Committees with especially heavy workloads (for example, Fellows

and Doctoral Dissertations) have more members. Committee members are selected by the ACM President based on nominations from the co-chairs as well as suggestions from committee chairs, SIG officers, and others.

Each selection committee sets its own rules of operation, including policies for carrying nominations over from year to year. Committees operate by email and conference calls. Moreover, committees representing the Turing, ACM-Infosys Foundation, and Fellows awards, hold annual day-long, in-person meetings.


A Work in Progress

ACM's awards, policies, and practices have been growing and evolving for over 40 years.

The Awards Committee meets annually to discuss new awards and operational issues; this event is often held in conjunction with the Awards Banquet. (For more information, view the committee's annual report <http://www.acm.org/about/annual-reports-current-fy/FY09%20Awards%20Annual%20Report.pdf?searchterm=Annual+Report+2009>.)

Staff at ACM HQ keeps the process operating smoothly, handling the necessary email correspondence; conference calls; scheduling; reports; Web site updates; and budgeting required for supporting the entire process.

While one should naturally aspire to recognition by an ACM award or elevation to an advanced member grade, there is another important role for consideration: You can nominate or endorse someone whose work you feel is worthy of acknowledgment. The selection process can only be as good as the nominations received. There is considerable variation in the number of nominations submitted for the different awards, but every selection committee would rather have too many good nominations than too few.

We welcome inquiries and suggestions for improving the awards process. Please contact us via Rosemary McGuinness, the ACM Awards Committee Liaison, mcguinness@hq.acm.org. 

Calvin Gottlieb and James Horning are co-chairs of ACM's Awards Committee.

© 2010 ACM 0001-0782/10/0400 \$10.00

^a To be held this year on Saturday, June 26, in San Francisco, CA.

Computing Paradigm Not a Branch of Science

BEGINNING WITH THE headline, “Computing’s Paradigm,” The Profession of IT Viewpoint by Peter J. Denning and Peter A. Freeman (Dec. 2009) reflected some confusion with respect to Thomas Kuhn’s notion of “paradigm” (a set of social and institutional norms that regulate “normal science” over a period of time). Paradigms, said Kuhn, are incommensurable but determined by the social discourse of the environment in which science develops.

The crux of the matter seems to be that computing can’t be viewed as a branch of science since it doesn’t deal with nature but with an artifact, namely the computer. For guidance, we reflect on at least one scientific antecedent—thermodynamics, which originated from the need to understand the steam engine but is distinguished from steam engineering by its search for general principles, detached from a specific machine. The Carnot cycle and entropy theorem are scientific results, not feats of engineering.

The metatheoretical problem of computing seems mainly semiotic. Suppose, 200 years ago, somebody had created a discipline called, say, Thermozap, that included the study of the Carnot cycle and the building of new steam engines. Somebody might have come up with the insoluble problem of whether the new discipline was science or engineering. It was neither but rather a hodgepodge of things better left separated.

Computing is in a similar situation. There is an area (call it Knuth-Dijkstra computing) that studies scientific problems posed by the existence of computing devices. Thermodynamics was part of physics because steam engines use physical forces. Computing devices are formal machines, so Knuth-Dijkstra computing is a mathematical discipline. Then there is the computing discipline that builds systems (call it Denning-Freeman computing), which is definitely part of engineering. The error is in thinking they are the same. Both refer to the same device,

generically called “computer,” but is a misleading connection, since the two disciplines describe the computer in different ways—a formal model of computation in Knuth-Dijkstra computing, an actual machine in Denning-Freeman computing.

Denning and Freeman proposed a “framework” that takes the side of engineering computing (why I call it Denning-Freeman computing), describing development of an engineering system and leaving no doubt as to the envisioned nature of the discipline. All the purportedly different fields they proposed—from robotics to information processing in DNA—are actually different applications of the same paradigm. To consider them different would be like saying quantum physics is different for nuclear plants and for semiconductors. The physics is the same; what changes is the engineering process of its application, as in computing.

The abstract problem of symbol manipulation is mathematical and the subject of computing science. The instantiation of the symbol-manipulation model in useful systems is a problem for the engineering of computing, a discipline that is theoretically, methodologically, and conceptually separated from the mathematical study of symbol manipulation.

Simone Santini, Madrid, Spain

I wish to suggest ways to improve Peter J. Denning’s and Peter A. Freeman’s proposed computing paradigm in their Viewpoint “Computing’s Paradigm” (Dec. 2009). While I accept the tentative five phases—initiation, conceptualization, realization, evaluation, and action—in the proposed paradigm, they are, in practice, incomplete.

While I agree with initiation (the existential argument followed by conceptualization) as the design argument, three additional phases are missing: The first is a phase 0 I call understanding (or problem understanding). Before one can pose the existential (Denning’s and Freeman’s initiation), a phase must address (problem) understanding, a

key element in all complex computing domains. Moreover, understanding is associated with modeling, a key aspect of understanding. One cannot determine whether a system can be built or represented without the understanding needed to pose hypotheses, theses, or formal requirements. Understanding is often not addressed very well by beginning computing researchers and developers, especially as it pertains to information processes.

The second missing element of conceptualization is an explicit statement about bounded rationality, per Herbert Simon (http://en.wikipedia.org/wiki/Bounded_rationality), a concept based on the fact that the rationality of individuals is limited by the information they possess, the cognitive limitations of their minds, and the finite amount of time they have to make decisions. Bounded rationality addresses the tentative nature of design and discovery as an evolving set of decisions posed against multiple criteria derived from understanding and initiation. The results from conceptualization, or design, must always be understood as both tentative and knowledge-limited.

Finally, a phase missing from evaluation and action is “technology readiness” (http://en.wikipedia.org/wiki/Technology_readiness_level), especially in deploying real systems. A new technology, when first invented or conceptualized is not suitable for immediate application. It is instead usually subject to experimentation, refinement, and increasingly realistic contextual testing. When proven, it can be incorporated into a deployed system or subsystem. All information processes are realized and embedded within the context of existing deployed systems. Therefore, technology readiness of a posed information process must stand as a separate phase between evaluation and action.

1. Simon, H. Bounded Rationality and Organizational Learning. *Organization Science* 2, 1 (1991), 125–134.

2. Simon, H. A mechanism for social selection and successful altruism. *Science* 250, 4988 (1990), 1665–1668.

3. Simon, H. A behavioral model of rational choice. In *Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in a Social Setting*. John Wiley & Sons, Inc., New York, 1957.

David C. Rine, Fairfax, VA

Authors' Response:

Our argument concerned computing's "belief system." Kuhn discussed belief systems in science. Whether or not we were true to Kuhn is irrelevant to our argument.

Santini says computing is about computers. We disagree. Computing is about information processes, and computers are machines that implement information processes. There are natural, as well as artificial, information processes. Computing is as much about computers as astronomy is about telescopes.

Computing does not separate neatly into math and engineering, as Santini claims. Computing increasingly employs experimental (scientific) methods to test hypotheses about complex information processes.

Santini's desire to parse computing into separate elements will fail, just as all such previous attempts have failed. Our collective concern with information processes keeps pulling all the elements together, no matter how hard we try to separate them.

Peter Denning, Monterey, CA

Peter Freeman, Atlanta, GA

Hold the Accusations That Limit Scientific Innovation

I applaud the debate on MapReduce between "MapReduce and Parallel DBMSs: Friends or Foes?" by Michael Stonebraker et al. and "MapReduce: A Flexible Data Processing Tool" by Jeffrey Dean and Sanjay Ghemawat (Jan. 2010). But I strongly object to the former's criticism of the MapReduce designers, saying "Engineers should stand on the shoulders of those who went before, rather than on their toes." Creating an alternate method is not stepping on anyone's toes. Such accusations, besides being unjust, impede science.

Jonathan Grier, Lakewood, NJ

Authors' Response:

As we noted in the article, the Map

phase of a MapReduce computation is essentially a filter and a group-by operation in SQL, while the Reduce phase is largely a target-list computation in SQL. When user-defined functions are included in SQL (as they are in many commercial implementations), the functionality provided by parallel SQL DBMSs and MapReduce implementations appears to be the same.

The parallel DBMS literature, dating from the 1980s, includes hundreds of articles on implementation tactics. Our comment about "standing on the shoulders..." was meant to suggest that any new implementation effort should carefully review the prior literature to learn what past results are available, then add to the store of total knowledge.

The MapReduce team seemed not to have done this exercise. Hence the comment.

Michael Stonebraker, Daniel Abadi, David J. DeWitt, Sam Madden, Erik Paulson, Andrew Pavlo, Alexander Rasin, Cambridge, MA

Even in the Classroom, a Click Is Just a Click

The news item "Web Used for Final Exams in Denmark" (Jan. 2010) gave the impression that such an approach was never tried before. I have taught computer- and network-security-related classes for the past eight years, incorporating the Internet as a tool students use during class, including on quizzes and exams. I am sure I am not the only instructor in the U.S. allowing students to use the Internet for research and comprehension in the classroom.

Is Europe just now discovering the value of Internet searches in education? There is no reason to require that students memorize details accessible at the click of a mouse, when they might better spend their time analyzing and comprehending. The old way of requiring that students memorize facts from textbooks should give way to methods of learning more in tune with the Y generation.

Moreover, exams should be tailored so students don't just regurgitate facts, but make facts accessible over the Internet, then require students show they have comprehended them to solve problems. This new

paradigm in testing emphasizes comprehension over memorization.

Bela Erdelyi, Lincroft, NJ

How to Honor the Heroes of CS

Communications cover article "Amir Pnueli Ahead of His Time" (Jan. 2010) mourned the passing of Amir Pnueli in November 2009. Likewise, *Communications* mourned (Nov. 2008), along with the rest of the computer science community, the disappearance and passing of Jim Gray. Tragic as these events are, they are sure to be followed by others, as computer science is no longer in its infancy but well past middle age. I see the risk that *Communications* covers (and articles) could turn into a gallery of the revered heroes of our science who will be passing away in ever greater numbers. *Communications* could instead honor its icons by, perhaps, adding an obituary column, even as a permanent feature.

Panos Louridas, Athens, Greece

Editor's Response:

Communications does indeed publish obituaries to note the passing of prominent computer scientists. In certain cases, however, the Editorial Board deems the event to be deserving of further recognition. Jim Gray was in full vigor when he disappeared without a trace in January 2007, as was Amir Pnueli when he passed away in November 2009. In both cases there was a sense of unusual or unexpected tragedy, which explains the degree of coverage in Communications.

Corrections

In the article "Amir Pnueli: Ahead of His Time" (Jan. 2010), it was reported that Pnueli was born in Nahalal, Israel, in 1941. The State of Israel was yet to be declared in 1941. Nahalal was then in the British Mandate of Palestine.

The article also noted that Pnueli worked with David Harel on Statecharts. The Statecharts formalism was developed by Harel. Pnueli was involved in the development of Statemate, a software system implementing the Statecharts formalism. □

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

© 2010 ACM 0001-0782/10/0400 \$10.00

In the Virtual Extension

Communications' *Virtual Extension* brings more quality articles to ACM members. These articles are now available in the ACM Digital Library.

Choc'late: A Framework for Specification-based Testing

Pak-Lok Poon, Sau-Fun Tang, T.H. Tse, and T.Y. Chen

Software testing based on informal specifications has remained popular. The authors present a CHOiCe reLATion framEwork (CHOC'LATE) that allows testers to systematically reenact an unstructured informal specification in a more formal representation—choice relation table—from which a test suite can be generated automatically. Unlike formal specifications, the choice relation table is easy to understand by software developers with little training. Furthermore, CHOC'LATE incorporates mechanisms for consistency checking, automatic deductions, and prioritization of choices for test suite generation. Because of these merits, the authors contend that CHOC'LATE will have a significant contribution to software quality assurance in the industry.

Coming Next Month in COMMUNICATIONS

*Beyond Total Capture:
A Critique of Lifelogging*

Energy-Efficient Algorithms

*Enhanced Debugging
with Traces*

*Principles of Robust
Timing over the Internet*

*Cloud Computing
and Electricity*

*Is Mobile Email
Addiction Overlooked?*

And the latest news on relational databases, cloud computing and developing nations, and how CS aids astronomy.

Designing for Collective Intelligence

Dawn Gregg

Collective intelligence is a fundamentally different way of viewing how applications can support human interaction and decision making. Historically, applications have focused on improving the productivity of individuals, providing tools and data to fulfill specific job functions. Under the collective intelligence paradigm, the focus is on harnessing the intelligence of groups of people to enable greater productivity and better decisions than are possible by individuals working in isolation. The processes involved in designing and implementing specialized collective intelligence applications are discussed in the context of DDtrac, a Web-based application that allows for the collection and summary of special education data.

Data Mining and Revenue Management Methodologies in College Admissions

Surye Rebbapragada, Amit Basu, and John Semple

While colleges *want* to admit the best students, competition with other colleges and the limited admission time window create a complex decision problem. This article presents a novel approach to the college admission process by partitioning the process into two phases in which data mining and revenue management—two powerful technologies—can be applied to find win-win solutions for both applicants and colleges. Using the process outlined by the authors, student applications can be processed dynamically, and the college can select the best possible candidate at each point throughout the admission season.

WWW Recycling for a Better World

Stefano Ferretti, Marco Furini, Claudio E. Palazzi, Marco Rocchetti, and Paola Salomoni

Is the World Wide Web becoming World Wide Waste? Indeed, by comparing the immense benefits that Web 2.0 could bring to society, with its factual employment, one could provocatively change the meaning of the acronym WWW. The authors propose to redesign the utilization paradigm of Web 2.0 and, in general, of the Internet in order to recycle unused parts of Web 2.0 into altruistic bricks that can be appropriately

rerouted and composed for alternative (unselfish) employment.

Individual Resistance to IT Innovations

Rhoda C. Joseph

Adoption of an information technology (IT) innovation is a much more attractive and frequently examined area to study than non-adoption. This is mainly due to the pro innovation bias that is found in the existing IT and information systems literature. Companies spend millions of dollars advertising their new gadgets to consumers. However, many IT innovations face varying degrees of resistance in their lifetimes. As researchers and IT professionals it is important to understand the subtle nuances of technology resistance and actively engage in strategies to better understand the individual needs of users.

A Tale of Two Internet Service Providers

Robert J. Aalberts, Percy S. Poon, and Paul D. Thistle

Taking a risk management perspective, this article introduces a potentially far-reaching case from New Jersey: *Doe v. XYZ Co.* The Doe case establishes for the first time a legal duty for employers to protect third parties (non-employees) from illegal behavior by any employee conducted on a workplace computer. The article is instructive because it contrasts the flawed management style found in Doe with the facts in *Delfino v. Agilent Technologies, Inc.* where management effectively handled an employee's unlawful activity thus preventing costly legal problems.

Capstone Programming Courses Considered Harmful

M. Keith Wright

Enrollment in U.S. computer-related degree programs has plummeted in the past decade as has the demand for U.S. programmers. This article examines the steady decline in programmer demand from the viewpoint of the author's experiences as a professional programmer for many Fortune 500 companies. Ideas are presented on how to redesign computer curriculums to address modern economic needs, including replacing capstone (senior) programming courses with internships that emphasize ethical IT service-level management and standard platform training.

ACM's Online Books & Courses Programs!

Helping Members Meet Today's Career Challenges

NEW! Over 2,500 Online Courses in Multiple Languages Plus 1,000 Virtual Labs from Element K!



ACM's new Online Course Collection includes over **2,500 online courses in multiple languages, 1,000 virtual labs, e-reference tools, and offline capability.** Program highlights:

The ACM E-Learning Catalog - round-the-clock access to 2,500+ online courses on a wide range of computing and business topics, in multiple languages.

Exclusive vLab® Virtual Labs - 1,000 unique vLab® exercises place users on systems using real hardware and software allowing them to gain important job-related experience.

Reference Tools - an e-Reference Library extends technical knowledge outside of the classroom, plus online Executive Summaries and quick reference cards to answer on-the-job questions instantly.

Offline Player - members can access assessments and self-study courses offline, anywhere and anytime, without a live Internet connection.

A downloadable Quick Reference Guide and a 15-minute site orientation course for new users are also available to help members get started.

The ACM Online Course Program is open to ACM Professional and Student Members.

600 Online Books from Safari

ACM members are eligible for a **special 40% savings** offer to upgrade to a Premium or Full Library subscription.

For more details visit:

http://pd.acm.org/books/about_sel.cfm

The ACM Online Books Collection includes **full access to 600 online books** from Safari® Books Online, featuring leading publishers including O'Reilly. Safari puts a complete IT and business e-reference library right on your desktop. Available to ACM Professional Members, Safari will help you zero in on exactly the information you need, right when you need it.

Safari
Books Online



Association for
Computing Machinery

Advancing Computing as a Science & Profession

500 Online Books from Books24x7

All Professional and Student Members also have **full access to 500 online books** from Books24x7®, in ACM's rotating collection of complete unabridged books on the hottest computing topics. This virtual library puts information at your fingertips. Search, bookmark, or read cover-to-cover. Your bookshelf allows for quick retrieval and bookmarks let you easily return to specific places in a book.



pd.acm.org
www.acm.org/join

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 excerpts from selected posts.



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

DOI:10.1145/1721654.1721659

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

SQL Databases v. NoSQL Databases

Michael Stonebraker considers several performance arguments in favor of NoSQL databases—and finds them insufficient.



From Michael Stonebraker's "The NoSQL Discussion Has Nothing to Do With SQL"

<http://cacm.acm.org/blogs/blog-cacm/50678>

Recently, there has been a lot of buzz about NoSQL databases. In fact, there were at least two conferences on the topic in 2009, one on each coast. Seemingly, this buzz comes from people who are proponents of:

- ▶ document-style stores in which a database record consists of a collection of key-value pairs plus a payload. Examples of this class of system include CouchDB and MongoDB, and we call such systems *document stores* for simplicity;

- ▶ key-value stores whose records consist of key-payload pairs. Usually, these are implemented by distributed hash tables, and we call these *key-value stores* for simplicity. Examples include MemcacheDB and Dynamo.

In either case, one usually gets a low-level, record-at-a-time database management system (DBMS) interface instead of SQL. Hence, this

group identifies itself as advocating NoSQL.

There are two possible reasons to move to either of these alternate DBMS technologies: performance and flexibility.

The performance argument goes something like the following: I started with MySQL for my data storage needs and over time found performance to be inadequate. My options were:

1. "Shard" my data to partition it across several sites, giving me a serious headache managing distributed data in my application or
2. Abandon MySQL and pay big licensing fees for an enterprise SQL DBMS or move to something other than a SQL DBMS.

The flexibility argument goes something like the following: My data does not conform to a rigid relational schema. Hence, I can't be bound by the structure of a RDBMS and need something more flexible.

This blog posting considers the performance argument; a subsequent posting will address the flexibility argument.

For simplicity, we will focus this discussion on the workloads for which

NoSQL databases are most often considered: update- and lookup-intensive online transaction processing (OLTP) workloads, not query-intensive, data-warehousing workloads. We do not consider document repositories or other specialized workloads for which NoSQL systems may be well suited.

There are two ways to improve OLTP performance; namely, provide automatic sharding over a shared-nothing processing environment and improve per-server OLTP performance.

In the first case, one improves performance by providing scalability as nodes are added to a computing environment; in the second case, one improves the performance of individual nodes.

Every serious SQL DBMS—such as Greenplum, Aster Data, Vertica, ParAccel, and others—written in the last 10 years has provided shared nothing scalability, and any new effort would be remiss if it did not do likewise. Hence, this component of performance should be "table stakes" for any DBMS. In my opinion, nobody should ever run a DBMS that does not provide automatic sharding over computing nodes.

As a result, this posting continues about the other component; namely, single-node OLTP performance. The overhead associated with OLTP databases in traditional SQL systems has little to do with SQL, which is why "NoSQL" is such a misnomer.

Instead, the major overhead in an OLTP SQL DBMS is communicating with the DBMS using ODBC or

JDBC. Essentially *all* applications that are performance-sensitive use a stored-procedure interface to run application logic inside the DBMS and avoid the crippling overhead of back-and-forth communication between the application and the DBMS. The other alternative is to run the DBMS in the same address space as the application, thereby giving up any pretense of access control or security. Such *embeddable* DBMSs are reasonable in some environments, but not for mainstream OLTP, where security is a big deal.

Using either stored procedures or embedding, the useful work component is a very small percentage of the total transaction cost for today's OLTP databases, which usually fit in main memory. Instead, a recent paper¹ calculated that total OLTP time was divided almost equally between the following four overhead components:

Logging

Traditional databases write everything twice—once to the database and once to the log. Moreover, the log must be forced to disk, to guarantee transaction durability. Logging is, therefore, an expensive operation.

Locking

Before touching a record, a transaction must set a lock on it in the lock table. This is an overhead-intensive operation.

Latching

Updates to shared data structures, such as B-trees, the lock table, and resource tables, must be done carefully in a multithreaded environment. Typically, this is done with short-term duration latches, which are another considerable source of overhead.

Buffer Management

Data in traditional systems is stored on fixed-size disk pages. A buffer pool manages which set of disk pages is cached in memory at any given time. Moreover, records must be located on pages and the field boundaries identified. Again, these operations are overhead intensive.

If you eliminate any one of the above overhead components, you speed up a DBMS by 25%. Eliminate three and your speedup is limited by a factor of

two. You must get rid of all four to run a DBMS a lot faster.

Although the NoSQL systems have a variety of different features, there are some common themes. First, many NoSQL systems manage data that is distributed across multiple sites, and provide the “table stakes” noted above. Obviously, a well-designed multisite system, whether based on SQL or something else, is way more scalable than a single-site system.

Second, many NoSQL systems are disk-based and retain a buffer pool as well as a multithreaded architecture. This will leave intact two of the four sources of overhead noted above.

Concerning transactions, there is often support for only single record transactions and an eventual consistency replica system, which assumes that transactions are commutative. In effect, the “gold standard” of ACID transactions is sacrificed for performance.

However, the net-net is that the single-node performance of a NoSQL, disk-based, non-ACID, multithreaded system is limited to be a modest factor faster than a well-designed stored-procedure SQL OLTP engine. In essence, ACID transactions are jettisoned for a modest performance boost, and this performance boost has nothing to do with SQL.

However, it is possible to have one's cake and eat it too. To go fast, one needs to have a stored procedure interface to a runtime system, which compiles a high-level language (for example, SQL) into low-level code. Moreover, one has to get rid of all of the above four sources of overhead.

A recent project² clearly indicated that this is doable, and showed blazing performance on TPC-C. Watch for commercial versions of these and similar ideas with open source packaging. Hence, I fully expect very high speed, open-source SQL engines in the near future that provide automatic sharding. Moreover, they will continue to provide ACID transactions along with the increased programmer productivity, lower maintenance, and better data independence afforded by SQL. Hence, high performance does not require jettisoning either SQL or ACID transactions.

In summary, blinding performance depends on removing overhead. Such

overhead has nothing to do with SQL, but instead revolves around traditional implementations of ACID transactions, multithreading, and disk management. To go wildly faster, one must remove all four sources of the overhead discussed above. This is possible in either a SQL context or some other context.

References

1. S. Harizopoulos, et. al., “OLTP Through the Looking Glass, and What We Found There,” *Proc. 2008 SIGMOD Conference*, Vancouver, B.C., June 2008.
2. M. Stonebraker, et. al., “The End of an Architectural Era (It's Time for a Complete Rewrite),” *Proc. 2007 VLDB Conference*, Vienna, Austria, Sept. 2007.

Disclosure: Michael Stonebraker is associated with four startups that are either producers or consumers of database technology. Hence, his opinions should be considered in this light.

Reader's comment

You seem to leave out several other sub-categories of the NoSQL movement in your discussion. For example: Google's BigTable (and clones) as well as graph databases. Considering those in addition, would that change your point of view?

—Johannes Ernst

Blogger's Reply

I am a huge fan of “One size does not fit all.” There are several implementations of SQL engines with very different performance characteristics, along with a plethora of other engines. Besides the ones you mention, there are array stores such as Rasdaman and RDF stores such as Freebase. I applaud efforts to build DBMSs that are oriented toward particular market needs.

The purpose of the blog entry was to discuss the major actors in the NoSQL movement (as I see it) as they relate to bread-and-butter online transaction processing (OLTP). My conclusion is that “NoSQL” really means “No disk” or “No ACID” or “No threading,” i.e., speed in the OLTP market does not come from abandoning SQL. The efforts you describe, as well as the ones in the above paragraphs, are not focused on OLTP. My blog comments were restricted to OLTP, as I thought I made clear.

—Michael Stonebraker

Michael Stonebraker is an adjunct professor at the Massachusetts Institute of Technology.



DOI:10.1145/1721654.1721660

David Roman

Going Mobile

Mobile devices are quickly becoming ubiquitous around the world and the resulting usage of the “Mobile Internet” is rapidly increasing, as is the need for Web sites that serve up content on the small screen efficiently and effectively. Organizations that ignore this trend will miss opportunities to capture and keep users engaged. Gartner Inc. predicts that mobile phones will overtake PCs as the most common Web access device worldwide by 2013 (<http://www.gartner.com/it/page.jsp?id=1278413>). *Communications'* site will be making changes to better accommodate mobile users of portable platforms like smartphones, tablet PCs, and e-readers.



Web content designed for full-sized displays can sit poorly on mobile screens, and mobile networks and hardware may “present challenges for taking advantage of the conveniences of mobile devices for information access,” according to Dongsong Zhang, writing in “Web Content Adaption for Mobile Handheld Devices” (<http://cacm.acm.org/magazines/2007/2/5729>) in the February 2007 issue of *Communications* (<http://cacm.acm.org/magazines/2007/2>). Zhang discusses content adaptation, which the World Wide Web Consortium specifies in a set of best practices for delivering Web content on mobile devices (<http://www.w3.org/TR/mobile-bp/>). Indeed, *Communications'* site complies with W3C guidelines where applicable.

Communications' site performs ably on today's leading mobile systems. Content presentation is uncluttered and top articles are accessed simply on a BlackBerry, one user says, and Apple's multitouch interface makes it easy to select and enlarged articles with a few pokes and gestures on an iPhone. However, support for these and other mobile devices will be improved.

Another planned improvement for the site, following the lead of the larger Web, involves the greater use of multimedia content, especially video. Cisco Systems estimates that Internet video will account for over 60% of all consumer Internet traffic in 2013, and that all forms of video will account for 91% (http://newsroom.cisco.com/dlls/2009/prod_060909.html). Cisco hints at needed improvements to get mobile services, networks, and systems ready for video traffic. For its part, *Communications'* site will be ready for mobile readers.

ACM Member News

SHANGHAI JIAOTONG UNIVERSITY WINS 'THE BATTLE OF THE BRAINS'

A team of three students from Shanghai Jiaotong University needed a lot more than programming skills to win the 2010 ACM-ICPC World Finals, according to Yong Yu, coach of the winning team and a professor in the university's department of computer science and engineering.

“If you want to win, you must have tacit understanding and trust between team members, broad knowledge and its flexible use, mastery of programming skills, a stable psychological quality, and on-the-spot adaptability,” Yu said in an email interview. “Of course, you also need a little luck.”

Known as “The Battle of the Brains,” the 2010 ACM-ICPC World Finals, which was held in February at Harbin Engineering University in Harbin, China, featured 103 university teams competing in the final round. The teams used open standard technology to solve 11 problems of varying degrees of difficulty in the shortest time possible.

The competition, which is sponsored by IBM, was dominated by teams from China and Russia. The Shanghai Jiaotong University team successfully solved seven problems in 778 minutes, followed by Moscow State University, which solved seven problems in 940 minutes. The other teams in the top 10 are, in order, National Taiwan University (Taiwan), Taras Shevchenko Kiev National University (Ukraine), Petrozavodsk State University (Russia), Tsinghua University (China), Saratove State University (Russia), University of Warsaw (Poland), St. Petersburg State University (Russia), and Sun Yat-sen University (China).

With this year's victory, Shanghai Jiaotong University has captured first place for the third time in the last decade. Yu praised his team's ability to challenge their individual limits in terms of “will power, intelligence, and physical and psychological endurance.”

—Bob Violino

Data Streaming 2.0

In today's real-time Web, data streaming applications no longer have the luxury of making multiple passes over a recorded data set.

AS THE RISE of the real-time Web continues to fuel dizzying growth in the volume of packets moving around the global network, computer scientists are taking a fresh look at an old research problem: data streaming.

Traditionally, data streaming applications have worked by capturing data as it travels by and then storing it for later analysis, allowing developers to create algorithms that make multiple passes through a data set, using tried and tested methods like clustering, classification, and association mining.

With today's Internet, however, it is all but impossible to capture and store every passing bit. "Moore's law is not keeping up with our desire to record anything that can be recorded," notes Sudipto Guha, an associate professor of computer science at the University of Pennsylvania. Modern Web-focused applications like network monitoring, click log analysis, and fraud detection have created massive streams of live data where analysis must be carried out immediately.

In this real-time Web, data streaming applications no longer have the luxury of making multiple passes over a recorded data set. Instead, research-



ers have developed new, more efficient single-pass algorithms that are starting to approach the sophistication of their multiple-pass predecessors. As the research landscape evolves, some developers are also starting to explore whether these algorithms have applications beyond the traditional realm of data streaming.

Research into data streaming algorithms received an important boost at the turn of the millennium, when the sudden increase of denial-of-service attacks in the wake of the Year 2000 problem switchover prompted network software developers to investi-

gate new approaches to ensure the integrity of data on the network. At about the same time, the research community became interested in the problem as well, inspiring pioneering efforts like AT&T's Gigascope Internet monitoring project.

"In high-speed networks, you have only nanoseconds to handle a packet before the next one arrives," says Graham Cormode, a researcher at AT&T Labs. When a single Internet service provider (ISP) router may handle gigabytes of headers every hour, it's simply not practical to store data—even metadata—for later analysis. For example, if an ISP wants to determine the 90th percentile of Internet Protocol (IP) packet sizes over an IP packet stream, then traditional multiple-pass algorithms will inevitably fall short.

The huge volume of incoming data, coupled with a growing emphasis on real-time computation, has given researchers the impetus to explore new computation models that involve taking a single pass over the data. In this model, algorithms must make smart use of system resources to monitor a passing data stream.

Limited Memory Allocation

Perhaps the greatest challenge for single-pass algorithms involves making smart use of memory allocation. "There's always a limited amount of memory," says Nick Koudas, a faculty member in the computer science department at the University of Toronto, "and you can only look once." Further complications arise from the inevi-

table memory limitations involved in storing intermediate results, summaries, samples, synopses, or sketches that can be used to provide the final query answers.

To address these constraints, recent data-streaming research has focused on developing techniques based on sampling, sketches, and other methods to provide approximate answers. Many of these approaches can now give bounds on the approximation error, at times deterministic, and many times probabilistic. These bounds are typically a function of the memory space available to the algorithms. Thus, error analysis can offer developers an elegant way of making tradeoffs between accuracy and conserving additional memory.

In recent years, researchers have developed a number of algorithms for approximately answering queries and performing analysis like top-k elements, number of distinct elements, and quantiles. These algorithms are able to provide guaranteed error bounds on the returned answers, letting developers make intelligent tradeoffs between the amount of memory used and accuracy within the single-pass framework.

Modern single-pass algorithms are becoming sufficiently sophisticated so that they can now, in some cases, approach the performance of multiple pass algorithms. “Perhaps the big surprise of streaming is how much is possible with only one look at the data,” says Cormode. “We can approximate the entropy, find how many distinct items there are, or identify the most frequently occurring items.” These tasks are relatively straightforward with a traditional multiple-pass approach, but accomplishing the same objectives with a single-pass architecture—while managing system costs—calls for more sophisticated techniques. “The cost of these algorithms depends on how accurately the answer is approximated,” Cormode says. “It turns out that we can get accurate answers with only kilobytes of storage.”

The Best of Both Approaches

While developers have traditionally faced a binary choice in deciding whether to implement single-pass vs. multiple-pass algorithms, research-

Single-pass algorithms might have applications beyond the traditional realm of data streaming.

ers have recently started to explore techniques that mix the best of both approaches. For example, Cormode is interested in exploring how to mix linear passes with a partial indexing or reordering of data. “I don’t think the debate over how best to model this kind of computation is settled yet,” he says.

As the networking community continues to push the boundaries of these algorithms for traditional data-streaming applications, some researchers are also starting to explore how these approaches could be leveraged into other emerging research arenas. The University of Pennsylvania’s Guha sees potential applications in the realm of specialized hardware, such as like IBM’s cell processor and streaming graphics processing unit computation and ternary content-addressable memory systems. He is also interested in exploring the communication between different parts of a stream in a distributed network.

Some researchers are also starting to explore whether the principles of data streaming algorithms could shed light on the world of social networking, where users are collectively generating a rapidly expanding data stream of text updates and other social graph activity.

While online social networks seem to bear a surface-level resemblance to data networks, those similarities only stretch so far. Networking relies on highly structured data, whereas the social Web consists largely of unstructured text and an unpredictable hodgepodge of media objects like photos, videos, and audio streams. Working with such a diverse data set presents conceptual challenges for researchers accustomed to addressing the relatively cut-and-dried problems of data

streaming. “With streams of unstructured text, friend links, and media postings, the problems are much less clean to define,” says Cormode. “Indeed, defining what to compute is a big part of the problem!”

To mine the social Web effectively, algorithms may need to evolve to incorporate techniques from natural language processing and statistical modeling of temporal trends. Koudas and his team at the University of Toronto are currently exploring approaches to reconstructing chains of events from multiple sources across the Web by sorting out threads, subtopics, and other data points to analyze the evolution of stories in the news using data streaming principles. “The real-time Web is becoming a reality,” says Koudas. “Now the data point is a stream of text rather than a packet.”

As data streaming algorithms start to expand beyond the network layer and into the world of social media, will they evolve into new applications that help map the unpredictable terrain of online friendships, assorted jokes, and cute kitten photos? Will new streaming algorithms take shape to help make sense of the messy, unstructured facts of daily life? Perhaps if there’s one thing that data streaming algorithms can teach us, it is this: Data comes and goes, but the stream never ends. □

Further Reading

Zhang, J.

A survey on streaming algorithms for massive graphs. *Managing and Mining Graph Data*, Aggarwal, C.C. and Wang, H. (eds.), Springer, New York, 2010.

Aggarwal, C.C. (Ed.)

Data Streams: Models and Algorithms. Springer, New York, 2007.

Mathioudakis, M., Koudas, N., and Marbach, P.

Early online identification of attention gathering items in social media. *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, New York, Feb. 2010.

Cormode, G. and Hadjieleftheriou, M.

Finding the frequent items in streams of data. *Comm. of the ACM* 52, 10, October 2009.

Alex Wright is a writer and information architect who lives and works in Brooklyn, NY. Rajeev Rastogi, vice president and head of Yahoo! Labs Bangalore, contributed to the development of this article.

© 2010 ACM 0001-0782/10/0400 \$10.00

Robots Gear Up for Disaster Response

After 15 years of research, robots for search and rescue may be nearing prime time.

ON JANUARY 17, 1995, an earthquake near Kobe, Japan killed approximately 6,434 people, caused the collapse of 200,000 buildings, and resulted in \$102.5 billion in damages. Three months later a truck bomb exploded outside a federal government building in Oklahoma City, OK, and claimed 168 lives and damaged or destroyed 324 buildings within a 16-block radius. While it was an awfully memorable year for disasters, both of these tragedies set in motion a flurry of research in robotics that observers say could save countless lives in future disasters.

Indeed, developers of search and rescue robots say the technology—which spans such diverse disciplines as artificial intelligence, sensing, communications, materials, and mechanical engineering—is nearly ready for deployment. Applications could include search, reconnaissance and mapping, removing or shoring up rubble, delivery of supplies, medical treatment, and evacuation of casualties.

However, a host of technical challenges remain. Also, researchers are concerned that a lack of standards, scarce federal funding, and tepid interest from companies that don't yet see a big market for robotic rescuers stand in the way of the miniaturization, device hardening, and systems integration that are needed to make the technology mature.

Only one emergency response team in the U.S.—New Jersey Task Force One—so far owns a robot. And the robots tested by researchers in a handful of disasters in recent years have produced decidedly mixed performances. “We still don't know how to use these things,” says Robin Murphy, a professor of computer science and engineering and director of the Center for Robot-



A camera robot being inserted in a bore hole at Crandall Canyon Mine in Utah.

Assisted Search and Rescue at Texas A&M University. “Real disasters are infrequent, and every one is different. The robots never get used exactly the way you think they will, and they keep uncovering new bottlenecks and problems. So it's an emerging technology.”

Murphy says the devices are often tested in unrealistically robot-friendly labs or via simulations that don't quite duplicate the realities of real-life situations that involve dirt and sand, steep changes in elevation, or radio-blocking metal structures. Some of the most vexing problems seem simple yet remain frustratingly intractable. For example, at the Crandall Canyon Mine in Utah, where six miners and three rescue workers were killed in 2007, mud greatly hindered the effectiveness of the workers' camera robot. “We steered the robot to places where water was dripping and turned it face-up to rinse off some of the mud,” Murphy says. However, the cam-

era robot was eventually trapped by a rock slide, causing the robot's tether to snap and for it to be lost.

Howie Choset, associate professor of robotics at Carnegie Mellon University (CMU), specializes in snake robots, which are thin, legless devices with multiple joints. They can go places more traditional, track- or wheel-propelled robots can't, but the technology still needs work. “My last trial at a rubble pile in Texas didn't go so well,” Choset notes. “They didn't get over little obstacles I thought they should have. Our control laws are still not well defined; we don't have good feedback; we don't have enough sensing in the robots; and their skins have to be better designed.”

So while his mechanical snakes can perform remarkable feats such as crawling up the interior of a vertical pipe or swimming across a pool, Choset says a lack of funding stands in the way of making the snakes truly versatile and

robust. Choset needs, for instance, to develop more mechanical snake gaits. He'd like his snakes to be able to change from a vertical undulating gait to a side-winder gait on command or, better yet, to autonomously switch gaits to suit new conditions. And he'd like the mechanical snake to know how to execute one gait in its front segments and a different gait at the rear segments. "We have developed the greatest variety of snake gaits in the world," says Choset, "but a rubble pile has that many more situations than we can anticipate."

Asked if further animal study would help, Choset replies, "It's true you are inspired by biology, but snakes have 200 bones and the snake robot has just 15 links. Snakes have a material called muscles, but we are not going to be making muscles any time soon." And, he adds, snakes have marvelous sensors for heat and pressure in their skins, something else technology has yet to easily match.

Three Levels of Challenges

Users of search and rescue robots face challenges at three levels, says Sanjiv Singh, a research professor at CMU's Robotics Institute. At the lowest level lies information processing—getting and managing information about the environment. At the next level comes mobility—getting the robot to where it is needed. And at the highest level comes manipulation—enabling the robot to perform the appropriate physical task once it is in place. Singh's Ember project, partially funded by the U.S. National Science Foundation, seeks to aid first responders at the first two levels, and in situations that are dynamic, chaotic, and often providing poor visibility.

Inside a burning building, for example, it is unlikely that the structure's communication systems will remain working, and first responders won't have a map or plan for the building. Singh's group has developed technology whereby a firefighter or a robot can scatter smart radio beacons inside the building. Some beacons are stationary and some are attached to a human or robot. These nodes begin talking to each other and autonomously organize themselves into an ad hoc sensor network. The radios measure distances to each other and, using algorithms developed by Singh's

Miniaturization, device hardening, and systems integration are all needed for the maturing of search and rescue robots.

group, construct a map of their physical layout—or a map of conditions such as temperature—and track the movement of robots or people.

"Imagine there is a commander standing outside the building, and he looks at a screen and he can see where all his people are inside the building," Singh says. "And they can do this without any prior survey of the building, and without any power or prior communications infrastructure inside the building."

Constructing spatial maps from distance-only data has been feasible for some time. It's possible to draw a map showing the positions of cities in the U.S. solely from the intercity mileage table at the back of an atlas, Singh says. But his innovation was the development of algorithms—based on Kalman filtering, Markov methods, and Monte Carlo localization—that can do the job with a sparsely populated distance table.

Singh has also made progress at the second level of the hierarchy, the one dealing with robot mobility. He has developed a suite of search algorithms for teams of robots to use in spaces humans can't or don't want to go. Some are suited to looking for an immobile person, while others are geared to looking for moving people, such as an intruder. In the latter case, the robots might post "guards" at various locations in a building to spot the intruder's movement.

In addition, Singh's algorithms can be classified as "efficient" (find a target in the lowest expected time); "guaranteed" (clear the environment so capture is assured); or "constrained" (maintain robot positions that ensure network connectivity or line-of-sight communi-

cation). "You could combine these algorithms if you have a team of robots or a team of robots and humans," he says. Combining an efficient search with a guaranteed search would tend to minimize search time while still making sure the search ultimately succeeds.

Murphy, who has become a kind of evangelist for the search and rescue robotics community in the U.S., says the technical problems associated with the devices will be solved in due course. But she says strong government funding and support is needed if search and rescue robots are to see widespread use in fewer than 10 years. The standards being developed now at the National Institute of Standards and Technology will also be a big help, Murphy predicts.

Brilliant robotic technology exists, says Murphy, but it needs to be integrated into complete, robust systems, and sensors and other components must be made smaller, stronger, and cheaper. All of this requires corporate effort, she notes. "We are just inches away," Murphy says. "A lot of the software is just waiting for the hardware to catch up." ■

Further Reading

Murphy, R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., Erkmén, A.

Search and rescue robotics. *Springer Handbook of Robotics*, Siciliano, B. and Khatib, O. (eds.). Springer Science and Business Media, Secaucus, NJ, 2008.

Kumar, V., Rus, D., Singh, S.

Robot and sensor networks for first responders. *IEEE Pervasive Computing* 3, 4, October-December, 2004.

Shammas, E., Choset, H., Rizzi, A.

Geometric motion planning analysis for two classes of underactuated mechanical systems. *International Journal of Robotics Research* 26, 10, October 2007.

International Rescue System Institute

<http://www.rescuesystem.org/IRSweb/en/IRSU.html>

Biorobotics Laboratory, Carnegie Mellon University

<http://www.cs.cmu.edu/~biorobotics/>

Dekker, S.

The Field Guide to Understanding Human Error. Ashgate Publishing, Farnham, Surrey, U.K., 2006.

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

© 2010 ACM 0001-0782/10/0400 \$10.00

Spies Among Us?

Governments' practice of electronic surveillance—and the growing use of warrantless wiretapping—has observers deeply concerned.

THE BOUNDARY BETWEEN protection and privacy has always been a bit fuzzy. However, in an era of global terrorism, powerful drug cartels, and aggressive cybercriminals, it's no secret that government officials are tapping into technology in ever more sophisticated ways to monitor voice and data communications. "There are potential benefits but there are also potential risks," says Steven M. Bellovin, professor of computer science at Columbia University. "In many instances we're moving into uncharted territory."

Make no mistake, electronic surveillance—and the growing use of warrantless wiretapping—has caused deep concerns. Since New York's Twin Towers were destroyed during a terrorist attack in 2001, intelligence agencies around the world have embraced new and more powerful tools to intercept messages and track potential threats. The ultra-secretive nature of the monitoring combined with the dragnet-style approach to collecting data pushes laws and legal systems to the limit—and perhaps beyond.

It's no small matter. Government entities that intercept plots and identify plans are more likely to thwart attacks and save lives. In addition, data mining and predictive analytics tools provide clues that could help prevent a disaster. However, if data is used improperly—and historical as well as current evidence supports the fact that this scenario is likely—serious legal questions arise. Among other things, these center on the use and misuse of government secrecy; the extent of unilateral presidential authority; and the role of intermediaries, including telecommunications companies and Internet service providers, in protecting privacy and assisting law-enforcement agencies.

Moreover, the use of warrantless surveillance could unleash chilling consequences, including Watergate-style



Mark Klein, the former AT&T technician who blew the whistle on the U.S. government's wide-ranging, ultra-secretive surveillance program to *The New York Times*.

attacks on political enemies and, for society, the curtailment of civil liberties and freedom. Says Susan Landau, a distinguished engineer at Sun Microsystems and co-author of *Privacy on the Line: The Politics of Wiretapping and Encryption*, "Cybersecurity measures must be taken seriously. There are consequences at all levels of society if these tools and systems are misused."

1984 +26?

The battle for intelligence information is nothing new, of course. For centuries, governments have worked to intercept communications—using informants, spies, eavesdropping, and whatever means necessary to achieve their goal. Law enforcement agencies, too, have embraced data collection methods, including wiretaps. However, in virtually all democracies, well-established laws control telephone tapping and other forms of wiretapping. In most instances, it's necessary to obtain a court order to eavesdrop.

The legal system in the U.S., for example, is well defined. The Fourth Amendment to the Constitution protects against unreasonable searches and seizures. What's more, various laws—including Title III in 1968 and the Foreign Intelligence Surveillance Act (FISA) in 1978—establish strict guidelines about the use of wiretaps. In fact, FISA was established partly as a response to the Watergate spying scandal and the resignation of President Nixon in 1974.

Nevertheless, protections haven't always prevented abuses. For example, the U.S. Federal Bureau of Investigation—under the leadership of J. Edgar Hoover—began assembling dossiers on spies, political activists, and others beginning in the 1930s. Eventually, Hoover used wiretapping and other surveillance tools to track more than 12,000 Americans, including political leaders and public figures. Hoover later launched campaigns to discredit those who spoke out against him and

the FBI. Frequently, he tapped into information collected through unauthorized spying to target enemies.

Fast forward to the present day and the stakes are much higher. Nuclear weapons, biological threats, and other forms of terrorism represent a widespread and persistent danger. Yet, technology also provides powerful tools for monitoring phone conversations, electronic messages, and data streaming across the Internet. This has led organizations like the National Security Agency (NSA) in the U.S. to pursue warrantless domestic wiretapping. “There is evidence that the NSA is monitoring the communications of millions of Americans,” says Kevin S. Bankston, senior staff attorney for the Electronic Frontier Foundation (EFF).

The EFF contends that the U.S. government, with the aid of AT&T and other telecommunications companies, has engaged in a massive domestic dragnet surveillance program over a nine-year span beginning since at least 2001. Under one claim, the telecom carrier databases of customer calling records appear to have been made available for examination by the NSA; under other claims, technical devices may have copied Internet traffic and made it available to the NSA.

This program came to light in January 2006. That’s when the *New York Times* broke a story about the existence of ultra-secretive surveillance and Mark Klein, a former AT&T technician

In the U.K., the Home Secretary, a cabinet minister, approves all wiretaps and no judge is required.

turned whistleblower, provided so far undisputed evidence that the telecom provider had set up a separate room run by the NSA at the company’s San Francisco offices. AT&T routed copies of Internet traffic to computers inside the room, he contends.

Not surprisingly, the fallout is growing. In January 2006, the EFF filed a class action lawsuit, *Hepting v. AT&T*, designed to curtail unwarranted eavesdropping. It alleges that the telecommunications provider violated customers’ privacy. Then, in September 2008, EFF filed a second lawsuit, *Jewel v. NSA*, which seeks to halt warrantless wiretapping and hold the government officials who used it accountable. Both *Hepting v. AT&T* and *Jewel v. NSA* have been dismissed by the courts and are being appealed by EFF. “We believe that these practices present a significant danger to a democratic society,” Bankston explains.

Although it refuses to comment on pending litigation, NSA stands behind its overall objectives. “[The agency] is committed to performing its mission under the rule of law and takes seriously its responsibilities to protect privacy rights,” says NSA spokesperson Marci Green. “All of our employees with mission responsibilities are thoroughly trained on the legal authorities and procedures that govern NSA’s activities.” AT&T, meanwhile, says only that it is “fully committed to protecting our customers’ privacy. We do not comment on matters of national security.”

Down to the Wire

Historically, wiretapping has been a fairly targeted activity. “You knew who you wanted to listen to and you targeted this person’s communications,” states Patrick Radden Keefe, author of *Character: Dispatches from the Secret World of Global Eavesdropping*. However, after the passage of the USA PATRIOT Act in 2001, even though the statute did not authorize dragnet surveillance, the scope and boundaries for surveillance shifted dramatically in the U.S.

Ratcheting up the stakes further is a quantum leap in technology. Today, data switches and telephone switches from major equipment manufacturers such as Cisco, Lucent, and Nortel have built-in data intercept capabilities; and data mining and analytics applications identify targeted packets—including

Obituaries

Andrew Booth, 91, Ray Solomonoff, 73, are Dead

Computer science lost two pioneering, seminal figures with the deaths of Andrew Donald Booth, 91, and Ray Solomonoff, 73, on November 29, 2009 and December 7, 2009, respectively.

A British computer scientist, physicist, and engineer, Andrew Booth is famous for Booth’s multiplication algorithm, which multiplies two signed binary numbers in two’s complement notation, and the invention of the magnetic storage device for computers. In the late 1940s, Booth realized that data storage would be an important

computer component if they were to become viable machines. In 1948 he demonstrated the successful operation of a rotating drum, a brass cylinder coated with nickel, as part of an automatic relay computer, and three years later developed an all-purpose electronic computer, Apec, one of the first electronic computers.

An American computer scientist, physicist, and mathematician, Ray Solomonoff is recognized for his invention of algorithmic probability and his founding role in the branch of artificial intelligence based on

machine learning, prediction, and probability. In the late 1950s, Solomonoff created probabilistic languages and their associated grammars, which led to the discovery in 1960 of algorithmic probability and his publication of the theorem that launched Kolmogorov complexity and significantly furthered the development of information theory.

In the early 1950s, Solomonoff, Marvin Minsky, John McCarthy, and others became interested in machine learning and, in 1956, they were among the 10 attendees

of the Dartmouth Summer Research Conference on Artificial Intelligence, a seminal event in the foundation of the field of artificial intelligence. Solomonoff pioneered the application of probability theory to solving artificial intelligence problems, which is now the dominant approach among artificial intelligence researchers, and developed the idea of the “infinity point,” a formula for the cost and the time required to create a machine that is vastly more intelligent than a group of humans.

—Jack Rosenberger

names and other words—while using sophisticated algorithms with predictive capabilities to provide insights into possible threats.

But that's only part of the story. The NSA "now monitors huge volumes of records of domestic emails and Internet searches as well as bank transfers, credit-card transactions, travel and telephone records," writes Bruce Schneier, a computer security expert, author, and blogger. In fact, according to the *Wall Street Journal*, the NSA receives this so-called "transactional" data from other agencies or private companies, and its software programs analyze the various transactions for suspicious patterns. Officials hand out promising leads to counterterrorism programs scattered across the U.S. government.

Still, no one outside the NSA and elite government circles appears to know exactly what's being used and how. And the issue affects an unknown and possibly increasing number of nations. In 2004, an unknown individual or individuals installed unauthorized software on the Vodafone Greece telephone network and illegally monitored the cell phone conversations of more than 100 persons, including prime minister Kostas Karamanlis and many leading government officials, for approximately seven months. "It was an extremely sophisticated attack that exploited lawful intercept software," Bellovin explains. "It was proof that these systems can be exploited. It has probably happened elsewhere and we simply don't know about it."

Indeed, the fear of hackers infiltrating intelligence systems isn't unwarranted. Insiders such as technicians, who have access to switches and other equipment, are of particular concern to security experts. "It's supposed to be impossible for an insider to gain access to these systems," Bellovin states. "It's most likely possible. There's a long history of phone phreaks breaking into telephone switches and hackers getting into highly secure government computing systems. So, you have to ask why it's not possible."

On the international scene, serious concerns have also taken root. In Italy, judges are allowed to order wiretaps but because their function is more akin to a prosecutor than an inde-

pendent entity—and there are no obstacles to obtaining a wiretap—many civil libertarians worry that there are too few checks and balances in place. In the United Kingdom, the Home Secretary, a cabinet minister, approves all wiretaps and no judge is required. And in the Netherlands police can tap any phone or computer network as long as the crime under investigation has at least a three-year prison term.

Sweden, Turkey, South Korea, Mongolia, and many other countries have also found themselves embroiled in recent wiretapping controversies. And in numerous countries—most notably China and Iran—wiretapping and other surveillance is simply a fact of life.

The debate over surveillance methods and civil liberties isn't about to disappear. The ongoing threat of terrorism translates into a desire among government leaders to build ever-more powerful snooping technology. While no one would argue that security and intelligence gathering should be tossed aside, many observers say that it's time to approach the issue responsibly. Concludes Bankston, "It's essential to respect privacy and adhere to the principles of a free society." ■

Further Reading

Diffie, W. and Landau, S. *Privacy on the Line: The Politics of Wiretapping and Encryption*. MIT Press, Cambridge, MA, 2007.

Risen, J. and Lichtblau, E. E-Mail surveillance renews concerns in Congress. *The New York Times*, June 16, 2009.

Schneier, B. NSA's domestic spying, March 26, 2008, http://www.schneier.com/blog/archives/2008/03/nsas_domestic_s.html.

Offices of the Inspector Generals, Dept. of Defense, Dept. of Justice, Central Intelligence Agency, National Security Agency, Office of the Director of National Intelligence *Unclassified Report on the President's Surveillance Program, July 20, 2009*, <http://www.fas.org/irp/eprint/psp.pdf>.

Electronic Frontier Foundation *NSA spying*, <http://www.eff.org/issues/nsa-spying>.

Samuel Greengard is an author and freelance writer based in West Linn, OR. Cindy Cohn, Electronic Frontier Foundation, contributed to the development of this article.

© 2010 ACM 0001-0782/10/0400 \$10.00

Milestones

Dan David Prize



Computer scientists Gordon E. Moore (left), Michael O. Rabin, and Leonard Kleinrock are

among the six recipients of the 2010 Dan David Prize, in the category of computers and telecommunications. The three laureates will split a prize of \$1 million and be honored in a ceremony at Tel Aviv University in May; Israeli President Shimon Peres is scheduled to be in attendance.

Moore, co-founder and Chairman Emeritus of Intel, was cited for his "remarkable intuition leading him to proclaim 'Moore's Law,' which has become the guiding principle for the semiconductor industry to deliver ever-more-powerful chips while decreasing the cost of electronics, and for his co-founding of the outstanding microprocessor pioneer, Intel."

Rabin, the Thomas J. Watson, Sr. Professor of Computer Science at Harvard, was cited for his "major research results, which have had and which will continue to have a great impact on the shape of computer and communication technology and, in particular, for his work on automata and complexity theory, on probabilistic algorithms, and on ways to improve privacy and create unbreakable ways to encrypt data, making secrecy, privacy, and protection ever more crucial to society."

Kleinrock, a professor of computer science at the University of California, Los Angeles, was cited for his "seminal research contributions in communication networks, establishing the fundamental principles upon which many of the most important aspects of information communications and the Internet are based."

The Dan David Prize is an annual international award and named after businessman and philanthropist Dan David, and is headquartered at Tel Aviv University. The prize recognizes and encourages innovative and interdisciplinary research and aims to promote the scientific, technological, and humanistic achievements that advance and improve the world.



Congratulations

2009 ACM Distinguished Members

ACM honors 84 new inductees as Distinguished Members in recognition of their contributions to both the practical and theoretical aspects of computing and information technology

2009 ACM Distinguished Educators

Mordechai Ben-Ari
Weizmann Institute of Science

Martin C. Carlisle
United States Air Force Academy

Thomas H. Cormen
Dartmouth College

Suzanne W. Dietrich
Arizona State University

Robert Geist
Clemson University

Charles F. Kelemen
Swarthmore College

Nancy R. Mead
Software Engineering Institute, CMU

Ramon C. Puigjaner
University of Balearic Islands

Henry M. Walker
Grinnell College

2009 ACM Distinguished Engineers

Eric Allman
Sendmail, Inc.

Jay Bayne
Milwaukee Institute

Tyrone Grandison
IBM Almaden Research Center

Mamdouh Ibrahim
IBM

Jeffrey T. Kreulen
IBM Research

James McGill
Morgan Stanley

Dwight Meglan
SimQuest

Jim Melton
Oracle Corp.

Milan Milenkovic
Intel

Shubhendu Mukherjee
Intel Corporation

Inderpal Singh Mumick
Kirusa

Brian M. Novack
AT&T

Pradeep Kumar Sinha
*Centre For Development Of
Advanced Computing (C-Dac)*

Bryan F. Smith
IBM

Paul A. Strassmann
George Mason University

William Tracz
Lockheed Martin IS&GS

Allen Wirfs-Brock
Microsoft Corp.

2009 ACM Distinguished Scientists

Ana Isabel Antón
North Carolina State University

Lars Arge
Aarhus University

Kellogg S. Booth
University of British Columbia

Christian Cachin
IBM Zurich Research Laboratory

Ludmila Cherkasova
Hewlett-Packard Laboratories

Mark Crovella
Boston University

Mary Czerwinski
Microsoft Research

Sunil R. Das
Troy University

Laura K. Dillon
Michigan State University

Richard Draves
Microsoft Research

Schahram Dustdar
Vienna University of Technology

Keith Edwards
Georgia Tech



2009 ACM Distinguished Scientists

continued

Carla Schlatter Ellis

Duke University

Ahmed Elmagarmid

Purdue University

Sally Fincher

The University of Kent

Rebecca E. Grinter

Georgia Institute of Technology

Manish Gupta

IBM India Research Laboratory

Jayant R. Haritsa

Indian Institute of Science

Bruce Hendrickson

Sandia National Laboratories

Michael A. Heroux

Sandia National Laboratories

Michael Hind

IBM T. J. Watson Research Center

Mikhail Borisovich Ignatyev

*St. Petersburg State University
of Aerospace Instrumentation*

John Karat

IBM T. J. Watson Research Center

James H. Kaufman

IBM Research Division

Stefanos Kaxiras

University of Patras, Greece

Andras Kornai

Harvard University

Ajay D Kshemkalyani

University of Illinois at Chicago

Tok Wang Ling

National University of Singapore

Jorge Lobo

IBM T. J. Watson Research Center

Anna Lubiw

University of Waterloo

Carlos J. P. Lucena

*Pontifical Catholic University
of Rio de Janeiro*

Rao Manneppalli

Lockheed Martin

Jose E. Moreira

IBM T.J. Watson Research Center

Ngoc Thanh Nguyen

Wroclaw University of Technology

James F. O'Brien

University of California, Berkeley

Harold Ossher

IBM T. J. Watson Research Center

Venkata N. Padmanabhan

Microsoft Research India

Fabio Paternò

CNR-ISTI

Naren Ramakrishnan

Virginia Tech

Ganesan Ramalingam

Microsoft Research India

Yong Rui

Microsoft China R&D Group

Pierangela Samarati

Universita' degli Studi di Milano

Cyrus Shahabi

University of Southern California

Sol M. Shatz

University of Illinois at Chicago

Prashant Shenoy

University of Massachusetts, Amherst

Frank M. Shipman, III

Texas A&M University

Peretz Shoval

Ben-Gurion University

Liuba Shrira

Brandeis University

Vugranam Sreedhar

IBM T. J. Watson Research Center

Peter J. Stuckey

University of Melbourne

Loren G. Terveen

University of Minnesota

Ron van der Meyden

University of New South Wales

Ellen M. Voorhees

*National Institute of Standards
and Technology*

David B. Whalley

Florida State University

Edward Wobber

Microsoft Research Silicon Valley

Cheng Xiang Zhai

*University of Illinois at
Urbana-Champaign*

Liang-Jie Zhang

IBM T.J. Watson Research Center

Michelle X. Zhou

IBM T. J. Watson Research Center



**Association for
Computing Machinery**

Advancing Computing as a Science & Profession

<http://distinguished.acm.org>

Emerging Markets Development 2.0: The IT-Enabled Transformation of International Development

The fundamental assumptions of international development are changing, increasingly putting the tools for a digital economy into the hands of the world's poor.

WHERE ARE THE AMAZON and eBay for international development? If we could find them, they might represent “Development 2.0”: new IT-enabled models that can transform the processes and structures of development. In this column, I will highlight some examples, and analyze how they are changing the way we “do development.”

The foundation for these changes has been the rapid diffusion of IT into the developing world. In 1998, less than one out of every 100 inhabitants in developing countries was an Internet user. By 2008, that figure was 22 out of every 100. In 1998, two of every 100 inhabitants in developing countries was a mobile phone subscriber. By 2008, that figure was 55 out of every 100.⁴ Shared usage takes this further: even in the world's poorest continent—Africa—an estimated two-thirds of the population now has access to a mobile phone.²

What happens when you start to connect the world's poor into the infrastructure for a digital economy? What happens is some of the basic assumptions about barriers to development might no longer apply. Some examples follow, divided into three categories.

New Relations

Connecting the excluded: The world's poor have historically been excluded and disconnected from information, and from potential suppliers and customers. Information technology can cut across these historical barriers and help connect the poor. For instance, as Mark Granovetter famously identified, if you want to get a job it is not the strong, close ties of your immediate family and friends that help (family and friends only know what you already know). Instead, it is the comparatively weaker ties of more distant social connections that are most advantageous. Yet the poor lack such connections.

IT can help by linking to a much wider social world through information exchanges. One such is Babajob (<http://www.babajob.com>), a networking site on which potential employers in urban India can post details of low-skilled jobs. To reach beyond those with Internet access, the system sends job alerts via SMS to those who might be looking for such a job, or who might know someone who is. At

What happens when you start to connect the world's poor into the infrastructure for a digital economy?

present, more than one million alerts are sent out every month, breaking through the traditional obstacles to information flow.

Disintermediating: Where the poor are not disconnected, they are often connected via gatekeepers of dubious quality. To access government services, they may have to go via a corrupt official. To access finance, they may have to rely on a usurer charging extortionate interest rates. To access agricultural assistance, they may require an extension officer who visits infrequently.

Many e-business models rely on “cutting out the middleman,” and IT offers the same facility for international development. The Bhoomi project has provided e-government services in India’s Karnataka state since 2001; for example issuing land ownership certificates to farmers who need these to obtain bank loans. An impact assessment shows it does what one would expect of an IT project: cutting error rates and improving service quality.¹

What it also does is disintermediate. Previously, farmers had to apply for their certificate via a government official. In approximately half of the cases, that official would demand a bribe before they would issue the certificate (averaging around US\$3; equivalent to a day’s income in rural India). After computerization, those officials have been significantly removed from the process, as farmers largely get their certificates issued online by visiting a local Internet kiosk. As a result, less than 1% report having to pay a bribe.

New Roles

Digital production: The poor have always been producers, largely of agricultural produce for their own subsistence or for sale, often at low prices. But they have lacked access to resources and capabilities that would help them break out of the grip of poverty. The diffusion of IT is giving them access to the means of production for a digital economy, offering radically new ways of earning a living.

Txteagle (<http://txteagle.com>) is an example that brings crowdsourcing to the mobile phone base in Kenya. It takes simple tasks suitable for a voice-and-SMS phone and outsources them to those who have both a mobile phone and time available. Examples of such

work include translation of text into local languages, transcription of audio clips, and input of survey data for development agencies working in a local area. Payment can be either airtime or cash using Kenya’s M-PESA mobile currency system, and there has been a specific focus on pushing work out to the rural poor.

Digital innovation: Having access to IT tools means some of those at the bottom of the pyramid have moved beyond production to innovation. They have appropriated the technology to such an extent that they start to do new things with it.



An M-PESA agent converting money into e-value cash in Masai Mara, Kenya.

Many such innovative uses are digital memes: ideas that originated somewhere; perhaps simultaneously in the slum areas surrounding most third-world cities that are crucibles of both poverty and creativity. Beeping (or flashing) is one such innovative usage for mobile phones: hanging up a call before it is answered. This has developed into a free messaging system. Street hawkers allocate different mobile phone ringtones to different customers, enabling a free “come sell to me” message.

Use of airtime as currency is a similar innovative development. Family members in distant cities remit “money” back home as an airtime transfer. This can then be used in the village to pay for goods and services from those traders who themselves have a mobile phone.

New Models

Collective Power: Many poor communities have a collective strength; for example, working together on farming tasks that no individual household alone could complete in time. Yet it can be difficult for them to express their collective power to outside bodies, such as government.

IT enables “crowdvoicing”: the capture of group knowledge and opinions within a community and its dissemination to a broader audience. Community radio has changed from a small-scale version of the one-way-broadcast approach of national radio stations.

Through SMS, phone calls, mobile phone clip recordings, and PC-based audio, community members can contribute content and have their voice heard on such radio stations.

One step further, and information technology can be used to turn that voice into actual decision-making power. In Belo Horizonte, Brazil, the city government allocated a US\$11 million decision to an online vote of ordinary citizens, who were given a choice between spending money on a new sports complex, a library, street renewal, or a commercial center regeneration project. This “e-participatory budgeting” initiative drew in more than 500,000 votes (the sports complex won); seven times more participants than seen with earlier non-IT-based participatory budgeting.⁵

Social Enterprise: Poor communities can be caught between the Scylla and Charybdis of business models. Commercial models may ignore them because they are seen to lack money and capabilities; where such models arrive they may bring dynamism and profits for the private sector, but not development. Developmental models—such as those brought by government—may lack dynamism, breed dependency, and be inefficient or even corrupt.

Social enterprise is seen to steer between the rock and the hard place, combining business drive with a concern for socioeconomic benefits within the community. IT has enabled the use of social enterprise; for example, allowing the growth of “social outsourcing,” which subcontracts IT work to the urban and rural poor in a deliberate attempt to create new livelihoods. The Kudumbashree initiative in India’s Kerala state has been a leader in this, diverting the state government’s data entry, digitization, IT training, and PC assembly and maintenance work to social enterprises created by groups of unemployed women from below-poverty-line families. The result to date has been the creation of approximately 2,500 new jobs that have pulled hundreds of households out of poverty, and created a base of enterprise that is now diversifying to find new clients.³

Conclusion

So how is all this happening? At root, it stems from the power of IT. Some of those powers are generic. IT can cut costs so dramatically that new ways of doing things become possible. The digitization of data is part of that, allowing reproduction and communication of information at virtually zero cost. And the increasingly ubiquitous connectivity of IT allows new relations and network structures to exist.

As Mark Cleverley outlined in his September 2009 *Communications Emerging Markets* column, general technical innovations will continue to drive this process forward. There are also specific innovations for the bottom of the pyramid that are helping. Some of these are technical. Movirtu’s MXShare provides device-independent mobile services for users unable to afford a real phone. It gives them a virtu-

The increasingly ubiquitous connectivity of IT allows new relations and network structures to exist.

al mobile phone number and account that can be accessed via a PIN from any borrowed or communal phone. The One-Laptop-Per-Child project, though often criticized, is putting digital processing and communication into the hands of millions of children for the first time; a vast social experiment that will no doubt catalyze many new Development 2.0 initiatives.

Development 2.0 is being enabled by business innovations, including simple ones like pre-paid mobile tariffs. These innovations broke the logjam of unaffordability, allowing the poor to buy airtime as and when money was available, and are now universal throughout the poorer parts of the developing world. Some innovations combine business and technology. For example, Kenya’s M-PESA system, mentioned earlier, provides a low-cost digital money platform for mobile phones, onto which all sorts of new financial applications previously denied to low-income users—saving, lending, insurance, and so forth—are being mounted.

In discussing Development 2.0, it is important we keep our feet on the ground. We have seen this in e-business. Talk of the “new economy” or “weightless economy” soon gave way to a realization that the old economy was still very much around. For every Amazon or eBay, there were hundreds more “clicks and bricks” operations representing a more incremental than transformative approach.

The same is true for international development. Dig behind the images of disintermediation, for example, and a rather different picture emerges. Farmers using the Bhoomi project still go through an intermediary, swapping

the local government official for the local Internet kiosk owner. Kiva operates peer-to-peer micro-lending allowing individual lenders in the U.S. to invest in entrepreneurs in the global south. But to make the global financing chain work, first Kiva itself and then a series of local partner organizations must sit between lender and borrower.

For some examples, Development 2.0 may be too optimistic. Perhaps these are Development 1.5 examples at present, with a promise of greater change to come. But it is not yet known which will be the amazon.com success and which will be the beenz.com failure. Indeed, in a further echo of the dot-com boom and bust some of the initiatives cited in this column are long on media hype and public relations efforts but short on facts and figures about actual impacts—how many poor people actually involved; how much money saved; how much income generated. Hopefully we’ll see some solid and objective research emerging on these in the not too distant future.

In the meantime, we can celebrate the fact that the foundations and assumptions of international development are changing. The tools for a digital economy are now—and will increasingly be—in the hands of the world’s poor. Our view of them can start to migrate: from seeing them as victims to seeing them first as consumers, then producers, then innovators of a digital age. And, as we do so, changing our views on the processes and structures of socioeconomic development: from Development 1.0 to Development 2.0. ■

References

1. Bhatnagar, S.C. and Singh, N. *Results from a Study of Impact of E-Government Projects in India*. ICTD2009, Doha, 2009.
2. Heeks, R.B. *Beyond Subscriptions: Actual Ownership, Use and Non-Use of Mobiles in Developing Countries*. ICT4D blog, 2009; <http://ict4dblog.wordpress.com>
3. Heeks, R.B. and Arun, S. *IT Social Outsourcing as a Development Tool*. Development Informatics Group, IDPM, University of Manchester, U.K., 2007; <http://www.womenicenterprise.org/IT%20SocialOutsourcing%20Kerala%20Paper.doc>
4. ITU. *ICT Statistics Database*, International Telecommunications Union, Geneva, 2009; <http://www.itu.int/ITU-D/icteye/Indicators/Indicators.aspx>
5. Peixoto, T. *e-Participatory Budgeting*, University of Zurich, 2008; <http://edc.unige.ch/edcadmin/images/Tiago.pdf>

Richard Heeks (richard.heeks@manchester.ac.uk) is the director of the Centre for Development Informatics at the University of Manchester, U.K.; <http://www.manchester.ac.uk/cdi>

Copyright held by author.



Historical Reflections

Be Careful What You Wish For

Reflections on the decline of mathematical tables.

FOR SOME PEOPLE it's typewriters. For other people it's mechanical calculating machines that bring a nostalgic tear to the eye. For me it's mathematical tables. The sight—even the smell—of a set of four-figure tables transports me to my distant school and college days. You can still find mathematical tables—their yellowed pages filled with decimal digits and not much else—in secondhand book stores and occasionally on eBay. I once thought I might like to collect mathematical tables, but then I discovered from the *Index to Mathematical Tables*¹ that many hundreds of tables have been published. Even a selective collection would prove burdensome, if not grounds for divorce. Ironically, the *Index to Mathematical Tables*, a monumental bibliographical endeavor, was published in 1962, just as tables were going out of business.

Mathematical tables were excruciatingly tedious to calculate. Take for example logarithmic tables, which ruled the calculating roost for about 300 years. Logarithms were invented by the great Scottish philosopher John Napier around 1614, and a practical table of logarithms to base 10, the *Arithmetica Logarithmica*, was calculated by the English mathematician Henry Briggs and published in 1624. While the logarithm was an invention of genius, computing them to 14 decimal places was assuredly a labor of Hercules. It is said that when Napier and Briggs first met “almost one quarter



Part of the collection of books of mathematical tables assembled by Charles Babbage.

of an hour was spent, each beholding the other with admiration, before one word was spoken.”

Logarithms were so laborious to calculate that subsequent tables were not recomputed but were compiled from the existing canons. The raw logarithms would be reduced to four, five, six, or seven decimal places and conveniently arranged and printed using the best ty-

pography of the day. It is said that the most accurate table of logarithms ever produced was Charles Babbage's seven-figure tables. Babbage was somewhat a connoisseur of tables and owned approximately 300 volumes. Like his predecessors, he did not recompute the logarithms but copied them from existing tables. Where there was a discrepancy in his sources he would recom-

pute the offending entry. It was almost a literary exercise—an anthology of the best logarithms, so to speak.

But Babbage did produce some astronomical and actuarial tables from scratch. Babbage used computers for the calculations—not machines but human drudges, because “computer” was then an occupation, not a machine. It was while he was checking his astronomical tables that he made the oft-quoted remark “I wish to God these calculations had been executed by steam.” Shortly after, in 1824, he secured funding from the British government to build a “difference engine” for calculating tables (powered by hand, not steam). Babbage’s engine was a magnificent failure. He overdesigned, mismanaged, and nothing materialized except a prototype and some plans for a full-scale machine. Babbage’s difference engine was finally completed in 1991 at the London Science Museum, in time to celebrate the bicentenary of Babbage’s birth. Amazingly, it worked beautifully.

Although a few difference engines along Babbage’s lines were built in the second half of the 19th century, they tended to be fickle and expensive and there was no real market for them. So, for the first half of the 20th century most tables were produced by human computers, sometimes using a mechanical desk calculator or a punched card machine, but as often as not just plain pencil and paper.

The biggest human computing organization in the world was established in New York in 1938 and is described by David Grier in his remarkable book *When Computers Were Human*.² The so-called Mathematical Tables Project was a depression-era, make-work project for Roosevelt’s Works Progress Administration. It was intended to employ out-of-work clerks and bookkeepers in the useful work of making mathematical tables. Most of the project’s human computers used nothing more than colored pencils and pre-printed calculating sheets. At its peak in World War II, the project employed 450 computers producing tables for the war effort. Table making was then a flourishing business. In 1943, a new journal—*Mathematical Tables and Other Aids to Computation*—recorded the practical and theoretical advances being made.

Another massive wartime table-

making project was established by the Moore School of Electrical Engineering, University of Pennsylvania, and the Ballistics Research Laboratory at the Aberdeen Proving Ground, Maryland. The war had created an unprecedented demand for “firing tables” for newly developed artillery and for existing weapons to be deployed in new theaters of war. A firing table provided the elevation and azimuth to enable a gun to be aimed with accuracy for a given target distance. A team of 200 female computers, equipped with mechanical calculating machines, helped to produce the tables. But this was not nearly enough computing power. Each firing table contained about 3,000 entries, and each entry took about one or two person-days to compute. Two Moore School academics, John Mauchly and Presper Eckert, proposed the construction of an electronic computer to perform these ballistics calculations—this was the ENIAC, the Electronic Numerical Integrator and Computer. Eckert and Mauchly got the go-ahead to build the ENIAC in spring 1943, although it was not completed until late 1945 and so was too late to help in the war effort. While ENIAC was not the first, it was certainly the most famous early electronic digital computer.

Before the ENIAC was finished, however, it was realized that although it could calculate ballistics tables to perfection, it was otherwise rather limited in the kind of computations it could perform. In the summer of 1944 the Princeton mathematician John von Neumann learned of the ENIAC, and worked with the Moore School team to design a better machine. Dubbed the EDVAC, for Electronic Discrete Variable Automatic Computer, it was a design of such versatility that its architecture has underpinned computer design ever since. It became known as the stored-program computer.

In the 1950s the new stored-program computers started to proliferate. Suddenly table making was made easy. Among other computational tasks, computers began to grind out tables as never before. We can see from the *Index of Mathematical Tables* that in the 1950s more tables were published than in any previous decade. But of course, if you had access to a computer you no longer had much need for mathemati-

cal tables. For example, trigonometrical and hyperbolic functions could be computed using a simple subroutine to compute values on-the-fly in the course of a program.

At first it was not entirely clear whether computers would stimulate the production and consumption of tables or eliminate them entirely. An aging generation of table makers clung to the former hope. But the writing was on the wall. By the end of the 1950s, most big academic and research institutions had a computer, and it was clear that by the end of the 1960s they all would. In the late 1940s *Mathematical Tables and Other Aids to Computation* had become the journal of record of the ACM, but it seemed like the tail was wagging the dog and in 1954 ACM launched its own *Journal of the ACM*. In 1959 *Mathematical Tables and Other Aids to Computation* bent to the wind and renamed itself the *Mathematics of Computation*. Over the next two decades the publication of new tables slowed to a trickle.

Four-figure tables still found a place in school and undergraduate studies, but the advent of the electronic calculators in the 1970s finally put an end to them. I recall an occasion in 1990 when I saw in a thrift store a two-foot-high stack of brand-new four-figure tables. On inspection it turned out they were printed in 1978. I suppose they had been stored for a decade in the publisher’s warehouse and had now been remaindered—with one last stop in a thrift store before finally being pulped. I was powerless to prevent their fate, though I rescued one for my bookshelves for the princely sum of 10 pence.

It is one of the great ironies of computing that in the 1950s what all go-ahead table makers wanted was a digital computer, but within a decade the computer had made both them and their tables obsolete. As a wise man once cautioned, “be careful what you wish for.” ■

References

1. Fletcher, A., Miller, J.C.P., Rosenhead, L., and Comrie, L.J. *An Index to Mathematical Tables*. Addison-Wesley, 1962.
2. Grier, D.A. *When Computers Were Human*, Princeton University Press, 2005.

Martin Campbell-Kelly (M.Campbell-Kelly@warwick.ac.uk) is a professor in the Department of Computer Science at the University of Warwick, where he specializes in the history of computing.

Copyright held by author.



Technology Strategy and Management

Cloud Computing and SaaS as New Computing Platforms

To become an industry platform, vendors must open their infrastructure technology to other product companies.

THERE IS NO doubt that software as a service (SaaS) as well as the more general infrastructure technology that facilitates this type of software delivery and pricing—cloud computing—are becoming new platforms for enterprise and personal computing.^a They compete with traditional desktop or handheld computers (including smartphones) that run applications directly on the devices. We can see all the platform concepts discussed in my previous *Communications* column (“The Evolution of Platform Thinking,” January 2010): direct and indirect network effects at the ecosystem level as well as firms offering infrastructure in addition to products, open versus closed systems, and conflicts of interest emerging between platform leaders and their complementors.

But SaaS and cloud computing rise to the level of an industry platform only when firms open their technology to other industry players, including complementors and potential competitors, rather than simply using the Web as an alternative delivery and pricing mechanism for what used to be pack-



aged software products. Whether SaaS and the cloud competition are “winner take all” markets (like Microsoft in desktop operating systems or the VHS format in home VCRs) or “winner take most” (such as Google in Internet search) remains to be seen. But we can analyze this question through the lens of platform dynamics.

The ideas of SaaS and cloud comput-

ing have emerged gradually.^b In fact, delivering software applications over a network is an old idea but, in the past,

^b The analysis here draws heavily on R. Bhattacharjee, “An Analysis of the Cloud Computing Platforms” (Cambridge, MA: Unpublished Master’s Thesis in System Design and Management, Massachusetts Institute of Technology, June 2009). Also see B. Hayes, “Cloud Computing,” *Commun. ACM* 51, 7 (July 2008), 9–11.

^a This column is based on portions of my forthcoming book, *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Uncertain World* (Oxford University Press, 2010).

has not reached the level of an industry platform. The concept goes back to time-sharing in the 1960s and 1970s, as well as application hosting in the 1980s and 1990s. Then we saw an increasing number of firms in the 1990s and 2000s deliver what used to be packaged software applications from a new platform—the Web—and usually for free. These applications ranged from email to calendars, groupware, online publishing, simple word processing, and many other common consumer and even business applications. Advances in networks as well as virtualization technology have made Web delivery possible regardless of the type of computer the user purchased. But, again, only when vendors open their SaaS or cloud infrastructure technology to other product companies do we have an industry platform.

For example, Salesforce.com created a customer relationship management (CRM) product and configured it not as packaged software but as software delivered over servers and accessed through a browser. When it did this, it created its own in-house platform for delivering the software as a service to its customers. But then it created AppExchange as an open integration platform for other application companies that built products utilizing some features in the Salesforce CRM product. When it did this, Salesforce.com created a new industry platform, or rather a platform wannabe because there are competitors. Salesforce then extended the open platform concept with Force.com, a development and deployment environment using Salesforce's SaaS infrastructure. Amazon (Elastic Compute Cloud, known as EC2) and Google (Google App Engine) also have opened up their SaaS or cloud infrastructures to host outside applications as well as their own productized online services.

By the end of 2008, Amazon was already the most popular general-purpose cloud platform, with over 400,000 developers registered to use its Web services.^c Amazon has become so attractive because it has a rich infrastructure to support online retailing operations and has made these services available to its cloud users—data storage, com-

puting resources, messaging, content management, and billing.

Network Effects

SaaS and cloud computing platforms exhibit *direct* network effects to the extent they have specific application programming interfaces (APIs) or Web services that encourage application developers to tailor their applications or that make it difficult for users of these applications to switch platforms. The direct network effects do not seem as powerful as between Windows and applications written for PCs, or between particular smartphone operating systems like Symbian, Blackberry, or Palm, and applications written for those environments.

The SaaS and cloud programming interfaces and technical standards for exchanging data and logic are usually simple and standardized, relying on the Internet HTTP protocol. But some APIs and Web services are specific to individual SaaS/cloud platforms. For example, many real estate companies or retail shops have built applications that incorporate Google Maps—tying the applications to Google's platform. Other companies have built e-commerce applications using facilities for handling payments provided by Amazon—tying the applications to Amazon's platform.

Cloud and SaaS platforms exhibit *indirect* network effects to the extent that the popularity of one platform over another with developers makes the platform more attractive to other developers or users. As more applications appear on a particular platform, they attract more application developers in a positive feedback loop. SaaS/cloud platform competitors also can try to attract end users by making use of their platforms free, perhaps with funding

The product firms seem to offer SaaS and the cloud as another mode of delivery and pricing.

from advertisements. The application companies generally pay a fee depending on usage, data storage, or some other criteria. Platform vendors can charge high or low fees to attract developers, or make some aspects of their platforms free. This is another version of the “free, but not free” strategy that we have seen in PC and Internet software.

Some firms, such as Salesforce.com (with Force.com and VisulaForce) and Bungee Labs, have taken the SaaS and cloud platform ideas further than just providing an environment to launch applications. They also provide services and program development tools that help companies build new applications within these competing platform environments. Developers can also usually integrate with Web services from other vendors, even though some Web services or APIs may be specific to the platform. This is another version of an “open, but not open,” or “closed, but not closed” strategy.

In short, we see SaaS and cloud platforms appearing in multiple levels: First, we see the general technology of the Internet and virtualization making SaaS technically possible. Then we see companies utilizing this technology to offer SaaS or cloud versions of their products. Finally, we see some firms not only offering SaaS versions of their products (now Web-based services) but opening up their technology to allow other application developers to build and launch applications from these platforms.

SaaS or cloud platforms also appear to be efficient for both users and vendors. Multiple customers can use the same facilities and thereby increase utilization rates for the hardware and the networks. For example, Amazon and Google have enormous data centers that they do not fully utilize. They can launch their own products (automated services) while also hosting applications from other companies, without sacrificing security of the different “tenants.” Hosts such as Amazon, Google, and Salesforce.com generally guarantee security for their hosting customers through detailed service level agreements (SLAs).

Effects for Traditional Software Vendors

On the other hand, there are some negative consequences of the SaaS and cloud platforms for traditional software

^c See <http://gigaom.com/2008/10/09/amazon-cuts-prices-on-s3/>, cited in Bhattacharjee.

product companies and users. Most software product companies today offer Web-based hosted versions of their applications. These in-house SaaS platforms only account for a few percent of sales at big vendors such as SAP, Oracle, or Microsoft, but the demand is rising. SaaS and cloud platforms are especially popular among startup enterprise software companies.^d

The product firms seem to offer SaaS and the cloud as another mode of delivery and pricing. But, in this model, customers usually do not pay separate maintenance and product license fees, which account for about two-thirds of the revenues of the major product firms. Nor do the customers have to deal with customizing versions of the software or migrating patches—which traditionally have generated service revenues for the product companies and account for most of the other one-third of revenues for these firms. That work is all done for them via the SaaS/cloud infrastructure and the one price. There are sometimes issues of performance of applications over a shared, dispersed network. Some enterprise customers are also concerned about security of their data, proprietary knowledge in their applications somehow leaking to competitors, or the SaaS/cloud platform failing.

We also have potential conflicts of interest when application software companies move their products to a SaaS or cloud platform infrastructure for delivery and pricing with their users but also open up their platforms to other application companies whose products are potentially complementary (for example, Salesforce.com). There is less conflict when we have pure infrastructure provisioning from companies that have excess computing capacity on the Web but are not specifically application product vendors (for example, Amazon).

But when a company tries to play both sides of this market, conflicts can occur. For example, Google's App Engine now includes Salesforce.com's APIs as part of its platform. A company can write an application, launch it on Google's App Engine, and use features from Google (such as search or Google Maps) as well as features from Salesforce's CRM prod-

SaaS and the cloud replace some traditional software products but will not eliminate them anytime soon.

uct. However, if Google decides to build its own CRM product, then we have a potential conflict of interest. Salesforce will have to rely on Google to maintain neutrality.

Microsoft, the largest software product company, for years has been preparing (albeit reluctantly) for SaaS and cloud computing as an alternative to traditional packaged software. It created Windows Live and Office Live and has more than a decade of experience with the online MSN network, which delivers content as well as software products and product upgrades. Now, Microsoft has created a cloud platform called Windows Azure that will compete with Amazon and Google. Early indications are that Azure will be relatively neutral to the extent that application developers should be able to use various programming languages and not just the .NET environment. Developers should also be able to incorporate features from other Web services platforms.

But Microsoft is also packaging its own online services and products into Azure and clearly gives preference to its own products and services. For example, everything on Windows Live and Office Live will be available as well as Microsoft SQL Server services, Microsoft CRM services, .NET services, and Sharepoint Services. Microsoft is therefore making it possible for customers to use various Microsoft products as Web services rather than buying the packaged software. Customers should be able to integrate the Microsoft services with products of other vendors, but exactly how open the new platform will be remains to be seen. The other issue for Microsoft is that usage of the Azure cloud will reduce demand for Windows

desktop and servers. Microsoft expects customers as well as some application companies will build applications using the Azure Web services and running the whole system on the Azure platform rather than buying more PCs or servers bundled with Windows.

Conclusion

SaaS and the cloud are clearly new platforms for computing. They replace some traditional software products but will not eliminate them anytime soon. While it is relatively easy for a software product company to create a hosted version of its products, delivering these products over an outside SaaS platform like Amazon, Google, AppExchange, or Windows Azure requires rewriting at least some and maybe most of the code to use the different interfaces and available services.

Although the SaaS/cloud delivery and pricing model has many advantages, it also has disadvantages for product vendors and users. The transition, therefore, is likely to be gradual and partial, as companies create new versions of products that seem well suited to the new platform delivery and pricing models. Similarly, users have many customized applications and data stored in proprietary databases. They would all have trouble switching to a SaaS/cloud platform quickly but surely can do so gradually if the economics make more sense.

Finally, as long as the SaaS/cloud vendors maintain some differentiation among their platform offerings, direct network efforts are not too powerful, and switching is not too difficult or expensive for application developers or users, then we will probably continue to see multiple SaaS/cloud platforms coexist.^e This is what we have experienced in video game consoles (with Sony, Microsoft, and Nintendo) and smartphone operating systems and handsets (with RIM, Apple, Nokia, Palm, Microsoft, and Google). ■

e For the conditions that make and do not make "winner take all" markets, see T. Eisenmann, G. Parker, and M.W. Van Alstyne, "Strategies for Two-Sided Markets," *Harvard Business Review* 84, 10 (Oct. 2006), 92–101.

Michael Cusumano (cusumano@mit.edu) is Sloan Management Review Distinguished Professor of Management and Engineering Systems at the MIT Sloan School of Management and School of Engineering in Cambridge, MA.

Copyright held by author.

d M.A. Cusumano, "The Changing Software Business: Moving from Products to Services," *IEEE Computer* 41, 1 (Jan. 2008), 20–27.

Viewpoint

When Network Neutrality Met Privacy

Incorporating the consideration of privacy into the ongoing debate concerning network neutrality.

WHAT IF THE U.S. Congress passed a new law prohibiting Internet Service Providers (ISPs) from looking at any part of any packet they route except for the destination IP addresses? The law would accomplish three things.

First, the new law would grant users an extraordinary amount of privacy. By keeping prying eyes (and packet sniffers) away from virtually every part of every packet, it would encapsulate all Internet communications within private tunnels, not of encryption but of law.

Second, the law would mandate a form of network neutrality, the now-well-known principle that ISPs should not be allowed to treat packets differently based on application, content, or source.^a For several years, legions of fierce advocates on both sides have debated whether the U.S. government should impose some form of mandatory Net neutrality, with their attention focused lately on the FCC, which is considering such a rule. Despite these years of debate, very few have examined the underexplored relationship between two important Internet values: Net neutrality and privacy.

An ISP that wants to treat packets differently based on application, con-

tent, or source must first peer deeply enough into those packets to determine their application, content, or source. If our hypothetical law prohibits an ISP from examining this type of information, then all packets will look alike and discrimination will be impossible. A privacy-respecting network is a neutral network.

Third, sadly, users would enjoy better privacy and mandatory Net neutrality for a fleetingly short time, because the law would also break the Internet. ISPs have good reasons for peeking at a little bit more than just IP addresses on an Internet full of threats. If they

could not, those threats would strangle the Internet, grinding traffic to a halt under the weight of too many viruses, hackers, and worms.

Fortunately, Congress has never passed such an extreme law, but two decades ago, it enacted a much more measured network privacy law that deserves to be recognized as the world's first mandatory Net neutrality law.

The Electronic Communications Privacy Act of 1986

The Federal Electronic Communications Privacy Act (ECPA), originally enacted in 1986, prohibits the unjust-



Protesters at Net neutrality rally in May 2008.

^a Commentators have defined Net neutrality in dozens of subtly different ways. This column uses the definition stated here in the text.

tified interception of information on a computer network. Violators face both significant civil liability and possible criminal punishment.^{b,c} In rare and extreme cases, the FBI can charge people who use packet sniffers as felony wiretappers.

Historically, ISPs have not worried much about ECPA, because Congress built a number of exceptions into the law permitting traditional forms of provider monitoring. Thus, ISPs can monitor to “protect rights and property,” for the “rendition of service,” and with the consent of their users.

Lately, ISPs have begun to test the limits of these exceptions, engaging in new forms of unprecedented, invasive monitoring designed to generate new revenue sources by monitoring users on behalf of the advertising and copyrighted content industries.¹⁻³ These plans may violate ECPA. Although the first two exceptions listed above are broad, they are not boundless, and ISPs will have trouble convincing courts that behavioral advertising or copyright policing have anything to do with “protect[ing] rights and property” or the “rendition of service.” Likewise, the third exception, consent, is no panacea. Courts interpreting ECPA should be skeptical about claims of user consent based on buried privacy policies, lengthy banners and terms of service full of microscopic type.

ECPA as Mandatory Net Neutrality

If ECPA forbids providers from aggressively expanding their methods of scrutiny, then it will also prevent providers from implementing aggressive new forms of packet discrimination. To take the most prominent Net neutrality squabble as an example, the FCC disciplined Comcast for slowing down and sometimes blocking its users’ BitTorrent connections for violating what the FCC calls “reasonable network management.”^d (Comcast has appealed.)

b 18 U.S.C. § 2511(1)(a).

c 18 U.S.C. § 3121.

d Memorandum Opinion and Order, In the Matter of Formal Complaint of Free Press and Public Knowledge Against Comcast Corporation for Secretly Degrading Peer-to-Peer Applications (Aug. 20, 2008); http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-08-183A1.pdf

Scrutiny without handling does not violate Net neutrality and handling without scrutiny does not necessarily implicate privacy.

What if, rather than complaining to the FCC, some of Comcast’s customers had instead sued Comcast for violating ECPA? So long as Comcast identified BitTorrent traffic by looking only at the TCP port numbers on packets, it probably did not violate ECPA. But if Comcast instead used deep packet inspection tools to capture a vast amount of data—and based on Comcast’s admitted relationship with deep packet inspection vendor Sandvine, it probably did—then it much likelier violated ECPA. Thus, a court considering an ECPA lawsuit might have found Comcast liable for throttling BitTorrent, whether or not this qualified as reasonable network management, as an undue intrusion on user privacy.

This example reveals that federal privacy law has mandated a form of Net neutrality for nearly a quarter of a century. This seems surprising given that the Net neutrality debate itself is barely five years old. Just as surprising, this conclusion takes a central premise of the debate, that we need a new law before we can have mandatory Net neutrality, and flips it on its head. Those who want mandatory Net neutrality already have it; those who oppose it need to convince Congress to amend ECPA.

But Is This Really Net Neutrality?

Net neutrality’s champions need not celebrate too quickly, however, because ECPA overlaps only incompletely with their ideals. Net neutrality focuses almost exclusively on the *handling* of packets. The worst thing a provider can do is block traffic, and

Calendar of Events

April 15-17

ACM SE '10: ACM Southeast Regional Conference, University MS, Contact: H Conrad Cunningham Email: cunningham@cs.olemiss.edu

April 16-17

Consortium for Computing Sciences Colleges (CCSC) Northeastern, West Hartford, CT, Contact: Dr William M Mitchell, Email: willmitchell@lightbound.com

April 18-21

SIGUCCS Management Symposium, Victoria, BC Canada, Sponsored: SIGUCCS, Contact: Charles Patrick Kohrman II, Email: cpk1@psu.edu

April 19-21

Tenth International Symposium on Functional and Logic Programming, Sendai, Japan, Contact: Naoki Kobayashi, Email: koba@eccei.tohoku.ac.jp

April 19-23

International Parallel and Distributed Processing Symposium, Atlanta, GA, Contact: Pitts William, Email: pitts.bill@gmail.com

April 23-24

Consortium for Computing Sciences in Colleges (CCSC) South Central, Austin, TX, Contact: Dr William M Mitchell, Email: willmitchell@lightbound.com

April 24-28

8th Annual IEEE/ACM International Symposium on Code Generation and Optimization, Toronto, ON Canada, Sponsored: SIGMICRO, SIGPLAN, Contact: Andreas I Moshovos, Email: moshovos@eecg.toronto.edu

April 26-30

The 19th International World Wide Web Conference, Raleigh, NC, Contact: Stephanie Smith, Email: smith_s@hq.acm.org



ACM's *interactions* magazine explores critical relationships between experiences, people, and technology, showcasing emerging innovations and industry leaders from around the world across important applications of design thinking and the broadening field of the interaction design. Our readers represent a growing community of practice that is of increasing and vital global importance.

interactions
<http://www.acm.org/subscribe>



Privacy brings in an entirely different frame of reference, one composed of values that have nothing to do with innovation and economic prosperity.

slowing traffic is nearly as bad. ECPA-imposed network privacy—let us call it *Net non-scrutiny*—focuses instead almost entirely on a provider's *scrutiny* of communications. The worst thing a provider can do is capture the contents of communications. Scrutiny without handling does not violate Net neutrality and handling without scrutiny does not necessarily implicate privacy.

Toward Resolving the Net Neutrality Debate

Privacy and Net neutrality are not mutually exclusive, of course. ECPA's mandatory Net non-scrutiny does not preclude lawmakers from mandating Net neutrality as well. To make Net non-scrutiny look a bit more like Net neutrality, Congress could amend ECPA to clarify and restrict exceptions to the law, for example by getting rid of the consent exception for ISP monitoring. In fact, doing this might serve as a substitute for mandating Net neutrality, because Net non-scrutiny might serve as an acceptable compromise solution, perhaps striking a balance that gives Net neutrality's advocates much of what they desire while giving providers the freedom to manage their networks.

Finally, notice how arguing for Net neutrality through Net non-scrutiny alters the terms of the debate. Proponents of neutrality argue mostly about its benefits for innovation and economic growth. Sometimes, they clothe these arguments in the

language of "freedom," but by this they usually mean a narrow, market-drenched conception of freedom. Opponents of neutrality argue also only on economic terms. The debate has taken place too often on insularly economic terms with values lined up on both sides internal to this economic frame. The arguments for and against mandatory neutrality raise especially difficult economic questions, because they require predicting the effect of complex inputs on a complex industry dominated by new technology, and as a result the Net neutrality debate has devolved into a bare-knuckles economics brawl. Neither side has landed a knockout punch, however, and both sides admit that their predictions might be wrong.

Recasting the debate as one about ensuring the proper amount of privacy makes an intractable debate much more tractable. Privacy brings in an entirely different frame of reference, one composed of values that have nothing to do with innovation and economic prosperity. Stacked up against privacy, there is more space between competing visions of ISP behavior: doing X might make it difficult to deploy next-generation video applications, but it will protect user privacy in return.

These are not easy conflicts to resolve, to be sure, but we will find these competing values easier to weigh and contrast, because it is not economics all the way down. We will find it easier to compare the significance of one value versus another, and we will be able to better predict how choosing one over the other will play politically. In this case, there is virtue in comparing apples and oranges. **C**

References

1. Hansell, S. Charter will monitor customers' Web surfing to target ads. *New York Times Bits Blog* (May 14, 2008); <http://bits.blogs.nytimes.com/2008/05/14/charter-will-monitor-customers-web-surfing-to-target-ads/>
2. Stone, B. AT&T and other I.S.P.'s may be getting ready to filter. *New York Times Bits Blog* (Jan. 8, 2008); <http://bits.blogs.nytimes.com/2008/01/08/att-and-other-isps-may-be-getting-ready-to-filter/>
3. Story, L. A company promises the deepest data mining yet. *New York Times* (Mar. 20, 2008).

Paul Ohm (paul.ohm@colorado.edu) is an associate professor of Law and Telecommunications at the University of Colorado specializing in computer crime law, information privacy, criminal procedure, and intellectual property.

Copyright held by author.



Kode Vicious

The Data-Structure Canon

*Data structures are part of the foundation of computer science.
It pays to revisit them from time to time.*

Dear KV,

In most areas of science there are a few basic underlying laws that inform the rest of the study of a given subject. Physics, chemistry, and electrical engineering all have these basic equations. What are the basic equations in computer science? Or is computer science baseless?

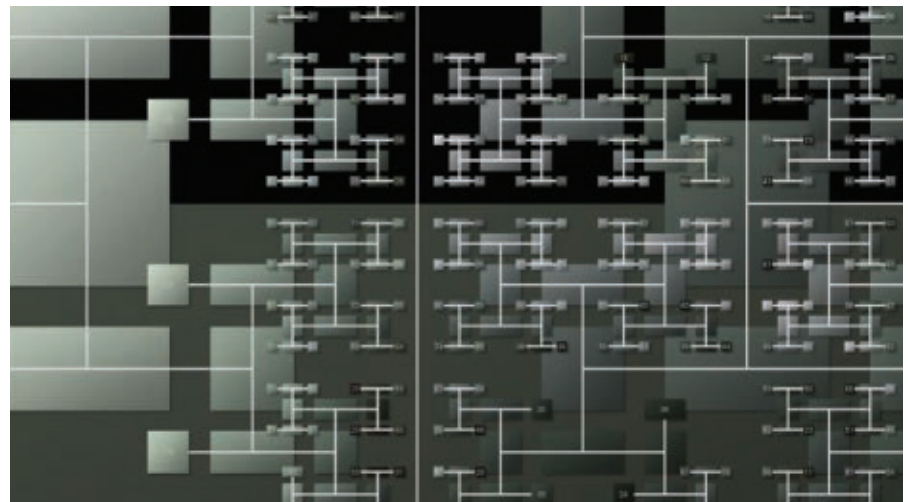
Looking for a Base

Dear Baseless,

While I'm sure that several mathematically minded readers will take me to task for this, what pops into my mind when I read your question are the basic data structures used in programming, rather than a set of equations. It's likely this reflects the way I look at software, which is as a set of structures, or perhaps it just reflects the neural damage I've suffered through reading code written by hundreds of monkeys. I'm not really in a position to say at the moment.

There are certainly plenty of joke equations, mostly having to do with the ratio between quality of code, and cost and speed of its production, but those have never been adequately specified, at least not so that they could be considered physical laws such as Ohm's.

The idea of a software canon—a set of ideas, algorithms, or data structures that are worthy of repeated study—seems at first to be somewhat ludicrous. After all, this is not a church: churches have canons; rational people, such as



software engineers, do not. So, let's not think of these as sacred concepts, merely the basic ones to which we should return on a regular basis so as to regain our footing in the heady world of objects, methods, and algorithms.

If there were any set of basics that I wanted to hammer into software engineers and programmers, it would be the basic data structures, their features and pitfalls. Here, then, is a brief overview of the data-structure canon, which, unfortunately, is metaphorical rather than actual. I mean, no one is going to let you have a real canon (or cannon) in the office—trust me, I've asked.

The basic data structures are the array, list, tree, and hash table. That seems like a short list, and I'm sure some of the more experienced readers of this column are warming up their

mail applications to write me nasty notes about how there are more basic data structures than just these four. To these people I say, "Go for it!"

What I find most interesting about this list is how many programmers take it for granted and how few of them understand the implications of using one data structure versus another. Try as I might, I cannot lay all the blame at the feet of the programmers using these data structures; a good deal of the blame has to go to their teachers and the legions of poorly written "how to program" books that emphasize how to *use* a data structure rather than the underlying implications of what happens when they are used. Abstraction is a fine thing—it makes understanding large systems easier—but it also can obscure important facts: in

particular, the performance implications of certain choices.

The array is not only the most basic data structure used in computer programs, but it is also probably the most abused and misunderstood as well. There are very few languages, other than assembler, that do not have some easy-to-use form of array. The array's salient features are ease of element insertion, retrieval, and deletion, as well as its ability to contain only elements of the same data type. Arrays are most often preallocated, which is to say that once you declare an array of a certain size, that memory is allocated to the array and the array can be no larger or smaller than its initial allocation. The static nature of arrays leads to all kinds of problems, some of which are hidden by tricks such as creating vectors, which are really glorified arrays that are reallocated behind your back when they grow.

One of the array's hidden weaknesses is that, when used as a stack variable, it can take a toll on the performance of an application, particularly if it is used as a dumping ground for the program's data. One programmer I worked with would allocate multiple-megabyte arrays on the stack, never thinking of the extra overhead this incurred, nor that his code would stop functioning on a system that was not a huge honking server.

Of course, you might argue that everyone now has a huge honking server and that stack or system memory just doesn't matter; but amusingly enough for you—not for those of us who worked on this code—it was eventually ported to an embedded device. The funny thing was that it crashed, pretty often, and debugging embedded devices is not very fun when they're located in a remote environment where they're supposed to run unattended for years, rather than crashing unattended after only a few days. Do not shove everything into an array just because you can, and don't put big arrays on the stack just because you can. Studying how arrays are implemented on your system or in the language you use can definitely pay off in the long run, and relearning this knowledge from time to time seems to be required for just about everyone.

Lists are what people use when they think they're too clever to use arrays. Many modern languages have built-

Do not shove everything into an array just because you can, and don't put big arrays on the stack just because you can.

in support for lists, and one ancient language was built seemingly just to process them, as if once you had a screwdriver (list), you could make every problem look like a screw (list to process). Lists are more expensive than arrays when you're inserting, finding, or removing elements, but they do have the advantage that they need be no longer than the sum of their elements.

The most (actually least) fun thing about lists is that everyone and his brother thinks they should create their own macros, class, or library to handle them. A common assignment for burgeoning computer programmers is to have them create their own list code as a way of teaching them about pointers, when they are still taught about such things. Unfortunately, the experience of having created their own list implementation, and having it actually work, leads them to think that theirs is the best implementation of a list that ever was. They show it to all their friends, post it on the Internet, or perhaps create an open source project around it. The fact is the list classes already in existence are likely good enough for most people. If you think they're not, then test yours with a reasonable benchmark, the creation of which will tell you more about what you know about how lists work than writing a new data structure will.

Trees are for people who have mastered, or think they have mastered, lists. A well-tended tree generally has a lower insertion, deletion, and retrieval time than a similar-size list because of how searches are performed on trees versus lists. The problem with a tree is knowing how to tend one. Some

programmers seemingly have black thumbs, which means their trees are simply lists where the head is named "root." They don't consider how the data they're putting in the tree will be organized once their list insertion routine returns. I would be much happier if people who decide to use or implement tree structures would consider how the data in them is organized.

Last, we come to the hash table. The purpose of a hash table is to give you a flexible data structure that is faster to access than a list or a tree. Here, the biggest challenge is selecting or designing the hashing function. Choose the wrong hash function and you'll wind up with a list (see the discussion of trees in the previous paragraph). I would tell you that you should just use a well-known hash-table data structure and leave it at that, but, again, the quality of the hash depends on the data. Some hashing functions are good for some types of data and some are good for others, and so you have to pick your hash carefully.

Given the different complexities I've just mentioned, I think you'll see that these four data structures, which form the basis of most computer programs, are worth studying time and again. A good place to start is Knuth's *The Art of Computer Programming, Volume 1*, Chapter 2, which covers all of these subjects more seriously and in depth. When you're done reading all that he has to say, please write me, though I should be retired by then.

KV

Related articles on queue.acm.org

Advice to a Newbie

Kode Vicious

<http://queue.acm.org/detail.cfm?id=1242495>

Kode Vicious Bugs Out

Kode Vicious

<http://queue.acm.org/detail.cfm?id=1127862>

API Design Matters

Michi Henning

<http://queue.acm.org/detail.cfm?id=1255422>

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and a member of the ACM *Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

Copyright held by author.



Association for
Computing Machinery

Advancing Computing as a Science & Profession



You've come a long way. Share what you've learned.



ACM has partnered with MentorNet, the award-winning nonprofit e-mentoring network in engineering, science and mathematics. MentorNet's award-winning **One-on-One Mentoring Programs** pair ACM student members with mentors from industry, government, higher education, and other sectors.

- Communicate by email about career goals, course work, and many other topics.
- Spend just **20 minutes a week** - and make a huge difference in a student's life.
- Take part in a lively online community of professionals and students all over the world.



Make a difference to a student in your field.

Sign up today at: www.mentornet.net

Find out more at: www.acm.org/mentornet

MentorNet's sponsors include 3M Foundation, ACM, Alcoa Foundation, Agilent Technologies, Amylin Pharmaceuticals, Bechtel Group Foundation, Cisco Systems, Hewlett-Packard Company, IBM Corporation, Intel Foundation, Lockheed Martin Space Systems, National Science Foundation, Naval Research Laboratory, NVIDIA, Sandia National Laboratories, Schlumberger, S.D. Bechtel, Jr. Foundation, Texas Instruments, and The Henry Luce Foundation.

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

What can be done to make cooling systems in data centers more energy efficient?

BY ANDY WOODS

Cooling the Data Center

POWER GENERATION ACCOUNTS for about 40% to 45% of the primary energy supply in the U.S. and the U.K., and a good fraction is used to heat, cool, and ventilate buildings. A new and growing challenge in this sector concerns computer data centers and other equipment used to cool computer data systems. On the order of six billion kilowatt hours of power was used in data centers in 2006 in the U.S., representing about 1.5% of the country's electricity consumption. Of this power demand, much more than 20% is typically used for cooling the computer equipment, but some newer installations have managed to reduce consumption through a series of innovations in the

design of data-center cooling systems, as well as improvements in the software and hardware.

The need to control power consumption is of increasing importance as computing power grows and large data centers house increasingly dense arrays of servers. A number of different systems can be adopted to provide cooling for these large, energy-intense buildings, and opportunities continue to emerge for improving the energy efficiency of cooling schemes.

This article reviews some of the generic approaches to cooling and identifies opportunities for further innovation. In assessing the energy demand for cooling, there are both external considerations, relating to the climate and the associated heat losses between the building and the exterior, and internal considerations, relating to the method of cooling to be adopted, and the associated challenges of achieving efficient heat exchange.

Reference Model

In the models and simple calculations presented here, I have adopted some simplified assumptions about power generation in data centers, although the numbers can be scaled to different system sizes. The engineering design of servers includes both the horizontal rack server and the vertically stacked rack servers, which can have energy densities of 10 kW per square meter of floor area. Hence, for a 4,000-square-meter data center, this represents a tremendous heat load up to 350GW hours per year.

Two approaches can be taken to cool the servers: water or air. Air has the advantage that it can be passed directly through the space containing the equipment, and hence may be easy to circulate. Unless the airflow path is targeted to the regions of high heat production, however, air-cooled systems will likely involve spatial variations in temperature. Therefore, to maintain the equipment at the correct operating temperature, the main volume of air will need to be somewhat colder. Of-



CHILLED WATER SUPPLY

CHILLED WATER RETURN


SUPPLY

RETURN


ten there is not only a main supply of air to the space, or racks of servers, but also small fans adjacent to equipment with high heat-generation rates that drive large volumes of the main-space air over these specific pieces of equipment. The generation of temperature gradients in the space will still tend to happen if the servers are arranged in clusters. This can lead to a less efficient cooling system if the exterior temperature is higher than the temperature required in the main space; the inefficiency arises from the increase in heat transfer through the fabric of the building as the interior temperature is reduced, which in turn requires additional cooling.

There have been numerous attempts to provide cooling that is targeted more directly to the hot equipment, resulting in smaller temperature gradients. One approach to help achieve such targeted cooling is through the use of water as the heat-exchange fluid since it has a large specific heat and very high density relative to air, thus requiring much less volume flux than air per unit of heat flux being transferred. By locating a cooling coil adjacent to hot equipment, the water can directly convect away the heat load. If the coil has some cooling fins, air passed through the fins will also be cooled by the water circulating through the coil. This provides the opportunity for hybrid cooling, whereby small fans may blow chilled air from the chilling coil to the remainder of the equipment, while the water coil itself is responsible for the heat exchange with the high-heat load equipment. One caveat about water heat exchange, however, is that it requires careful construction so as not to damage the IT systems in case of failure or leakage.

Simple considerations illustrate the typical volume flow rates of air through the system. For example, the American Society of Heating, Refrigerating, and Air Conditioning Engineers suggests that servers be supplied with cooling air in the temperature range of 20°–25°C (68°–77°F), while the heated air vented from the systems may be as warm as 35°C (95°F). A heat load of 10kW per square meter requires an airflow rate of 1–2 cubic meters per second per square meter of the floor area. Scaling this up to a 4,000-square-meter data center would require an air-circulation rate



There are a number of energy-efficient ways of rejecting the heat from the building. One critical issue to consider is exchanging the interior air directly with the exterior air.



of 4,000–8,000 cubic meters per second. This air needs to be directed carefully through ducts or plena to ensure there are no recirculation or stagnant regions that will then overheat. With a water-cooling system, the specific heat is much higher, so for the same temperature change we would need only 1–2 cubic meters per second of water distributed through the 4,000-square-meter data center. Here the challenge is in distributing this water effectively throughout the system to limit the possibility of overheating; this might involve a novel rack design with built-in cooled water panels, for example.

Once the air or water has passed through a part of the server system and heated up, a heat-exchange or heat-rejection system needs to transfer this heat flux out of the data center and provide a new source of cooled air or water, which can then circulate through the server system again. Let's now explore some possible approaches for rejection of this heat flux from the data center. First I assess designs for air-to-air cooling, assuming that either the system is air-cooled or the water is cooled by a water-to-air heat-exchange system within the building, with this heat then rejected with an air-to-air device.

Air Cooling: Climatic Challenges

There are a number of energy-efficient ways of rejecting the heat from the building. One critical issue to consider is exchanging the interior air directly with the exterior air; owing to the challenges of dust removal and humidity conditioning of the air, it may be that such direct exchange of air is restricted; air-to-air heat transfer between the interior and exterior may then be achieved using heat exchangers. If external air can be brought directly into the data center, then the heat exchanger can be simplified to, or combined with, the direct exchange of air; the merits of heat exchange or air exchange compared with refrigeration/heating lies in the different energy costs for pumping air through a space or heat exchanger compared with the energy required to cool the interior air using a refrigeration cycle.

Noting the complexity of temperature gradients within the data center, we assess the ideal situation in which the cooling air is able to access directly

the regions of high heat load, thereby minimizing the large temperature gradients that can develop. One key issue relates to the geographical location of the data center and the local climate. A building located in a region with a temperate climate may have considerable energy savings available compared with a building in a hot climate in which the temperature exceeds the desired temperature of the inflow air.

In essence, if the external temperature is lower than $20^{\circ}\text{--}25^{\circ}\text{C}$ ($68^{\circ}\text{--}77^{\circ}\text{F}$), it may be possible to use a direct heat-exchange system to reduce the temperature of the interior air, which is recirculated through the space by passing through a heat exchanger also connected to the external air (see Figure 1). Such direct heat exchange can provide an effective means of cooling the air, since the heat exchanger requires only mechanical power to drive the air through the system. Furthermore, if the external air can be brought directly into the interior, some of the mechanical work associated with driving the air through a heat exchanger (for example, a thermal wheel) can be reduced, although in cold external conditions, it may be necessary to preheat the incoming air to a comfortable or acceptable temperature. If this is not achieved through a heat exchanger with the heated interior air, it can be achieved by direct mixing with the interior air.^a

If the exterior air temperature rises above the temperature range, $20^{\circ}\text{--}25^{\circ}\text{C}$ ($68^{\circ}\text{--}77^{\circ}\text{F}$), then the recirculated air may need some direct cooling through a heat pump or other refrigeration system. As long as the exterior temperature is not in excess of the temperature of the hot outflow air, however, it may be possible to use direct heat exchange to reduce the temperature of the air in the space and thereby complement the use of the heat pump, which would then provide the last phases of cooling (Figure 2). The seasonal fluctuation in the temperature of the exterior can therefore be enormously significant, since it limits the range of conditions for which direct heat exchange rather than refrigeration is required. In a predominantly winter climate, the direct heat-exchange approach may be very effective and can lead to an efficient sys-

^a A. Woods et al. *Energy and Buildings*, Elsevier, 2009.

Figure 1. Schematic of a direct heat-exchange system for cooling a data center in winter.

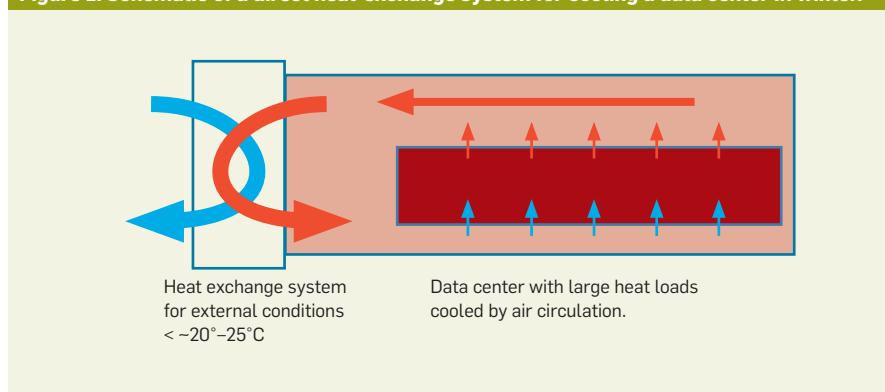
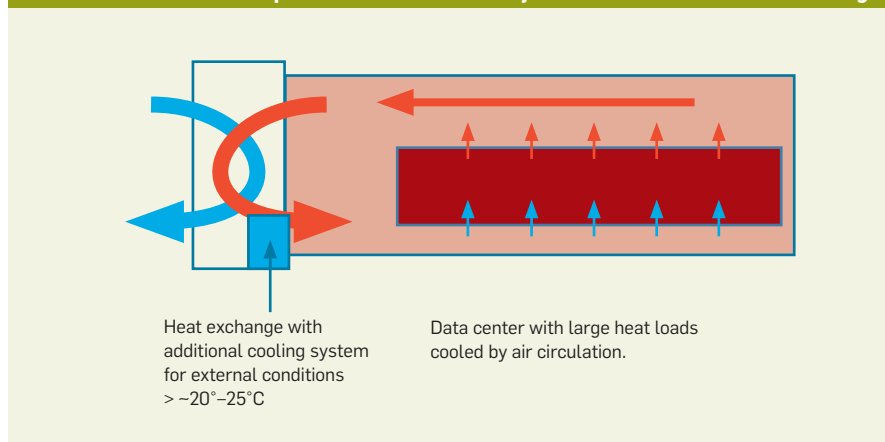


Figure 2. Schematic of a hybrid heat-exchange system, using some refrigeration in mild summer conditions to complement the direct heat rejection in the air-to-air heat exchanger.



tem, which may need a backup cooling system only for very hot days. One of the challenges of such a system is the capital cost of including both a heat pump and a heat-exchange system, although the operating costs can be substantially reduced if the heat pump is used for only a small fraction of the year.

For example, the climate in Berkeley, CA, typically has only 330 hours when the temperature exceeds 77°F (25°C), as seen in Figure 3; and 730 hours with a temperature in excess of 72°F (22°C), for which a simple heat exchanger would need supplementary cooling capacity. Since data centers typically run continuously, this represents less than 8% to 9% of the year, and so a direct heat-exchange system (Figure 1) may have considerable merit.

With a direct heat-exchange system, the interior air transfers its heat to the exterior, for example, by passing through a series of parallel plates for air-to-air heat exchange or by using a rotating heat-exchange wheel. There is then no thermodynamic work required for cooling, although there is mechani-

cal work driving the air through the heat-exchange system. In contrast, in hotter conditions this approach would make it possible to some exchange heat with the exterior, but then additional cooling would be required to bring the temperature of the inflowing air to $20^{\circ}\text{--}25^{\circ}\text{C}$ ($68^{\circ}\text{--}77^{\circ}\text{F}$).

In an air-to-air heat exchanger the main energy consumed is in the work done to drive the air through the heat exchanger. Such heat exchangers can be relatively efficient. For example, a rotary-wheel heat exchanger may involve a pressure loss of about 200–300 Pa (pascal), and with a flow rate of 4,000 cubic meters per second and hence many such heat exchangers, this would involve a work load of about 0.8MW–1.2MW (or perhaps more), depending on the efficiency and number of fans (see Figure 4).

This would then allow for cooling loads of 10MW to be achieved using about 10% of this cooling load in mechanical work for the heat exchange—although it is important to recognize there would be additional losses in

the overall efficiency of the system, perhaps associated with losses in ducts and with the efficiency of the air-pumping system. It is worth noting that we would expect this headline energy-consumption figure to be greater than but of comparable magnitude to the energy required to supply and mix air directly from the outside in the case that direct air-exchange ventilation is possible without a heat exchanger.

Given the inefficiencies in practical implementation of this approach, we may expect the energy cost of direct heat-exchange cooling to be in excess of 10%–20% of the cooling load; this can be compared with the energy loss in a cooling device—for example, a heat

pump, which might operate with a COP factor of 1:4 or 1:5, representing about 20%–25% of the cooling load (Figure 5). This system of direct mechanical air recirculation, with heat rejection to the exterior through a heat-pump or chiller system, will need to be used when the exterior is warmer than the interior (that is, $> 25^{\circ}\text{C}$, 77°F), so that direct exchange is no longer viable. In running a refrigeration system, there is a balance between the amount to which the air temperature is reduced and the flow rate of the air, so as to achieve the same overall cooling flux. The optimal balance between the reduction in temperature and the flow rate depends on the energy required by the fans, which

increases as the flow rate increases, and the efficiency of the cooling heat pump, which typically decreases as the temperature difference across the heat pump increases. By optimizing the couple system, the most energy-efficient mode of operation of the mechanical cooling may be found.

These simple observations, based on the external climatic conditions and the internal heat loads, identify the main challenge in data centers to be the circulation of large volumes of air (or cooling fluid) to remove the heat; this uses a considerable amount of energy, and in hot conditions, the need to chill the recirculated air in addition to the pumping work increases this energy load considerably.

The typical savings that may be achieved by adopting the above ideas, relative to use of a refrigeration cycle throughout the year, are possibly very substantial; using direct heat exchange when external conditions are colder could represent a large percentage of savings in the energy for cooling. The ideas presented here may be able to reduce that energy consumption substantially, but this depends on the ambient temperature; some schemes that adopt this general approach have been installed, including those by Intel and the KyotoCooling system, but there seems to be tremendous potential for much more widespread adoption of the design principle.

This analysis, however, does point to the need to locate data centers in cooler climatic zones, where the exterior temperature may allow for direct heat exchange for a greater part of the year. Indeed, as the number of hours for which cooling is required increases, the energy consumed in the data center increases. If the number of hours increases by a fraction x of the total year, then the energy consumption may increase by an amount on the order of $0.2x$ of the total energy consumed in powering the IT equipment, assuming the use of direct cooling accounts for about one-quarter of the energy consumption. This points to the benefits of locating data centers in more northerly or colder climatic zones.

Interior Design Considerations for Air Cooling

Let's now turn to the design of the in-

Figure 3. Example data on the number of hours the temperature exceeds different values per year in Berkeley, California.

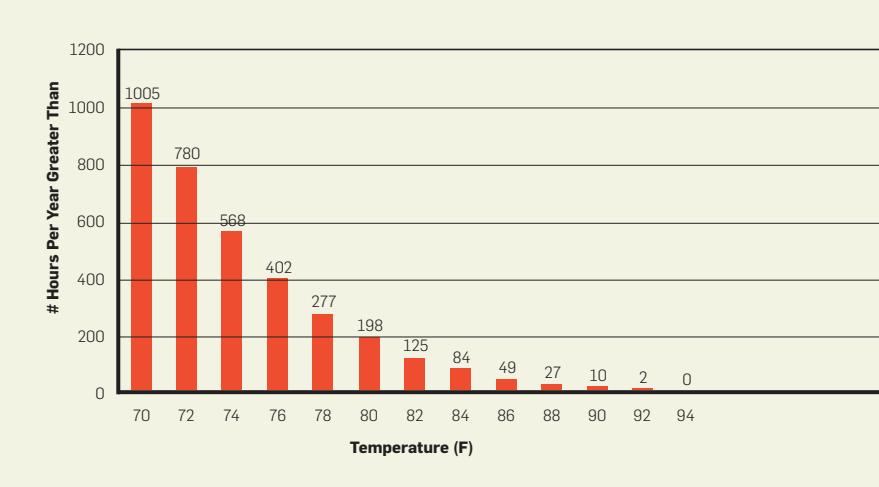
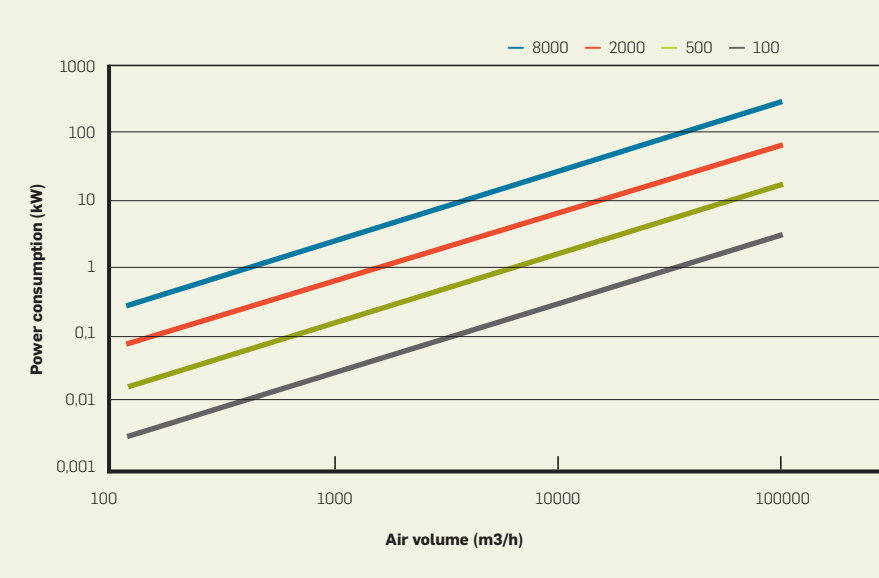


Figure 4. Illustration of the fan power required for a given flow rate of air through a building. This is relevant for the power consumption in a direct heat-exchange system, where the power to drive air through the heat exchanger is the main load on the system.



terior to help achieve more energy savings. One of the main challenges within the data center is the distribution of cooled air to achieve the cooling of each server. As noted earlier, if not all of the chilled air reaches the equipment with high heat load, then temperature gradients become established as heat is transferred from this equipment to the main airstream; although in equilibrium the outflowing mixed air stream will be the same, the equipment may be hotter than in a “well-mixed” model for a given temperature. The main challenge with such stratification is that in order to keep the equipment within the desired temperature, the surrounding air needs to be cooler than the desired equipment temperature. In hot external conditions, maintaining the main space at lower temperatures, in order to keep the equipment at the required temperature, may lead to overcooling; this is because the heat gains to the space through the insulated envelope of the building will increase as the interior temperature falls.

Depending on the heat transfer across the walls and ceiling, with 10°C (50°F) of additional cooling, this may lead to an additional heat gain of up to 20–30 watts per square meter of the data center. In a 4,000-square-meter data center, this represents nearly 0.1MW of power for cooling. Although this is only a small percentage of the total load, it is nonetheless significant. In addition, any air that is exchanged with the exterior for ventilation purposes will need to be cooled a further 10°C (50°F), again increasing the cooling load.

Often underfloor air-distribution systems are used to provide a uniform supply of air through the equipment being cooled. One approach is to supply the air through a “cold” corridor and extract the hot air from the adjacent “hot” corridor, between successive rows of servers. This creates, therefore, a cross-flow across the rows of cooling equipment, transferring heat to the circulating air; the idea is to recycle air through the cooling plant below the floor. This configuration, however, sets up temperature gradients.

Figure 6 shows a data center with rows of servers, with cold air entering through the floor and passing through the racks of servers and then exiting through the next hot corridor back to

Figure 5. A schematic approach to the cooling of a data center using an air-to-air refrigeration cycle.

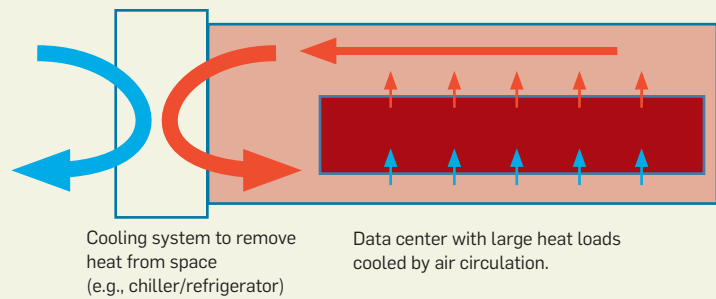


Figure 6. Schematic of an underfloor cooling system.

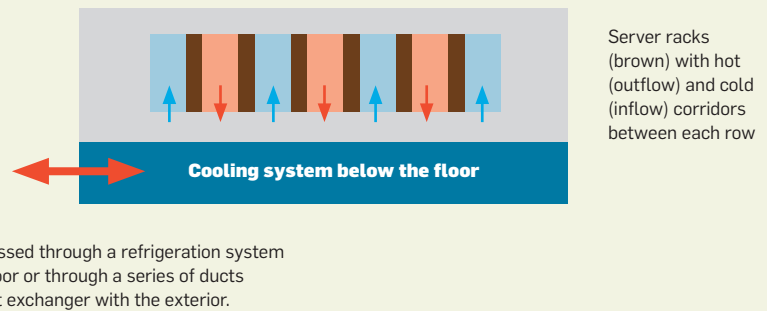
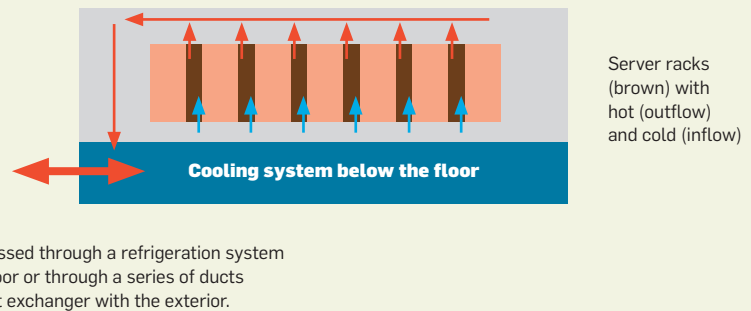


Figure 7. Schematic of a pure upflow system with a cooling unit in the floor of the data center.



the cooling system below the floor. The hot return air may be cooled by a refrigeration plant located below the floor, or by using a direct heat exchanger system with the exterior air, depending on the external temperature, as discussed earlier. The yellow arrow in Figure 6 shows how the heat-exchange system vents the heat to the exterior, perhaps through a wet-coil system.

This design, which involves a series of inflow and outflow corridors, is already in use in some data centers, but


there may be opportunities to enhance the energy efficiency of the system.

One challenge is that the flows are running against gravity in the hot corridors if the air is extracted through the floor, and any mixing of the hot and cold air zones then requires an even greater flow rate past the servers to avoid overheating. One resolution of this problem may be to supply air at a low level to the server rack and extract directly above the top of the server rack. The air can then be ducted into


the refrigeration unit and resupplied to the space (Figure 7). This can lead to substantial stratification in that the air entering the equipment racks is cold, while that leaving at a high level has been heated by heat exchange; such a configuration is beneficial from an energy perspective, since the cold air passes over the hot equipment and warms up, venting from the space at a much higher temperature. The vertical extent of the heat load requires careful limits, however, so that the equipment at the high level does not become unsustainably hot owing to the ascent of heat from the lower level.

This design reduces the chance of heating the cold-supply air, except by heat exchange from the server, and hence can lead to reduced airflows. Ideally, this system would be combined with the direct heat exchange to the exterior (figures 1 and 2) so that the refrigeration unit is used as little as possible for cooling (such as, only in very hot external temperature conditions).

Another area for potential enhancement of the energy efficiency is the geometrical arrangement of the equipment itself, and the associated control of the use of the machines depending on the computing load. One of the key messages stated earlier related to the development of temperature gradients between the heat-generating equipment and the air in the main space; this situation may require lowering the main-space temperature to maintain the heat transfer from the equipment so it can operate within the correct range of operating temperatures. Stacking and localizing equipment may enhance the development of such temperature gradients or may require larger local air circulation rates. Using an upflow displacement scheme and distributing the equipment that is generating heat across the floor space will minimize the buildup of such temperature gradients, and, hence, either the degree of cooling required at any time or the flow rate of the cooling air through specific supply pathways. The integration of the server switching and virtualization software with the energy efficiency of cooling the system offers the potential to develop strategies of hardware use that can lead to substantial savings in the cooling loads and the airflow-rate requirements for a given




In an energy-efficient world, the primary use of the servers would be minimized so as to reduce the primary heat generation.



activity of the hardware, especially in conditions where some latency exists in the data-center hardware capacity.

Summary

This simplified picture of computer data centers has illustrated the considerable variation in energy performance of the center based on the design of the cooling system for the servers. In an energy-efficient world, the primary use of the servers would be minimized so as to reduce the primary heat generation. That is a software challenge that can be addressed using virtual server technology and other approaches to minimize the use of computing resources without compromising on the speed of the system. Given a heat load, however, it is clear that with careful design the convective flow patterns of the air in the data server centers, coupled with the use of a hybrid heat-exchange/refrigeration system can substantially reduce the additional energy required to cool the servers in operation.

In addition to the design of the direct heat-exchange system and the fans, there are many options for optimizing the energy efficiency of the refrigeration system, including the use of ground-source heat pumps and air-to-air heat pumps. Given the large expansion in data centers for Internet and bank service providers, optimizing the design of their cooling systems has the potential for major energy savings, as well as significantly reducing the operating costs of the centers. 

Related articles on queue.acm.org

A Conversation with Steve Furber

<http://queue.acm.org/detail.cfm?id=1716385>

Power-Efficient Software

Eric Saxe

<http://queue.acm.org/detail.cfm?id=1698225>

Powering Down

Matthew Garrett

<http://queue.acm.org/detail.cfm?id=1331293>

Andy Woods is head of the BP Institute at the University of Cambridge, England. His research interests lie in developing mathematical and experimental models of fluid flow and heat transfer in natural systems, including energy efficiency and natural ventilation in buildings and the thermodynamics of heat pumps, as well as the dynamics of flows in porous rocks for geothermal power generation, oil and gas production, and carbon dioxide sequestration.



The battle is bigger than most of us realize.

BY MACHE CREEGER

CTO Roundtable: Malware Defense

AS ALL MANNER of information assets migrate online, malware has kept on track to become a huge source of individual threats. As security professionals close off points of access, attackers develop more sophisticated attacks in a continuously evolving game of cat and mouse. Today, profit models from malware are comparable to any seen in the legitimate world.

But there's hope. Some studies have shown that while 25% of consumer-facing PCs are infected by some sort of malware, the infection rate of the commercial PC sector is around half that rate. This difference is most likely a direct result of the efforts of security professionals working in commercial sites to defend against these threats. This CTO Roundtable panel—our fourth—is split between users and vendors. During the course of our conversation on malware defense, we plan to educate readers about the scope of the malware threat today, the types of frameworks needed to address it, and how to minimize overall risk of breach.

Leveraging the academic roots and

vendor-neutral focus of ACM, CTO Roundtables help define and articulate general best-practices consensus on newly emerging commercial technologies. Through our panels, we provide practitioners with valuable access to objective and unbiased expert advice on what folks should and should not focus on in the next one to three years.

—Mache Creeger

Participants

Michael Barrett is the CISO (chief information security officer) of PayPal.

Jeff Green is head of the threat research unit at McAfee Lab, where his role has been unifying the siloed research around email, Web site, spam,

phishing, and malware to support businesses around remediation management, patch management, risk, and compliance.

Vlad Gorelik is vice president of engineering at AVG Technologies. He is responsible for what AVG calls behavioral chronologies—nonsignature malware defenses. He has been working on nontraditional malware defenses for over five years.

Vincent Weafer is vice president for security response at Symantec. His group focuses on malware defense, handling phishing, frauds, threat intelligence, URL data feeds, and reputation.

Opinder Bawa is the CIO for the University of California, San Francisco (UCSF) School of Medicine.

Steve Bourne is CTO at El Dorado Ventures, where he helps assess venture-capital investment opportunities. Prior to El Dorado, Bourne worked in software engineering management at Cisco, Sun, DEC, and Silicon Graphics. He is a past president of ACM and chairs both the ACM Professions Board and the ACM *Queue* editorial board.

Mache Creeger is the moderator for the CTO Roundtable series. He is the principal of Emergent Technology Associates providing marketing and business developing consulting services to technology companies focused on enterprise infrastructure.

CREEGER: Let's start off the discussion by providing an assessment of the malware threat as you see it today.

WEAFER: The past 12 months in the malware-threat landscape have been a natural evolution of the past couple of years. We have seen a huge explosion in the volume of new malware. We've also seen evolution in terms of the sophistication of malware, new data-mining techniques, and new methods of self-protection that have really changed the threat landscape. The attacker's ability to get smarter tools easily and use them faster with less technical skill has changed a lot of what we're seeing.

We are not looking at a single pandemic threat but a huge explosion of individual threats. What you get on one machine is completely different from what you get on another machine. Each infection has a unique signature. You're served up a unique piece of mal-



VINCENT WEAFER

We have seen a huge explosion in the volume of new malware. We've also seen evolution in terms of malware sophistication, new data-mining techniques, and new methods of self-protection that have changed the threat landscape. The attacker's ability to get smarter tools easily and use them faster with less technical skill has changed a lot of what we are seeing.



ware that may not be seen by anyone else in the world. Threats have gone from global to local to personalized.

GORELİK: I agree. You are also seeing automated tools for distributing malware: for finding sites that are vulnerable, attacking them, and turning them into distribution sites. Once you start automating, it becomes a much broader distribution model. You start seeing ways to reach many more people in one shot and more innovative profit models and social engineering techniques. These models are fairly sophisticated and on par with marketing seen in the legitimate world.

BARRETT: About four years ago, when I was at American Express, I asked the question, how many desktop PCs on the Internet have been compromised? It was surprisingly hard to get an answer, as industry does not seem to track that issue. About a year ago, I succeeded when I talked to some folks at Georgia Tech. Based on their research, they believed the number was almost exactly 12%.

What's interesting about this is if you talk to consumer-facing ISPs, the numbers are more in the 25% range. The difference between the total number and the consumer-facing segment suggests that field-security practitioners working in the nonconsumer sector are actually having an impact.

BAWA: A lot of malware that we see is twofold. We see general malware, things on our computers that are trying to capture information such as a Social Security number and send it to some collection point. These are more generic, general attacks at a possibly superficial level. We also find a lot of very specific, custom software components that will say, "If they exist, go through these databases, scan for this information, and send me these two million transactions."

In the medical industry we see a complete set of problems—from downloading movies, to targeting databases for Social Security numbers, and the latest trend, which is stealing medical information.

If someone stole 100 Social Security numbers, that person could sell them on the street for around \$3 to \$4 each. If someone steals a person's medical information, the thief can sell it for thousands of dollars to someone in the



Steve Bourne



Mache Creeger



Vincent Weafer



Vlad Gorelik



Opinder Bawa



Jeff Green



Michael Barrett



The CTOs

United States who needs to see a doctor to get an operation.

GORELIK: This demonstrates that malware is a business. At the end of the day, these guys are there to make money. If they see high-margin information they can steal for leverage, they're going to go after it. As the market gets saturated by things such as Social Security numbers and credit card numbers, their values drop, so people move to higher-value targets.

WEAFER: If I can combine your physical address with your financial information or email address, it's more valuable to the attackers. Refining content from low-value information that is raw and without context to high-val-

ue specialized content is definitely a growing trend.

While we as vendors are seeing an increase in both sophistication and volume, paradoxically user awareness is dropping. Apart from Conficker, which has gotten a lot of attention, the opposite has occurred over the past few years. The perception is that the malware and spyware problems of the early 2000s have gone away and things are safer. You are seeing people with less awareness and with the perception that they don't need to worry. The challenge is how, in a nonalarmist way, to keep people aware of the dangers.

BARRETT: There is good data suggesting that a lot of people think their PCs

are protected when they are not. They either mistakenly assume that because a PC came with a preinstalled AV (antivirus) product, its protection lasts a lifetime, or worse, they think they have paid for a subscription to update it and have not. That turns out to be quite a systemic problem.

GORELIK: In fact, it is even nastier. We are seeing quite a bit of fake antivirus protection being pushed out. It looks real and sometimes even has a small AV engine with a small set of signatures. In reality, though, it's also infecting the host machine.

BAWA: Older institutions have machines that run Windows 2000, Windows 95, and Windows 98. They are

typically in laboratories connected to specialized equipment and are not easily upgraded. Generally, there is a low awareness of the implications of that vulnerability.

CREEGER: What can people do to avoid these threats?

BAWA: SCO (Santa Cruz Operation) focused on the SMB (small to medium business) market. SMBs with fewer than 50 employees usually have an IT provider they trust—not a national chain but just a local mom-and-pop shop. At more than 50 employees, companies hire a system administrator/IT manager/IT person. That person usually tries his or her best, but has a very difficult task keeping everything running and updated and often is not effective. The segment between 50 and about 250 employees is not being well served. Companies with fewer than 50 or more than 250 employees have viable options, but inside that range, companies have very limited options for effective IT support.

GREEN: We just did a broad survey of small to medium companies. The results clearly stated that in this segment, the average IT administrator spends less than one hour a month on security.

CREEGER: Aren't people afraid? Since folks don't see the direct results of poor security in their face all the time, is malware more of an insidious type of threat today?

WEAFER: With the silent nature of these current attacks, many people are obviously unaware of the risk or damage to their personal data. Moreover, there are many more who just don't care. They can still open their documents, still effectively work, so they choose not to worry. There is certainly a lack of awareness with regard to some of the newer attack techniques. There's the perception that what I don't see won't hurt me.

CREEGER: You're saying that malware writers are getting more sophisticated. They've learned that if they minimize the direct impact to the computing platform so that the effects of their attacks are not "in your face," then they can get much more value from those machines.

WEAFER: Absolutely. I call it slow and low. If you attack a machine slowly and silently, you'll extract maximum profit

over a longer period of time than doing something aggressively, being seen, and then being stopped quickly.

BARRETT: At PayPal we have concluded that it isn't just malware that is the problem; the average Internet consumer has had no training on what represents safe behavior. We have to get out and help teach them that. We have concluded that putting words on Web pages is a fairly poor educational medium. We try to simplify our customer safety messages and take them to where consumers hang out.

CREEGER: When I talk to other experts, they say it goes way beyond that. If you visit questionable Web sites, shady areas of less than ethical intent—such as pornography or pirated software (warez) sites—you don't have to execute object code from some random email message or visit a Web site from a link provided by some phishing email. The fact that you've just visited the site is enough to get an infection.

BARRETT: That may be true if you have not kept your PC patched. In our set of messages around PC safety, we say three things:

- ▶ Run a modern operating system. If you use Windows 95 to surf the Internet and visit one of those Web sites, you have a very good chance of being compromised.

- ▶ Keep those updates turned on. Not being up to date on operating-system patches was the mistake made by all the Conficker victims.

- ▶ Also run an up-to-date antivirus program with the latest security updates.

Do those three things and you substantially lower your risk of infection. Are you 100 percent safe? No, but you're not 100% safe crossing the street either.

BAWA: I have a different perspective. There are some things users can be taught and there will be a real change. Generally speaking, security is not one of them. If we're counting on users to do the right thing at all times, it's never going to happen. Very simply, users expect that the operating system and the environment are secure and that they are protected by antivirus software that comes with malware protection. They believe protection is what they purchased and if it's not doing the job, they are not getting what they paid for.

BARRETT: I'm sympathetic to your point of view, but would say that societies can choose what represents safe and unsafe behavior and can legislate against deliberately unsafe behavior. As a society, we believe it is important to drive safely on the road. If you go flying down a busy city street at excessive speed and kill somebody, you will probably go to jail for vehicular homicide.

There are no intrinsic reasons why we can't impose limits on unsafe behavior that actually jeopardizes others on the Internet. The most insidious thing about malware is that it isn't a threat just to you; it is also a threat to Internet safety at large.

GORELIK: One of the issues with your analogy is that driving down a crowded city street at high speed has direct and immediate consequences. Malware does not work that way. If you do get infected, and even if you clean it up, you might not see the direct consequences of that infection or be able to tie that infection to consequences suffered by someone else.

It is very difficult for a user to make that connection. The impact of malware infection is disconnected compared with something much more physical and tangible. I agree that education would help, but there are limits to its effectiveness.

BARRETT: Too many people find out how to protect themselves on the Internet through ad hoc means such as talking to people at cocktail parties. They may find out things such as avoiding phishing scams by looking out for poorly spelled email messages and Web sites without any graphics. Often that advice may have been true years ago but is no longer valid today.

Recently we were told by one of our security providers that a number of our customers had shown up on a Sinowal trojan drop server list (<http://news.bbc.co.uk/2/hi/technology/7701227.stm>). We had a big file of these and we are notifying those customers saying the following:

- ▶ We're terribly sorry, we believe your PC has been compromised because these are your details and they match all the data we store on you.

- ▶ This is how you probably got infected.

- ▶ Go here for advice on how to fix your PC.

► By the way, some malware is so nasty you're going to have to clear the machine and rebuild it from base hardware or take it to somebody who actually can rewrite the master boot record.

► Here's a PayPal security key to help protect you.

It's a whole get-well kit. I think we're probably one of the first companies to attempt this. As an industry we are going to have to do more of this kind of outreach.

CREEGER: Is that possible today given the architecture of existing operating systems? I have been told there's some really nasty stuff out there and if you poke it the wrong way it will crash the machine.

WEAFER: The vast majority of stuff does not require that kind of drastic action. While there's absolutely some really bad malware out there, there is still an enormous amount of fairly basic stuff that can be taken care of without a bare-metal rebuild.

One thing that's increasingly happening is that Web sites, particularly those owned by small businesses, are being compromised. These sites are in turn being used to attack others. So it's not just the problem of the desktop owner, but now the Web site owner as well. Among the biggest infection vectors we see today are threats being delivered onto user systems from compromised Web sites. If you are an owner of a Web site, you need to pay attention to this problem. This is really challenging for small business because most of those people have no idea what Web site security really entails.

Some hosting companies will do security scans as part of their service, or you can go to specialist boutiques or service providers. Depending on the type of infection, however, malware removal by this means could be very difficult.

CREEGER: It sounds like a small to medium business has to understand a lot of detail in order to address the risks we have been talking about. Are there people who can take over that function? For small to medium business owners responsible for the company's Web sites and PCs, what should they do and what are the trade-offs? What should they be thinking about? How big does an organization have to be before it hires an in-house expert?



OPINDER BAWA

Companies with fewer than 50 or more than 250 employees have viable options, but inside that range, companies have very limited options for effective IT support.



BAWA: We do a risk-to-reward security analysis for any potential insecurity we come across. This helps determine the cost benefit for any given solution to ensure we're focused on the highest risks at all times.

Consumers expect to buy one product that does everything. In the small to medium business market, companies start to understand that if they are setting up Web sites, they are going to need some protection but they don't know what that is. They turn to their small IT provider to provide that package. When you get to a \$100 million company, you are going to be concerned about DLP (data loss prevention). If you're a \$5 million company, you're not, because you know everybody and it's much more of a trusting environment.

BARRETT: One of the most basic defenses is to have good practices around patching. If you have an endpoint, keep it patched, and run up-to-date AV, then that goes a long way toward providing reasonable protection.

BAWA: People also want value. Once they understand they need these five different components for protection, they look for value. The one common question that all SMBs will ask is, "What product can I buy that gives me the most features for the best price?"

CREEGER: Trying to figure out what the value is for the money spent is a bit of a gamble. People naturally don't want to spend a lot of money, but they don't have any effective way of evaluating what their exposure is, given a particular level of spending.

GREEN: There isn't a strong enough appreciation for the volume and propensity of malware that we're dealing with. When you tell people the number of threats we see per day, their jaws just drop to the table.

WEAFER: It's the dynamic nature of malware attackers that if you defend against something, things will change. Security is very different from the more stable and predictable areas of computing such as storage and search. When one hole is plugged, attackers simply move on to attack different areas.

CREEGER: You are telling me that you have a highly adaptable adversary and that it is very difficult to assess risk and asset value. You're trying to plug as many leaks as you can with a fixed budget, and there's no real guarantee

that what you left out wasn't the critical thing that will sink your boat. Is there any good news here?

WEAFER: The good news is that people who take the basic commonsense actions seem to be less impacted on a consistent basis. This is the good hygiene story where the guy who just washes his hands before meals seems to get sick far less often. Even though new technology and new trends are being seen all the time, the basic best practices haven't changed all that much.

GORELIK: There are two kinds of attack models. One is to cast a wide net and pull in whatever you catch. As a user, if you do the basic commonsense things, you're probably reasonably protected. There are also guys who are going to target you because you have something of value. Security here should be more sophisticated and based on the impact of a successful breach.

BARRETT: This attack is based in the attacker's perceived value of the assets you have on your system and brings to mind the old joke about what to do if you are attacked by a bear in the wilderness. The answer is that you don't have to outrun the bear, you just have to outrun the guy next to you. To the extent that attackers perceive that you have juicy assets, but you seem to have done a half-decent job of locking things up, attackers may go try the bozo next door who has probably done a less competent job.

BOURNE: From a defensive perspective, are there things people were doing last year that have stopped working?

WEAFER: Up to recently we would say that if you avoided going down the dark alleys of the Internet—the pornography sites, for example—you were probably safe. That is not so true anymore because of the rise of cross-site scripting attacks that inject malicious code into otherwise normal Web pages.

GORELIK: You now have to be careful of legitimate software that has been repackaged to include other executables containing threats. Also, in the past people could safely download software that was not signed. At this point, I would not download anything that's not signed—and signed by an entity I trust. You have no way of guaranteeing that the integrity of the software has not been tampered with on the server.



VLAD GORELIK

I would not download anything that's not signed—and signed by an entity I trust. You have no way of guaranteeing that the integrity of the software has not been tampered with on the server.



I did an experiment and went to a reasonably well-known site and downloaded a whole bunch of stuff. When I first started seeing a large number of compromised executables, I thought that the problem was in our testing process. Sadly, it turned out that our process was just fine, but the files had really been infected.

BAWA: But it's not just about trusting. In October 2008 people visiting the Macys.com retail site got hijacked to a Web site that looked like Macys.com but wasn't.

BARRETT: I believe owners of consumer-facing Web sites have a responsibility not only to ensure that their infrastructure is secure, but also to demonstrate that theirs is a legitimate Web site. We pay a nontrivial uplift to have extended validation. When you go to PayPal.com with a modern operating system and browser, you get a green glow object in the address bar that says, PayPal, Inc.—not in the URL itself but beside it. These are the kinds of definitive user interface cues that consumers can and should be looking for.

WEAFER: I believe we will see more reputation-based security models. Black-and-white listing-based protection is a simpler version of this. You're seeing examples of reputation protection in browsers offering warnings about unsafe sites with file downloads showing whether the files were signed or from a trusted sources and whether email senders are on a known spam list. These are all examples of simple reputation-based security and imply a trend toward a more general model that provides the user with a full reputation on what they download or use. The user's appetite for risk then determines whether the transaction proceeds.

CREEGER: Could cloud computing help or hurt these types of issues? What does cloud computing do for these types of vulnerabilities? If people start moving their services out onto cloud platforms, are they better or worse off?

BARRETT: Cloud computing is very promising for cost-effective and burst capacity. There is a very large user base of organizations that are attracted to cloud computing where they deal with nonconfidential information. There are also people who represent more regulated industries, such as financial, that cannot just dump the data in

an outsourced cloud and not know its physical location. I have to know where my data resides because there are safe-harbor considerations I must maintain. So the data-location requirement is one issue with clouds.

A second issue is the ability to define an application's security requirements. If I have particular security requirements around my application, I don't want it to co-reside with someone else's application that has a different requirement set. We don't have the policy language yet to adequately describe everyone's security requirements. For cloud computing to work, that type of definitional information needs to be in place. It is not there today, but we will undoubtedly get to the point where we will have the proper risk vocabulary to address this issue.

Most of the cloud vendors at this point are using comparatively basic security patches that are perfectly functional, and as I said earlier, a little patching goes a long way. They are demonstrating to perspective clients their SAS70 (http://en.wikipedia.org/wiki/Statement_on_Auditing_Standards_No._70:_Service_Organizations) and you can review it to determine whether their controls are adequate.

CREEGER: What are the more interesting and surprising types of malware you have experienced?

BARRETT: ATM (automated teller machine) malware targets a specific vendor's ATMs. The bad guy wanders up to the ATM, opens up its externally accessible maintenance port, shoves in a USB stick, and takes it over. The victim happily uses the ATM while it does things the bank doesn't know and didn't intend. This is not at all common as yet and occurred only because somebody was able to steal the source code for that particular ATM.

BAWA: A service person came in and put malware on the point-of-sale devices for six or eight local San Francisco Bay area gas stations and captured all the debit cards.

BARRETT: For various reasons, debit cards are often more attractive targets than credit cards.

CREEGER: On the malware defense side, what surprisingly innovative things have vendors produced in the past few years?

WEAFER: Reputation-based security

is very interesting—cloud-based plus reputation-based. Reputation is a good way to deal with environments that are not heavily regulated and not able to be locked down.

BARRETT: One of the most interesting trends in the past year or so is the active research community letting machines get infected by bits of malware. They are trying to work out what the malware is being used for, how it's engineered, and what greater purpose it serves. The best example is the folks who took apart the Storm network and determined that its purpose was to spam Viagra ads. They really understood the mechanics of how that particular botnet was put together.

WEAFER: There's a lot more ongoing research evaluating the ecosystem and looking at every part of the crime life cycle—not just the technology pieces but who benefits from the crimes. Who is making money, how are they doing it, where is it happening, and how are they controlling it? Ultimately, we are looking for effective levers to combat malware. Sometimes these levers may be technology based, sometimes policy based.

BARRETT: Malware isn't fundamentally a technical problem; it's a crime problem. Even though we and other financial services firms have seen a lot of consumers have their credentials stolen, the average level of victimization is usually very low. The criminal market has so many credentials available at this point that the bottleneck is not in acquiring more credentials, it's in monetizing them.

CREEGER: Can ISPs provide effective malware defense?


WEAFER: ISPs need to engage their users and provide not only technology help but also policy help around this problem. The ISP is for many people the closest thing they have to a touch point on the Internet. When a user infection is identified, the ISP should engage the user in a positive dialogue and work to resolve the problem for the good of the entire community. Letting the problem fester or moving it to another vendor should not be a viable alternative.

BARRETT: The force at play is what economists call negative externalities. Decisions are made by one party, but the costs are borne by others. The costs resulting from users browsing dubious

Web sites on unprotected machines and getting infected are borne not just by the ISP but by the Web sites that may get attacked from the infected user. That user's browsing behavior has real and measurable costs to those businesses. The argument here is simply, what is the set of economic and regulatory policies that will influence users to better align their decisions with its cost impact on the rest of the community?

CREEGER: It's the same argument as public health.

BAWA: I have a very different opinion on this. Living in the technology world, you become analytical: How do I fix? What are the data points? and so on. I think security and some of the issues we're facing are more like influenza, where every year there's going to be another strain no matter what you do. You can do everything right and there will still be another strain. Technology and policy have solved certain things very well: CRM (customer relationship management) systems, supply chain, iTunes...Security is an environment that is ever changing, and unless we understand that, we're going to be chasing our tail for a long time.

CREEGER: I want to thank everyone for sharing your experiences and perspectives. It is clear to me that malware defense is an evolving area that is extremely dynamic and has equal measures of both art and science. This discussion has given our audience a better understanding of the threat landscape and an appreciation for what they should be thinking about to reduce the risk of compromise. Hopefully it will encourage folks to evaluate their security architectures to be more in tune with the risks they face. 

Related articles on queue.acm.org

Cybercrime 2.0: When the Cloud Turns Dark

Niels Provos, Maheeb Abu Rajab
and Panayiotis Mavrommatis
<http://queue.acm.org/detail.cfm?id=1517412>

Criminal Code: The Making of a Cybercriminal

Thomas Wadlow and Vlad Gorelik
<http://queue.acm.org/detail.cfm?id=1180192>

Cybercrime: An Epidemic

Team Cymru
<http://queue.acm.org/detail.cfm?id=1180190>

© 2010 ACM 0001-0782/10/0400 \$10.00

Clearing the clouds away from the true potential and obstacles posed by this computing capability.

BY MICHAEL ARMBRUST, ARMANDO FOX, REAN GRIFFITH, ANTHONY D. JOSEPH, RANDY KATZ, ANDY KONWINSKI, GUNHO LEE, DAVID PATTERSON, ARIEL RABKIN, ION STOICA, AND MATEI ZAHARIA

A View of Cloud Computing

CLOUD COMPUTING, THE long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. They need not be concerned about overprovisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or underprovisioning for one that becomes wildly popular, thus missing potential customers and revenue. Moreover, companies with large batch-oriented tasks can get results as quickly as their programs can scale, since using 1,000 servers for one hour costs no more than using one server for 1,000

hours. This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT.

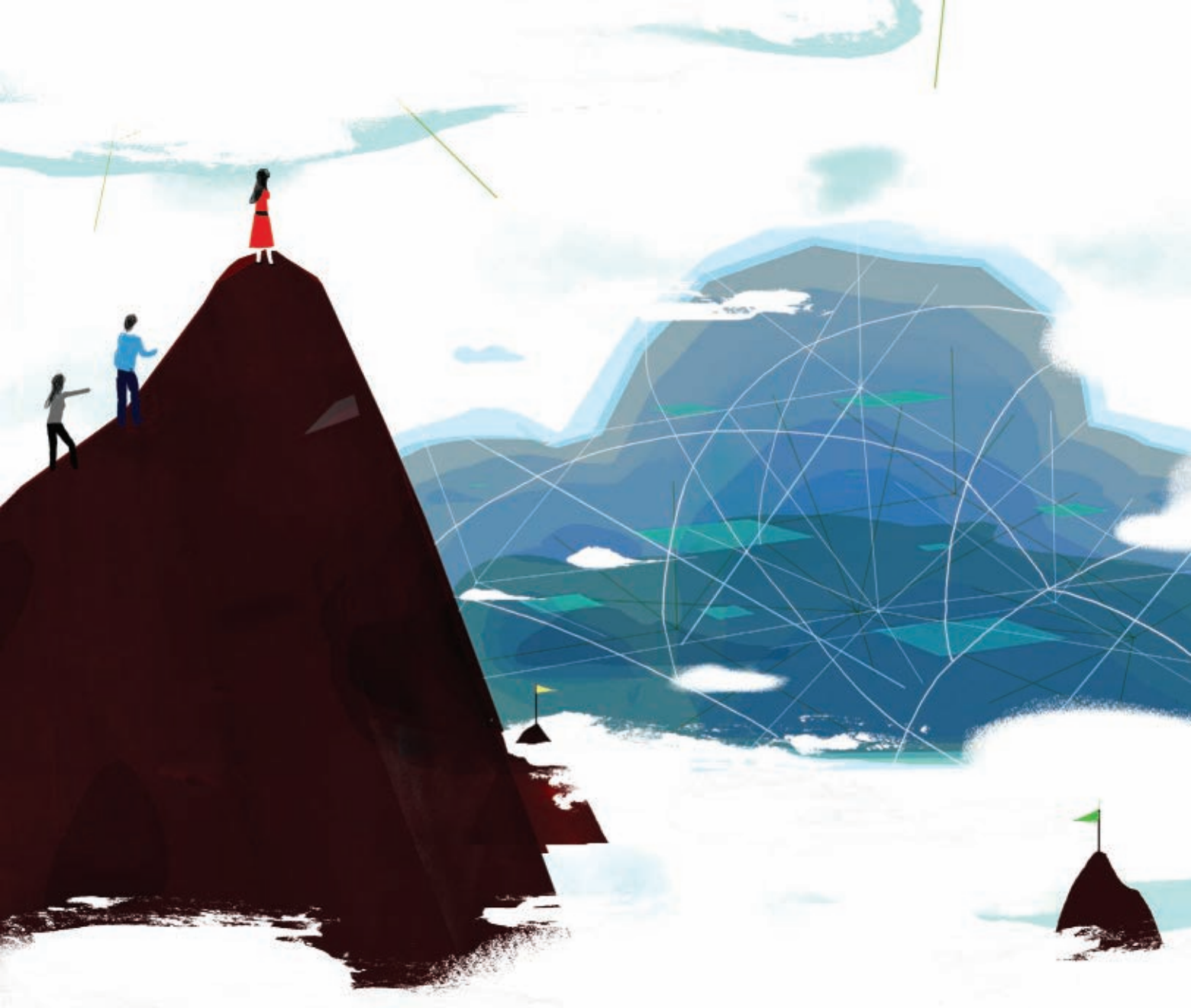
As a result, cloud computing is a popular topic for blogging and white papers and has been featured in the title of workshops, conferences, and even magazines. Nevertheless, confusion remains about exactly what it is and when it's useful, causing Oracle's CEO Larry Ellison to vent his frustration: "The interesting thing about cloud computing is that we've redefined cloud computing to include everything that we already do.... I don't understand what we would do differently in the light of cloud computing other than change the wording of some of our ads."

Our goal in this article is to reduce that confusion by clarifying terms, providing simple figures to quantify comparisons between of cloud and conventional computing, and identifying the top technical and non-technical obstacles and opportunities of cloud computing. (Armburst et al⁴ is a more detailed version of this article.)

Defining Cloud Computing

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. The services themselves have long been referred to as Software as a Service (SaaS).^a Some vendors use terms such as IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) to describe their products, but we eschew these because accepted definitions for them still vary widely. The line between "low-level" infrastructure and a higher-level "platform" is not crisp. We believe the two are more alike than different, and we consider them together. Similarly, the

^a For the purposes of this article, we use the term Software as a Service to mean applications delivered over the Internet. The broadest definition would encompass any on demand software, including those that run software locally but control use via remote software licensing.



related term “grid computing,” from the high-performance computing community, suggests protocols to offer shared computation and storage over long distances, but those protocols did not lead to a software environment that grew beyond its community.

The data center hardware and software is what we will call a *cloud*. When a cloud is made available in a pay-as-you-go manner to the general public, we call it a *public cloud*; the service being sold is *utility computing*. We use the term *private cloud* to refer to internal data centers of a business or other organization, not made available to the general public, when they are large enough to benefit from the advantages of cloud computing that we discuss here. Thus, cloud computing is the sum of SaaS and utility computing,

but does not include small or medium-sized data centers, even if these rely on virtualization for management. People can be users or providers of SaaS, or users or providers of utility computing. We focus on SaaS providers (cloud users) and cloud providers, which have received less attention than SaaS users. Figure 1 makes provider-user relationships clear. In some cases, the same actor can play multiple roles. For instance, a cloud provider might also host its own customer-facing services on cloud infrastructure.

From a hardware provisioning and pricing point of view, three aspects are new in cloud computing.

► The appearance of infinite computing resources available on demand, quickly enough to follow load surges, thereby eliminating the need for cloud

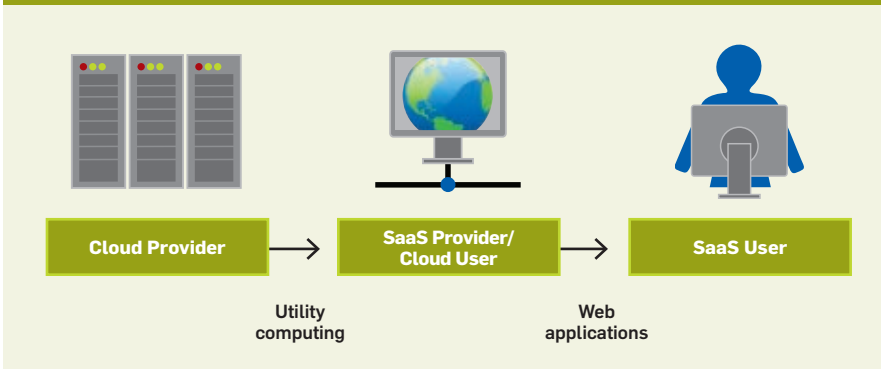
computing users to plan far ahead for provisioning.

► The elimination of an up-front commitment by cloud users, thereby allowing companies to start small and increase hardware resources only when there is an increase in their needs.^b

► The ability to pay for use of computing resources on a short-term basis as needed (for example, processors by the hour and storage by the day) and release them as needed, thereby rewarding conservation by letting machines and storage go when they are no longer useful.

^b Note, however, that upfront commitments can still be used to reduce per-usage charges. For example, Amazon Web Services also offers long-term rental of servers, which they call reserved instances.

Figure 1. Users and providers of cloud computing. We focus on cloud computing's effects on cloud providers and SaaS providers/cloud users. The top level can be recursive, in that SaaS providers can also be a SaaS users via mashups.



We argue that the construction and operation of extremely large-scale, commodity-computer data centers at low-cost locations was the key necessary enabler of cloud computing, for they uncovered the factors of 5 to 7 decrease in cost of electricity, network bandwidth, operations, software, and hardware available at these very large economies of scale. These factors, combined with statistical multiplexing to increase utilization compared to traditional data centers, meant that cloud computing could offer services below the costs of a medium-sized data center and yet still make a good profit.

Our proposed definition allows us to clearly identify certain installations as examples and non-examples of cloud computing. Consider a public-facing Internet service hosted on an ISP who can allocate more machines to the service given four hours notice. Since load surges on the public Internet can happen much more quickly than that (Animoto saw its load double every 12 hours for nearly three days), this is not cloud computing. In contrast, consider an internal enterprise data center whose applications are modified only with significant advance notice to administrators. In this scenario, large load surges on the scale of minutes are highly unlikely, so as long as allocation can track expected load increases, this scenario fulfills one of the necessary conditions for operating as a cloud. The enterprise data center may still fail to meet other conditions for being a cloud, however, such as the appearance of infinite resources or fine-grained billing. A private data center may also not benefit from the economies of scale that make public clouds financially attractive.

Omitting private clouds from cloud computing has led to considerable debate in the blogosphere. We believe the confusion and skepticism illustrated by Larry Ellison's quote occurs when the advantages of public clouds are also claimed for medium-sized data centers. Except for extremely large data centers of hundreds of thousands of machines, such as those that might be operated by Google or Microsoft, most data centers enjoy only a subset of the potential advantages of public clouds, as Table 1 shows. We therefore believe that including traditional data centers in the definition of cloud computing will lead to exaggerated claims for smaller, so-called private clouds, which is why we exclude them. However, here we describe how so-called private clouds can get more of the benefits of public clouds through *surge computing* or *hybrid cloud computing*.

Classes of Utility Computing

Any application needs a model of computation, a model of storage, and a model of communication. The statistical multiplexing necessary to achieve elasticity and the appearance of infinite capacity available on demand requires automatic allocation and management. In practice, this is done with virtualization of some sort. Our view is that different utility computing offerings will be distinguished based on the cloud system software's level of abstraction and the level of management of the resources.

Amazon EC2 is at one end of the spectrum. An EC2 instance looks much like physical hardware, and users can control nearly the entire software stack, from the kernel upward.

This low level makes it inherently difficult for Amazon to offer automatic scalability and failover because the semantics associated with replication and other state management issues are highly application-dependent. At the other extreme of the spectrum are application domain-specific platforms such as Google AppEngine, which is targeted exclusively at traditional Web applications, enforcing an application structure of clean separation between a stateless computation tier and a stateful storage tier. AppEngine's impressive automatic scaling and high-availability mechanisms, and the proprietary MegaStore data storage available to AppEngine applications, all rely on these constraints. Applications for Microsoft's Azure are written using the .NET libraries, and compiled to the Common Language Runtime, a language-independent managed environment. The framework is significantly more flexible than AppEngine's, but still constrains the user's choice of storage model and application structure. Thus, Azure is intermediate between application frameworks like AppEngine and hardware virtual machines like EC2.

Cloud Computing Economics

We see three particularly compelling use cases that favor utility computing over conventional hosting. A first case is when demand for a service varies with time. For example, provisioning a data center for the peak load it must sustain a few days per month leads to underutilization at other times. Instead, cloud computing lets an organization pay by the hour for computing resources, potentially leading to cost savings even if the hourly rate to rent a machine from a cloud provider is higher than the rate to own one. A second case is when demand is unknown in advance. For example, a Web startup will need to support a spike in demand when it becomes popular, followed potentially by a reduction once some visitors turn away. Finally, organizations that perform batch analytics can use the "cost associativity" of cloud computing to finish computations faster: using 1,000 EC2 machines for one hour costs the same as using one machine for 1,000 hours.

Although the economic appeal of

cloud computing is often described as “converting capital expenses to operating expenses” (CapEx to OpEx), we believe the phrase “pay as you go” more directly captures the economic benefit to the buyer. Hours purchased via cloud computing can be distributed non-uniformly in time (for example, use 100 server-hours today and no server-hours tomorrow, and still pay only for 100); in the networking community, this way of selling bandwidth is already known as usage-based pricing.^c In addition, the absence of up-front capital expense allows capital to be redirected to core business investment.

Therefore, even if Amazon’s pay-as-you-go pricing was more expensive than buying and depreciating a comparable server over the same period, we argue that the cost is outweighed by the extremely important cloud computing economic benefits of elasticity and transference of risk, especially the risks of overprovisioning (underutilization) and underprovisioning (saturation).

We start with elasticity. The key observation is that cloud computing’s ability to add or remove resources at a fine grain (one server at a time with EC2) and with a lead time of minutes rather than weeks allows matching resources to workload much more closely. Real world estimates of average server utilization in data centers range from 5% to 20%.^{15,17} This may sound

shockingly low, but it is consistent with the observation that for many services the peak workload exceeds the average by factors of 2 to 10. Since few users deliberately provision for less than the expected peak, resources are idle at nonpeak times. The more pronounced the variation, the more the waste.

For example, Figure 2a assumes our service has a predictable demand where the peak requires 500 servers at noon but the trough requires only 100 servers at midnight. As long as the average utilization over a whole day is 300 servers, the actual cost per day (area under the curve) is $300 \times 24 = 7,200$ server hours; but since we must provision to the peak of 500 servers, we pay for $500 \times 24 = 12,000$ server-hours, a factor of 1.7 more. Therefore, as long as the pay-as-you-go cost per server-hour over three years (typical amortization time) is less than 1.7 times the cost of buying the server, utility computing is cheaper.

In fact, this example underestimates the benefits of elasticity, because in addition to simple diurnal patterns, most services also experience seasonal or other periodic demand variation (for example, e-commerce in December and photo sharing sites after holidays) as well as some unexpected demand bursts due to external events (for example, news events). Since it can take weeks to acquire and rack new equipment, to handle such spikes you must provision for them in advance. We already saw that even if service operators predict the spike sizes correctly, capacity is wasted, and if they overestimate the spike they provision for, it’s even worse.

They may also underestimate the spike (Figure 2b), however, accidental-

ly turning away excess users. While the cost of overprovisioning is easily measured, the cost of underprovisioning is more difficult to measure yet potentially equally serious: not only do rejected users generate zero revenue, they may never come back. For example, Friendster’s decline in popularity relative to competitors Facebook and MySpace is believed to have resulted partly from user dissatisfaction with slow response times (up to 40 seconds).¹⁶ Figure 2c aims to capture this behavior: Users will desert an underprovisioned service until the peak user load equals the data center’s usable capacity, at which point users again receive acceptable service.

For a simplified example, assume that users of a hypothetical site fall into two classes: active users (those who use the site regularly) and defectors (those who abandon the site or are turned away from the site due to poor performance). Further, suppose that 10% of active users who receive poor service due to underprovisioning are “permanently lost” opportunities (become defectors), that is, users who would have remained regular visitors with a better experience. The site is initially provisioned to handle an expected peak of 400,000 users (1,000 users per server \times 400 servers), but unexpected positive press drives 500,000 users in the first hour. Of the 100,000 who are turned away or receive bad service, by our assumption 10,000 of them are permanently lost, leaving an active user base of 390,000. The next hour sees 250,000 new unique users. The first 10,000 do fine, but the site is still overcapacity by 240,000 users. This results in 24,000 additional defections, leaving 376,000 permanent users. If this pattern continues, after $\lg(500,000)$ or 19 hours, the number of new users will approach zero and the site will be at capacity in steady state. Clearly, the service operator has collected less than 400,000 users’ worth of steady revenue during those 19 hours, however, again illustrating the underutilization argument—to say nothing of the bad reputation from the disgruntled users.

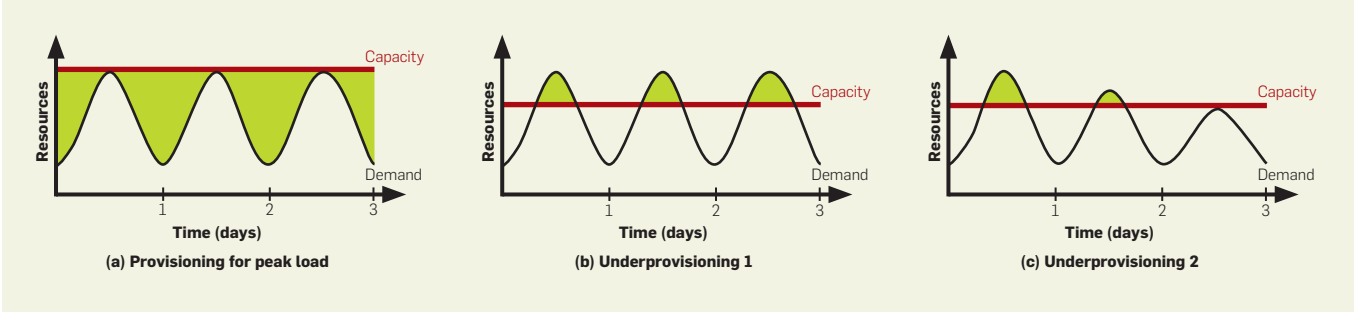
Do such scenarios really occur in practice? When Animoto³ made its service available via Facebook, it experienced a demand surge that resulted in growing from 50 servers to 3,500 servers in three days. Even if the average

c Usage-based pricing is not renting. Renting a resource involves paying a negotiated cost to have the resource over some time period, whether or not you use the resource. Pay-as-you-go involves metering usage and charging based on actual use, independently of the time period over which the usage occurs.

Table 1. Comparing public clouds and private data centers.

Advantage	Public Cloud	Conventional Data Center
Appearance of infinite computing resources on demand	Yes	No
Elimination of an up-front commitment by Cloud users	Yes	No
Ability to pay for use of computing resources on a short-term basis as needed	Yes	No
Economies of scale due to very large data centers	Yes	Usually not
Higher utilization by multiplexing of workloads from different organizations	Yes	Depends on company size
Simplify operation and increase utilization via resource virtualization	Yes	No

Figure 2. (a) Even if peak load can be correctly anticipated, without elasticity we waste resources (shaded area) during nonpeak times. (b) Underprovisioning case 1: potential revenue from users not served (shaded area) is sacrificed. (c) Underprovisioning case 2: some users desert the site permanently after experiencing poor service; this attrition and possible negative press result in a permanent loss of a portion of the revenue stream.



utilization of each server was low, no one could have foreseen that resource needs would suddenly double every 12 hours for three days. After the peak subsided, traffic fell to a lower level. So in this real-world example, scale-up elasticity was not a cost optimization but an operational requirement, and scale-down elasticity allowed the steady-state expenditure to more closely match the steady-state workload.

Top 10 Obstacles and Opportunities for Cloud Computing

Table 2 summarizes our ranked list of critical obstacles to growth of cloud computing. The first three affect adoption, the next five affect growth, and the last two are policy and business obstacles. Each obstacle is paired with an opportunity to overcome that obstacle, ranging from product development to research projects.

Number 1. Business Continuity and Service Availability

Organizations worry about whether utility computing services will have adequate availability, and this makes some wary of cloud computing. Ironically, existing SaaS products have set a high standard in this regard. Google Search has a reputation for being highly available, to the point that even a small disruption is picked up by major news sources.¹¹

Users expect similar availability from new services, which is difficult to do. Table 3 shows recorded outages for Amazon Simple Storage Service (S3), AppEngine and Gmail in 2008, and explanations for the outages. Note that despite the negative publicity due to these outages, few enterprise IT infrastructures are as good. Techni-

cal issues of availability aside, a cloud provider could suffer outages for non-technical reasons, including going out of business or being the target of regulatory action (a recent example of the latter occurred last year, as we describe later).

Although they have not done so, cloud vendors could offer specialized hardware and software techniques in order to deliver higher reliability, presumably at a high price. This reliability could then be sold to users as a service-level agreement. But this approach only goes so far. The high-availability computing community has long followed the mantra “no single point of failure,” yet the management of a cloud computing service by a single company is in fact a single point of failure. Even if the company has multiple data centers in different geographic regions using different network providers, it may have common software infrastructure and accounting systems, or the company may even go out of business. Large cus-

tomers will be reluctant to migrate to cloud computing without a business-continuity strategy for such situations. We believe the best chance for independent software stacks is for them to be provided by different companies, as it has been difficult for one company to justify creating and maintain two stacks in the name of software dependability. Just as large Internet service providers use multiple network providers so that failure by a single company will not take them off the air, we believe the only plausible solution to very high availability is multiple cloud computing providers.

Number 2. Data Lock-In

Software stacks have improved interoperability among platforms, but the storage APIs for cloud computing are still essentially proprietary, or at least have not been the subject of active standardization. Thus, customers cannot easily extract their data and programs from one site to run on another. Con-

Table 2. Top 10 obstacles to and opportunities for growth of cloud computing.

Obstacle	Opportunity
1 Availability/Business Continuity	Use Multiple Cloud Providers
2 Data Lock-In	Standardize APIs; Compatible SW to enable Surge or Hybrid Cloud Computing
3 Data Confidentiality and Auditability	Deploy Encryption, VLANs, Firewalls
4 Data Transfer Bottlenecks	FedExing Disks; Higher BW Switches
5 Performance Unpredictability	Improved VM Support; Flash Memory; Gang Schedule VMs
6 Scalable Storage	Invent Scalable Store
7 Bugs in Large Distributed Systems	Invent Debugger that relies on Distributed VMs
8 Scaling Quickly	Invent Auto-Scaler that relies on ML; Snapshots for Conservation
9 Reputation Fate Sharing	Offer reputation-guarding services like those for email
10 Software Licensing	Pay-for-use licenses

cern about the difficulty of extracting data from the cloud is preventing some organizations from adopting cloud computing. Customer lock-in may be attractive to cloud computing providers, but their users are vulnerable to price increases, to reliability problems, or even to providers going out of business.

For example, an online storage service called The Linkup shut down on Aug. 8, 2008 after losing access as much as 45% of customer data.⁶ The Linkup, in turn, had relied on the online storage service Nirvanix to store customer data, which led to finger pointing between the two organizations as to why customer data was lost. Meanwhile, The Linkup's 20,000 users were told the service was no longer available and were urged to try out another storage site.

One solution would be to standardize the APIs^d in such a way that a SaaS developer could deploy services and data across multiple cloud computing providers so that the failure of a single company would not take all copies of customer data with it. One might worry that this would lead to a "race-to-the-bottom" of cloud pricing and flatten the profits of cloud computing providers. We offer two arguments to allay this fear.

First, the quality of a service matters as well as the price, so customers may not jump to the lowest-cost service. Some Internet service providers today cost a factor of 10 more than others because they are more dependable and offer extra services to improve usability.

Second, in addition to mitigating data lock-in concerns, standardization of APIs enables a new usage model in which the same software infrastructure can be used in an internal data center and in a public cloud. Such an option could enable hybrid cloud computing or surge computing in which the public cloud is used to capture the extra tasks that cannot be easily run in the data center (or private cloud) due to temporarily heavy workloads. This option could significantly expand the cloud computing market. Indeed, open-source reimplementations of proprietary cloud APIs, such as Euca-

lyptus and HyperTable, are first steps in enabling surge computing.

Number 3. Data Confidentiality/Auditability

Despite most companies outsourcing payroll and many companies using external email services to hold sensitive information, security is one of the most often-cited objections to cloud computing; analysts and skeptical companies ask "who would trust their essential data out there somewhere?" There are also requirements for auditability, in the sense of Sarbanes-Oxley and Health and Human Services Health Insurance Portability and Accountability Act (HIPAA) regulations that must be provided for corporate data to be moved to the cloud.

Cloud users face security threats both from outside and inside the cloud. Many of the security issues involved in protecting clouds from outside threats are similar to those already facing large data centers. In the cloud, however, this responsibility is divided among potentially many parties, including the cloud user, the cloud vendor, and any third-party vendors that users rely on for security-sensitive software or configurations.

The cloud user is responsible for application-level security. The cloud provider is responsible for physical security, and likely for enforcing external firewall policies. Security for intermediate layers of the software stack is shared between the user and the operator; the lower the level of abstraction exposed to the user, the more responsibility goes with it. Amazon EC2 users have more technical responsibility (that is, must implement or procure more of the necessary functionality themselves) for their security than do Azure users, who in turn have more responsibilities than AppEngine customers. This user responsibility, in turn, can be outsourced to third parties who

sell specialty security services. The homogeneity and standardized interfaces of platforms like EC2 make it possible for a company to offer, say, configuration management or firewall rule analysis as value-added services.

While cloud computing may make external-facing security easier, it does pose the new problem of internal-facing security. Cloud providers must guard against theft or denial-of-service attacks by users. Users need to be protected from one another.

The primary security mechanism in today's clouds is virtualization. It is a powerful defense, and protects against most attempts by users to attack one another or the underlying cloud infrastructure. However, not all resources are virtualized and not all virtualization environments are bug-free. Virtualization software has been known to contain bugs that allow virtualized code to "break loose" to some extent. Incorrect network virtualization may allow user code access to sensitive portions of the provider's infrastructure, or to the resources of other users. These challenges, though, are similar to those involved in managing large non-cloud data centers, where different applications need to be protected from one another. Any large Internet service will need to ensure that a single security hole doesn't compromise everything else.

One last security concern is protecting the cloud user against the provider. The provider will by definition control the "bottom layer" of the software stack, which effectively circumvents most known security techniques. Absent radical improvements in security technology, we expect that users will use contracts and courts, rather than clever security engineering, to guard against provider malfeasance. The one important exception is the risk of inadvertent data loss. It's difficult to imagine Amazon spying on the contents of virtual machine memory; it's easy to

Table 3. Outages in AWS, AppEngine, and gmail service and outage duration date.

Service and Outage	Duration	Date
S3 outage: authentication service overload leading to unavailability ¹⁷	2 hours	2/15/08
S3 outage: Single bit error leading to gossip protocol blowup ¹⁸	6–8 hours	7/20/08
AppEngine partial outage: programming error ¹⁹	5 hours	6/17/08
Gmail: site unavailable due to outage in contacts system ¹¹	1.5 hours	8/11/08

^d Data Liberation Front; <http://dataliberation.org>

ACM Journal on Computing and Cultural Heritage



JOCCH publishes papers of significant and lasting value in all areas relating to the use of ICT in support of Cultural Heritage, seeking to combine the best of computing science with real attention to any aspect of the cultural heritage sector.



www.acm.org/jocch
www.acm.org/subscribe



Association for
Computing Machinery

imagine a hard disk being disposed of without being wiped, or a permissions bug making data visible improperly.

This is a problem in non-cloud contexts as well. The standard defense, user-level encryption, is also effective in the cloud. This is already common for high-value data outside the cloud, and both tools and expertise are readily available. This approach was successfully used by TC3, a health care company with access to sensitive patient records and health care claims, when moving their HIPAA-compliant application to AWS.³

Similarly, auditability could be added as an additional layer beyond the reach of the virtualized guest OS, providing facilities arguably more secure than those built into the applications themselves and centralizing the software responsibilities related to confidentiality and auditability into a single logical layer. Such a new feature reinforces the cloud computing perspective of changing our focus from specific hardware to the virtualized capabilities being provided.

Number 4. Data Transfer Bottlenecks

Applications continue to become more data-intensive. If we assume applications may be “pulled apart” across the boundaries of clouds, this may complicate data placement and transport. At \$100 to \$150 per terabyte transferred, these costs can quickly add up, making data transfer costs an important issue. Cloud users and cloud providers have to think about the implications of placement and traffic at every level of the system if they want to minimize costs. This kind of reasoning can be seen in Amazon’s development of its new cloudfront service.

One opportunity to overcome the high cost of Internet transfers is to ship disks. Jim Gray found the cheapest way to send a lot of data is to ship disks or even whole computers.¹⁰ While this does not address every use case, it effectively handles the case of large delay-tolerant point-to-point transfers, such as importing large data sets.

To quantify the argument, assume that we want to ship 10TB from U.C. Berkeley to Amazon in Seattle, WA. Garfinkel⁹ measured bandwidth to S3 from three sites and found an average write

bandwidth of 5Mbits/sec to 18Mbits/sec. Suppose we get 20Mbits/sec over a WAN link. It would take

$$10 * 1012 \text{ Bytes} / (20 \times 10^6 \text{ bits/second}) \\ = (8 \times 1013) / (2 \times 10^7) \text{ seconds} = 4,000,000 \text{ seconds,}$$

which is more than 45 days. If we instead sent 10 1TB disks via overnight shipping, it would take less than a day to transfer 10TB, yielding an effective bandwidth of about 1,500Mbit/sec. For example, AWS⁸ recently started offering such a service, called Import/Export.

Number 5. Performance Unpredictability

Our experience is that multiple virtual machines (VMs) can share CPUs and main memory surprisingly well in cloud computing, but that network and disk I/O sharing is more problematic. As a result, different EC2 instances vary more in their I/O performance than in main memory performance. We measured 75 EC2 instances running the STREAM memory benchmark.¹⁴ The mean bandwidth is 1,355Mbytes/sec., with a standard deviation across instances of just 52Mbytes/sec, less than or about 4% of the mean. We also measured the average disk bandwidth for 75 EC2 instances each writing 1GB files to local disk. The mean disk write bandwidth is nearly 55Mbytes per second with a standard deviation across instances of a little over 9Mbytes/sec, or about 16% of the mean. This demonstrates the problem of I/O interference between virtual machines.

One opportunity is to improve architectures and operating systems to efficiently virtualize interrupts and I/O channels. Note that IBM mainframes and operating systems largely overcame these problems in the 1980s, so we have successful examples from which to learn.

Another possibility is that flash memory will decrease I/O interference. Flash is semiconductor memory that preserves information when powered off like mechanical hard disks, but since it has no moving parts, it is much faster to access (microseconds vs. milliseconds) and uses less energy. Flash memory can sustain many more I/Os per second per gigabyte of storage than disks, so multiple virtual machines


with conflicting random I/O workloads could coexist better on the same physical computer without the interference we see with mechanical disks.

Another unpredictability obstacle concerns the scheduling of virtual machines for some classes of batch processing programs, specifically for high-performance computing. Given that high-performance computing (HPC) is used to justify government purchases of \$100M supercomputer centers with 10,000 to 1,000,000 processors, there are many tasks with parallelism that can benefit from elastic computing. Today, many of these tasks are run on small clusters, which are often poorly utilized. There could be a significant savings in running these tasks on large clusters in the cloud instead. Cost associativity means there is no cost penalty for using 20 times as much computing for 1/20th the time. Potential applications that could benefit include those with very high potential financial returns—financial analysis, petroleum exploration, movie animation—that would value a 20x speedup even if there were a cost premium.


The obstacle to attracting HPC is not the use of clusters; most parallel computing today is done in large clusters using the message-passing interface MPI. The problem is that many HPC applications need to ensure that all the threads of a program are running simultaneously, and today's virtual machines and operating systems do not provide a programmer-visible way to ensure this. Thus, the opportunity to overcome this obstacle is to offer something like “gang scheduling” for cloud computing. The relatively tight timing coordination expected in traditional gang scheduling may be challenging to achieve in a cloud computing environment due to the performance unpredictability just described.

Number 6: Scalable Storage

Earlier, we identified three properties whose combination gives cloud computing its appeal: short-term usage (which implies scaling down as well as up when demand drops), no upfront cost, and infinite capacity on demand. While it's straightforward what this means when applied to computation, it's less clear how to apply it to persistent storage.



Just as large ISPs use multiple network providers so that failure by a single company will not take them off the air, we believe the only plausible solution to very high availability is multiple cloud computing providers.



There have been many attempts to answer this question, varying in the richness of the query and storage API's, the performance guarantees offered, and the resulting consistency semantics. The opportunity, which is still an open research problem, is to create a storage system that would not only meet existing programmer expectations in regard to durability, high availability, and the ability to manage and query data, but combine them with the cloud advantages of scaling arbitrarily up and down on demand.

Number 7: Bugs in Large-Scale Distributed Systems

One of the difficult challenges in cloud computing is removing errors in these very large-scale distributed systems. A common occurrence is that these bugs cannot be reproduced in smaller configurations, so the debugging must occur at scale in the production data centers.

One opportunity may be the reliance on virtual machines in cloud computing. Many traditional SaaS providers developed their infrastructure without using VMs, either because they preceded the recent popularity of VMs or because they felt they could not afford the performance hit of VMs. Since VMs are de rigueur in utility computing, that level of virtualization may make it possible to capture valuable information in ways that are implausible without VMs.

Number 8: Scaling Quickly

Pay-as-you-go certainly applies to storage and to network bandwidth, both of which count bytes used. Computation is slightly different, depending on the virtualization level. Google AppEngine automatically scales in response to load increases and decreases, and users are charged by the cycles used. AWS charges by the hour for the number of instances you occupy, even if your machine is idle.

The opportunity is then to automatically scale quickly up and down in response to load in order to save money, but without violating service-level agreements. Indeed, one focus of the UC Berkeley Reliable Adaptive Distributed Systems Laboratory is the pervasive and aggressive use of statistical machine learning as a diagnostic and predictive tool to allow dynamic scaling, automatic reaction to perfor-

mance and correctness problems, and automatically managing many other aspects of these systems.

Another reason for scaling is to conserve resources as well as money. Since an idle computer uses about two-thirds of the power of a busy computer, careful use of resources could reduce the impact of data centers on the environment, which is currently receiving a great deal of negative attention. Cloud computing providers already perform careful and low-overhead accounting of resource consumption. By imposing fine-grained costs, utility computing encourages programmers to pay attention to efficiency (that is, releasing and acquiring resources only when necessary), and allows more direct measurement of operational and development inefficiencies.

Being aware of costs is the first step to conservation, but configuration hassles make it tempting to leave machines idle overnight so that startup time is zero when developers return to work the next day. A fast and easy-to-use snapshot/restart tool might further encourage conservation of computing resources.

Number 9: Reputation Fate Sharing

One customer's bad behavior can affect the reputation of others using the same cloud. For instance, blacklisting of EC2 IP addresses¹³ by spam-prevention services may limit which applications can be effectively hosted. An opportunity would be to create reputation-guarding services similar to the "trusted email" services currently offered (for a fee) to services hosted on smaller ISP's, which experience a microcosm of this problem.

Another legal issue is the question of transfer of legal liability—cloud computing providers would want customers to be liable and not them (such as, the company sending the spam should be held liable, not Amazon). In March 2009, the FBI raided a Dallas data center because a company whose services were hosted there was being investigated for possible criminal activity, but a number of "innocent bystander" companies hosted in the same facility suffered days of unexpected downtime, and some went out of business.⁷

Number 10: Software Licensing

Current software licenses commonly

restrict the computers on which the software can run. Users pay for the software and then pay an annual maintenance fee. Indeed, SAP announced that it would increase its annual maintenance fee to at least 22% of the purchase price of the software, which is close to Oracle's pricing.¹⁷ Hence, many cloud computing providers originally relied on open source software in part because the licensing model for commercial software is not a good match to utility computing.

The primary opportunity is either for open source to remain popular or simply for commercial software companies to change their licensing structure to better fit cloud computing. For example, Microsoft and Amazon now offer pay-as-you-go software licensing for Windows Server and Windows SQL Server on EC2. An EC2 instance running Microsoft Windows costs \$0.15 per hour instead of \$0.10 per hour for the open source alternative. IBM also announced pay-as-you-go pricing for hosted IBM software in conjunction with EC2, at prices ranging from \$0.38 per hour for DB2 Express to \$6.39 per hour for IBM WebSphere with Lotus Web Content Management Server.

Conclusion

We predict cloud computing will grow, so developers should take it into account. Regardless of whether a cloud provider sells services at a low level of abstraction like EC2 or a higher level like AppEngine, we believe computing, storage, and networking must all focus on horizontal scalability of virtualized resources rather than on single node performance. Moreover:


1. Applications software needs to both scale down rapidly as well as scale up, which is a new requirement. Such software also needs a pay-for-use licensing model to match needs of cloud computing.

2. Infrastructure software must be aware that it is no longer running on bare metal but on VMs. Moreover, metering and billing need to be built in from the start.

3. Hardware systems should be designed at the scale of a container (at least a dozen racks), which will be the minimum purchase size. Cost of operation will match performance and cost of purchase in importance, rewarding

energy proportionality⁵ by putting idle portions of the memory, disk, and network into low-power mode. Processors should work well with VMs and flash memory should be added to the memory hierarchy, and LAN switches and WAN routers must improve in bandwidth and cost.

Acknowledgments

This research is supported in part by gifts from Google, Microsoft, Sun Microsystems, Amazon Web Services, Cisco Systems, Cloudera, eBay, Facebook, Fujitsu, Hewlett-Packard, Intel, Network Appliances, SAP, VMWare, Yahoo! and by matching funds from the University of California Industry/University Cooperative Research Program (UC Discovery) grant COM07-10240 and by the National Science Foundation Grant #CNS-0509559. 

The authors are associated with the UC Berkeley Reliable Adaptive Distributed Systems Laboratory (RAD Lab).

References

1. Amazon.com CEO Jeff Bezos on Animoto (Apr. 2008); <http://blog.animoto.com/2008/04/21/amazon-ceo-jeff-bezos-on-animoto/>.
2. Amazon S3 Team. Amazon S3 Availability Event (July 20, 2008); <http://status.aws.amazon.com/s3-20080720.html>.
3. Amazon Web Services. TC3 Health Case Study; <http://aws.amazon.com/solutions/case-studies/tc3-health/>.
4. Armbrust, M., et al. Above the clouds: A Berkeley view of cloud computing. Tech. Rep. UCB/Eecs-2009-28, EECS Department, U.C. Berkeley, Feb 2009.
5. Barroso, L.A., and Holze, U. The case for energy-proportional computing. *IEEE Computer* 40, 12 (Dec. 2007).
6. Brodtkin, J. Loss of customer data spurs closure of online storage service 'The Linkup.' *Network World* (Aug. 2008).
7. Fink, J. FBI agents raid Dallas computer business (Apr. 2009); http://cbs11tv.com/local/Core_IPNetworks.2.974706.html.
8. Freedom OSS. Large data set transfer to the cloud (Apr. 2009); <http://freedomoss.com/clouddataingestion>.
9. Garfinkel, S. An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS. Tech. Rep. TR-08-07, Harvard University, Aug. 2007.
10. Gray, J., and Patterson, D. A conversation with Jim Gray. *ACM Queue* 1, 4 (2003), 8–17.
11. Helft, M. Google confirms problems with reaching its services (May 14, 2009).
12. Jackson, T. We feel your pain, and we're sorry (Aug. 2008); <http://gmailblog.blogspot.com/2008/08/we-feel-your-pain-and-were-sorry.html>.
13. Krebs, B. Amazon: Hey spammers, Get off my cloud! *Washington Post* (July 1 2008).
14. McCalpin, J. Memory bandwidth and machine balance in current high performance computers. *IEEE Technical Committee on Computer Architecture Newsletter* (1995), 19–25.
15. Rangan, K. The Cloud Wars: \$100+ Billion at Stake. Tech. Rep., Merrill Lynch, May 2008.
16. Rivlin, G. Wallflower at the Web Party (Oct. 15, 2006).
17. Siegele, L. Let It Rise: A Special Report on Corporate IT. *The Economist* (Oct. 2008).
18. Stern, A. Update from Amazon Regarding Friday's S3 Downtime. CenterNetworks (Feb. 2008); <http://www.centernetworks.com/amazon-s3-downtime-update>.
19. Wilson, S. AppEngine Outage. CIO Weblog (June 2008); <http://www.cio-weblog.com/50226711/appengine\outage.php>.



acmqueue has now moved completely online!

acmqueue is guided and written by distinguished and widely known industry experts. The newly expanded site also offers more content and unique features such as *planetqueue* blogs by *queue* authors who “unlock” important content from the ACM Digital Library and provide commentary; **videos**; downloadable **audio**; **roundtable discussions**; plus unique *acmqueue* **case studies**.

acmqueue provides a critical perspective on current and emerging technologies by bridging the worlds of journalism and peer review journals. Its distinguished Editorial Board of experts makes sure that *acmqueue*'s high quality content dives deep into the technical challenges and critical questions software engineers should be thinking about.



Visit today!

<http://queue.acm.org/>

DOI:10.1145/1721654.1721673

Prior work on power management reflects recurring themes that can be leveraged to make future systems more energy efficient.

BY PARTHASARATHY RANGANATHAN

Recipe for Efficiency: Principles of Power-Aware Computing

POWER AND ENERGY are key design considerations across a spectrum of computing solutions, from supercomputers and data centers to handheld phones and other mobile computers. A large body of work focuses on managing power and improving energy efficiency. While prior work is easily summarized in two words—“Avoid waste!”—the challenge is figuring out where and why waste happens and determining how to avoid it. In this article, I discuss how, at a general level, many inefficiencies, or waste, stem from the inherent way system architects address the complex trade-offs in the system-design process. I discuss common design practices that lead to power

inefficiencies in typical systems and provide an intuitive categorization of high-level approaches to addressing them. The goal is to provide practitioners—whether in systems, packaging, algorithms, user interfaces, or databases—a set of tools, or “recipes,” to systematically reason about and optimize power in their respective domains.

If you are a user of any kind of computing device, chances are you can share a personal anecdote about the importance of power management in helping control the electricity (energy) it consumes. On mobile devices, this translates directly into how long the battery lasts under typical usage. The battery is often the largest and heaviest component of the system, so improved battery life also enables smaller and lighter devices. Additionally, with the increasing convergence of functionality on a single mobile device (such as phone + mp3 player + camera + Web browser), battery life is a key constraint on its utility. Indeed, longer battery life is often the highest-ranked metric in user studies of requirements for future mobile devices, trumping even increased functionality and richer applications.

Power management is also important for tethered devices (connected to a power supply). The electricity consumption of computing equipment in a typical U.S. household runs to several hundred dollars per year. This cost is vastly multiplied in business enterprises. For example, servers in Google’s data centers have been estimated to consume millions of dollars

>> key insights

- **The energy efficiency of today’s systems can be improved by at least an order of magnitude.**
- **A holistic look at how systems use power and heat reveals new “recipes” to help optimize consumption and avoid wasting precious resources for a given task.**
- **Future power management will include nontraditional approaches, including crossing individual layers of design and spending more power to save power.**

ILLUSTRATION BY JEAN-FRANCOIS PODEVIN



Overview of previous work on power management.

$$P = C \cdot V_{dd}^2 \cdot F_{0..1} + T_{sc} \cdot V_{dd} \cdot I_{peak} \cdot F_{0..1} + V_{dd} \cdot I_{leakage}$$

Average power, peak power, power density, energy-delay...

Circuits

- Voltage scaling/islands**
- Clock gating/routing**
Clock-tree distribution, half-swing clocks
- Redesigned latches/flip-flops**
pin-ordering, gate restructuring, topology restructuring, balanced delay paths, optimized bit transactions
- Redesigned memory cells**
Low-power SRAM cells, reduced bit-line swing, multi-Vt, bit-line/word-line isolation/segmentation
- Other optimizations**
Transistor resizing, GALs, low-power logic

Architecture

- Voltage/freq scaling**
- Gating**
Pipeline, clock, functional units, branch prediction, data path
- Power-efficient cores**
- Split windows, throttling...**
- Bank partitioning**
- Cache redesign**
Sequential, MRU, hash-rehash, column-associative, filter cache, sub-banking, divided word line, block buffers, multi-divided module, scratch
- Low-power states**
- DRAM refresh control**
- Data packing/compression**
- Platforms architecture**
Gray, bus-invert, address-increment, intercomponent power shifting, heterogeneity, power-aware load allocation

Compiler, Systems

- Switching control**
Register relabeling, operand swapping, instruction scheduling
- Memory access reduce**
Locality optimizations, register allocation
- Power-mode control**
- CPU/resource schedule**
- Memory/disk control**
Disk spinning, page allocation, memory mapping, memory bank control
- Networking**
Power-aware routing, proximity-based routing, balancing hop count...
- Distributed computing**
Mobile-agents placement, network-driven computation
- Distributed computing**
Energy-aware resource allocation, VM migration, temperature-aware workload placement, ensemble power management

in electricity costs per year.¹⁰ IT analysis firm IDC (<http://www.idc.com/>) estimates the total worldwide spending on power management for enterprises was likely a staggering \$40 billion in 2009. Increased power consumption can also lead to increased complexity in the design of power supplies (and power distribution and backup units in larger systems) that also add costs.

Another challenge associated with power consumption in systems is the waste heat they generate; consequently, the term “power management” also includes the heat management in systems. Such heat is often a greater problem than the amount of electricity being consumed. To prevent the heat from affecting the user or the system’s electronics, systems require increasingly complex thermal packaging and heat-extraction solutions, adding more costs. For large systems like supercomputers and data centers, such costs often mean an additional

dollar spent on cooling for every dollar spent on electricity. This effect is captured in a metric called “power usage effectiveness,” or PUE,¹³ developed by the Green Grid, a global consortium of IT companies seeking to improve energy efficiency in data centers. Heat dissipation in systems also has implications for the compaction and density of computing systems, as in blade-server configurations.

Studies, most notably concerning servers and hard-disk failures, have shown that operating electronics at temperatures that exceed their operational range can lead to significant degradation of reliability; for example, the Uptime Institute, an industry organization that tracks data-center trends (<http://www.uptimeinstitute.org/>), has identified a 50% increased chance of server failure for each 10°C increase over 20°C¹⁵; similar statistics have also been shown over hard-disk lifetimes.¹⁴


Finally, power management in computing systems has environmental implications. Computing equipment in the U.S. alone is estimated to consume more than 20 million gigajoules of energy per year, the equivalent of four-million tons of carbon-dioxide emissions into the atmosphere.¹⁰ Federal agencies have identified energy-consumption implications for air quality, national security, climate change, and electricity-grid reliability, motivating several initiatives worldwide from governmental agencies, including the Environmental Protection Agency in the U.S. (<http://www.epa.gov/>), Intelligent Energy Europe (ec.europa.eu/energy/intelligent/), Market Transformation Program in the U.K. (<http://efficient-products.defra.gov.uk/cms/market-transformation-programme/>), and Top Runner (http://www.eccj.or.jp/top_runner/index.html) in Japan, and from industry consortiums, including SPEC (<http://www.spec.org/>), Green-

Grid (<http://www.thegreengrid.org/>), and TPC (http://www.tpc.org/tpc_energy/default.asp) on improving energy efficiency, or minimizing the amount of energy consumed for a given task.


The importance of power management is only likely to increase in the future. On mobile devices, there is a widening gap between advances in battery capacity and anticipated increases in mobile-device functionality. New battery technologies (such as fuel cells) might address it, but designing more power-efficient systems will still be important. Energy-review data from the U.S. Department of Energy (<http://www.eia.doe.gov/>) points to steadily increasing costs for electricity. Indeed, for data centers, several reports indicate that costs associated with power and cooling could easily overtake hardware costs.^{2,14} Increased compaction (such as in future predicted blade servers) will increase power densities by an order of magnitude within the next decade, and the increased densities will start hitting the physical limits of practical air-cooled solutions. Research is ongoing in alternate cooling technologies (such as efficient liquid cooling), but it will still be important to be efficient about generating heat in the first place. All of this requires better power management.

How to Respond

Much prior work looked at power management and energy efficiency; the figure here outlines key illustrative solutions in the literature across different levels of the solution stack in process technology and circuits, architecture and platforms, and applications and systems design. A detailed discussion of the specific optimizations is not my intent here, and, indeed, several tutorial articles^{6,10,11} and conferences that focus solely on power, including the International Symposium on Low Power Electronics and Design (<http://www.islped.org/>) and the Workshop on Power Aware Computing and Systems (aka HotPower; <http://www.sigops.org/sosp/sosp09/hotpower.html>), provide good overviews of the state of the art in power management. This rich body of work examining power management and energy efficiency can be broadly categorized across different levels of



The goal is to provide practitioners a set of tools, or “recipes,” to systematically reason about and optimize power in their respective domains.



the solution stack (such as hardware and software), stages of the life cycle (such as design and runtime), components of the system (such as CPU, cache, memory, display, interconnect, peripherals, and distributed systems), target domains (such as mobile devices, wireless networks, and high-end servers), and metrics (such as battery life and worst-case power). Much prior work concerns electrical and computer systems engineering, with a relatively smaller amount in the core areas of computer science. The prior focus on power and energy challenges at the hardware and systems levels is natural and central, but, in the future, significant improvements in power and energy efficiency are likely to result from also rethinking algorithms and applications at higher levels of the solution stack. Indeed, discussions in the past few years on the future of power management focused this way.^{9,12}

In spite of the seemingly rich diversity of prior work on power management, at a high level, the common theme across all solutions is “Avoid wasted energy!” Where the solutions differ is in the identification and intuition needed for specific sources of inefficiency, along with the specific mechanisms and policies needed to target these inefficiencies. This observation raises interesting questions: What general recurring high-level trends lead to these inefficiencies at different levels of the system? And what common recurring high-level approaches are customized in the context of specific scenarios? The ability to answer supports the beginnings of a structure to think about power management in a more systematic manner and potentially identify opportunities for energy efficiency beyond traditional platform-centric domains.


Sources of Waste

It is easy to imagine that there is a certain minimum amount of electrical energy needed to perform a certain task and a corresponding minimum amount of heat that must be extracted to avoid thermal problems. For example, R.N. Mayo et al.⁸ performed simple experiments to measure the energy consumption of common mobile tasks (such as listening to music, making a phone call, sending email


and text messages, and browsing the Web) implemented on different devices (such as cellphones, MP3 players, laptops, and PCs) and observed two notable results: There is a significant difference in energy efficiency, often 10- to a hundredfold, across different systems performing the same task. And there are variations in the user experience across devices, but even when focused on duplicating the functionality of the best-performing system, these experiments showed it was impossible to do so at the same energy level on a different worse-performing system.

Why do some designs introduce additional inefficiencies over and above the actual energy required for a given task? My observation is that these inefficiencies are often introduced when the system design must reconcile complex trade-offs that are difficult to avoid. For example, systems are often designed for the most general case, most aggressive workload performance, and worst-case risk tolerance. Such designs can lead to resource overprovisioning to better handle transient peaks and offer redundancy in the case of failure. Moreover, individual components of a broader system are often designed by different teams (even by different vendors) without consideration for their use with one another. Individual functions of a system are also designed modularly, often without factoring their interactions with one another, adding further inefficiencies. Further, traditional designs focus primarily on system performance. This approach has sometimes led to resource-wasteful designs to extract small improvements in performance; with today's emphasis on energy costs, these small improvements are often overshadowed by the costs of power and heat extraction. Similarly, additional inefficiencies are introduced when the system design takes a narrow view of performance (vs. actual end-user requirements) or fails to address total cost of ownership, including design and operational costs.

General-purpose solutions. General-purpose systems often provide a better consumer experience; for example, most users prefer to carry a single converged mobile device rather than sev-



An insidious problem is when each layer of the stack makes worst-case assumptions about other layers in the stack, leading to compound inefficiencies.



eral separate devices (such as phone, camera, and MP3 player or GPS unit). Additionally, the exigencies of volume economics further motivate vendors to develop general-purpose systems; a product that sells in the millions of units is usually cheaper to make than, say, a product that sells in the hundreds of units.

By definition, general-purpose systems must be designed to provide good performance for a multitude of different applications. This requirement results in designers using the “union” of maximum requirements of all application classes. For example, a laptop that targets a DVD-playback application might incorporate a high-resolution display and powerful graphics processor. When the same laptop is used for another task (such as reading email), the high-power characteristics of the display and graphics processor might not be needed. However, when the laptop is designed for both workloads, most designs typically include a display with the characteristics of the most aggressive application use, in this case, a high-resolution display that plays DVD movies well. Lacking adequate design thought into how energy consumption might be adapted to different kinds of tasks, such an approach often leads to significant power inefficiencies. Another example is in the data center, where optimizing for both mission-critical and non-mission-critical servers in the same facility can lead to significant inefficiencies in terms of cooling costs. Similar conflicting optimizations occur when legacy solutions must be supported on newer systems.

Planning for peaks and growth. Most workloads go through different phases when they require different performance levels from the system. For example, several studies have reported that the average server utilization in live data centers can be low (often 10%–30%). Mobile systems have also been found to spend a significant fraction of their time in idle mode or using only a small fraction of their resources.

However, most benchmarks (the basis of system design) are typically structured to stress worst-case performance workloads irrespective of how the system is likely to be used in prac-

tice. Consequently, many systems are optimized for the peak-performance scenario. In the absence of designs that proportionally scale their energy with resource utilization, the result can be significant inefficiencies. For example, many power supplies are optimized for peak conversion efficiency at high loads. When these systems are operated at low loads, the efficiency of conversion can drop dramatically, leading to power inefficiencies.

Similar overprovisioning occurs when planning for the future. Most computing systems are designed for three-to-five-year depreciation cycles, and in the case of larger installations, like data centers, even longer. Systems must be designed to ensure that sufficient capacity is built in to meet incremental growth needs. On many systems, overprovisioning also leads to inefficiencies when the system is not operating at the resource-utilization capacities that account for future growth. For example, a data center with cooling provisioned for one megawatt of operational power, but operating at only 100 kilowatts of power consumption, is significantly more inefficient than a data center with cooling provisioned for, say, 150 kilowatts of operational power and

operating at 100 kilowatts of actual power consumption.

Design process structure. Current system-design approaches generally follow a structured process. System functionality is divided across multiple hardware components (CPU, chipset, memory, networking, and disk in a single system or different individual systems in a cluster) and software components (firmware, virtualization layer, operating systems, and applications). Even within a component (such as the networking stack), there are often multiple layers with well-defined abstractions and interfaces. Power management is usually implemented within these well-defined layers but often without consideration for the interaction across the layers. However, such modular designs or local optimizations might be suboptimal for global efficiency without communication across layers. An insidious problem is when each layer of the stack makes worst-case assumptions about other layers in the stack, leading to compound inefficiencies.

Information exchange across layers often enables better power optimization. For example, a power-management optimization at the physical layer of a wireless communication

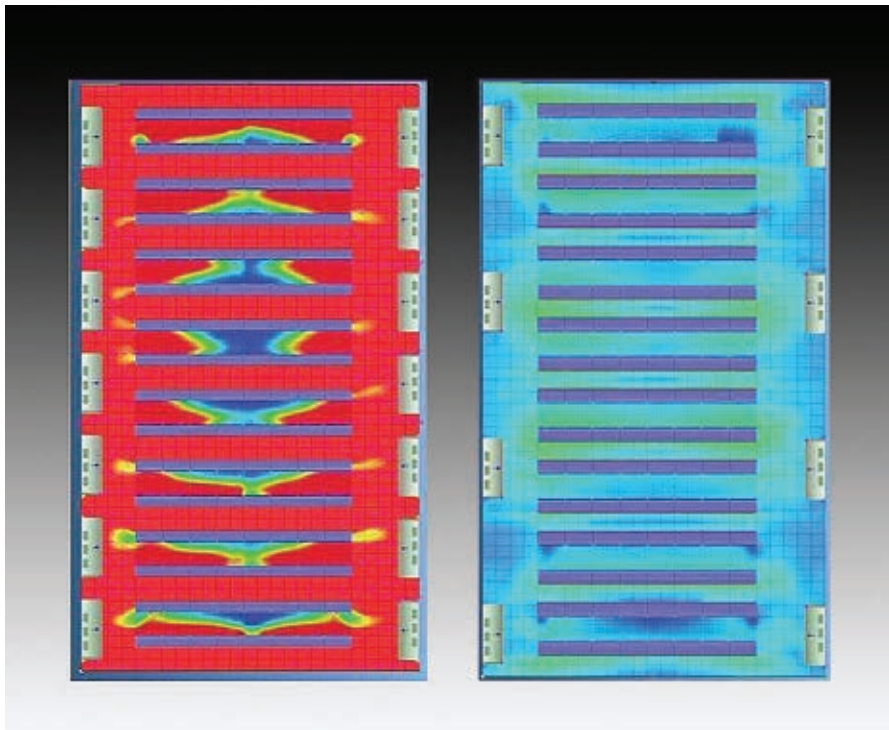
protocol that is aware of higher-level application activity can be more efficient than one that is oblivious to higher-level application activity. Similarly, a power-management solution that optimizes at an ensemble level (such as across different components in a system or different systems in a cluster) can be more efficient.

Similar problems exist at other boundaries of the system architecture. For example, the power management of servers is handled by the IT department, while the cooling infrastructure is often handled by a separate facilities department. This organizational structure can lead to inefficiencies as well. For example, a cooling solution that is aware of the nonuniformities in power consumption (and consequent heat generation) can be more efficient than a solution that is not.

Inefficiencies that result from layering can also be found at other places in the overall solution architecture. For example, in a classic client-server architecture, selectively exchanging information between the clients and servers has been shown to be beneficial for energy optimizations at both levels.

Tethered-system hangover. In this final design practice that leads to inefficiencies, the inefficiencies are mainly a reflection of the relentless drive to achieve higher performance, often following the assumption that there is no constraint on power, particularly by tethered systems (plugged into a power supply when in use) with no immediate consideration of battery life. For example, historically, many processor-architecture designs have included optimizations that achieved incremental performance improvements inconsistent with the amount of additional power consumed to implement the solutions. Similar trade-offs are seen in designs for high availability at the expense of energy (such as triple modular redundancy running three concurrent executions of the same task to ensure no possibility of downtime).

Additional examples include designs with user interfaces that identify the content of interest to the user; expending energy in these areas can be more energy efficient than designs that focus on metrics like refresh rate.



IBM uses thermal analysis to test and create “green” configurations of its iDataPlex system to deliver optimal energy efficiency, as shown on the right.


Similarly, designs that focus on energy delay may be significantly more energy efficient but with only a marginal difference in performance from pure performance-centric designs. In general, several significant power inefficiencies in today's systems stem from a design focus that does not sufficiently address total cost of ownership and ultimate end-user experience, but rather focuses disproportionately on one or more narrow metrics.

How to Reduce Waste


Once these inefficiencies are identified, the next step is to identify approaches to reduce them that fall into 10 broad categories:

Use a more power-efficient alternative. These approaches include replacing a system component with a more power-efficient alternative that performs the same task with less energy. For example, more energy-efficient nonvolatile memory can replace a disk drive, and optics can replace conventional networking. A more power-efficient alternative might sometimes involve adding the right hooks to enable the approaches discussed later. For example, replacing a display with a single backlight with an alternate display that provides more fine-grain control of power can, in turn, enable power optimizations that turn off unused portions of the display. Choosing a power-efficient alternative often involves other trade-offs, possibly due to costs or performance; otherwise, the design would have used the power-efficient option in the first place.

Create "energy proportionality" by scaling down energy for unused resources. These approaches involve turning off or dialing-down unused resources proportional to system usage, often called "energy proportionality"² or "energy scale-down."⁸ Automatically turning off unused resources requires algorithms that respond to the consequences of turning off or turning down a system (such as by understanding how long it takes to bring the system back on again). If a single component or system lacks the option to be scaled-down, the optimization is sometimes applied at the ensemble level; examples of ensemble-level scale-down include changing traffic routing to turn off unused switches



Decades ago, Nobel physicist Richard Feynman implied we should be able to achieve the computational power of a billion desktop-class processors in the power consumption of a single typical handheld device.



and virtual-machine consolidation to coalesce workloads into a smaller subset of systems in a data center.

Match work to power-efficient option. These approaches are complementary to the preceding approach—energy proportionality—but, rather than having the resources adapt when not fully utilized for a given task, they match tasks to the resources most appropriate to the size of the task. An example is the intelligent use of heterogeneity to improve power efficiency (such as scheduling for asymmetric and heterogeneous multicore processors). Matching work to resources implies there is a choice of resources for a given task. In cluster or multicore environments, the choice exists naturally, but other designs might need to explicitly introduce multiple operation modes with different power-performance trade-offs.

Piggyback or overlap energy events. These approaches seek to combine multiple tasks into a single energy event. For example, multiple reads coalescing on a single disk spin can reduce total disk energy. Prefetching data in predictable access streams or using a shared cache across multiple processes are other examples where such an approach saves energy. Disaggregating or decomposing system functionality into smaller subtasks can help increase the benefits from energy piggybacking by avoiding duplication of energy consumption for similar subtasks across different larger tasks.

Clarify and focus on required functionality. These approaches produce solutions specific to the actual constraints on the design without trying to be too general-purpose or future-proof. For example, special-purpose solutions (such as graphics processors) can be more energy-efficient for their intended workloads. Similarly, designs that seek to provide for future growth by adding modular building blocks can be more energy efficient compared to a single monolithic future-proof design.

Cross layers and broaden the scope of the solution space. Rather than having individual solutions address power management at a local level, focusing on the problem holistically is likely to achieve better efficiencies. Examples

where such an approach have been shown to be effective include scheduling across an ensemble of systems or system components and facilities-aware IT scheduling (such as temperature-aware workload placement). Exchanging information across multiple layers of the networking stack has also been shown to be beneficial for energy efficiency.

Trade off some other metric for energy. These approaches achieve better energy efficiency by marginally compromising some other aspect of desired functionality. An interesting example involves trading off fidelity in image rendering in DVD playback for extended player battery life. Also in this category are optimizations for improved energy delay where improvements in energy consumption significantly outweigh degradations in delay.

Trade off uncommon-case efficiency for common-case efficiency. These approaches seek to improve overall energy efficiency by explicitly allowing degradation in energy efficiency for rare cases and to improve energy efficiency in common cases. For example, a server power supply could be optimized for peak efficiency at normal light loads, even if it leads to degraded power efficiency at infrequent peak loads.

Spend someone else's power. These approaches take a more local view of energy efficiency but at the expense of the energy-efficiency of a different remote system. For example, a complex computation in a battery-constrained mobile device can be offloaded to a remote server in the “cloud,” potentially improving the energy efficiency of the mobile device. Approaches that scavenge energy from, say, excess heat or mechanical movement to improve overall energy efficiency also fall in this category.

Spend power to save power. A final category proactively performs tasks that address overall energy efficiency, even though these tasks may themselves consume additional energy. Examples include a garbage collector that periodically reduces the memory footprint to allow memory banks to be switched to lower-power states and a compression algorithm that enables the use of less energy for communication and storage.

The first five categories are well studied and found throughout existing power optimizations. The other five are less common but likely to be important in the future. Combinations are also possible.

Finally, irrespective of which approach is used to improve power efficiency, any solution must include three key architectural elements:

- ▶ Rich measurement and monitoring infrastructure;
- ▶ Accurate analysis tools and models that predict resource use, identify trends and causal relationships, and provide prescriptive feedback; and
- ▶ Control algorithms and policies that leverage the analysis to control power (and heat), ideally coordinated with one another.

From a design point of view, system support is needed at all levels—hardware, software, and application—to facilitate measurement, analysis, control, and cross-layer information sharing and coordination.

Looking Ahead

In spite of all this research and innovation, power management still has a long way to go. By way of illustration, several decades ago, Nobel physicist Richard Feynman estimated that, based on the physical limits on the power costs to information transfer,⁵ a staggering 10^{18} -bit operations per second can be achieved for one watt of power consumption. In terms easier to relate to, this implies we should be able to achieve the computational power of a billion desktop-class processors in the power consumption of a single typical handheld device. This is a data point on the theoretical physics of energy consumption, but the bound still points to the tremendous potential for improved energy efficiency in current systems. Furthermore, when going beyond energy consumption in the operation of computing devices to the energy consumption in the supply-and-demand side of the overall IT ecosystem (cradle-to-cradle³), the potential is enormous.

The energy efficiency of today's systems can be improved by at least an order of magnitude through systematic examination of their inherent inefficiencies and rethinking of their designs. In particular, in addition

to the large body of work in electrical and computer engineering, a new emerging science of power management can play a key role⁹ across the broader computer science community. I hope the discussions here—on the design practices that lead to common inefficiencies and the main solution approaches for addressing them—provide a starting framework toward systematically thinking about other new ideas in new domains that will help achieve the improvements. ■

References

1. Anderson, D., Dykes, J., and Riedel, E. More than an interface: SCSI vs. ATA. In *Proceedings of the Second Usenix Conference on File and Storage Technologies* (San Francisco, CA, Mar. 31–Apr. 2, 2003), 245–256.
2. Barroso, L.A. and Hölzle, U. The case for energy-proportional computing. *IEEE Computer* 40, 12 (Dec. 2007), 33–37.
3. Chandrakant, P. *Dematerializing the Ecosystem*. Keynote at the Sixth USENIX Conference on File and Storage Technologies (San Jose, CA, Feb. 26–29, 2008); <http://www.usenix.org/events/fast08/tech/patel.pdf>
4. Cole, G. *Estimating Drive Reliability in Desktop Computers and Consumer Electronics*. Tech. Paper TP-338.1. Seagate Technology, Nov. 2000.
5. Feynman, R. *Feynman Lectures on Computation*. Westview Press, 2000.
6. Irwin, M.J. and Vijaykrishnan, N. Low-power design: From soup to nuts. Tutorial at the International Symposium on Computer Architecture (Vancouver, B.C., June 10–14, 2000); <http://www.cse.psu.edu/research/mdl>
7. Lefurgy, C., Rajamani, K., Rawson, F., Felter, W., Kistler, M., and Keller, T.W. Energy management for commercial servers. *IEEE Computer* 36, 12 (Dec. 2003), 39–48.
8. Mayo, R.N. and Ranganathan, P. Energy consumption in mobile devices: Why future systems need requirements-aware energy scale-down. In *Proceedings of the Workshop on Power-Aware Computing Systems* (San Diego, CA, 2003), 26–40.
9. National Science Foundation. Workshop on the Science of Power Management (Arlington, VA, Apr. 9–10, 2009); <http://scipm.cs.vt.edu/>
10. Patel, C. and Ranganathan, P. Enterprise power and cooling: A chip-to-data-center perspective. In *Proceedings of Hot Chips 19* (Palo Alto, CA, Aug. 20, 2007); <http://www.hotchips.org/archives/hc19/>
11. Rajamani, K., Lefurgy, C., Ghiasi, S., Rubio, J.C., Hanson, H., and Keller, T. Power management for computer systems and datacenters. In *Proceedings of the 13th International Symposium on Low-Power Electronics and Design* (Bangalore, Aug. 11–13, 2008); <http://www.islped.org/X2008/Rajamani.pdf>
12. Ranganathan, P. (moderator). Power Management from Cores to Data Centers: Where Are We Going to Get the Next 10X? Panel at International Symposium on Low-Power Electronic Devices (Bangalore, 2008); <http://www.islped.org/X2008/>
13. Rawson, A., Pflueger, J., and Cader, T. (C. Belady, Ed.). *The Green Grid Data Center Power Efficiency Metrics: Power Usage Effectiveness and DCIE*. The Green Grid, 2007; www.thegreengrid.org
14. Shankland, S. Power could cost more than servers, Google warns. *CNET News* (Dec. 9, 2005); http://news.cnet.com/Power-could-cost-more-than-servers,-Google-warns/2100-1010_3-5988090.html
15. Sullivan, R.F. *Alternating Cold and Hot Aisles Provides More Reliable Cooling for Server Farms*. White paper, Uptime Institute, 2000; <http://www.dataclean.com/pdf/AlternColdnew.pdf>

Parthasarathy Ranganathan (Partha.Ranganathan@hp.com) is a distinguished technologist in Hewlett-Packard Labs, Palo Alto, CA.

DOI:10.1145/1721654.1721674

Cryptographic protocols safeguard the privacy of user queries to public databases.

BY SERGEY YEKHANIN

Private Information Retrieval

THE UBIQUITY OF the Internet means a plethora of online public databases and an indispensable resource for retrieving up-to-date information. But it also poses a significant risk to user privacy, since a malicious database owner may monitor user queries and infer what the user is after. Indeed, in cases where user intentions are to be kept secret, users are often cautious about accessing public databases. For example, investors querying a stock-market database for the current market value of certain stocks might prefer not to reveal their interest in the stocks because it could inadvertently influence their price. Alternatively, companies might want to search for certain patents without revealing the patents' identities.

Private information retrieval (PIR) schemes are cryptographic protocols designed to safeguard the privacy of database users. They allow clients to retrieve records from public databases while completely hiding the identity of the retrieved records from database owners. The possibility of retrieving

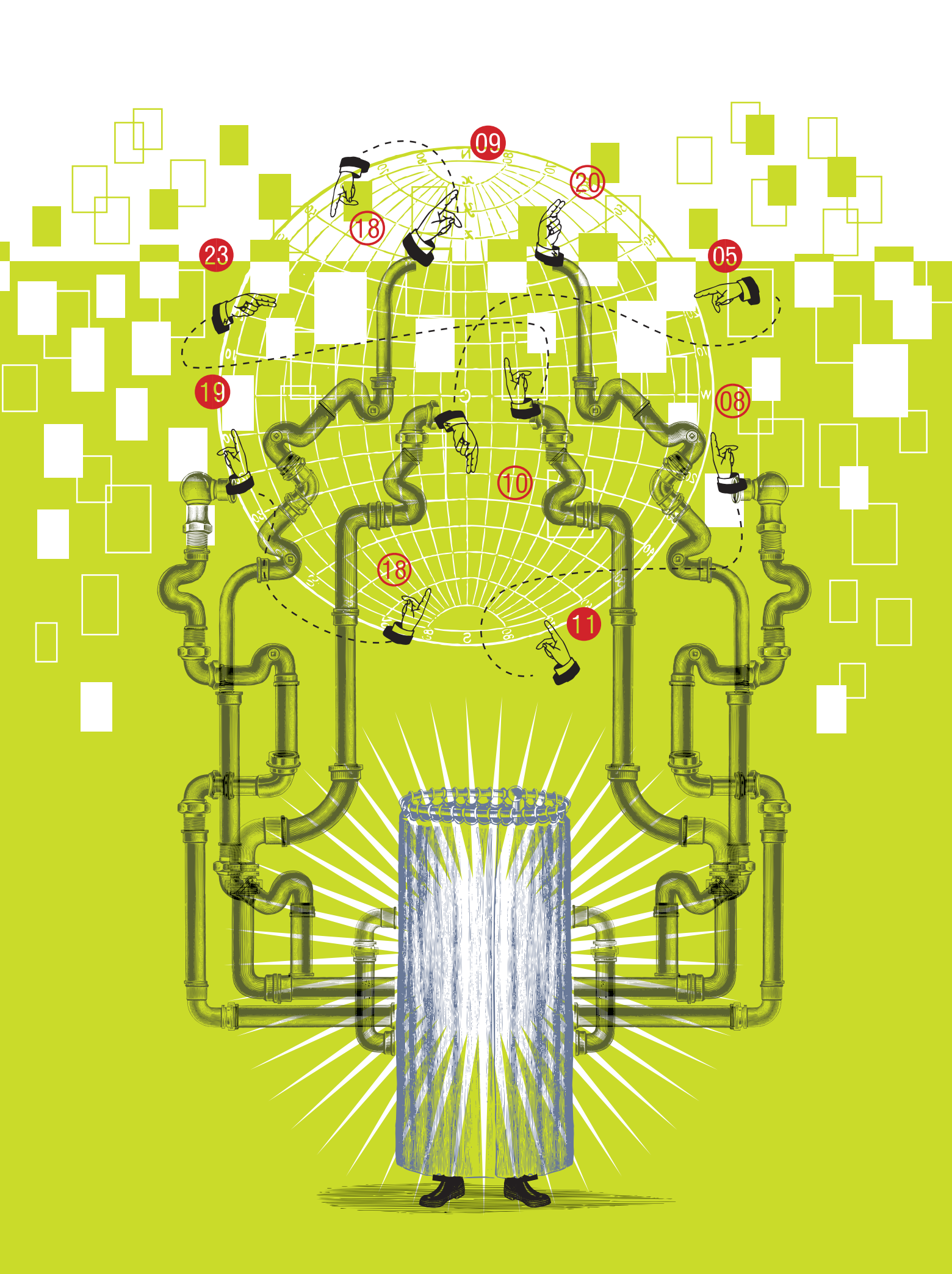
database records without revealing their identities to the owner of the database may seem beyond hope. Note, however, that a trivial solution is available: When users want a single record, they can ask for a copy of the whole database. This solution involves enormous communication overhead and is likely to be unacceptable. It turns out⁷ that for users who want to keep their privacy fully protected (in the “information-theoretic” sense), this trivial solution is optimal.

Fortunately, the negative result applies only to databases stored on a single server, rather than those replicated across several servers. In 1995, Chor et al.⁷ came up with PIR schemes that enable private retrieval of records from replicated databases, with a non-trivially small amount of communication. In such protocols, users query each server holding the database. The protocol ensures that each individual server (by observing only the query it receives) gets no information about the identity of the items of user interest. Chor et al.'s seminal paper⁷ triggered an important and extensive body of follow-up work.^{1,3,4,8,12,18,21,23,24}

Computational PIR. PIR schemes discussed here are often called “information theoretic” because they provide an absolute guarantee that each server participating in protocol execution gets no information about what users are after. PIR has also been studied in a computational setting.^{6,10,15,16} Computational PIR schemes are similar to their information-theoretic counterparts but provide a weaker

» key insights

- **User privacy is at risk whenever owners of publicly available network resources are able to trace, record, and mine user activity.**
- **The digital ecosystem involves not only such threats to personal privacy but also promising cures.**
- **Information-theoretic protocols decompose each user's query into several subqueries, ensuring they include no information about the user's intent.**



guarantee. Specifically, they ensure only that a server cannot get any information about user intent unless it solves a certain computationally hard problem (such as factoring a large random integer). Providing privacy or security guarantees based on computational hardness assumptions is common in modern cryptography. In contrast to information-theoretic PIR schemes, computational PIR protocols with low communication overhead exist (under standard assumptions), even when a database is stored on a single server (see the figure here). However, for typical real-life parameters, known computational protocols are less efficient than known information-theoretic ones.

Locally decodable codes. PIR schemes are intimately related to a special class of error-correcting codes called “locally decodable codes,” or LDCs, that, on their own, are objects of interest. All recent constructions of information-theoretic PIR schemes work by first constructing LDCs, then converting them to PIRs.

Error-correcting codes help ensure reliable transmission of information over noisy channels, as well as reliable storage of information on a medium that may be partially corrupted over time or whose reading device is subject to errors. Such codes allow one to add redundancy, or bit strings, to messages, encoding them into longer bit strings, called “codewords,” in a way that the message can still be recovered even if a certain fraction of the codeword bits are corrupted. In typical applications of error-correcting codes the message is first partitioned into small blocks, each of which is then encoded separately.

This encoding strategy allows efficient random-access retrieval of the information, since one must decode only the portion of data in which one is interested. Unfortunately, this strategy yields poor noise resilience, since, when even a single block (out of possibly tens of thousands) is completely corrupted, some information is lost. In view of this limitation it would seem preferable to encode the whole message into a single codeword of an error-correcting code. Such a solution improves the robustness to noise but is hardly satisfactory, since one needs to look at the whole codeword in order to recover any particular bit of the message (at least when using classical error-correcting codes). Such decoding complexity is prohibitive for today’s massive data sets.

LDCs simultaneously provide efficient random-access retrieval and high noise resilience by allowing reliable reconstruction of an arbitrary bit of the message from looking at only a small number of randomly chosen codeword bits. Local decodability comes at the price of certain loss in terms of code efficiency. Specifically, LDCs require longer codeword lengths than their classical counterparts. Though LDCs were discussed in the literature^{2,19} in the early 1990s, the first formal definition of LDCs was given in 2000 by Katz and Trevisan.¹³ Further work on the efficiency of LDCs was covered in.^{3,4,8,12,14,18,24}

Outline. Computational and information theoretic PIR schemes rely on different sets of techniques. The main focus in this article is information-theoretic schemes, reserving the term PIR for information-theoretic protocols. In the following section we make

the notions of information-theoretic PIR and LDCs more concrete, explaining the relationship between them. Later, we demonstrate how nontrivial PIR is possible, offering an exposition of one of the simplest and earliest schemes. We then present the most basic PIR protocol of the current generation. Finally, we discuss computational PIR protocols and an early single-server scheme due to Kushilevitz and Ostrovsky.¹⁵

Preliminaries

Beginning with a (slightly informal) definition of PIR schemes, we model the database as an n -bit string x that is replicated between a small number k of non-communicating servers S_1, \dots, S_k . The user holds an index i (an integer between 1 and n) and is interested in obtaining the value of the bit x_i . To achieve this goal, the user tosses several random coins, queries each server, and receives replies from which the desired bit x_i is computed. The query to each server is distributed independently of i ; therefore, each server gets no information about what the user is after.

Note that user queries are not necessarily requests for particular individual bits or sets of database bits. Rather, they specify functions computed by the servers; for example, a query may specify a set of indices between 1 and n , and the server’s response may be the XOR of the database bits stored at these indices.

Historically, the main parameter of interest in PIR schemes was communication complexity, or a function of n measuring the total number of bits communicated between user and servers, maximized over all choices of x in $\{0, 1\}^n$, i in 1 to n , and the user’s random coin tosses.

The most efficient two-server PIR protocols available today have communication complexity of $O(n^{1/3})$. These protocols, which are due to Chor et al.,⁷ have never been improved upon. However, PIR schemes involving three or more servers have seen improvement.^{1,4,7,8,12,24} The best-known are due to Efremenko⁸ and Itoh and Suzuki.¹² Efremenko’s three-server PIR schemes require $f(n)$ bits of communication, where $f(n)$ is a certain function that grows slower than any



Two-server information-theoretic PIR scheme. To retrieve a database record, the user queries two servers, each of which stores a copy of the database, but individual queries carry no information about what the user is after. The desired record is obtained from the servers’ combined responses.

polynomial in n but faster than any power of the logarithm of n . The best lower bound for the communication complexity of two-server PIR is $5 \log n$ due to Wehner and de Wolf.²¹ Closing the gap between upper and lower bounds for PIR communication complexity is a major open problem.


Now we address a more detailed treatment of LDCs that have three parameters: k , δ , and ε . A (k, δ, ε) -LDC encodes n -bit messages x to N -bit codewords $C(x)$, such that for every i between 1 and n the bit x_i can be recovered with probability $1 - \varepsilon$ by a randomized decoding procedure that reads only k codeword bits, even after the codeword $C(x)$ is adversarially corrupted in up to δN locations. Here, the probability is only over the decoder's random coin tosses.

We illustrate the notion of LDCs by presenting a $(2, \delta, 2\delta)$ -Hadamard LDC that encodes n -bit messages to 2^n -bit codewords. In what follows, let $[n]$ denote the set $\{1, \dots, n\}$.


Hadamard code. Every coordinate in the Hadamard code corresponds to one (of 2^n) subsets of $[n]$ and stores the XOR of the corresponding bits of the message x . Let y be an (adversarially corrupted) encoding of x . Given an index $i \in [n]$ and y , the Hadamard decoder picks a set S in $[n]$ uniformly at random and outputs the XOR of the two coordinates of y corresponding to sets S and $S \Delta \{i\}$. (Here, Δ denotes the symmetric difference of sets (such as $\{1, 4, 5\} \Delta \{4\} = \{1, 5\}$, and $\{1, 4, 5\} \Delta \{2\} = \{1, 2, 4, 5\}$). It's not difficult to verify that both decoders' queries go to uncorrupted locations if y differs from the correct encoding of x in at most δ fraction of coordinates than with probability $1 - 2\delta$. In such cases, the decoder correctly recovers the i -th bit of x .

The Hadamard code allows for a super-fast recovery of the message bits (such as, given a codeword corrupted in 0.1 fraction of coordinates, one is able to recover any bit of the message with probability 0.8 by reading only two codeword bits) at a price of very large codeword length. Designing LDCs with optimal, or smallest possible, codeword length is a major challenge.

The definition of LDCs implies that every bit x_i of the message can



The protocol ensures that every individual server (by observing only the query it receives) gets no information about the identity of the items of user interest.



be recovered from many different k -tuples of codeword bits. Note that such k -tuples cannot all belong to a small ($o(N)$ -size) subset S of codeword coordinates, since corrupting the coordinates in S could lead to the high probability of a decoding error. Thus the distribution of every individual query of the decoder must be somewhat close to a uniform distribution on the codeword bits. Indeed, many known LDCs have decoders whose individual queries are distributed perfectly uniformly; such LDCs are called “perfectly smooth.”

There is a strong relationship between LDCs and PIR schemes. Short LDCs yield efficient PIR schemes and vice versa. Here, we demonstrate the flavor of this relationship, presenting a general procedure that obtains a k -server PIR scheme from any perfectly smooth k -query LDC.

Let C be a perfectly smooth LDC encoding n -bit messages to N -bit codewords. At the preprocessing stage, servers S_1, \dots, S_k encode the n -bit database x with the code C . Next, a user interested in obtaining the i -th bit of x tosses random coins and generates a k -tuple of queries (q_1, \dots, q_k) , such that x_i can be computed from $C(x)_{q_1}, \dots, C(x)_{q_k}$. For every j in $[k]$, the user sends the query q_j to the server S_j . Each server S_j responds with a one-bit answer $C(x)_{q_j}$. The user combines the servers' responses to obtain x_i .

Verifying that the protocol is private is straightforward, since for every j in $[k]$ the query q_j is uniformly distributed over the set of codeword coordinates; the total communication is given by $k(\log N + 1)$.

Early PIR Scheme

Let d be a small integer. Our goal here is to demonstrate how nontrivial PIR is possible by presenting a simple $(d+1)$ -server scheme with $O(n^{1/d})$ communication to access an n -bit database. The key idea behind this scheme is polynomial interpolation in a finite-field setting.

We begin with some technical background. Let $p > d$ be a prime. It is well known that addition and multiplication of numbers $\{0, \dots, p-1\}$ modulo p satisfy the standard identities that one is used to over the real numbers. That is, numbers $\{0, \dots, p-1\}$ form a finite

field with respect to these operations. This field is denoted by F_p . In what follows we deal with polynomials defined over finite fields. Such polynomials have all algebraic properties that one is used to with polynomials over the real numbers. Specifically, a univariate polynomial over F_p of degree d is uniquely determined by its values at any $d + 1$ points.

Let m be a large integer. Let E_1, \dots, E_n be a certain collection of n vectors over F_p of dimension m . The collection is fixed and independent of the n -bit database x . We assume the collection is known to both the servers and the user. On the preprocessing stage of the PIR protocol, each of $(d + 1)$ servers represents the database x by the same degree d polynomial f in m variables. The key property of such a polynomial is that for every i in $[n] : f(E_i) = x_i$. In order to ensure that such a polynomial f exists we choose m to be reasonably large compared to n . Setting $m = O(n^{1/d})$ suffices.

Now suppose the user wants to retrieve the i -th bit of the database and knows the collection of vectors E_1, \dots, E_n . The user's goal is thus to recover the value of the polynomial f (held by the servers) at E_i . Obviously, the user cannot explicitly request the value of f at E_i from any of the servers, since such a request would ruin the privacy of the protocol; that is, some server will get to know which database bit the user is after. Instead, the user obtains the value of $f(E_i)$ indirectly, relying on the rich structure of local dependencies between the evaluations of a low-degree polynomial f at multiple points. Specifically, the user generates a randomized collection of m -dimensional vectors P_1, \dots, P_{d+1} over F_p such that:

1. Each of the vectors P_λ is individually uniformly random and thus provides no information about E_i ; and
2. The values of any degree d polynomial (including the polynomial f) at P_1, \dots, P_{d+1} determine the value of the polynomial at E_i .

The user sends each server one of the vectors P_1, \dots, P_{d+1} . The servers then evaluate the polynomial f at the vectors they receive and return the values they obtain back to the user. The user combines the values $f(P_1), \dots, f(P_{d+1})$ to get the desired value $f(E_i)$. The protocol is perfectly private, and the com-

munication amounts to sending $(d + 1)$ vectors of dimension m to the servers and a single value back to the user. Here, we elaborate on how vectors P_1, \dots, P_{d+1} are chosen.

The user picks an m -dimensional vector V uniformly at random and for every λ between 1 and $d+1$ sets $P_\lambda = E_i + \lambda V$. Clearly, every individual vector P_λ is uniformly random. To see that values of $f(P_1), \dots, f(P_{d+1})$ determine the value of $f(E_i)$ consider a univariate polynomial $g(\lambda) = f(E_i + \lambda V)$. Note that the degree of g is at most d . Therefore the values of g at $d + 1$ points determine the polynomial g uniquely. It remains to notice that $g(\lambda) = f(P_\lambda)$ and $g(0) = f(E_i) = x_i$.

This PIR scheme is a classical example of PIR schemes. The ideas behind it can be traced back to Babai et al.² The idea of obtaining PIR schemes and LDCs through polynomials was used extensively thereafter.^{1,3,7} The most powerful constructions in this framework are due to Beimel et al.⁴; see also Woodruff and Yekhanin.²³

Modern LDCs and PIRs

In 2007, the author of the current article proposed a new approach to designing LDCs and PIR schemes,²⁴ leading to development of today's most effective codes and schemes.^{8,12,18} The new constructions are not based on polynomials; their key technical ingredient is a design of a large family of sets with restricted intersections. Here, we offer a bird's-eye view of the construction of LDCs. Such codes are easily turned into PIR schemes.

Let k be a small integer. Here, we design a binary k -query LDC C that encodes n -bit messages to codewords. This construction involves two steps: The first is a reduction to a problem of constructing a certain family of sets with restricted intersections; the second is an algebraic construction of the desired family.

Step 1. Construct an F_2 -linear code C , so:

► C is an F_2 -linear map. For any two messages x_1, x_2 in F_2^n we have $C(x_1 + x_2) = C(x_1) + C(x_2)$, where the sums of vectors are computed modulo two in each coordinate; and

► The decoding algorithm proceeds by tossing random coins, reading a certain k -tuple of coordinates of the cor-

rupted codeword and outputting the XOR of the values in these coordinates.

For i in $[n]$, let e_i denote a binary n -dimensional vector, whose unique non-zero coordinate is i . Observe that every linear LDC admits a combinatorial description. That is, to define a linear LDC it is sufficient to specify for every i in $[n]$:

► A set T_i of coordinates of $C(e_i)$ that are set to 1. Such sets completely specify the encoding, since for any message x , $C(x) = \sum_{i:x_i=1} C(e_i)$; and

► A family Q_i of k -size subsets of codeword coordinates that can be read by a decoding algorithm while reconstructing the i -th message bit.

Not every collection of sets T_i and families Q_i yields an LDC. Certain combinatorial constraints must be satisfied. The basic rationale for these constraints is the following:

1. The decoding must be correct in case no codeword bits have been corrupted. This implies that for every i, j in $[n]$ and any k -set S in Q_i the size of $S \cap T_j$ must be odd if $i = j$ and even otherwise; and

2. The distribution of individual queries of the decoding algorithm must be close to uniform. This implies that for every i in $[n]$, the union of the k -sets in Q_i must be large relative to the number of codeword coordinates.

Step 2. Design a collection of sets T_i and families Q_i that satisfy these constraints. This is where most of the technical work occurs. The construction is supported by a strong geometric intuition. We consider a bijection between the set of codeword coordinates and a set of m -dimensional vectors over a finite field of cardinality k . We choose sets T_i to be unions of certain parallel hyperplanes in the m -dimensional linear space over F_k and families Q_i to be certain families of lines. (Note that lines over F_k have only k points.) We use basic algebra to argue about the intersection sizes.

Computational PIR

Computational PIR schemes are attractive because they avoid the need to maintain replicated copies of a database and do not compromise user privacy against colluding servers. Here, we share a high-level overview (the basic version) of the Kushilevitz and

Ostrovsky scheme that relies on the standard hardness assumption¹⁵—the “quadratic residuosity assumption.”

Let m be a positive integer. A number a is called a “quadratic residue,” or QR, modulo m , if there exists an integer x such that $x^2 = a \pmod m$. Otherwise, a is called a “quadratic non-residue,” or QNR, modulo m . The QR assumption states that it is computationally hard to distinguish numbers that are QRs modulo m from those that are not, unless one knows the factorization of m .

The protocol. The server stores the n -bit database x in a square matrix of size s by s for $s = \sqrt{n}$.

$$\begin{bmatrix} X_{11} \cdots X_{1j} \cdots X_{1s} \\ \vdots \\ X_{i1} \cdots X_{ij} \cdots X_{is} \\ \vdots \\ X_{s1} \cdots X_{sj} \cdots X_{ss} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ b_i \\ \vdots \\ a_s \end{bmatrix}$$

Suppose the database user is interested in obtaining the value x_{ij} for some i, j between 1 and s . The user would select a large integer m together with its factorization at random and generate $s - 1$ random QRs $a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_s$ modulo m , as well as a single random QNR b_i . The user would then send the string $a_1, \dots, a_{i-1}, b_i, a_{i+1}, \dots, a_s$ and the integer m to the server.

The QR assumption implies that the server cannot distinguish b_i from integers $\{a_l\}$ and thus observes only a string of s random-looking integers u_1, \dots, u_s modulo m (one per each row of the database). The server responds with s integers π_1, \dots, π_s (one per each column of the database). Here each π_h is equal to a product of integers u_l for all l such that $x_{lh} = 1$; formally, $\pi_h = \prod_l \mid_{x_{lh}=1} u_l \pmod m$.

Verifying that the value of π_j is going to be a QR modulo m if $x_{ij} = 0$ and is going to be a QNR modulo m is not hard if $x_{ij} = 1$, since a product of two QRs is a QR and the product of a QR with a QNR is a QNR. The user need only check whether π_j is a QR, which is easy, since the user knows the factorization of m .

The total amount of communication in this PIR scheme is roughly $O(\sqrt{n})$. Better protocols are available; see Gentry and Ramzan¹⁰ and Lipmaa¹⁶ for the most efficient computational PIR schemes.

Conclusion

With nearly 15 years of development, the PIR field has grown large and deep, with many subareas and connections to other fields. For more details, see Ostrovsky and Sketch¹⁷ for a survey of computational PIR and Trevisan²⁰ and Yekhanin²⁵ for a survey of information theoretic PIR; see also Gasarch.⁹

Here, we have concentrated on a single (the most studied) aspect of PIR schemes—their communications complexity. Another important aspect of PIR schemes is the amount of computation servers must perform in order to respond to user queries.^{5,11,23} We are hopeful that further progress will lead to the wide practical deployment of these schemes.

Acknowledgments

We would like to thank Martin Abadi, Alex Andoni, Mihai Budiu, Yuval Ishai, Sumit Nain, and Madhu Sudan and the anonymous referees for helpful comments regarding this article. □

References

1. Ambainis, A. Upper bound on the communication complexity of private information retrieval. In *Proceedings of the Automata, Languages and Programming, 24th International Colloquium LNCS 1256* (Bologna, Italy, July 7–11). Springer, 1997, 401–407.
2. Babai, L., Fortnow, L., Levin, L., and Szegedy, M. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing* (New Orleans, LA, May 5–8). ACM Press, New York, 1991, 21–31.
3. Beimel, A., Ishai, Y., and Kushilevitz, E. General constructions for information-theoretic private information retrieval. *Journal of Computer and System Sciences* 71, 2 (2005), 213–247.
4. Beimel, A., Ishai, Y., Kushilevitz, E., and Raymond, J.F. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In *Proceedings of the 43rd Symposium on Foundations of Computer Science* (Vancouver, B.C., Nov. 16–19). IEEE Computer Society, 2002, 261–270.
5. Beimel, A., Ishai, Y., and Malkin, T. Reducing the servers’ computation in private information retrieval: PIR with preprocessing. In *Proceedings of the 20th Annual International Cryptology Conference LNCS 1880* (Santa Barbara, CA, Aug. 20–24). Springer, 2000, 55–73.
6. Cachin, C., Micali, S., and Stadler, M. Computationally private information retrieval with polylogarithmic communication. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques LNCS 1592* (Prague, Czech Republic, May 2–6). Springer, 1999, 402–414.
7. Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. Private information retrieval. *Journal of the ACM* 45, 6 (1998), 965–981.
8. Efremenko, K. 3-query locally decodable codes of subexponential length. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing* (Bethesda, MD, May 31–June 2). ACM Press, New York, 2009, 39–44.
9. Gasarch, W. Web page on private information retrieval; <http://www.cs.umd.edu/~gasarch/pir/pir.html>
10. Gentry, C. and Ramzan, Z. Single-database private information retrieval with constant communication rate. In *Proceedings of the 32nd International Colloquium on Automata, Languages and*

- Programming LNCS 3580* (Lisbon, Portugal, July 11–15). Springer, 2005, 803–815.
11. Ishai, Y., Kushilevitz, E., Ostrovsky, R., and Sahai, A. Batch codes and their applications. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing* (Chicago, June 13–16). ACM Press, New York, 2004, 262–271.
12. Itoh, T. and Suzuki, Y. New constructions for query-efficient locally decodable codes of subexponential length. *IEICE Transactions on Information and Systems E93-D*, 2 (2010), 263–270.
13. Katz, J. and Trevisan, L. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing* (Portland, OR, May 21–23). ACM Press, New York, 2000, 80–86.
14. Kerenidis, I. and de Wolf, R. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Sciences* 69, 3 (2004), 395–420.
15. Kushilevitz, E. and Ostrovsky, R. Replication is not needed: Single-database computationally private information. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science* (Miami Beach, FL, Oct. 19–22). IEEE Computer Society, 1997, 364–373.
16. Lipmaa, H. An oblivious transfer protocol with log-squared communication. In *Proceedings of the 11th International Conference on Information Security LNCS 5222* (Taipei, Sept. 15–18). Springer, 2008, 314–328.
17. Ostrovsky, R. and Sketch, W.E. III A survey of single database private information retrieval: Techniques and applications. In *Proceedings of the 10th International Conference on Practice and Theory in Public-Key Cryptography LNCS 4450* (Beijing, Apr. 16–20). Springer, 2007, 393–411.
18. Raghavendra, P. A Note on Yekhanin’s Locally Decodable Codes Technical Report TR07-016. Electronic Colloquium on Computational Complexity, 2007.
19. Sudan, M. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. Ph.D. thesis, University of California, Berkeley, 1992.
20. Trevisan, L. Some applications of coding theory in computational complexity. *Quaderni di Matematica* 13 (2004), 347–424.
21. Wehner, S. and de Wolf, R. Improved lower bounds for locally decodable codes and private information retrieval. In *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming LNCS 3580* (Lisbon, Portugal, July 11–15). Springer, 2005, 1424–1436.
22. Woodruff, D. *New Lower Bounds for General Locally Decodable Codes Technical Report TR07-006*. Electronic Colloquium on Computational Complexity, 2007.
23. Woodruff, D. and Yekhanin, S. A geometric approach to information theoretic private information retrieval. *SIAM Journal on Computing* 47, 4 (2007), 1046–1056.
24. Yekhanin, S. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM* 55, 1 (2008), 1–14.
25. Yekhanin, S. *Locally Decodable Codes and Private Information Retrieval Schemes*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 2007; <http://research.microsoft.com/en-us/um/people/yekhanin/Papers/phdthesis.pdf>

Sergey Yekhanin (yekhanin@microsoft.com) is a researcher in Microsoft Research Silicon Valley, Mountain View, CA.

Combining the paradigm features of both logic and functional programming makes for some powerful implementations.

BY SERGIO ANTOY AND MICHAEL HANUS

Functional Logic Programming

THE EVOLUTION OF programming languages is the stepwise introduction of abstractions hiding the underlying computer hardware and the details of program execution. Assembly languages introduce mnemonic instructions and symbolic labels for hiding machine codes and addresses. Fortran introduces arrays and expressions in standard mathematical notation for hiding registers. Algol-like languages introduce structured statements for hiding *gotos* and jump labels. Object-oriented languages introduce visibility levels and encapsulation for hiding the representation of data and the management of memory. Along these lines, declarative languages—the most prominent representatives of which are functional and logic languages—hide the order of evaluation by removing assignment and other control statements. A declarative program is a set of logical statements describing properties of the application domain. The execution of a declarative program is the computation of the value(s) of an expression with

respect to these properties. Thus, the programming effort shifts from encoding the steps for computing a result to structuring the application data and the relationships between the application components.

Declarative languages are similar to formal specification languages, but with a significant difference: they are *executable*. The language that describes the properties of the application domain is restricted to ensure the existence of an efficient evaluation strategy. Different formalisms lead to different classes of declarative languages. *Functional languages* are based on the notion of mathematical function; a functional program is a set of functions that operate on data structures and are defined by equations using case distinction and recursion. *Logic languages* are based on predicate logic; a logic program is a set of predicates defined by restricted forms of logic formulas, such as Horn clauses (implications).

Both kinds of languages have similar motivations but provide different features. For example, functional languages provide efficient, demand-driven evaluation strategies that support infinite structures, whereas logic languages provide nondeterminism and predicates with multiple input/output modes that offer code reuse. Since all these features are useful for developing software, it is worthwhile to amal-

» key insights

- Professionals involved in software development should follow the potential capabilities of functional logic programming, as it is an emerging programming paradigm offering novel ways to develop software.
- Functional logic programming supports specification, prototyping, and application programming within a single language. It is the basis of a practical and convenient approach to produce high-quality software.
- Functional logic programming is terse yet clear, supports rapid development by avoiding some tedious tasks, and allows incremental refinements to improve efficiency. It makes programming fun without sacrificing reliability.



Generative artist and programmer David Bollinger uses math and algorithms to create artwork. The illustrations in this article reflect n -unit cubes recursively subdivided based on patterns defined by the greatest common divisor among the coordinate axes.

gamate them into a single language. *Functional logic languages* combine the features of both paradigms in a conservative manner. Programs that do not use the features of one paradigm behave as programs of the other paradigm. In addition to the convenience of using the features of both paradigms within a single language, the combination has additional advantages. For instance, the demand-driven evaluation of functional programming applied to nondeterministic operations of logic programming leads to more efficient search strategies. The effort to develop languages intended to meet this goal has produced a substantial amount of research spanning two decades (see Hanus²² for a recent survey). These achievements are reviewed here.

The concrete syntax of the examples is Curry,²⁷ but the design of the code and most of our considerations about the programs are valid for any other functional logic language, for example, *TOY*,³⁰ based on reduction for functional evaluation and on narrowing for the instantiation of unbound logic variables.

A common trait of functional and logic languages is the distinction between data constructors and defined operations. This distinction is also essential for functional logic languages to support reasonably efficient execution mechanisms. *Data constructors* build the data underlying an application, whereas *defined operations* manipulate the data. For instance, `True` and `False` construct (are) Boolean values. A simple example of a more

complex data structure is a list of values. We denote by `[]` the empty list and by `(x:xs)` a non-empty list where `x` is the first element and `xs` is the list of remaining elements. We create longer lists by nested applications of constructors: `(1:(2:(3:[])))` denotes a list with elements 1, 2, and 3. Since lists are ubiquitous structures in declarative languages, syntactic sugar is usually provided to ease writing lists; thus, `[1,2,3]` abbreviates the previous list structure.

A further feature common to many functional and logic languages is the definition of operations by *pattern matching*. Functions are specified by different equations for different argument values. For instance, a function, “++”, returning the concatenation of two input lists is defined as follows

(here we use infix notation for “++”):

```
[] ++ ys = ys
(x:xs) ++ ys = x : (xs ++ ys)
```

The first equation is applicable to calls to “++” with an empty list as the first argument. The second equation is used to evaluate calls with non-empty lists. The pattern $(x:xs)$ of the second equation combines case distinction (the list is not empty) and selectors (let x and xs be the head and tail of the argument list) in a compact

way. The meaning of this definition is purely equational; expressions are evaluated by replacing left-hand sides by corresponding right-hand sides (after substituting the equation’s variables with corresponding actual arguments). These replacement steps are also called *reductions*, and the defining equations are often called *rules*, since they are applied from left to right.

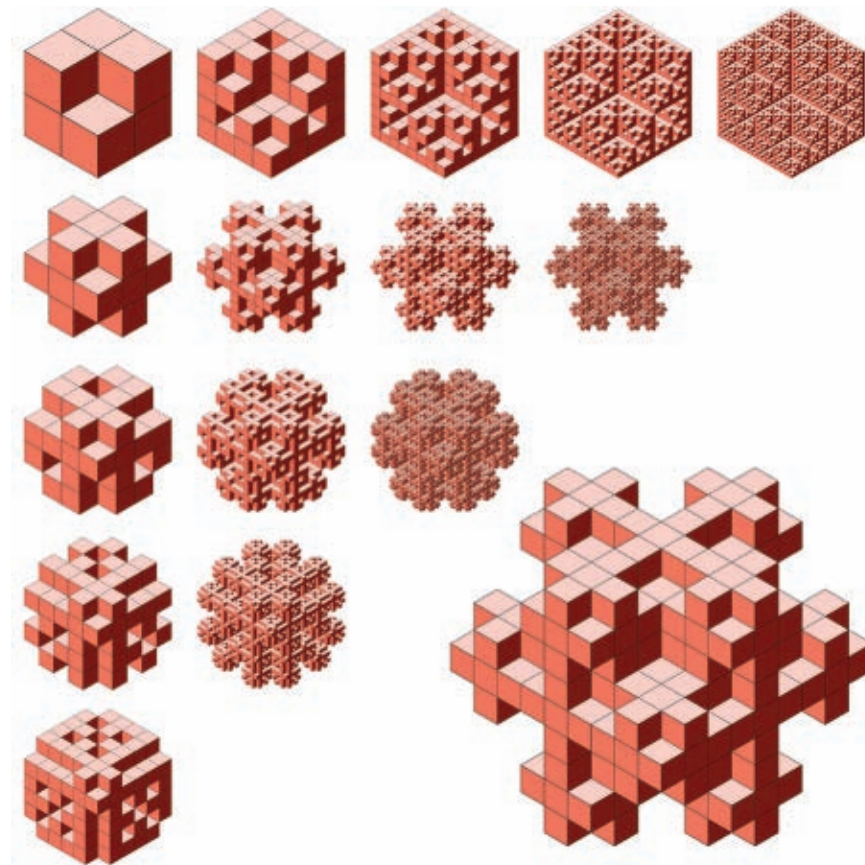
```
(xs ++ ys) ++ zs = xs ++ (ys ++ zs)
```

Such an equation describes a property (associativity) of the operation “++” rather than a constructive definition about its evaluation. The restriction to constructor-based rules is an important factor to support execution

the conditions on e constrain the values that e may assume. For instance, consider the equation $zs++[2] ::= [1,2]$. (We use the symbol “ $::=$ ” for equations that should be solved in order to syntactically distinguish them from equations that define operations.) This equation states that we are interested only in the values for the variable zs that satisfy the equation. In this case we have to replace or instantiate zs with the list $[1]$. The combination of variable instantiation and term reduction is called *narrowing*, originally introduced in automated theorem proving³⁷ and first proposed for programming in Reddy.³⁵ In general, there might be infinitely many instantiations for the free variables of an expression. The research on functional logic languages has led to reasonable narrowing strategies that avoid a blind guessing of all the potential values of a variable.⁴

Good strategies, as discussed later, perform only “constructive” guessing steps: They instantiate free variables only if the instantiation is necessary to sustain a computation and only with those values that are demanded to perform a reduction step. For instance, in the evaluation of $zs++[2]$, zs is replaced either by the empty list $[]$ or by a non-empty list $(x:xs)$ where the head and tail are again unknown. Either instantiation enables a reduction step with one of the rule defining “++”. Not every instantiation will lead to a result; thus, some instantiation might be later discarded. Good narrowing strategies ensure the cost of an instantiation is incurred only when a guess for an unknown value is necessary. Expressions without unknowns or with unknowns that do not need to be known are evaluated as in functional languages.

The capability to compute with unknowns supports new forms of code reuse. As we have seen in the equation, we can compute the prefix of a list by narrowing. Similarly, the equation $zs++[e] ::= [1,2,3]$ is solved by instantiating e with the last element of the right-hand side list (and zs with the remaining list). Thus, we can reuse the concatenation operation “++” to compute the last element of a list. We can also define an explicit function `last` for this purpose:



with efficient strategies. “Execution” in functional languages means reducing expressions containing defined operations to *values* (that is, expressions without defined operations) by applying defining equations (“replacing equals by equals”).

The *logic* component of functional logic languages comes into play when computing with incomplete information. The problem is to evaluate an expression containing a subexpression e such that the value of e is unknown, but it is known that e must satisfy certain conditions. In a computation, e is represented by a free variable and

The problem is to evaluate an expression containing a subexpression e such that the value of e is unknown, but it is known that e must satisfy certain conditions. In a computation, e is represented by a free variable and

is represented by a free variable and

```
last xs | zs++[e] := xs
  = e
  where zs,e free
```

Here, we add both a *condition* to a defining rule, so that this rule is applicable only if the condition (the equation between “|” and “:=”) can be solved, and a declaration of the variables introduced by the rule (the *where ... free* clause). Although code reuse originating from the evaluation of operations with unknown arguments is well known in logic programming, functional logic languages provide additional structures for reusing code. For instance, it is apparent from the rule defining `last` that this rule is applicable only if the actual argument has a form that matches the result of narrowing `zs++[e]`. Thus, we can reformulate that rule as:

```
last (zs++[e]) = e
```

Note that purely functional languages, such as Haskell, do not allow this rule because it is not constructor-based; rather it contains a *functional pattern*, that is, a pattern with a defined function inside. When a rule of this kind is applied to some function call, the functional pattern is evaluated (by narrowing) to some value (which likely contains variables), which is then unified with the actual argument. Since the functional pattern `zs++[e]` can be narrowed to infinitely many values (`[e]` for `zs=[]`, `[x1,e]` for `zs=[x1],...`), it abbreviates an infinite number of ordinary patterns. Similarly to narrowing, it is not necessary to guess all these patterns at runtime. Instead, the necessary patterns can be computed in a demand-driven manner during pattern matching.⁷

These examples show the potential of functional logic languages: writing programs as clear specifications that abstract from the evaluation order, contain unknowns, and reuse defined operations in various ways. Of course, there is a price to pay for these advanced features. Dealing with unknowns requires the consideration of different alternatives during runtime; in other words, computations might be nondeterministic. This nondeterminism is *don't-know*, and finding all the solutions of a problem may require considering

A common trait of functional and logic languages is the distinction between data constructors and defined operations. This distinction is also essential for functional logic languages to support reasonably efficient execution mechanisms.

many nondeterministic alternatives. To make nondeterminism practically useful, the number of alternatives that must be considered by the execution of a program should be contained. This is provided by the evaluation strategy discussed in the “Strategy” section.

Curry

Curry²⁷ is a functional logic language developed by an international community of researchers to produce a standard for research, teaching, and application of functional logic programming (details can be found at www.curry-language.org). Here, we give an overview of Curry with emphasis on aspects relevant to functional logic programming.

The syntax of Curry borrows heavily from that of Haskell.³⁴ In fact, Curry mainly introduces a single syntactic extension (the declaration of free variables) with respect to Haskell, although the underlying evaluation strategy is different (as we will show). Thus, a Curry *program* is a set of definitions of *data types* and operations on values of these types. A data type is declared by enumerating all its constructors with the respective argument types. For example:

```
data Bool = True | False
data BTree a = Leaf a
             | Branch (BTree a) (BTree a)
```

The type `BTree` is *polymorphic*, that is, the type variable `a` ranges over all possible type expressions, and it has two constructors `Leaf` and `Branch`. Operations on polymorphic data structures are often generic. For instance, the following operation computes the number of nodes (branches and leaves) of a binary tree, where the (optional) first line defines the type signature:

```
size :: BTree a -> Int
size (Leaf _) = 1
size (Branch t1 t2) =
  1 + size t1 + size t2
```

Thus, `size (Branch (Leaf 1) (Leaf 3))` evaluates to 3. As in Haskell, the names of (type) variables and operations usually start with lowercase letters, whereas the names of type and data constructors start with an uppercase letter. The application of f to e is

denoted by juxtaposition ($f e$), except for infix operators such as “+”. Curry, in contrast to Haskell, may use the operation `size` to compute binary trees of a particular size. For example, if t is a free variable, the evaluation of `size t := 3` instantiates t to the tree structure `(Branch (Leaf x1) (Leaf x2))`, where $x1$ and $x2$ are free variables that remain unspecified because their values do not affect the equation.

As discussed earlier, finding solutions to an underspecified problem requires the nondeterministic evaluation of some expression. The ability to perform nondeterministic computations supports an interesting language feature, namely the definition of *nondeterministic operations*. These operations deliver more than one result with the intended meaning that one result is as good as any other. The archetype of nondeterministic operations is the binary infix operation “?”, called *choice*, that returns one of its arguments:


```
x ? y = x
x ? y = y
```

Thus, we can define the flipping of a coin as:


```
coin = 0 ? 1
```

Nondeterministic operations are unusual at first glance, but they are quite useful for programming. Note that the result of a nondeterministic operation is a single (indeterminate) value rather than the set of all possible values. Thus, the value of `coin` is not the set $\{0, 1\}$, but one element of this set. The “single-element” view is important to exploit demand-driven evaluation strategies for the efficient evaluation of such operations. For instance, it might not be necessary to compute all results of a nondeterministic operation if the context demands only values of a particular shape. A concrete example of this advantage is given in the following paragraphs. It should be noted that there exists a declarative foundation (such as, model-theoretic, big-step, and fixpoint semantics) for nondeterministic operations.¹⁹

To show a slightly more interesting example involving nondeterministic operations, consider the definition of



Although the call-time choice resembles an eager or call-by-value evaluation behavior, it fits well into the framework of demand-driven or lazy evaluation where arguments are shared to avoid the repeated evaluation of the same expression.



an operation that inserts an element into a list at an indeterminate position:

```
insert x ys = x : ys
insert x (y:ys) = y : insert x ys
```

Since both rules are applicable in evaluating a call to `insert` with a non-empty list, `insert 0 [1,2]` evaluates to any of $[0,1,2]$, $[1,0,2]$, or $[1,2,0]$. Thus, `insert` supports a straightforward definition of a permutation of a list by inserting the first element at some position of a permutation of the tail of the list:

```
perm [] = []
perm (x:xs) = insert x (perm xs)
```

Permutations are useful for specifying properties of other operations. For instance, a property of a sort operation on lists is that the result of sorting is a permutation of the input list where all the elements are in ascending order. Since functional logic languages can deal with several results of an operation as well as failing alternatives, it is reasonable to specify sorted lists by an operation `sorted` that is the identity only on sorted lists (and fails on lists that are not sorted):

```
sorted [] = []
sorted [x] = [x]
sorted (x:y:ys) | x <= y
                = x : sorted(y:ys)
```

Altogether, the expression “`sorted (perm xs)`” specifies the sorted permutation of a list xs . Since Curry evaluates nondeterministic definitions, this specification of sorting is executable. Although it seems that a complete strategy has to compute *all* the permutations, a good strategy does better than that. Modern functional logic languages, including Curry, use demand-driven strategies that do not always evaluate expressions completely. For instance, the argument `(perm xs)` of the expression `sorted (perm xs)` is evaluated only as demanded by `sorted`. If a permutation starts with out of order elements, as in `2:1:perm ys`, `sorted` will fail on this expression without further evaluating `perm ys`, that is, the evaluation of all the permutations of ys is avoided. This demand-driven search strategy reduces the

overall complexity of the computation compared to a simple generate-and-test strategy. Compared to other sorting algorithms, this program is still inefficient since we have not used specific knowledge about sorting collections of objects efficiently.

Nondeterministic operations promote concise definitions, as shown by `perm`. It is interesting to observe that narrowing, that is, computation with free variables that are nondeterministically instantiated, and the evaluation of nondeterministic operations without free variables, have the same expressive power. For instance, one can replace a free variable of type `Bool` in an expression by the nondeterministic operation `genBool` that is defined by

```
genBool = True ? False
```

so that it evaluates to any value of type `Bool`. The equivalence of narrowing with free variables and the evaluation of such nondeterministic generator functions is formally stated in Antoy and Hanus⁸ and the basis of a recent Curry implementation.¹³

Nevertheless, modern functional logic languages provide both features because both are convenient for programming. There is a subtle aspect to consider when nondeterministic expressions are passed as arguments to operations. Consider the operation:

```
double x = x+x
```

and the expression “`double coin`.” Evaluating the argument `coin` (to 0 or 1) before passing it to `double` yields 0 and 2 as results. If the argument `coin` is passed unevaluated to `double`, we obtain in one rewrite step the expression `coin+coin` which has four possible rewrite derivations resulting in the values 0, 1 (twice), and 2. The former behavior is referred to as *call-time choice* semantics²⁸ since the choice of the value of a nondeterministic argument is made at call time, whereas the latter is referred to as *need-time choice* semantics since the choice is made only when needed.

Although the call-time choice resembles an eager or call-by-value evaluation behavior, it fits well into the framework of demand-driven or lazy

evaluation where arguments are shared to avoid the repeated evaluation of the same expression. For instance, the actual argument (for example, `coin`) associated to the formal parameter `x` in the rule “`double x = x+x`” is not duplicated in the right-hand side. Rather both occurrences of `x` refer to the same term, which consequently is evaluated only once. This technique, called *sharing*, is essential to obtain efficient (and optimal) evaluation strategies. The call-time choice is the semantics usually adopted by current functional logic languages since it satisfies the principle of “least astonishment” in most situations.¹⁹

Curry also supports the use of constraint solvers, since the condition of a rule is not restricted to a Boolean expression as in Haskell. The condition of a rule can be any *constraint* that must be solved before applying the rule. A constraint is any expression of the predefined type `Success`. The type `Success` is a type without constructors but with a few basic constraints that can be combined into larger expressions. We have already seen the constraint operation “`=:`”, which is a function that maps its arguments into the type `Success`. Furthermore, there is a conjunction operation on constraints, “`&`”, that evaluates its two arguments concurrently. Beyond these basic constraints, some Curry implementations also offer more general constraint structures, such as arithmetic constraints on real numbers or finite domain constraints, together with appropriate constraint solvers. This enables the implementation of very effective algorithms to solve specific constraints. In this way, programs access and combine predefined constraints in a high-level manner.

Curry has a variety of other language features for high-level general-purpose programming. Similarly to Haskell, it is strongly typed with a polymorphic type system for reliable programming. Generic programming is also obtained through higher-order operations. Curry supports modules and offers input/output using the monadic approach like Haskell.³⁴ Moreover, it predefines primitive operations to encapsulate search, that is, to embed the nondeterministic search for solutions into purely functional computations by passing some or all the solutions into list struc-

tures. The combined functional and logic programming features of Curry have been shown useful in diverse applications: for example, to provide high-level APIs to access and manipulate databases,¹² to construct graphical user interfaces,²⁶ and to support Web programming.^{21,26} These developments were instrumental in identifying new design patterns that are specific to functional logic programming.⁶

Strategy

A crucial semantic component of a functional logic programming language is its evaluation strategy. Informally, the strategy tells which subexpression of an expression should be evaluated first. For non-strict semantics, such as demand-driven semantics that do not evaluate every subexpression, the problem is non-trivial. For functional logic languages the inherent difficulties of the problem are compounded by the presence of incomplete information in the form of free variables. The formalization of an evaluation strategy requires a formal model of computation. Various models have been proposed for functional logic programming. Rewriting systems⁹ are the model that more than any other has promoted significant progress in this area.

A functional logic computation is the repeated transformation, by narrowing steps, of an expression e . An *elementary step* of e involves a subexpression of e . This subexpression either is a redex (*reducible expression*) or is instantiated to obtain a redex. This redex is then reduced according to some rewrite rule of the program. The strategy produces the subexpression, the optional instantiation, and the rewrite rule that constitute a step. The strategy may produce several elementary steps of an expression. Steps that do not “interfere” with each other can be combined into a *multistep* and executed concurrently.

A computation of an expression terminates when the expression does not allow further steps. This expression is called a *normal form*. If a normal form contains occurrences of defined operations, such as a division by zero or the head of an empty list, it is called a *failure*, and the corresponding computation is said to *fail*. Otherwise, the normal form is called a *result* or *value* and the corresponding computation is said

to succeed. The nondeterministic nature of functional logic programming may lead to multiple outcomes of a computation. An expression may have several distinct results and several distinct failures. Failures are discarded, but failing computations may be both desirable and explicitly planned in the design of some programs.⁶ The primary task of the strategy is to ensure that every result of an expression is eventually obtained. Not surprisingly, for functional logic programs it is undecidable to tell whether a computation

The operation `take` makes an initial distinction on its first argument. The cases on this argument are zero and non-zero. In the non-zero case, the operation makes a subsequent distinction on its second argument. The cases on this argument are null and non-null. A definitional tree of the operation `take` (Antoy,⁴ Example 8) encodes these distinctions, the order in which they are made, and the cases that they consider.

Definitional trees guide the evaluation strategy similarly to case expres-

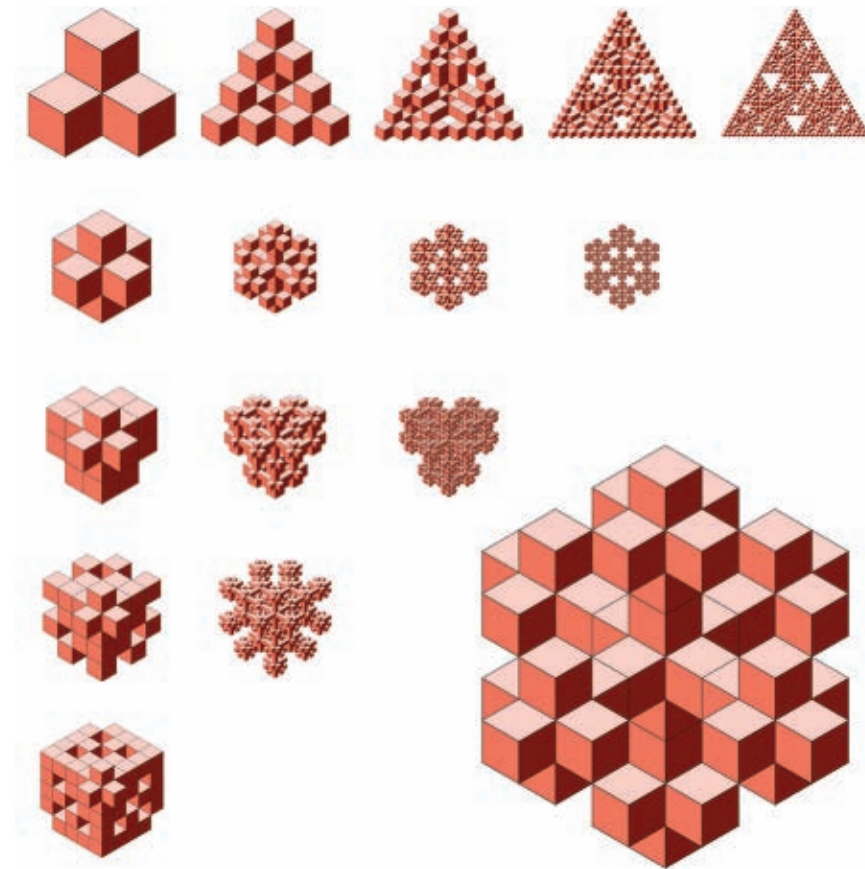
sions of functional languages, but there are significant differences. Case expressions are explicitly coded by the programmer, whereas definitional trees are inferred from the rules defining an operation. This difference becomes apparent in some situation where both strategies are applicable. For example, consider the following operation:

Haskell the same computation does not terminate on non-terminating t due to the fixed left-to-right evaluation of demanded arguments.

To fully support nondeterminism, in a functional logic program the textual order of the rules defining an operation is irrelevant. A consequence of this stipulation is that not every operation has a definitional tree, and definitional trees may have different characteristics. This has prompted a classification of functional logic programs according to the existence and/or the kinds of the definitional trees of their operations. Implementations of functional logic languages have grown more sophisticated over time. The modern approach transforms a program of a source language, which allows operations without a definitional tree, functional patterns, and/or partially applied functions, into a semantically equivalent program of a core language in which every operation has a definitional tree.^{3,8} Each such tree is compiled into code or byte code that implements a finite state automaton that determines both the evaluation of the arguments of a function application and the instantiation of free variables when necessary to perform a reduction.

Essential properties of an evaluation strategy are soundness (each computed result is correct with respect to an equational model of the program) and completeness (for each correct result, a corresponding value or a representative of this value is computed). Sound and complete strategies support a high-level abstract view of programming: The programmer states the intended properties of the application domain rather than the effects of individual computation steps.

Very strong properties are known for the evaluation strategy of Curry's core language. In particular, *needed narrowing*⁵ is sound and complete, computes only needed steps (each computation has minimal length), and computes only disjoint instantiations (no computation is unnecessarily repeated). The last two decades have seen a wealth of results in the area of strategies. Despite these achievements, we believe that more remains to be discovered. The promise lies in the development of strategies able to exploit the potential of parallel architectures. We briefly dis-



will terminate.

The strategy of contemporary implementations of functional logic languages is based on a formalism called a *definitional tree*. A definitional tree² is a hierarchical structure of the rewrite rules defining an operation of a program. For example, consider the operation that returns a prefix of a given length of a list, where for the purpose of illustration natural numbers are represented by Zero and Successor.

```
take Z _ = []
take (S n) [] = []
take (S n) (x:xs) = x : take n xs
```

In Curry, the computation of `f t 1` results in `1` even if t does not terminate, whereas in the lazy functional language

```
f 0 0 = 0
f _ 1 = 1
```

will terminate.

cuss some possibilities in the section “Looking Ahead.”

Programming

Functional logic programs mostly encode functional computations, but they can also encode nondeterministic computations and computations with incomplete information. This combination simplifies the design of some programs to a degree unparalleled by other programming paradigms. Simpler design leads to simpler proofs of program properties. We focus here on encoding a specification into a program and discussing the correctness of the program with respect to its specification. We keep our discussion informal, and out of necessity, we consider only small examples.

We start with an example where the formal specification can be directly written as a program. A regular expression over some alphabet is an expression of the following type (we omit empty regular expressions for simplicity):

```
data RE a = Lit a
          | Alt (RE a) (RE a)
          | Conc (RE a) (RE a)
          | Star (RE a)
```

Except for the syntax, this declaration is identical to the usual definition of regular expression. The type variable `a` ranges over the type of the alphabet; that is, we do not restrict our alphabet to characters only. A regular expression is one of the following expressions: a letter of the alphabet (`Lit`), a choice between two expressions (`Alt`), a concatenation of two expressions (`Conc`), or zero or more repetitions of an expression (`Star`). For instance, the regular expression `ab*` is given a name, `abstar`, and is defined as an expression of type `(RE Char)` as follows:

```
abstar = Conc (Lit 'a')
          (Star (Lit 'b'))
```

Derived constructors for regular expressions are defined as new operations, for example:

```
plus re = Conc re (Star re)
```

The language of a regular expression is defined by a mapping that takes a regular expression and yields

a set of words over the given alphabet. We prefer the functional logic programming style and define this mapping as a nondeterministic operation `sem` that yields any word (represented as a list) in the language of the given regular expression:

```
sem :: RE a -> [a]
sem (Lit c) = [c]
sem (Alt r s) = sem r ? sem s
sem (Conc r s) = sem r ++ sem s
sem (Star r) =
  [] ? sem (Conc r (Star r))
```

This is a concise specification of the semantics of regular expressions. Since it is a functional logic program, it is also executable so we can use it to generate words in the language of a regular expression. For instance, `sem abstar` evaluates to “a”, “ab”, “abb”, ... Therefore, `sem` can be also used to check whether a given word `w` is in the language of a given regular expression `re` through the equation `sem re == w`. The correctness of this claim stems from the soundness and completeness of the strategy, which guarantees the equation is satisfiable if and only if `w` is a value of `sem re`, hence, according to the definition of `sem`, if and only if `w` is in the language of `re`.

Moreover, we can check whether a string `s` contains a word generated by a regular expression `re` (similarly to the Unix’s `grep` utility) by solving the equation:

```
xs ++ sem re ++ ys == s
  where xs, ys free
```

If `s` contains a word `w` generated by a regular expression `re` as a substring, `w` must be a value of `sem re`, and there are sequences `xs` and `ys` such that the concatenation of `xs`, `w`, and `ys` is identical to `s`. The correctness of our `grep` program again follows from the soundness and completeness of the evaluation strategy. However, there is a noteworthy difference with the previous case. Earlier, the equation was *verified*, whereas in the case the equation is solved for `xs` and `ys`.

If a regular expression `r` contains the `Star` constructor, the language it generates is infinite so that `sem r` has infinitely many results. Since a demand-driven strategy computes

only the values that are required by the context, the previous equation might have a finite search space even in the presence of regular expressions with an infinite language. For instance, the equation

```
xs ++ sem abstar ++ ys == "abb"
  where xs, ys free
```

has a finite search space with exactly three solutions in `xs` and `ys`. Observe that we have used `sem` to generate regular expressions satisfying a given property.

For the next example let us consider an informal specification of the straight selection sort algorithm: given a non-empty sequence of elements, (1) select the smallest element, (2) sort the remaining ones, and (3) place the selected element in front of them. The only difficult-to-implement step of this algorithm is (1). Step (3) is trivial. Step (2) is obtained by recursion. Step (1) is more difficult because it is specified non-constructively. The specification must define the minimum of a sequence—in the obvious way—but will not say how to compute it. The algorithm to compute the minimum, a problem considerably more complicated than its definition, is left to the programmer. A functional logic programmer, however, can avoid the problem entirely coding a program that executes the specification.

The first rule of operation `sort` coded here implements the specification. The head of the rule employs a functional pattern. The input of `sort` is (evaluated to) a list, and one element of this list, `min`, is nondeterministically selected. The condition of the rule that ensures the selected element is indeed minimal in the list. The remaining code is straightforward. The operation `all`, defined in a standard library (*Prelude*) loaded with every program, returns `True` if its first argument, a predicate, is satisfied by every element of its second argument, a list. The expression `(min <=)` is called a *section*. It is equivalent to a function that takes an argument `x` and tells whether `min ≤ x`.

```
sort (u++[min]++v)
  | all (min <=) rest
  = min : sort rest
  where rest = u++v
sort [] = []
```

The assertion to establish the correctness of this program states that *sort l* is sorted and contains all and only the elements of *l*. The proof is by induction on the length of *l*. The base case is trivial. In the inductive case, if *sort l = x : xs*, then the head of the rule ensures that *x* is an element of *l*, and the condition of the rule ensures that *x* is not greater than any element of *xs*. The induction hypothesis ensures that *xs* contains all and only the elements of *l* except *x* and that *xs* is sorted. This entails the assertion.

Our final problem is the well-known *missionaries and cannibals puzzle*. Three missionaries and three cannibals want to cross a river with a boat that holds up to two people. Furthermore, the missionaries, if any, on either bank of the river cannot be outnumbered by the cannibals (otherwise, as the intuition hints, they would be eaten). A *state* of the puzzle is represented by the numbers of missionaries and cannibals and the presence of the boat on the initial bank of the river:

```
data State = State Int Int Bool
```

Thus, the initial state is “`State 3 3 True`”, but we do not construct it directly. Our program constructs a state only through a function, `makeState`, with the same signature as `State`. The function `makeState` applies `State` to its own arguments only if the resulting state is “sensible,” otherwise it simply fails. (This programming pattern is called “constrained constructor” in Antoy and Hanus.⁶) In this context, a state is sensible only if it is valid according to the numbers of missionaries and cannibals involved in the puzzle and is safe for the missionaries. In the following code, the operators `&&`, `||` and `==` are respectively the Boolean conjunction, disjunction and equality.

```
makeState m c b | valid && safe
                = State m c b
  where
    valid = 0<=m && m<=3 &&
           0<=c && c<=3
    safe  = m==3 || m==0 || m==c
```

For example, the initial and final states are constructed by:

```
initial = makeState 3 3 True
final   = makeState 0 0 False
```

There are a total of 10 different moves, five to cross the river in one direction and five in the other direction. Coding all these moves becomes particularly simple with nondeterminism and `makeState`. Both contribute to free the code from controlling which move should be executed.

```
move (State m c True)
  = makeState (m-2) c False -- 2 miss
  ? makeState (m-1) c False -- 1 miss
  ? makeState m (c-2) False -- 2 cann
  ? ...
```

A *solution* is a path through the state space from the initial to the final state. A *path* is a sequence of states each of which, except the first one, is obtained from the previous state by a move. The path is represented by a sequence of states. Our program adds new states at the front of a sequence that initially contains only the initial state, hence the order of the states in a path is reversed with respect to the moves that produce a solution. Our program constructs a path only through a function, `extendPath` that takes a non-empty path *p* and adds a state at the front of *p* so that the resulting path is sensible. In this context, a path is sensible only if it does not contain a cycle and each state of the path except the initial one is obtained with a move from the preceding state.

```
extendPath p | noCycle
              = next : p
  where
    next = move (head p)
    noCycle = all (next /=) p
```

Computing a solution of the puzzle becomes particularly simple with nondeterminism and `extendPath`. Our program simply extends a path until it produces the final state.

```
main = extendToFinal [initial]
extendToFinal p =
  if (head p == final) then p
  else extendToFinal (extendPath p)
```

The assertion to establish the correctness of this program states that operation `main` terminates with a so-

lution of the puzzle if and only if the puzzle has a solution. We discuss the properties of the program that inspire confidence in its correctness and would be key to a more rigorous proof.

The program maintains two invariants: (1) Every state constructed during an execution does not violate the problem’s constraints, and (2) the value returned by every invocation of operation `extendToFinal` is a path in the state space. Invariant (1) stems from the fact a state is constructed only by operation `makeState`, which ensures the legality of a state through conditions `valid` and `safe`. Invariant (2) stems from invariant (1) and the fact that a path is constructed only by operation `extendPath`, which ensures that the sequence of states it returns is indeed a path since only states originating from valid moves are added to a path. Operation `extendToFinal` keeps extending its argument, a path, unless the last inserted state is the goal, in which case it returns its argument. Operation `extendToFinal` is invoked by `main` with the initial state. Hence, if operation `extendToFinal` successfully terminates, it returns a solution of the problem. Operation `extendPath` invokes operation `move`, which is nondeterministic. The completeness of the evaluation strategy ensures that every move available in a state is chosen. Hence, if the problem has a solution, `extendToFinal`, and consequently `main`, will return this solution.

A further property of the program is that any solution returned by `extendToFinal` contains no repeated states. This condition stems from the fact that the paths returned by `extendToFinal` are constructed by operation `extendPath` which, through condition `noCycle`, fails to insert a state in a path that already contains that state.

Looking Behind


A motivation behind the emergence of functional logic programming was the desire to unify the two most prominent branches of the declarative paradigms, functional and logic programming. Much progress has been made toward this goal, and in the process the goal has somewhat changed. Unification in the form of a single common language accepted and used by the two communities, as perhaps some of us may have

envisioned, does not appear near. It may be difficult to understand all the reasons for this development, but investments, personal preferences, and specialization are certainly contributing factors. However, each community has become much more aware of the other, there are conferences in which papers from each paradigm are understood and equally enjoyed by all the participants, and efforts such as introducing logic variables in functional languages¹⁵ and functions in logic languages¹⁴ are indisputable evidence that the gap between the two paradigms has been narrowed. More interestingly, functional logic programming has become a paradigm in its own right with results whose depth and solidity rival those of the other declarative paradigms.


We sketched the concepts of Curry and used it for concrete examples. Curry is currently the only functional logic language that is based on strong theoretical foundations (for example, sound, complete, and optimal evaluation strategies^{4,5}) and has been used for a variety of projects, such as Web-based information systems, embedded systems programming, and e-learning systems.^{24,25} Curry has been also used from the very beginning to teach functional and logic programming concepts with a single programming language.²⁰ Nevertheless, there are various other languages with similar goals. We briefly discuss some of them and relate them to Curry.

The language *TOY*³⁰ has strong connections to Curry since it is based on similar foundations. In contrast to Curry, it supports more advanced constraint structures (for example, disequality constraints) but misses application-oriented libraries so that it has not been used for application programming. This is also the case for Escher,²⁹ a functional logic language where functions are deterministically evaluated in a demand-driven manner and logic features are modeled by simplification rules for logical expressions.

The language Oz³⁸ is based on a computation model that extends concurrent constraint programming with features for distributed programming and stateful computations. It supports functional notation, but operations used for goal solving must be defined with explicit disjunctions. Thus, functions used to solve equations must be



A crucial semantic component of a functional logic programming language is its evaluation strategy. Informally, the strategy tells which subexpression of an expression should be evaluated first.



defined differently from functions to rewrite expressions.

Since functions can be considered as specific relations, there are also approaches to extend logic languages with features for functional programming by adding syntactic sugar for functional notation. For instance, Casas et al.¹⁴ proposes to add functional notation to Ciao-Prolog which is translated by a preprocessor into Prolog. The functional logic language Mercury³⁹ restricts logic programming features in order to provide a highly efficient implementation. In particular, operations must have distinct modes so that their arguments are either completely known or unbound at call time. This inhibits the application of typical logic programming techniques, for example, computing with structures that are only partially known, so that some programming techniques for functional logic programming^{6,26} cannot be applied with Mercury. This condition has been relaxed in the language HAL,¹⁸ which adds constraint solving possibilities. However, both languages are based on a strict operational semantics that does not support optimal evaluation as in functional programming.

Some of the concepts and techniques developed for functional logic programming have migrated to other areas of computer science. A success story is Tim Sheard's Ω mega³⁶ system—a type system with an infinite hierarchy of computational levels that combines programming and reasoning. At the value level computation is performed by reduction. At all higher levels computation is performed by narrowing in the inductively sequential² systems using definitional trees to execute only needed steps.

Looking Ahead

Will functional logic programs ever execute as efficiently as imperative programs? We do not have the final answer, but the future is promising. Needed narrowing, the strategy for a core functional logic language, exhibits two distinct aspects of non-determinism: *don't care* and *don't know* choices. Don't care choices occur in some function calls when more than one subexpression needs to be evaluated, such as when both arguments of an equation or of an arithmetic binary

operation are demanded. Because the language is free of side effects, we do not care which argument is evaluated first. Therefore, it is viable to evaluate both arguments concurrently. A small amount of synchronization may be necessary to determine when the evaluation of a set of don't care choices is completed and to instantiate variables shared by these choices.

Don't know choices occur in some function calls when more than one alternative is available, such as when a free variable argument is instantiated

many problems, in particular non-numeric ones, functional logic programs have a degree of parallelism that is potentially higher than corresponding imperative programs. Future research will investigate this potential and attempt to exploit it for faster execution of functional logic programs.

The most important aids to the development of real applications, after a solid and efficient compiler, are a set of libraries for application programming and environments and tools that support program development and main-

profiling,¹¹ debugging,¹⁰ and testing.¹⁷

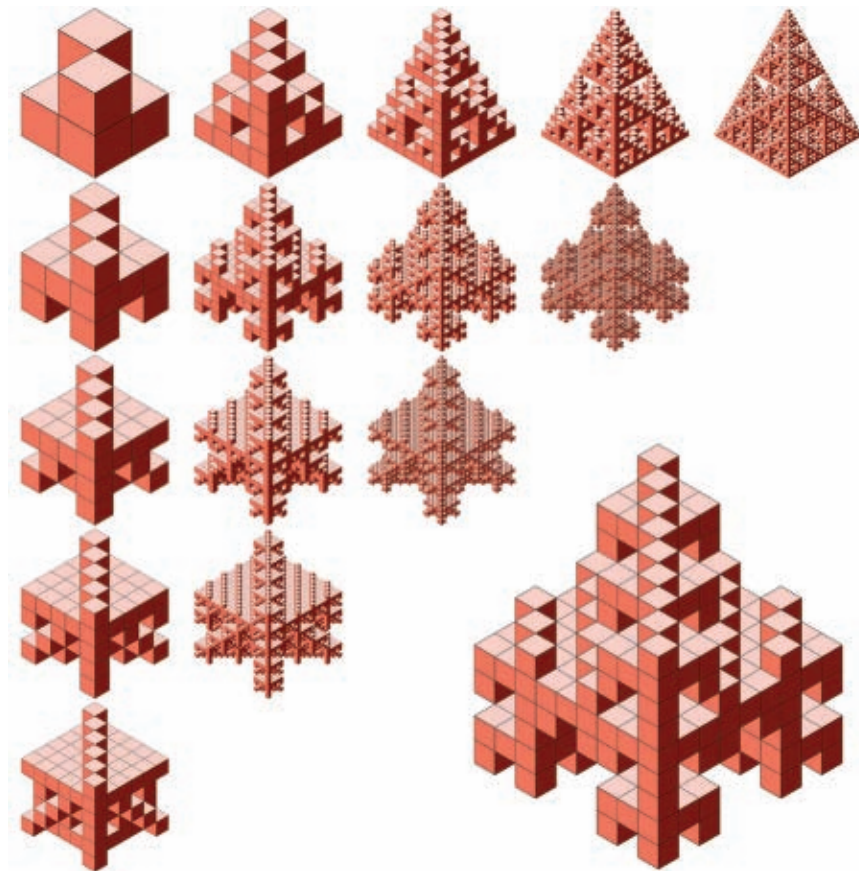
These initial efforts demonstrate the suitability of functional logic programming techniques for application programming. Some applications have been successfully developed, as we have mentioned. Due to the availability of Curry implementations with support for meta-programming, most of the current tools are implemented in Curry. These applications demonstrate the potential of functional logic programming. Future research and development in these areas will promote and ease the use of functional logic programs in practice.

The major asset of functional logic programming is the amalgamation of functional and logic programming features. Logic programming emphasizes nondeterminism, whereas functional programming is deterministic, that is, the input of a computation completely determines the output. This dichotomy creates a conflict when it is desirable to reason “functionally” about nondeterministic computations, for example, to select a minimal or maximal value among a set of results of a nondeterministic computation, or to print all these results in some order. Curry provides some features to encapsulate nondeterministic computations. For example, the set of results of a nondeterministic computation can be collected in some data structure and processed in its entirety. These features are useful and are used in some applications,^{13,31} but have limitations. Future research on the theory of these features and on their implementation should ease the encoding of complex problems into relatively simple functional logic programs.

Conclusion

Declarative languages describe programming tasks through high-level, concise, and executable abstractions. Other paradigms offer similar abstractions, but to a lesser degree. Compared to purely functional languages, functional logic languages allow nondeterministic descriptions with partial information that simplify encoding some problems into programs. Compared to purely logic languages, functional logic languages allow functional composition, which improves code modularity, reuse, and efficiency of execution.

Functional logic programs have the favor of executable specifications or



with different values and/or when an expression is reduced by different rules. Since we don't know in advance which alternative(s) will produce a solution (if we did, we would have coded a program that selects these alternatives only), all the alternatives must be sampled to ensure the completeness of the computation. Because each alternative is independent of any other alternative, in this situation too, the order of execution of the alternatives is irrelevant. As a consequence, both don't care and don't know choices can be executed concurrently. This is one area where the abstraction from the evaluation order pays off. For

tenance. Promising work has been started in both areas. Functional logic programming supports abstractions that lead to high-level, declarative programming interfaces. Libraries with interfaces with these characteristics ease the construction of reliable applications in various application areas, as shown by the various libraries developed for Curry. The formal foundations of functional logic languages are also useful for the development of environments and tools for reasoning about programs. Developments in these areas include tools for program analysis,²³ verification,¹⁶ partial evaluation,¹

high-level descriptions. Correctness proofs for these programs are simpler than those in other programming paradigms. These executable specifications are a good solution of some programming problems, for example, when the execution is efficient or when no other algorithm is known. Of course, the performance of this specification-oriented approach may not be satisfactory for some problems. Even in this situation, the effort to develop an initial prototype is not wasted. First, building one or more prototypes is a reasonable step for many software artifacts since a prototype clarifies problems and allows experimentation at a reduced cost. Second, a prototype can be refined with the introduction of more efficient data structures (for example, replace a list with a tree) or more efficient algorithms (for example, replace linear search with binary search). Since declarative languages also allow the implementation of efficient data structures,³³ the stepwise improvement of a prototype can be done without changing the implementation language. This methodology has the advantage that the initial prototype serves as a specification for verifying the improved program and/or as an oracle for testing it.

Last but not least, programming in a functional logic language is fun. As we move through the evolution of programming languages sketched in the introduction, we find that programming problems become easier and solutions more reliable. Evaluating an arithmetic expression directly through its standard notation without using registers, temporaries, and stacks is safer and more convenient. Structuring programs without labels and *gotos* is more elegant and increases the confidence in the correctness of the result. Legitimate concerns about the efficiency of these revolutionary innovations were raised at the times of their introductions.

History repeats itself. Programming with nondeterminism is exhilarating when a potentially difficult task is replaced by an almost effortless choice followed by a much simpler constraint. In our examples, this was to determine whether a string belongs to the language defined by a regular expression, or to find the minimum of a sequence, or to produce the solution of a puzzle without any concern for selecting the

moves. This approach is not always the best one in practice. For example, the nondeterministic choice of the minimum of a list may have to be replaced by an algorithm that is much faster for the computer to execute and much slower for the programmer to code and verify. But also in these cases, there is understanding and value in the prototypical implementation and satisfaction in its simplicity and elegance.

This work was partially supported by the German Research Council (DFG) grant Ha 2457/5-2, the DAAD grants D/06/29439 and D/08/11852, and the NSF grants CCR-0110496 and CCR-0218224. **C**

References

- Albert, E., Hanus, M. and Vidal, G. A practical partial evaluator for a multi-paradigm declarative language. *J. Functional and Logic Programming* 1 (2002).
- Antoy, S. Definitional trees. In *Proceedings of the 3rd International Conference on Algebraic and Logic Programming*. Springer LNCS 632, 1992, 143–157.
- Antoy, S. Constructor-based conditional narrowing. In *Proceedings of the 3rd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*. ACM Press, 2001, 199–206.
- Antoy, S. Evaluation strategies for functional logic programming. *J. Symbolic Computation* 40, 1 (2005), 875–903.
- Antoy, S., Echahed, R. and Hanus, M. A needed narrowing strategy. *J. ACM*, 47, 4 (2000), 776–822.
- Antoy, S. and Hanus, M. Functional logic design patterns. In *Proceedings of the 6th International Symposium on Functional and Logic*. Springer LNCS 2441, 2002, 67–87.
- Antoy, S. and Hanus, M. Declarative programming with function patterns. In *Proceedings of the International Symposium on Logic-based Program Synthesis and Transformation*. Springer LNCS 3901, 2005, 6–22.
- Antoy, S. and Hanus, M. Overlapping rules and logic variables in functional logic programs. In *Proceedings of the 22nd International Conference on Logic Programming*. Springer LNCS 4079, 2006, 87–101.
- Bezem, M., Klop, J.W. and de Vrijer, R. (eds.). *Term Rewriting Systems*. Cambridge University Press, 2003.
- Braßel, B., Fischer, S., Hanus, M., Huch, F. and Vidal, G. Lazy call-by-value evaluation. In *Proceedings of the 12th ACM SIGPLAN International Conference on Functional Programming*. ACM Press, 2007, 265–276.
- Braßel, B., Hanus, M., Huch, F., Silva, J. and Vidal, G. Run-time profiling of functional logic programs. In *Proceedings of the International Symposium on Logic-based Program Synthesis and Transformation*. Springer LNCS 3573, 2005, 182–197.
- Braßel, B., Hanus, M., and Müller, M. High-level database programming in Curry. In *Proceedings of the 10th International Symposium on Practical Aspects of Declarative Languages*. Springer LNCS 4902, 2008, 316–332.
- Braßel, B. and Huch, F. On a tighter integration of functional and logic programming. In *Proceedings of APLAS 2007*. Springer LNCS 4807, 2007, 122–138.
- Casas, A., Cabeza, D., and Hermenegildo, M.V. A syntactic approach to combining functional notation, lazy evaluation, and higher-order in LP systems. In *Proc. of the 8th International Symposium on Functional and Logic Programming*. Springer LNCS 3945, 2006, 146–162.
- Claessen, K. and Ljunglöf, P. Typed logical variables in Haskell. In *Proceedings ACM SIGPLAN Haskell Workshop*, Montreal, 2000.
- Clewa, J.M., Leach, J., and López-Fraguas, F.J. A logic programming approach to the verification of functional-logic programs. In *Proceedings of the 6th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*. ACM Press, 2004, 9–19.
- Fischer, S. and Kuchen, H. Systematic generation of glass-box test cases for functional logic programs. In *Proceedings of the 9th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*. ACM Press, 2007, 75–89.
- García de la Banda, M.J., Demoen, B., Marriott, K. and Stuckey, P.J. To the gates of HAL: A HAL tutorial. In *Proceedings of the 6th International Symposium on Functional and Logic Programming*. Springer LNCS 2441, 2002, 47–66.
- González-Moreno, J.C., Hortalá-González, M.T., López-Fraguas, F.J. and Rodríguez-Artalejo, M. An approach to declarative programming based on a rewriting logic. *Journal of Logic Programming*, 1999, 40–47.
- Hanus, M. Teaching functional and logic programming with a single computation model. In *Proceedings of the 9th International Symposium on Programming Languages, Implementations, Logics, and Programs*. Springer LNCS 1292, 1997, 335–350.
- Hanus, M. Type-oriented construction of Web user interfaces. In *Proceedings of the 8th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*. ACM Press, 2006, 27–38.
- Hanus, M. Multi-paradigm declarative languages. In *Proceedings of the International Conference on Logic Programming*. Springer LNCS 4670, 2007, 45–75.
- Hanus, M. Call pattern analysis for functional logic programs. In *Proceedings of the 10th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*. ACM Press, 2008, 67–78.
- Hanus, M. and Höppner, K. Programming autonomous robots in Curry. *Electronic Notes in Theoretical Computer Science*, 76, 2002.
- Hanus, M. and Huch, F. An open system to support Web-based learning. In *Proceedings 12th International Workshop on Functional and (Constraint) Logic Programming*, 2003, 269–282.
- Hanus, M. and Kluß, C. Declarative programming of user interfaces. In *Proceedings of the 11th International Symposium on Practical Aspects of Declarative Languages*. Springer LNCS 5418, 2009, 16–30.
- Hanus, M., Ed. Curry: An integrated functional logic language (vers. 0.8.2, 2006); <http://www.curry-language.org/>.
- Hussmann, H. Nondeterministic algebraic specifications and nonconfluent term rewriting. *Journal of Logic Programming*, 1992, 237–255.
- Lloyd, J. Programming in an integrated functional and logic language. *J. Functional and Logic Programming* 3, 1 (1999), 1–49.
- López-Fraguas, F. and Sánchez-Hernández, J. TOY: A multiparadigm declarative system. In *Proceedings of RTA'99*. Springer LNCS 1631, 1999, 244–247.
- Lux, W. Implementing encapsulated search for a lazy functional logic language. In *Proceedings of the 4th Fuji International Symposium on Functional and Logic Programming*. Springer LNCS 1722, 1999, 100–113.
- Milner, R., Tofte, M. and Harper, R. *The Definition of Standard ML*. MIT Press, 1990.
- Okasaki, C. *Purely Functional Data Structures*. Cambridge University Press, 1998.
- Peyton Jones, S. (Ed). *Haskell 98 Language and Libraries—The Revised Report*. Cambridge University Press, 2003.
- Reddy, U.S. Narrowing as the operational semantics of functional languages. In *Proceedings of the IEEE International Symposium on Logic in Computer Science* (Boston, 1985), 138–151.
- Sheard, T. Type-level computation using narrowing in mega. *Electron. Notes Theor. Comput. Sci.* 174, 7 (2007), 105–128.
- Stagle, J.R. Automated theorem-proving for theories with simplifiers, commutativity, and associativity. *J. ACM* 21, 4 (1974), 622–642.
- Smolka, G. The Oz programming model. *Computer Science Today: Recent Trends and Developments*. J. van Leeuwen (Ed). Springer LNCS 1000, 1995, 324–343.
- Somogyi, Z., Henderson, F., and Conway, T. The execution algorithm of Mercury, an efficient purely declarative logic programming language. *J. Logic Programming* 29, 1–3 (1996), 17–64.

Sergio Antoy (antoy@cs.pdx.edu) is Tektronix Professor at Portland State University Portland, OR.

Michael Hanus (mh@informatik.uni-kiel.de) is a professor of computer science at the University of Kiel, Germany.

© 2010 ACM 0001-0782/10/0400 \$10.00



Group Term Life Insurance**

10- or 20-Year Group Term Life Insurance*

Group Disability Income Insurance*

Group Accidental Death & Dismemberment Insurance*

Group Catastrophic Major Medical Insurance*

Group Dental Plan*

Long-Term Care Plan

Major Medical Insurance

Short-Term Medical Plan***

Who has time to think about insurance?

Today, it's likely you're busier than ever. So, the last thing you probably have on your mind is whether or not you are properly insured.

But in about the same time it takes to enjoy a cup of coffee, you can learn more about your ACM-sponsored group insurance program — a special member benefit that can help provide you financial security at economical group rates.

Take just a few minutes today to make sure you're properly insured.

Call Marsh Affinity Group Services at 1-800-503-9230 or visit www.personal-plans.com/acm.

3132851 35648 (7/07) © Seabury & Smith, Inc. 2007

The plans are subject to the terms, conditions, exclusions and limitations of the group policy. For costs and complete details of coverage, contact the plan administrator. Coverage may vary and may not be available in all states.

*Underwritten by The United States Life Insurance Company in the City of New York, a member company of American International Group, Inc.

**Underwritten by American General Assurance Company, a member company of American International Group, Inc.

***Coverage is available through Assurant Health and underwritten by Time Insurance Company.

AG5217

MARSH

Affinity Group Services
a service of Seabury & Smith

research highlights

P. 88

**Technical
Perspective
Creativity Helps
Influence Prediction
Precision**

By Padhraic Smyth
and Charles Elkan

P. 89

**Collaborative Filtering
with Temporal Dynamics**

By Yehuda Koren

P. 98

**Technical
Perspective
New Bar Set for
Intelligent Vehicles**

By Leslie Pack Kaelbling

P. 99

Toward Robotic Cars

By Sebastian Thrun

Technical Perspective

Creativity Helps Influence Prediction Precision

By Padhraic Smyth and Charles Elkan

THE PAST DECADE has seen an explosion of interest in machine learning and data mining, with significant advances in terms of both theoretical results and highly visible practical applications. One such application is that of automated recommender systems. Customers buy or rate items or products, and the ratings are stored in a database—indeed, there may be millions of customers and items. Past customer behaviors and preferences are then analyzed to automatically predict which items a customer is likely to rate highly or purchase, among items they have not already purchased or rated. Recommender systems of this type are now commonplace on the Web, for example, at sites such as Amazon.com for shopping and last.fm for music.

In October 2006, the Netflix company offered a prize of \$1 million dollars for the development of an algorithm that could significantly improve the accuracy of its in-house recommendation system. Netflix customers rate movies they have seen on a scale of 1 to 5. The recommendation problem is to suggest new movies to customers based on a large database of past ratings. One way of measuring the accuracy of a recommendation algorithm is to compute the average squared difference between ratings from the algorithm and actual customer ratings, on item-customer pairs not previously seen by the algorithm. Netflix stated it would award the \$1 million prize to the first person or team who could reduce the error rate of its in-house algorithm by a factor of 10%. For the competition, participants could download a large training set of data on which to develop their methods. This data set consisted of a highly sparse matrix of approximately 500,000 customers (rows) by 18,000 movies (columns), where less than 1% of the entries contained known ratings.

For three years (2006–2009), the competition was the focus of intense

activity among computer scientists, mathematicians, engineers, and statisticians around the globe. By the end there were over 40,000 registered competitors from over 150 different countries. Progress was initially quite rapid and by December 2006 a few teams already had algorithms that were capable of a 5% error reduction on unseen test data—halfway to the million dollars!

But, as with many interesting problems, achieving the second 5% of improvement was much more difficult than achieving the first 5%. As the rate of progress slowed in 2007 there was much speculation as to whether the 10% would ever be achievable. By fall 2007, Yehuda Koren, Robert Bell, and Chris Volinsky from AT&T Research had emerged among the leading contenders for the prize. This AT&T team won the initial “progress prize” (\$50,000) in October 2007 with an 8.4% reduction in error. Koren and colleagues continued to push closer to 10%, claiming the second progress prize (in collaboration with two Austrian university students) 12 months later, with a 9.4% reduction in error.

The following paper is an excellent example of how clever thinking about the fundamental factors that underlie a complex domain can lead to major improvement in prediction accuracy.

The following paper by Koren, now a senior research scientist at Yahoo! Research, was presented at the ACM SIGKDD Conference in Paris in June 2009. It is an excellent example of how clever thinking about the fundamental factors that underlie a complex domain can lead to major improvement in prediction accuracy. While the paper focuses on recommender systems, the thinking and strategy behind it provide useful lessons that are applicable to many data mining tasks.

A key aspect of this paper is the use of exploratory data analysis and visualization to uncover otherwise hidden information, in particular, to discover that the ratings data is non-stationary over time (as depicted in Figure 1). Koren proceeds to show that predictions can be systematically improved by adjusting for this non-stationarity. Multiple types of temporal variation in the data is incorporated into the equations for prediction models in a variety of ways, providing important improvement over predictions that do not include a temporal component.

This paper illustrates clearly how to combine statistical thinking (modeling of variation in a systematic manner) with computational methods (fitting these models to over 100 million ratings), to build sophisticated models on very large data sets. The author notes that “addressing temporal dynamics can have a more significant impact on accuracy than designing more complex learning algorithms.” This point is vital in the general context of predictive analytics—careful consideration of the basic factors that govern how data is generated (and collected) can lead to significantly more accurate predictions.

To bring closure to the \$1 million prize story (for readers who don't know how it ends) the grand prize was awarded in September 2009 to a team consisting of Yehuda Koren and six colleagues (the Bell Labs statisticians, the two students from Austria, and two engineers from Montreal). □

Padhraic Smyth is a professor in the department of computer science at the University of California, Irvine.

Charles Elkan is a professor in the department of computer science and engineering at the University of California, San Diego.

© 2010 ACM 0001-0782/10/0400 \$10.00

Collaborative Filtering with Temporal Dynamics

By Yehuda Koren

Abstract

Customer preferences for products are drifting over time. Product perception and popularity are constantly changing as new selection emerges. Similarly, customer inclinations are evolving, leading them to ever redefine their taste. Thus, modeling temporal dynamics is essential for designing recommender systems or general customer preference models. However, this raises unique challenges. Within the ecosystem intersecting multiple products and customers, many different characteristics are shifting simultaneously, while many of them influence each other and often those shifts are delicate and associated with a few data instances. This distinguishes the problem from concept drift explorations, where mostly a single concept is tracked. Classical time-window or instance decay approaches cannot work, as they lose too many signals when discarding data instances. A more sensitive approach is required, which can make better distinctions between transient effects and long-term patterns. We show how to model the time changing behavior throughout the life span of the data. Such a model allows us to exploit the relevant components of all data instances, while discarding only what is modeled as being irrelevant. Accordingly, we revamp two leading collaborative filtering recommendation approaches. Evaluation is made on a large movie-rating dataset underlying the Netflix Prize contest. Results are encouraging and better than those previously reported on this dataset. In particular, methods described in this paper play a significant role in the solution that won the Netflix contest.

1. INTRODUCTION

Modeling time drifting data is a central problem in data mining. Often, data is changing over time, and models should be continuously updated to reflect its present nature. The analysis of such data needs to find the right balance between discounting temporary effects that have very low impact on future behavior, while capturing longer term trends that reflect the inherent nature of the data. This led to many works on the problem, which is also widely known as *concept drift*; see, e.g., Schlimmer and Granger, and Widmer and Kubat.^{15,20}

Temporal changes in customer preferences bring unique modeling challenges. One kind of concept drift in this setup is the emergence of new products or services that change the focus of customers. Related to this are seasonal changes, or specific holidays, which lead to characteristic shopping patterns. All those changes influence the whole population, and are within the realm of traditional studies on concept drift. However, many of the changes in user behavior are

driven by localized factors. For example, a change in the family structure can drastically change shopping patterns. Likewise, individuals gradually change their taste in movies and music. Such changes cannot be captured by methods that seek a global concept drift. Instead, for each customer we are looking at different types of concept drifts, each occurs at a distinct time frame and is driven toward a different direction.

The need to model time changes at the level of each individual significantly reduces the amount of available data for detecting such changes. Thus we should resort to more accurate techniques than those that suffice for modeling global changes. For example, it would no longer be adequate to abandon or simply underweight far in time user transactions. The signal that can be extracted from those past actions might be invaluable for understanding the customer herself or be indirectly useful to modeling other customers. Yet, we need to distill long-term patterns while discounting transient noise. These considerations require a more sensitive methodology for addressing drifting customer preferences. It would not be adequate to concentrate on identifying and modeling just what is relevant to the present or the near future. Instead, we require an accurate modeling of each point in the past, which will allow us to distinguish between persistent signal that should be captured and noise that should be isolated from the longer term parts of the model.

Modeling user preferences is relevant to multiple applications ranging from spam filtering to market-basket analysis. Our main focus in the paper is on modeling user preferences for building a recommender system, but we believe that general lessons that we learn would apply to other applications as well. Automated recommendations are a very active research field.¹² Such systems analyze patterns of user interest in items or products to provide personalized recommendations of items that will suit a user's taste. We expect user preferences to change over time. The change may stem from multiple factors; some of these factors are fundamental while others are more circumstantial. For example, in a movie recommender system, users may change their preferred genre or adopt a new viewpoint on an actor or director. In addition, they may alter the appearance of their feedback. For example, in a system

A previous version of this paper appeared in the *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009), 447–456.

where users provide star ratings to products, a user that used to indicate a neutral preference by a “3 stars” input may now indicate dissatisfaction by the same “3 stars” feedback. Similarly, it is known that user feedback is influenced by anchoring, where current ratings should be taken as relative to other ratings given at the same short period. Finally, in many instances, systems cannot separate different household members accessing the same account, even though each member has a different taste and deserves a separate model. This creates a de facto multifaceted meta-user associated with the account. A way to distinguish between different persons is by assuming that time-adjacent accesses are being done by the same member (sometimes on behalf of other members), which can be naturally captured by a temporal model that assumes a drifting nature of a customer.

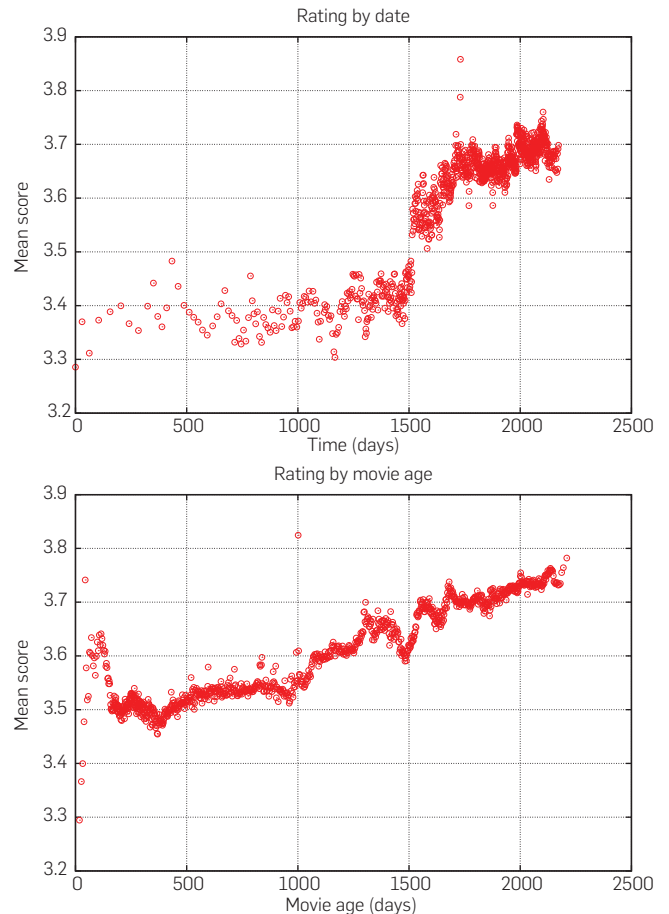
All these patterns and the likes should have made temporal modeling a predominant factor in building recommender systems. Nonetheless, with very few exceptions (e.g., Ding and Li, and Sugiyama et al.^{4,16}), the recommenders’ literature does not address temporal changes in user behavior. Perhaps this is because user behavior is composed of many different concept drifts, acting in different timeframes and directions, thus making common methodologies for dealing with concept drift and temporal data less successful. We show that capturing time drifting patterns in user behavior is essential for improving accuracy of recommenders. Our findings also give us hope that the insights from successful time modeling for recommenders will be useful in other data mining applications.

Our test bed is a large movie-rating dataset released by Netflix as the basis of a well-publicized competition.³ This dataset combines several merits for the task at hand. First, it is not a synthetic dataset, but contains user-movie ratings by real paying Netflix subscribers. In addition, its relatively large size—above 100 million date-stamped ratings—makes it a better proxy for real-life large-scale datasets, while putting a premium on computational efficiency. Finally, unlike some other dominant datasets, time effects are natural and are not introduced artificially. Two interesting (if not surprising) temporal effects that emerge within this dataset are shown in Figure 1. One effect is an abrupt shift of rating scale that happened in early 2004. At that time, the mean rating value jumped from around 3.4 stars to above 3.6 stars. Another significant effect is that ratings given to movies tend to increase with the movie age. That is, older movies receive higher ratings than newer ones. In Koren,⁸ we shed some light on the origins of these effects.

The major contribution of this work is presenting a methodology and specific techniques for modeling time drifting user preferences in the context of recommender systems. The proposed approaches are applied on the aforementioned extensively analyzed movie-ratings dataset, enabling us to firmly compare our methods with those reported recently. We show that by incorporating temporal information, we achieve best results reported so far, indicating the significance of uncovering temporal effects.

The rest of the paper is organized as follows. In the next section we describe basic notions and notation. Then, in

Figure 1. Two temporal effects emerging within the Netflix movie-rating dataset. Top: the average movie-rating made a sudden jump in early 2004 (1,500 days since the first rating in the dataset). Bottom: ratings tend to increase with the movie age at the time of the rating. Here, movie age is measured by the time span since its first rating event within the dataset. In both charts, each point averages 100,000 rating instances.



Section 3, our principles for addressing time changing user preferences are evolved. Those principles are then incorporated, in quite different ways, into two leading recommender techniques: factor modeling (Section 4) and item–item neighborhood modeling (Section 5).

2. PRELIMINARIES

2.1. Notation

We are given ratings for m users (aka customers) and n items (aka products). We reserve special indexing letters to distinguish users from items: for users u, v , and for items i, j . A rating r_{ui} indicates the preference by user u of item i , where high values mean stronger preference. For example, values can be integers ranging from 1 (star) indicating no interest to 5 (stars) indicating a strong interest. We distinguish predicted ratings from known ones, by using the notation \hat{r}_{ui} for the predicted value of r_{ui} .

The scalar t_{ui} denotes the time of rating r_{ui} . One can use different time units, based on what is appropriate for the

application at hand. For example, when time is measured in days, then t_{ui} counts the number of days elapsed since some early time point. Usually the vast majority of ratings are unknown. For example, in the Netflix data 99% of the possible ratings are missing because a user typically rates only a small portion of the movies. The (u, i) pairs for which r_{ui} is known are stored in the set $\mathcal{K} = \{(u, i) | r_{ui} \text{ is known}\}$, which is known as the *training set*.

Models for the rating data are learned by fitting the previously observed ratings. However, our goal is to generalize those in a way that allows us to predict future, unknown ratings. Thus, caution should be exercised to avoid overfitting the observed data. We achieve this by using a technique called *regularization*. Regularization restricts the complexity of the models, thereby preventing them from being too specialized to the observed data. We employ L2-regularization, which penalizes the magnitude of the learned parameters. Extent of regularization is controlled by constants which are denoted as: $\lambda_1, \lambda_2, \dots$

2.2. The Netflix data

We evaluated our algorithms on a movie-rating dataset of more than 100 million date-stamped ratings performed by about 480,000 anonymous Netflix customers on 17,770 movies between 31 December 1999 and 31 December 2005.³ Ratings are integers ranging between 1 and 5. On average, a movie receives 5,600 ratings, while a user rates 208 movies, with substantial variation around each of these averages. To maintain compatibility with results published by others, we adopted some common standards. We evaluated our methods on two comparable sets designed by Netflix: a holdout set (“Probe set”) and a test set (“Quiz set”), each of which contains over 1.4 million ratings. Reported results are on the test set, while experiments on the holdout set show the same findings. In our time-modeling context, it is important to note that the test instances of each user come later in time than his/her training instances. The quality of the results is measured by their root mean squared error (RMSE)

$$\sqrt{\sum_{(u,i) \in \text{TestSet}} (r_{ui} - \hat{r}_{ui})^2 / |\text{TestSet}|}$$

The Netflix data is part of the Netflix Prize contest, with the target of improving the accuracy of Netflix movie recommendations by 10%. The benchmark is Netflix’s proprietary system. Cinematch, which achieved an RMSE of 0.9514 on the test set. The grand prize was awarded to a team that managed to drive this RMSE to 0.8554 after almost 3 years of extensive efforts. Achievable RMSE values on the test set lie in a quite compressed range, as evident by the difficulty to win the grand prize. Nonetheless, there is evidence that small improvements in RMSE terms can have a significant impact on the quality of the top few presented recommendations.⁷ The algorithms described in this work played a central role in reaching the grand prize.

2.3. Collaborative filtering

Recommender systems are often based on *collaborative filtering* (CF), a term coined by the developers of the first recommender system—Tapestry.⁵ This technique relies only on past user behavior—e.g., their previous transactions or

product ratings—without requiring the creation of explicit profiles. CF analyzes relationships between users and interdependencies among products, in order to identify new user–item associations.

A major appeal of CF is that it is domain-free and avoids the need for extensive data collection. In addition, relying directly on user behavior allows uncovering complex and unexpected patterns that would be difficult or impossible to profile using known data attributes. As a consequence, CF attracted much of attention in the past decade, resulting in significant progress and being adopted by some successful commercial systems, including Amazon,¹⁰ TiVo,¹ and Netflix.

The two primary areas of CF are the *neighborhood methods* and *latent factor models*. The neighborhood methods are centered on computing the relationships between items or, alternatively, between users. The item-oriented approach evaluates the preference of a user to an item based on ratings of “neighboring” items by the same user. A product’s neighbors are other products that tend to be scored similarly when rated by the same user. For example, consider the movie “Saving Private Ryan.” Its neighbors might include other war movies, Spielberg movies, and Tom Hanks movies, among others. To predict a particular user’s rating for “Saving Private Ryan,” we would look for the movie’s nearest neighbors that were actually rated by that user. A dual to the item-oriented approach is user-oriented approach, which identifies like-minded users who can complement each other’s missing ratings.

Latent factor models comprise an alternative approach that tries to explain the ratings by characterizing both items and users on, say, 20–200 factors inferred from the pattern of ratings. For movies, factors discovered by the decomposition might measure obvious dimensions such as comedy vs. drama, amount of action, or orientation to children; less well-defined dimensions such as depth of character development or “quirkiness,” or completely uninterpretable dimensions. For users, each factor measures how much the user likes movies that score high on the corresponding movie factor. One of the most successful realizations of latent factor models is based on *matrix factorization*; see, e.g., Koren et al.⁹

3. TRACKING DRIFTING CUSTOMER PREFERENCES

One of the frequently mentioned examples of concept drift is changing customer preferences over time, e.g., “customer preferences change as new products and services become available.”⁶ This aspect of drifting customer preferences highlights a common paradigm in the literature of having global drifting concepts influencing the data as a whole. However, in many applications, including our focus application of recommender systems, we also face a more complicated form of concept drift where interconnected preferences of many users are drifting in different ways at different time points. This requires the learning algorithm to keep track of multiple changing concepts. In addition the typically low amount of data instances associated with individual customers calls for more concise and efficient learning methods, which maximize the utilization of signal in the data.

In a survey on the problem of concept drift, Tsymbal¹⁹ argues that three approaches can be distinguished in the literature. The *instance selection* approach discards instances that are less relevant to the current state of the system. A common variant is time-window approaches where only recent instances are considered. A possible disadvantage of this simple model is that it is giving the same significance to all instances within the considered time-window, while completely discarding all other instances. Equal significance might be reasonable when the time shift is abrupt, but less so when time shift is gradual. Thus, a refinement is *instance weighting* where instances are weighted based on their estimated relevance. Frequently, a time decay function is used, underweighting instances as they occur deeper into the past. The third approach is based on *ensemble learning*, which maintains a family of predictors that together produce the final outcome. Those predictors are weighted by their perceived relevance to the present time point, e.g., predictors that were more successful on recent instances get higher weights.

We performed extensive experiments with instance weighting schemes, trying different exponential time decay rates on both neighborhood and factor models. The consistent finding was that prediction quality improves as we moderate that time decay, reaching best quality when there is no decay at all. This finding is despite the fact that users do change their taste and rating scale over the years, as we show later. However, much of the old preferences still persist or, more importantly, help in establishing useful cross-user or cross-product patterns in the data. Thus, just underweighting past actions lose too many signals along with the lost noise, which is detrimental, given the scarcity of data per user.

As for ensemble learning, having multiple models, each of which considers only a fraction of the total behavior may miss those global patterns that can be identified only when considering the full scope of user behavior. What makes them even less appealing in our case is the need to keep track of the independent drifting behaviors of many customers. This, in turn, would require building a separate ensemble for each user. Such a separation will significantly complicate our ability to integrate information across users along multiple time points, which is the cornerstone of *collaborative filtering*. For example, an interesting relation between products can be established by related actions of many users, each of them at a totally different point of time. Capturing such a collective signal requires building a single model encompassing all users and items together.

All those considerations led us to the following guidelines we adopt for modeling drifting user preferences.

- We seek models that explain user behavior along the full extent of the time period, not only the present behavior (while subject to performance limitations). Such modeling is key to being able to extract signal from each time point, while neglecting only the noise.
- Multiple changing concepts should be captured. Some are user-dependent and some are item-dependent. Similarly, some are gradual while others are sudden.

- While we need to model separate drifting “concepts” or preferences per user and/or item, it is essential to combine all those concepts within a single framework. This combination allows modeling interactions crossing users and items thereby identifying higher level patterns.
- In general, we do not try to extrapolate future temporal dynamics, e.g., estimating future changes in a user’s preferences. Extrapolation could be very helpful but is seemingly too difficult, especially given a limited amount of known data. Rather than that, our goal is to capture past temporal patterns in order to isolate persistent signal from transient noise. The result, indeed, helps in predicting *future* behavior.

Now we turn to how these desirable principles are incorporated into two leading approaches to CF—matrix factorization and neighborhood methods.

4. TIME-AWARE FACTOR MODEL

4.1. The anatomy of a factor model

Matrix factorization is a well-recognized approach to CF.^{9,11,17} This approach lends itself well to an adequate modeling of temporal effects. Before we deal with those temporal effects, we would like to establish the foundations of a static factor model.

In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from patterns of item ratings. High correspondence between item and user factors leads to recommendation of an item to a user. More specifically, both users and items are mapped to a joint latent factor space of dimensionality f , such that ratings are modeled as inner products in that space. Accordingly, each user u is associated with a vector $p_u \in \mathbb{R}^f$ and each item i is associated with a vector $q_i \in \mathbb{R}^f$. A rating is predicted by the rule

$$\hat{r}_{ui} = q_i^T p_u = \left(\sum_{k=1}^f q_i[k] \cdot p_u[k] \right). \quad (1)$$

The major challenge is computing the mapping of each item and user to factor vectors $q_i, p_u \in \mathbb{R}^f$. After this mapping is accomplished, we can easily compute the ratings a user will give to any item by using Equation 1.

Such a model is closely related to singular value decomposition (SVD), which is a well-established technique for identifying latent semantic factors in the information retrieval. Applying SVD in the CF domain would require factoring the user–item rating matrix. Such a factorization raises difficulties due to the high portion of missing values, due to the sparseness in the user–item ratings matrix. Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting. Earlier works¹³ relied on imputation to fill in missing ratings and make the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, the data may be considerably distorted due to inaccurate imputation. Hence, more recent

works (e.g., Koren, Paterek, and Takacs et al.^{7,11,17}) suggested modeling directly only the observed ratings, while avoiding overfitting through an adequate regularized model. In order to learn the factor vectors (p_u and q_i), we minimize the regularized squared error on the set of known ratings:

$$\min_{q^*, p^*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - q_i^T p_u)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2). \quad (2)$$

Minimization is typically performed by stochastic gradient descent.

Model (1) tries to capture the interactions between users and items that produce the different rating values. However, much of the observed variation in rating values is due to effects associated with either users or items, independently of their interaction, which are known as biases. A prime example is that typical CF data exhibits large systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others. After all, some products are widely received as better (or worse) than others.

Thus, it would be unwise to explain the full rating value by an interaction of the form $q_i^T p_u$. Instead, we will try to identify the portion of these values that can be explained by individual user or item effects (biases). The separation of interaction and biases will allow us to subject only the true interaction portion of the data to factor modeling.

We will encapsulate those effects, which do not involve user-item interaction, within the *baseline predictors*. These baseline predictors tend to capture much of the observed signal, in particular much of the temporal dynamics within the data. Hence, it is vital to model them accurately, which enables better identification of the part of the signal that truly represents user-item interaction and should be subject to factorization.

A suitable way to construct a static baseline predictor is as follows. Denote by μ the overall average rating. A baseline predictor for an unknown rating r_{ui} is denoted by b_{ui} and accounts for the user and item main effects:

$$b_{ui} = \mu + b_u + b_i. \quad (3)$$

The parameters b_u and b_i indicate the observed deviations of user u and item i , respectively, from the average. For example, suppose that we want a baseline estimate for the rating of the movie Titanic by user Joe. Now, say that the average rating over all movies, μ , is 3.7 stars. Furthermore, Titanic is better than an average movie, so it tends to be rated 0.5 stars above the average. On the other hand, Joe is a critical user, who tends to rate 0.3 stars lower than the average. Thus, the baseline estimate for Titanic's rating by Joe would be 3.9 stars by calculating $3.7 - 0.3 + 0.5$.

The baseline predictor should be integrated back into the factor model. To achieve this we extend rule (1) to be

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u. \quad (4)$$

Here, the observed rating is separated to its four components: global average, item-bias, user-bias, and user-item interaction. The separation allows each component to explain only

the part of signal relevant to it. Learning is done analogously to before, by minimizing the squared error function

$$\min_{q^*, p^*, b^*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2). \quad (5)$$

Schemes along these lines were described in, e.g., Koren and Paterek.^{7,11}

The decomposition of a rating into distinct portions is convenient here, as it allows us to treat different temporal aspects in separation. More specifically, we identify the following effects: (1) user-biases (b_u) change over time; (2) item biases (b_i) change over time; and (3) user preferences (p_u) change over time. On the other hand, we would not expect a significant temporal variation of item characteristics (q_i), as items, unlike humans, are static in their nature. We start with a detailed discussion of the temporal effects that are contained within the baseline predictors.

4.2. Time changing baseline predictors

Much of the temporal variability is included within the baseline predictors, through two major temporal effects. The first addresses the fact that an item's popularity may change over time. For example, movies can go in and out of popularity as triggered by external events such as the appearance of an actor in a new movie. This is manifested in our models by treating the item bias b_i as a function of time. The second major temporal effect allows users to change their baseline ratings over time. For example, a user who tended to rate an average movie "4 stars," may now rate such a movie "3 stars." This may reflect several factors including a natural drift in a user's rating scale, the fact that ratings are given in relevance to other ratings that were given recently and also the fact that the identity of the rater within a household can change over time. Hence, in our models we take the parameter b_u as a function of time. This induces a template for a time sensitive baseline predictor for u 's rating of i at day t_{ui} :

$$b_{ui} = \mu + b_u(t_{ui}) + b_i(t_{ui}). \quad (6)$$

Here, $b_u(\cdot)$ and $b_i(\cdot)$ are real valued functions that change over time. The exact way to build these functions should reflect a reasonable way to parameterize the involving temporal changes. Our choice in the context of the movie-rating dataset demonstrates some typical considerations.

A major distinction is between temporal effects that span extended periods of time and more transient effects. In the movie-rating case, we do not expect movie likeability to fluctuate on a daily basis, but rather to change over more extended periods. On the other hand, we observe that user effects can change on a daily basis, reflecting inconsistencies natural to customer behavior. This requires finer time resolution when modeling user-biases compared with a lower resolution that suffices for capturing item-related time effects.

We start with our choice of time-changing item biases $b_i(t)$. We found it adequate to split the item biases into time-based bins, using a constant item bias for each time period. The decision of how to split the timeline into bins should

balance the desire to achieve finer resolution (hence, smaller bins) with the need for enough ratings per bin (hence, larger bins). For the movie-rating data, there is a wide variety of bin sizes that yield about the same accuracy. In our implementation, each bin corresponds to roughly 10 consecutive weeks of data, leading to 30 bins spanning all days in the dataset. A day t is associated with an integer $\text{Bin}(t)$ (a number between 1 and 30 in our data), such that the movie bias is split into a stationary part and a time changing part:

$$b_i(t) = b_i + b_{i, \text{Bin}(t)} \quad (7)$$

While binning the parameters works well on the items, it is more of a challenge on the users' side. On the one hand, we would like a finer resolution for users to detect very short-lived temporal effects. On the other hand, we do not expect enough ratings per user to produce reliable estimates for isolated bins. Different functional forms can be considered for parameterizing temporal user behavior, with varying complexity and accuracy.

One simple modeling choice uses a linear function to capture a possible gradual drift of user-bias. For each user u , we denote the mean date of rating by t_u . Now, if u rated a movie on day t , then the associated time deviation of this rating is defined as

$$\text{dev}_u(t) = \text{sign}(t - t_u) \cdot |t - t_u|^\beta.$$

Here $|t - t_u|$ measures the number of days between dates t and t_u . We set the value of β by cross-validation; in our implementation $\beta = 0.4$. We introduce a single new parameter for each user called α_u so that we get our first definition of a time-dependent user-bias

$$b_u^{(1)}(t) = b_u + \alpha_u \cdot \text{dev}_u(t). \quad (8)$$

A more flexible spline-based rule is described in Koren.⁸

A smooth function for modeling the user-bias meshes well with *gradual concept drift*. However, in many applications there are *sudden drifts* emerging as “spikes” associated with a single day or session. For example, in the movie-rating dataset we have found that multiple ratings, a user gives in a single day, tend to concentrate around a single value. Such an effect need not span more than a single day. The effect may reflect the mood of the user that day, the impact of ratings given in a single day on each other, or changes in the actual rater in multiperson accounts. To address such short-lived effects, we assign a single parameter per user and day, absorbing the day-specific variability. This parameter is denoted by $b_{u,t}$. Notice that in some applications the basic primitive time unit to work with can be shorter or longer than a day.

In the Netflix movie-rating data, a user rates on 40 different days on average. Thus, working with $b_{u,t}$ requires, on average, 40 parameters to describe each user-bias. It is expected that $b_{u,t}$ is inadequate as a stand-alone for capturing the user-bias, since it misses all sorts of signals that span more than a single day. Thus, it serves as an additive component within the previously described schemes. The time-linear model (8) becomes

$$b_{ui}^{(3)}(t) = b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t} + b_{i, \text{Bin}(t)}. \quad (9)$$

A baseline predictor on its own cannot yield personalized recommendations, as it misses all interactions between users and items. In a sense, it is capturing the portion of the data that is less relevant for establishing recommendations. Nonetheless, to better assess the relative merits of the various choices of time-dependent user-bias, we compare their accuracy as stand-alone predictors. In order to learn the involved parameters we minimize the associated regularized squared error by using stochastic gradient descent. For example, in our actual implementation we adopt rule (9) for modeling the drifting user-bias, thus arriving at the baseline predictor

$$b_{ui} = \mu + b_u + \alpha_u \cdot \text{dev}_u(t_{ui}) + b_{u,t_{ui}} + b_i + b_{i, \text{Bin}(t_{ui})}. \quad (10)$$

To learn the involved parameters, b_u , α_u , $b_{u,t}$, b_i , and $b_{i, \text{Bin}(t)}$, one should solve

$$\min \sum_{(u,i) \in \mathcal{R}} (r_{ui} - \mu - b_u - \alpha_u \text{dev}_u(t_{ui}) - b_{u,t_{ui}} - b_i - b_{i, \text{Bin}(t_{ui})})^2 + \lambda_\gamma (b_u^2 + \alpha_u^2 + b_{u,t_{ui}}^2 + b_i^2 + b_{i, \text{Bin}(t_{ui})}^2).$$

Here, the first term strives to construct parameters that fit the given ratings. The regularization term, $\lambda_\gamma (b_u^2 + \dots)$, avoids overfitting by penalizing the magnitudes of the parameters, assuming a neutral 0 prior. Learning is done by a stochastic gradient descent algorithm running 20–30 iterations, with $\lambda_\gamma = 0.01$.

Table 1 compares the ability of various suggested baseline predictors to explain signal in the data. As usual, the amount of captured signal is measured by the RMSE on the test set. As a reminder, test cases come later in time than the training cases for the same user, so predictions often involve extrapolation in terms of time. We code the predictors as follows:

- *static*, no temporal effects: $b_{ui} = \mu + b_u + b_i$,
- *mov*, accounting only for movie-related temporal effects: $b_{ui} = \mu + b_u + b_i + b_{i, \text{Bin}(t_{ui})}$,
- *linear*, linear modeling of user-biases: $b_{ui} = \mu + b_u + \alpha_u \cdot \text{dev}_u(t_{ui}) + b_i + b_{i, \text{Bin}(t_{ui})}$, and
- *linear+*, linear modeling of user-biases and single day effect: $b_{ui} = \mu + b_u + \alpha_u \cdot \text{dev}_u(t_{ui}) + b_{u,t_{ui}} + b_i + b_{i, \text{Bin}(t_{ui})}$.

The table shows that while temporal movie effects reside in the data (lowering RMSE from 0.9799 to 0.9771), the drift in user-biases is much more influential. In particular, sudden changes in user-biases, which are captured by the per-day parameters, are most significant.

Beyond the temporal effects described so far, one can

Table 1. Comparing baseline predictors capturing main movie and user effects. As temporal modeling becomes more accurate, prediction accuracy improves (lowering RMSE).

Model	Static	Mov	Linear	Linear+
RMSE	0.9799	0.9771	0.9731	0.9605

use the same methodology to capture more effects. A prime example is capturing periodic effects. For example, some products may be more popular in specific seasons or near certain holidays. Similarly, different types of television or radio shows are popular throughout different segments of the day (known as “dayparting”). Periodic effects can be found also on the user side. As an example, a user may have different attitudes or buying patterns during the weekend compared to the working week. A way to model such periodic effects is to dedicate a parameter for the combinations of time periods with items or users. This way, the item bias of (7) becomes

$$b_i(t) = b_i + b_{i, \text{Bin}(t)} + b_{i, \text{period}(t)}.$$

For example, if we try to capture the change of item bias with the season of the year, then $\text{period}(t) \in \{\text{fall, winter, spring, summer}\}$. Similarly, recurring user effects may be modeled by modifying (9) to be

$$b_u(t) = b_u + \alpha_u \cdot \text{dev}_u(t) + b_{ut} + b_{u, \text{period}(t)}.$$

However, we have not found periodic effects with a significant predictive power within the movie-rating dataset, thus our reported results do not include those.

Another temporal effect within the scope of basic predictors is related to the changing scale of user ratings. While $b_i(t)$ is a user-independent measure for the merit of item i at time t , users tend to respond to such a measure differently. For example, different users employ different rating scales, and a single user can change his rating scale over time. Accordingly, the raw value of the movie bias is not completely user-independent. To address this, we add a time-dependent scaling feature to the baseline predictors, denoted by $c_u(t)$. Thus, the baseline predictor (10) becomes

$$b_{ui} = \mu + b_u + \alpha_u \cdot \text{dev}_u(t_{ui}) + b_{u,t_{ui}} + (b_i + b_{i, \text{Bin}(t_{ui})}) \cdot c_u(t_{ui}). \quad (11)$$

All discussed ways to implement $b_u(t)$ would be valid for implementing $c_u(t)$ as well. We chose to dedicate a separate parameter per day, resulting in: $c_u(t) = c_u + c_{ut}$. As usual, c_u is the stable part of $c_u(t)$, whereas c_{ut} represents day-specific variability. Adding the multiplicative factor $c_u(t)$ to the baseline predictor lowers RMSE to 0.9555. Interestingly, this basic model, which captures just main effects disregarding user–item interactions, can explain almost as much of the data variability as the commercial Netflix Cinematch recommender system, whose published RMSE on the same test set is 0.9514.³

4.3. Time changing factor model

In Section 4.2 we discussed the way time affects baseline predictors. However, as hinted earlier, temporal dynamics go beyond this, they also affect user preferences and thereby the interaction between users and items. Users change their preferences over time. For example, a fan of the “psychological thrillers” genre may become a fan of “crime dramas” a year later. Similarly, humans change their perception on certain actors and directors. This effect is modeled by taking the user factors (the vector p_u) as a function of time. Once

again, we need to model those changes at the very fine level of a daily basis, while facing the built-in scarcity of user ratings. In fact, these temporal effects are the hardest to capture, because preferences are not as pronounced as main effects (user-biases), but are split over many factors.

We modeled each component of the user preferences $p_u(t)^T = (p_u(t)[1], p_u(t)[2], \dots, p_u(t)[f])$ in the same way that we treated user-biases. Within the movie-rating dataset, we have found modeling after (9) effective, leading to

$$p_u(t)[k] = p_{uk} + \alpha_{uk} \cdot \text{dev}_u(t) + p_{ukt} \quad k=1, \dots, f. \quad (12)$$

Here p_{uk} captures the stationary portion of the factor, $\alpha_{uk} \cdot \text{dev}_u(t)$ approximates a possible portion that changes linearly over time, and p_{ukt} absorbs the very local, day-specific variability.

At this point, we can tie all pieces together and extend the SVD factor model (4) by incorporating the time changing parameters. The resulting model will be denoted as *timeSVD*, where the prediction rule is as follows:

$$\hat{r}_{ui} = \mu + b_i(t_{ui}) + b_u(t_{ui}) + q_i^T p_u(t_{ui}). \quad (13)$$

The exact definitions of the time drifting parameters $b_i(t)$, $b_u(t)$, and $p_u(t)$ were given in Equations 7, 9, and 12. Learning is performed by minimizing the associated squared error function on the training set using a regularized stochastic gradient descent algorithm. The procedure is analogous to the one involving the original SVD algorithm. Time complexity per iteration is still linear with the input size, while wall clock running time is approximately doubled compared to SVD, due to the extra overhead required for updating the temporal parameters. Importantly, convergence rate was not affected by the temporal parameterization, and the process converges in around 30 iterations.

4.4. Comparison

The factor model we are using in practice is slightly more involved than the one described so far. The model, which is known as SVD++,⁷ offers an improved accuracy by also accounting for the more implicit information recorded by which items were rated (regardless of their rating value). While details of the SVD++ algorithm are beyond the scope of this article, they do not influence the introduction of temporal effects, and the model is extended to account for temporal effects following exactly the same procedure described in this section. The resulting model is known as *timeSVD++*, and is described in Koren.⁸

In Table 2 we compare results of three matrix factorization algorithms. First is SVD, the plain matrix factorization algorithm. Second is the SVD++ method, which improves upon SVD by incorporating a kind of implicit feedback. Third is *timeSVD++*, which also accounts for temporal effects. The three methods are compared over a range of factorization dimensions (f). All benefit from a growing number of factor dimensions that enables them to better express complex movie–user interactions. Addressing implicit feedback by the SVD++ model leads to accuracy gains within the movie-rating dataset. Yet, the improvement delivered by

Table 2. Comparison of three factor models: prediction accuracy is measured by RMSE (lower is better) for varying factor dimensionality (f). For all models accuracy improves with growing number of dimensions. Most significant accuracy gains are achieved by addressing the temporal dynamics in the data through the timeSVD++ model.

Model	$f = 10$	$f = 20$	$f = 50$	$f = 100$	$f = 200$
SVD	0.9140	0.9074	0.9046	0.9025	0.9009
SVD++	0.9131	0.9032	0.8952	0.8924	0.8911
timeSVD++	0.8971	0.8891	0.8824	0.8805	0.8799

timeSVD++ over SVD++ is consistently more significant. We are not aware of any single algorithm in the literature that could deliver such accuracy. We attribute this to the importance of properly addressing temporal effects. Further evidence of the importance of capturing temporal dynamics is the fact that a timeSVD++ model of dimension 10 is already more accurate than an SVD model of dimension 200. Similarly, a timeSVD++ model of dimension 20 is enough to outperform an SVD++ model of dimension 200.

4.5. Predicting future days

Our models include day-specific parameters. An apparent question would be how these models can be used for predicting ratings in the future, on new dates for which we cannot train the day-specific parameters? The simple answer is that for those future (untrained) dates, the day-specific parameters should take their default value. In particular for Equation 11, $c_u(t_{ui})$ is set to c_u , and $b_{u,t_{ui}}$ is set to zero. Yet, one wonders, if we cannot use the day-specific parameters for predicting the future, why are they good at all? After all, prediction is interesting only when it is about the future. To further sharpen the question, we should mention the fact that the Netflix test sets include many ratings on dates for which we have no other rating by the same user and hence day-specific parameters cannot be exploited.

To answer this, notice that our temporal modeling makes no attempt to capture future changes. All it is trying to do is to capture transient temporal effects, which had a significant influence on past user feedback. When such effects are identified, they must be tuned down, so that we can model the more enduring signal. This allows our model to better capture the long-term characteristics of the data, while letting dedicated parameters absorb short-term fluctuations. For example, if a user gave many higher than usual ratings on a particular single day, our models discount those by accounting for a possible day-specific good mood, which does not reflect the longer term behavior of this user. This way, the day-specific parameters contribute to cleaning the data, which improves prediction of future dates.

5. TEMPORAL DYNAMICS AT NEIGHBORHOOD MODELS

The most common approach to CF is based on neighborhood models. While typically less accurate than their factorization counterparts, neighborhood methods enjoy popularity thanks to some of their merits, such as explaining the

reasoning behind computed recommendations, and seamlessly accounting for new entered ratings.

Recently, we suggested an item-item model based on global optimization,⁷ which will enable us here to capture time dynamics in a principled manner. The static model, without temporal dynamics, is centered on the following prediction rule:

$$\hat{r}_{ui} = \mu + b_i + b_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) w_{ij} + c_{ij}. \quad (14)$$

Here, the set $R(u)$ contains the items rated by user u . The item-item weights w_{ij} and c_{ij} represent the adjustments we need to make to the predicted rating of item i , given a known rating of item j . It was proven greatly beneficial to use two sets of item-item weights: one (the w_{ij} s) is related to the values of the ratings, and the other disregards the rating value, considering only which items were rated (the c_{ij} s). These weights are automatically learned from the data together with the biases b_i and b_u . The constants b_{uj} are pre-computed according to Equation 3. Recall that $R(u)$ is the set of items rated by user u .

When adapting rule (14) to address temporal dynamics, two components should be considered separately. First component, $\mu + b_i + b_u$, corresponds to the the baseline predictor portion. Typically, this component explains most variability in the observed signal. Second component, $|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) w_{ij} + c_{ij}$, captures the more informative signal, which deals with user-item interaction. As for the baseline part, nothing changes from the factor model, and we replace it with $\mu + b_i(t_{ui}) + b_u(t_{ui})$, according to Equations 7 and 9. However, capturing temporal dynamics within the interaction part requires a different strategy.

Item-item weights (w_{ij} and c_{ij}) reflect inherent item characteristics and are not expected to drift over time. The learning process should capture unbiased long-term values, without being too affected from drifting aspects. Indeed, the time changing nature of the data can mask much of the longer term item-item relationships if not treated adequately. For instance, a user rating both items i and j high within a short time period is a good indicator for relating them, thereby pushing higher the value of w_{ij} . On the other hand, if those two ratings are given 5 years apart, while the user's taste (if not her identity) could considerably change, this provides less evidence of any relation between the items. On top of this, we would argue that those considerations are pretty much user dependent; some users are more consistent than others and allow relating their longer term actions.

Our goal here is to distill accurate values for the item-item weights, despite the interfering temporal effects. First we need to parameterize the decaying relations between two items rated by user u . We adopt exponential decay formed by the function $e^{-\beta_u \cdot \Delta t}$, where $\beta_u > 0$ controls the user-specific decay rate and should be learned from the data. We also experimented with other decay forms, like the computationally cheaper $(1 + \beta_u \Delta t)^{-1}$, which resulted in about the same accuracy, with an improved running time.

This leads to the prediction rule

$$\hat{r}_{ui} = \mu + b_i(t_{ui}) + b_u(t_{ui}) + |\mathbf{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathbf{R}(u)} e^{-\beta_u \cdot |t_{ui} - t_{uj}|} ((r_{uj} - b_{uj})w_{ij} + c_{ij}). \quad (15)$$

The involved parameters, $b_i(t_{ui}) = b_i + b_{i, \text{Bin}(t_{ui})}$, $b_u(t_{ui}) = b_u + \alpha_u \cdot \text{dev}_u(t_{ui}) + b_{u, t_{ui}}$, β_u , w_{ij} and c_{ij} , are learned by minimizing the associated regularized squared error

$$\sum_{(u,i) \in \mathcal{K}} \left(r_{ui} - \mu - b_i - b_{i, \text{Bin}(t_{ui})} - b_u - \alpha_u \text{dev}_u(t_{ui}) - b_{u, t_{ui}} - |\mathbf{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathbf{R}(u)} e^{-\beta_u \cdot |t_{ui} - t_{uj}|} ((r_{uj} - b_{uj})w_{ij} + c_{ij}) \right)^2 + \lambda_{12} (b_i^2 + b_{i, \text{Bin}(t_{ui})}^2 + b_u^2 + \alpha_u^2 + b_{u, t_{ui}}^2 + w_{ij}^2 + c_{ij}^2). \quad (16)$$

Minimization is performed by stochastic gradient descent. As in the factor case, properly considering temporal dynamics improves the accuracy of the neighborhood model within the movie-ratings dataset. The RMSE decreases from 0.9002⁷ to 0.8885. To our best knowledge, this is significantly better than previously known results by neighborhood methods. To put this in some perspective, this result is even better than those reported by using hybrid approaches such as applying a neighborhood approach on residuals of other algorithms.^{2, 11, 18} A lesson is that addressing temporal dynamics in the data can have a more significant impact on accuracy than designing more complex learning algorithms.

We would like to highlight an interesting point related to the basic methodology described in Section 3. Let u be a user whose preferences are quickly drifting (β_u is large). Hence, old ratings by u should not be very influential on his status at the current time t . One could be tempted to decay the weight of u 's older ratings, leading to "instance weighting" through a cost function like

$$\sum_{(u,i) \in \mathcal{K}} e^{-\beta_u \cdot |t - t_{ui}|} \left(r_{ui} - \mu - b_i - b_{i, \text{Bin}(t_{ui})} - b_u - \alpha_u \text{dev}_u(t_{ui}) - b_{u, t_{ui}} - |\mathbf{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathbf{R}(u)} ((r_{uj} - b_{uj})w_{ij} + c_{ij}) \right)^2 + \lambda_{12} (\dots).$$

Such a function is focused at the *current* state of the user (at time t), while de-emphasizing past actions. We would argue against this choice, and opt for equally weighting the prediction error at all past ratings as in Equation 16, thereby modeling *all* past user behavior. Therefore, equal-weighting allows us to exploit the signal at each of the past ratings, a signal that is extracted as item-item weights. Learning those weights would equally benefit from all ratings by a user. In other words, we can deduce that two items are related if users rated them similarly within a short time frame, even if this happened long ago.

6. CONCLUSION

Tracking the temporal dynamics of customer preferences to products raises unique challenges. Each user and product potentially goes through a distinct series of changes in their characteristics. Moreover, we often need to model all those

changes within a single model thereby interconnecting users (or, products) to each other to identify communal patterns of behavior. A mere decay of older instances or usage of multiple separate models lose too many signals, thus degrading prediction accuracy. The solution we adopted is to model the temporal dynamics along the whole time period, allowing us to intelligently separate transient factors from lasting ones. We applied this methodology to two leading recommender techniques. In a factorization model, we modeled the way user and product characteristics change over time, in order to distill longer term trends from noisy patterns. In an item-item neighborhood model, we showed how the more fundamental relations among items can be revealed by learning how influence between two items rated by a user decays over time. In both factorization and neighborhood models, the inclusion of temporal dynamics proved very useful in improving quality of predictions, more than various algorithmic enhancements. This led to the best results published so far on a widely analyzed movie-rating dataset. □

References

- Ali, K., van Stam, W. Tivo: making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004), 394–401.
- Bell, R., Koren, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. *IEEE International Conference on Data Mining (ICDM'07)* (2007), 43–52.
- Bennet, J., Lanning, S. The Netflix Prize. *KDD Cup and Workshop, 2007*. www.netflixprize.com.
- Ding, Y., Li, X. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM'04)* (2004), 485–492.
- Goldberg, D., Nichols, D., Oki, B.M., Terry, D. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35 (1992), 61–70.
- Kolter, J.Z., Maloof, M.A. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proceedings of the IEEE Conference on Data Mining (ICDM'03)* (2003), 123–130.
- Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)* (2008), 426–434.
- Koren, Y. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)* (2009), 447–456.
- Koren, Y., Bell, R., Volinsky, C. Matrix factorization techniques for recommender systems. *IEEE Comput. Intell.* 42 (2009), 30–37.
- Linden, G., Smith, B., York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.* 7 (2003), 76–80.
- Paterek, A. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of the KDD Cup and Workshop* (2007).
- Pu, P., Bridge, D.G., Mobasher, B., Ricci, F. (eds.). In *Proceedings of the 2008 ACM Conference on Recommender Systems* (2008).
- Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J. Application of dimensionality reduction in recommender system—A case study. *WEBKDD'2000*.
- Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on the World Wide Web* (2001), 285–295.
- Schlimmer, J., Granger, R. Beyond incremental processing: Tracking concept drift. In *Proceedings of the 5th National Conference on Artificial Intelligence* (1986), 502–507.
- Sugiyama, K., Hatano, K., Yoshikawa, M. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th International Conference on World Wide Web (WWW'04)* (2004), 675–684.
- Takacs, G., Pillaszy, I., Nemeth, B., Tikk, D. Major components of the gravity recommendation system. *SIGKDD Explor.* 9 (2007), 80–84.
- Toscher, A., Jährer, M., Legenstein, R. Improved neighborhood-based algorithms for large-scale recommender systems. *KDD'08 Workshop on Large Scale Recommenders Systems and the Netflix Prize* (2008).
- Tsymbal, A. The problem of concept drift: Definitions and related work. Technical Report TCD-CS-2004-15. Trinity College Dublin, 2004.
- Widmer, G., Kubat, M. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.* 23, 69 (1996), 101.

Yehuda Koren (yehuda@yahoo-inc.com),
Yahoo! Research, Haifa, Israel.

Technical Perspective

New Bar Set for Intelligent Vehicles

By Leslie Pack Kaelbling

THE DARPA CHALLENGES for autonomous driving have stimulated impressive technology demonstrations, showcased recent fundamental advances in artificial intelligence, and highlighted important directions for future research.

The following paper by Sebastian Thrun gives us a glimpse into the design and implementation of two winning DARPA grand challenge entries. Those robotic systems were enabled by advances that have been taking place in AI over the last 25 years in representation, learning, and control.

It is as important for robots, as for humans, to know what they don't know. Explicit representation of uncertainty in terms of probability, and incremental incorporation of information using Bayesian techniques, enables construction of systems that are robust and effective. Probabilistic estimation methods have long been used in relatively constrained applications, such as target tracking. Their extension to highly complex spaces, such as maps of surrounding obstacles and the intentions of other vehicles, is crucial to the success of intelligent systems embedded in the physical world. Such systems incorporate information from a variety of sensors; indeed, successful fusion of sensor information from inertial navigation sources, vision, and range sensing, was a crucial aspect of the robotic cars highlighted here. In the future, such systems will estimate abstract quantities, such as the goals and satisfaction level of their human drivers, and integrate information ranging from engine noise to weather reports to the content of human passengers' conversation into a comprehensive picture of the world around them.

Estimation techniques are ultimately built on models that tell us how perceptual inputs are related to the underlying state of the world and how future states of the world are related the previous ones. Traditionally, such models have been built by hand: expert sys-

tems models were specified with logical rules representing intuitive knowledge; and estimation systems were specified with linear-Gaussian distributions representing physical laws and the results of empirical tests. These models are either too brittle or too representationally limited to support intelligent robotics.

Machine learning has provided a methodology for acquiring models from data: these models can be highly non-linear, are grounded in real-world data, and can be updated incrementally as the surrounding world changes. This article provides us with a wonderful example of learning: it is difficult to make a visual detector for whether terrain can be driven on that works in all lighting conditions, terrain types, and so on. However, in any given conditions, it is apparently not too difficult to find a simple model that discriminates between drivable and non-drivable terrain. Thus, the Stanley system uses an online self-supervised learning system to continuously adapt a model that predicts drivability based on visual input. The self-supervised aspect is particularly interesting. The vehicle uses short-

The systems described in this paper exhibit important steps toward the goal of truly integrated, robust, intelligent systems that interact with the physical world.

range laser sensing to label the terrain near to it as drivable; this data is used to learn a model that then allows visual prediction of drivability over much longer distances. Today, virtually any system operating on data from the real world—speech, vision, language—uses models learned from data to make robust estimates and predictions.

In the end, what matters is how the system behaves. The vehicle must behave robustly, selecting appropriate actions by taking into consideration the state of the environment, from road friction to pedestrians to the driver's desires, as well as its own uncertainty about the state. This article describes planning and control systems, operating at different levels of abstraction, each of which attempts to select actions that maximize expected utility: that is, that will perform well, on average, given the robot's uncertainty. In addition, actions may be chosen for the explicit purpose of reducing uncertainty, by aiming cameras in appropriate directions or driving to previously unexplored locations. Robustness also depends on time-sensitive decision-making: when an obstacle suddenly appears, the vehicle must stop immediately, without time to weigh all of its options in detail. The vehicles described here have multitiered control systems that ensure quick reaction to changing environmental conditions while allowing deeper deliberation for higher-level decisions.

Thrun describes systems that exhibit important steps toward the goal of truly integrated, robust, intelligent systems that interact with the physical world. They will motivate researchers, old and new, to push forward to generally intelligent and robust robots that manage ubiquitous uncertainty, human interaction, and complex, competing objectives to substantially improve the safety, enjoyability, and efficiency of transportation. These same fundamental techniques will, in addition, provide the basis for revolutionary advances in intelligent robots for homes, hospitals, and factories. **□**

Leslie Pack Kaelbling is a professor of computer science and engineering at the Massachusetts Institute of Technology, Cambridge, MA.

© 2010 ACM 0001-0782/10/0400 \$10.00

Toward Robotic Cars

By Sebastian Thrun

This article advocates self-driving, robotic technology for cars. Recent challenges organized by DARPA have induced a significant advance in technology for autopilots for cars; similar to those already used in aircraft and marine vessels. This article reviews this technology, and argues that enormous societal benefits can be reaped by deploying this emerging technology in the marketplace. It lays out a vision for deployment, and discusses some of the key remaining technology obstacles.

1. INTRODUCTION

Perhaps no invention has influenced the 20th century more than the automobile. Most of us use cars daily, as our primary mode of transportation. In fact, there are presently over 800 million cars on the road worldwide.¹¹ Car-related expenses constitute the second highest spending of the average American family, as 87% of the working population commute solely by car. As a result, cars consume approximately 34% of the Nation's energy.^{9,15}

Despite the importance of the automobile, insufficient innovation has occurred in past decades. Today, cars are *grossly inefficient* when it comes to basic resources, such as human health, energy, and human productivity. In the United States alone, 42,000 people die annually in nearly 6 million traffic accidents.⁹ Traffic jams account for 3.7 billion wasted hours of human time and 2.3 billion wasted gallons of fuel.⁵ And because of our strong emphasis on individual car ownership, cars are utilized less than 4% of their lifetime, wasting precious natural resources and space when not in use. The societal costs of our wasteful utilization of cars are truly staggering!

This article advocates robotic technology for making cars more efficient. The author conjectures that with suitable development efforts, robotic cars will critically enhance driver safety. They will reduce traffic congestion by reduced vehicle spacing and smoother driving. And they will increase resource utilization by enabling entirely new car sharing models.

The bulk of this article focuses on specific prototype vehicles. State of the art in robotic technology was showcased in a recent series of DARPA Challenges.^{1,2} The DARPA Grand Challenge required autonomous driving along desert trails. In 2005, Stanford's robot Stanley won this challenge. Two years later, Carnegie Mellon University's robot Boss won the Urban Challenge. This challenge required driverless vehicles to traverse 60 miles of urban streets.

This article reviews some of the critical technology behind these vehicles. It then lays out a technology road-map for building affordable and reliable robotic cars.

2. THE DARPA CHALLENGES

The DARPA Challenges were unique experiments by DARPA. To spur technology innovation, DARPA issues a series of

competitions endowed with significant prize money. The original Grand Challenge, announced in 2003 and first executed in 2004, required driverless robotic cars to navigate a 142 mile-long course through the Mojave desert. DARPA offered 2\$M for the fastest team that could navigate the course within a 10 h time limit. Even though the course followed well-defined desert trails, all participating robots failed within the first few miles. This outcome was often interpreted as evidence that the technology was not ready for prime time.¹²

DARPA repeated the Grand Challenge in 2005, albeit along a new route, and double the prize money. Just as in the previous year, the now-132 miles long route led through flats, dry lake beds, and along treacherous mountain passes. Out of 195 registered teams, DARPA selected 23 finalists. Four robots returned within the allotted time, with Stanford's Stanley claiming first place; see Figure 1a. The fact that four robots finished indicated a significant advance in technology within just 18 months.

In 2007, DARPA organized a new competition, the Urban Challenge. This competition took place in a mock city with paved roads. Eleven robots and about three dozen conventional vehicles navigated in a maze of city roads. When vehicles met, they had to obey California traffic rules. Carnegie Mellon University's robot "Boss," shown in Figure 1b, claimed first place, and Stanford's robot "Junior" came in second; see Figure 1c. The Urban Challenge added the challenge of other traffic. The problem solving capabilities required in the Urban Challenge were also more demanding than in the Grand Challenge, as robots had to make choices which way to travel.

Even though both competitions are far simpler than everyday driving, these challenges were milestones for the field of robotics. The capabilities demonstrated here went significantly beyond what was previously possible. Behind these advances were solid innovations in a number of core technologies that shall be discussed in turn.

3. TECHNOLOGY

3.1. Vehicle hardware

Stanley and Junior, shown in Figure 1a and c, are based on a 2004 Volkswagen Touareg R5, and a 2006 Volkswagen Passat Wagon, respectively. Both vehicles utilize custom interfaces to enable direct, electronic actuation of throttle, brakes, gear shifting, and steering. Vehicle data are communicated to Linux-based computer systems through CAN bus interfaces.

A previous version of this paper, "Stanley: The Robot that Won the DARPA Grand Challenge," was published in the *Journal of Robotic Systems* (September 2006).

Figure 1. Autonomous robots in the DARPA Challenges: (a) Stanford's robot Stanley, winner of the Grand Challenge; (b) CMU's Boss, winner of the Urban Challenge (photograph courtesy of Carnegie Mellon University, Tartan Team); (c) Junior, runner-up in the Urban Challenge; and (d) UC Berkeley's Ghost rider motorcycle (photograph courtesy of Anthony Levandowski, Blue Team).



utilizes five different laser range finders; the primary sensor is a rotating laser that scans the environment at 64 scan lines and 10Hz. Both vehicles are also equipped with radar sensors, for long range obstacle detection. The roof racks hold the antennae for the GPS inertial navigation system (INS). Computers are all mounted in the vehicle trunks. Stanley uses six Pentium M blades connected by Gigabit Ethernet, whereas Junior employs two Intel quad cores, all running Linux.

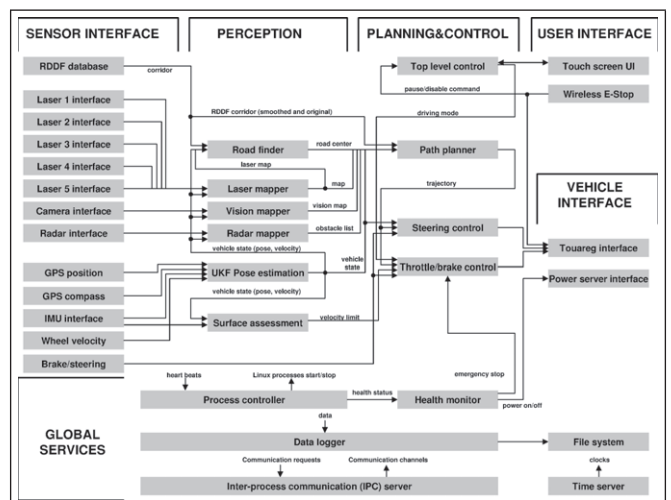
Clearly, both vehicles are conceptually similar. Perhaps the most important difference is that Stanley's environment sensors are pointed forward whereas Junior's sensors face in all directions around the vehicle. This design acknowledges the fact that Stanley only needs to sense the road ahead, whereas Junior must be aware of traffic in every direction.

3.2. Software architecture

Software is the primary contribution of both vehicles, as software is the *key* to robotic driving. Autonomous driving software generally factors into three main functional areas: perception, planning, and control. Perception addresses the problem of mapping sensor data into internal beliefs and predictions about the environment. Planning addresses the problem of making driving decisions. Control then actuates the steering wheel, throttle, brake, and other vehicle controls. Additional software modules interface to the vehicle and its sensors, and provide overarching services such as data logging and watchdog functionality.

Both vehicle's software architecture is modular. Modules run asynchronously and push data from sensors to actuators in a pipeline-fashion. Figure 2 depicts a diagram for Stanley's software; Junior's software is similar. The modularity of the software maximizes its flexibility in situations where the actual process times for data are unknown; it also minimizes the reaction time to new sensor data (which, in both vehicles, is about 300 ms).

Figure 2. Flowchart of Stanley software. The two dozen modules push data through a pipeline, comprising perception, planning, and control. Additional modules provide the IO to the vehicle, and global services such as logging.



Nearly all relevant sensors are mounted on custom roof racks. In Stanley's case, five scanning laser range finders and a color camera are mounted pointing forward into the driving direction of the vehicle, for road recognition. Junior

3.3. Sensor preprocessing

The “early” stage of perception requires data preprocessing and fusion. The most common form of fusion arises in the vehicle pose estimation, where “pose” comprises the vehicle coordinates, orientation (yaw, roll, pitch), and velocity. This is achieved via Kalman filters that integrate GPS measurements, wheel odometry, and inertial measurements.¹³

Further preprocessing takes place for the environment sensor data (laser, radar, camera images). Stanley integrates laser data over time into a 3-D point cloud, as illustrated in Figure 3. The point cloud is then analyzed for vertical obstacles, resulting in 2-D maps as shown in Figure 3. Because of the noise in sensor measurements, the actual test for the presence of a vertical obstacle is a probabilistic test.¹⁴ This test computes the probability of the presence of an obstacle, considering potential pose measurement errors. When this probability exceeds a threshold, the map is marked “occupied.”

A similar analysis takes place in Junior. Figure 4 illustrates a scan analysis, where adjacent scan lines are analyzed for obstacles as small as curbs.

Perhaps one of the most innovative elements of autonomous driving pertains to the fusion of multiple sensors. Stanley, in particular, is equipped with laser sensors whose range only extends to approximately 26 m. At this range, it is impossible to see obstacles in time to avoid them.

Adaptive vision addresses this problem.³ Figure 5 depicts camera images segmented into drivable and undrivable terrain. This segmentation relies on the laser data. The adaptive vision software extracts a small drivable area right in front of the robot, using the laser obstacle map. This area is then used to train the computer vision system, to recognize similar color and texture distributions anywhere in the

Figure 3. Stanley integrates data from multiple lasers over time. The resulting point cloud analyzed for vertical obstacles, which are avoided.

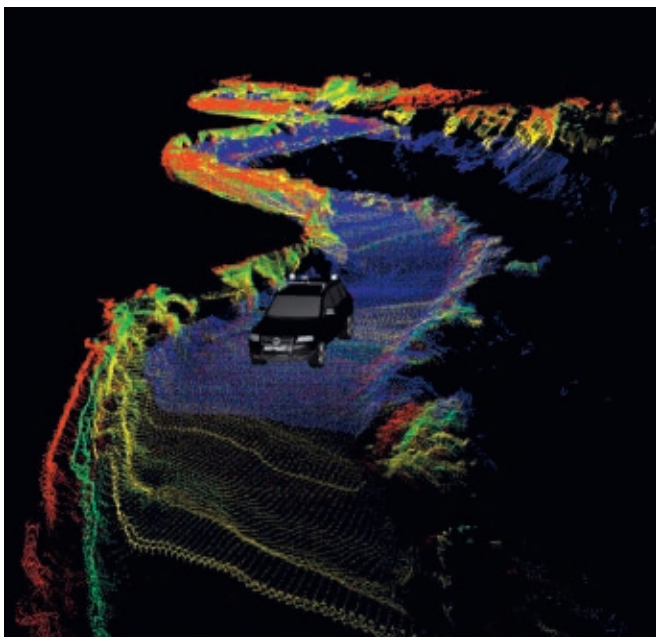


Figure 4. Junior analyzes 3-D scans acquired through laser range finder with 64 scan lines. Shown here is a single laser scan, along with the corresponding camera view of the vehicle.

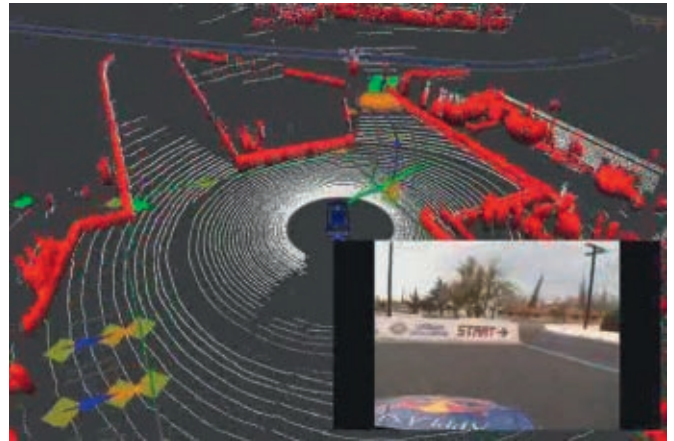


Figure 5. Camera image analysis is based on adaptive vision, which leveraged short-range laser data to train the system to recognize similar-looking terrain at greater distance.



image. The adaptation is performed ten times a second, so that the robot continuously adapt to the present terrain conditions. Adaptive vision enhances the obstacle detection range by up to 200 m, and it was essential in Stanley's ability to travel safely at speeds of up to almost 40 mph.

3.4. Localization

In both challenges, DARPA supplied contestants with maps of the environment. Figure 6 shows the Urban Challenge map. The maps contained detailed information about the drivable road area, plus data on speed limits and intersection handling.

Localization addresses the problem of establishing correspondence between the robot's present location, and the map. At first glance, the INS system may appear sufficient to do so; however, the typical INS estimation error can be a meter or more, which exceeds the acceptable error in most cases. Consequently, both robots relate features

Figure 6. Course map provided by DARPA, here shown with our data processing tool.



visible in the laser scans to map features, to further refine localization.

Stanley’s localization only addresses the lateral location of the robot relative to the map. Figure 7 illustrates the analysis of the terrain for a discrete set of vertical offsets. Localization then adjusts the estimated INS pose estimates such that the center line of the road in the map aligns with the center of the drivable corridor. As a result, Stanley tends to stay centered on the road (unless, of course, the robot swerves to avoid an obstacle).

Junior’s localization is essentially identical, but using infrared remission values of the laser in addition to range-based obstacle features. Infrared remission facilitates

Figure 7. Localization uses momentary sensor data to estimate the location of the robot relative to the map with centimeter precision. In Stanley, the localization only estimated the lateral location, as indicated by the lateral offset bars.

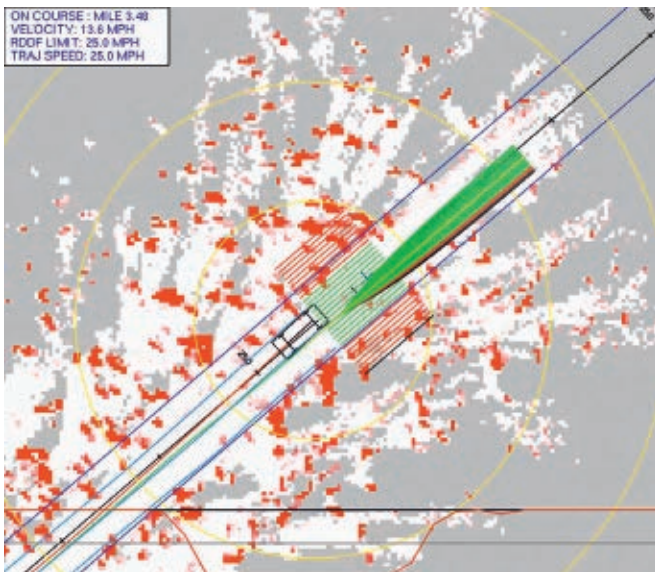
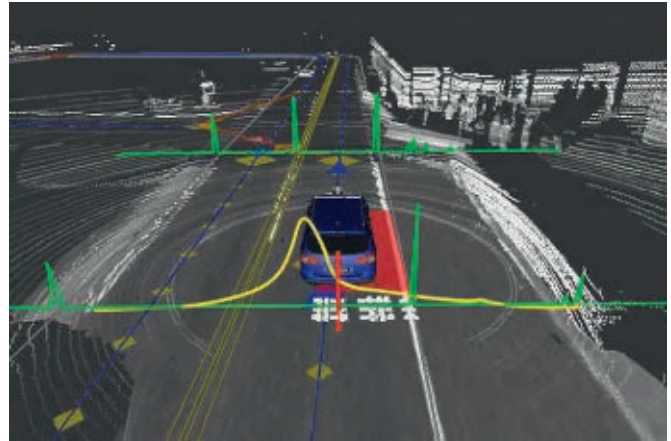


Figure 8. Lateral localization in Junior, based on infrared remission values acquired by the laser. The yellow graph depicts the posterior lateral position estimate.



the detection of lane markings, which are not detectable with range. Figure 8 illustrates an infrared remission scan, superimposed with the localization results. The yellow curve in this figure represents the posterior distribution over the lateral offset to the map, as estimated by fusing INS and remission values. In this specific instance, the localizer reduces the GPS error from about a meter to a few centimeters.

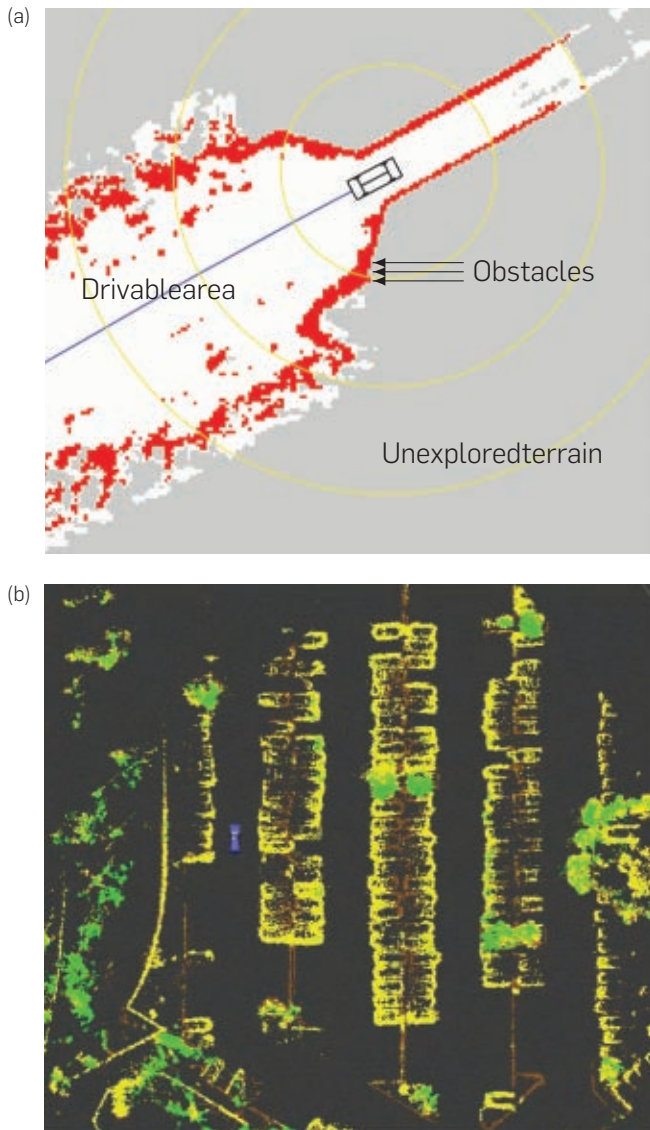
3.5. Obstacle tracking

Roads are full of obstacles. Many are static, such as ruts and berms in the desert, or curbs and parked vehicles in an urban environment. To avoid such static obstacles, both vehicles build local occupancy grid maps⁸ that maintain the location of static obstacles. Figure 9 shows examples of a maps built by both robots. Whereas Stanley distinguished only three types of terrain—drivable, occupied, and unexplored—Junior also categorizes the type of obstacles by height, which leads to an approximate distinction of curbs, cars, and tall trees, as illustrated in Figure 9b.

Equally relevant is the tracking of moving objects such as cars, which play a major role in urban driving. The key element of detecting moving objects is *temporal differencing*. If two subsequent laser scans mark a region as free in one scan, and occupied in another, then this joint observation constitutes a potential “witness” of a moving object. For the situation depicted in Figure 10a, Figure 10b illustrates such an analysis. Here scan points colored red or green correspond to such witnesses. The set of witnesses is then filtered (e.g., points outside the drivable map are removed) and moving objects are then tracked using particle filters. Figure 10c depicts an example result in which Junior finds and tracks four vehicles.

Further vehicle tracking is provided using radar sensors. To this end, Junior possesses three radar detectors, one pointing straight ahead, and two pointing to each side. The radars provide redundancy in moving object detection, and hence enhance the vehicle’s reliability when merging into moving traffic; however, they are only used

Figure 9. (a) Mapping in Stanley, where terrain is classified as either drivable (white), obstacles (dark gray), or unknown (light gray). (b) Junior build more elaborate maps that distinguish curbs from vertical obstacles and overhanging trees.

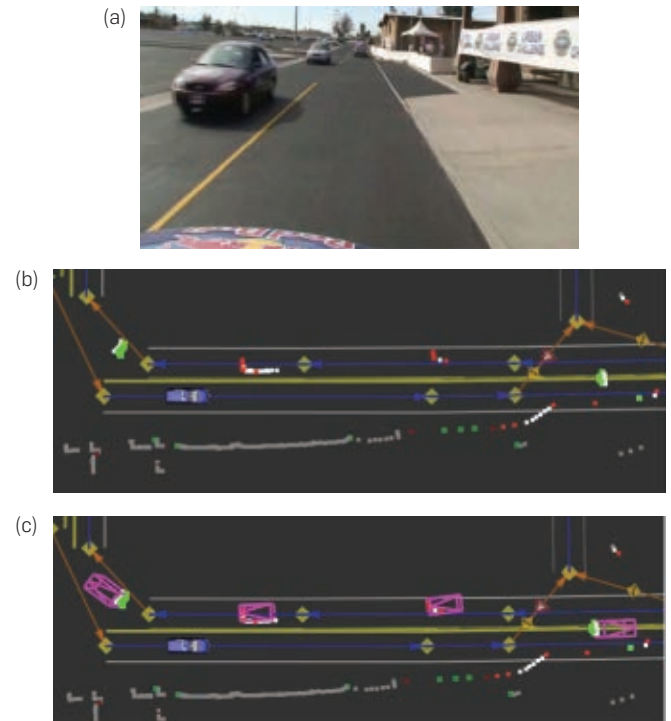


when the robot is standing still and attempting to merge into moving traffic.

3.6. Path planning

Driving decisions are made using path planning methods. Figure 11a illustrates a basic path planning technique, used by Stanley. This approach rolls out multiple trajectories to determine one that maximizes a plurality of criteria. These criteria minimize the risk of collision, but also favor the road centers over paths closer to the periphery. Search is performed along two dimensions: the amount of which the robot adjusts its trajectory laterally, and the speed at which this adjustment is carried out. Junior's basic path planning technique builds on the same idea, as illustrated in Figure 11b.

Figure 10. (a) Camera image of a scene in the urban challenge with oncoming traffic. (b) Scan differencing in this situation detects moving obstacles. (c) A particle filter tracks moving objects, as indicated by the boxes surrounding cars.



Planning in the Urban Challenge was substantially more demanding than in the Grand Challenge. In particular, the Urban Challenge required vehicles to choose their own path, and to navigate unstructured parking lots. For global path selection, Junior used a dynamic-programming-based global shortest path planner, which calculates the expected drive time to a goal location from any point in the environment. Hill climbing in this dynamic-programming function yields paths with the shortest expected travel time.

However, the momentary traffic situation may not permit driving the globally optimal path. Thus, Junior also considers local but discrete decisions, such as the lane change shown in Figure 11b, or the discrete turn decision illustrated in Figure 11c. In doing so, Junior minimizes its driving time in the context of the actual traffic situation. This approach permits Junior to react quickly and adequately to unforeseen situations, such as road blocks.

For “unstructured navigation” in parking lots, Junior uses a fast, modified version of the A* algorithm.⁴ This algorithm searches shortest paths relative to the vehicle's map, using search trees like the one shown in Figure 12. The specific modification of conventional A* pertains to the fact that robot states are continuous, and consequentially, conventional A* is not guaranteed to find realizable paths. However, by caching continuous waypoints in search nodes expanded by A*, one can guarantee that any path found is indeed realizable.

A* planning usually requires less than a second, and is performed on the fly, as Junior maps its environment.

Figure 11. (a) Stanley rolls out potential paths to avoid collisions with obstacles. (b) Junior does the same, but also considers discrete choices such as lane changes. (c) Complex set of potential paths in a multi-intersection situation.

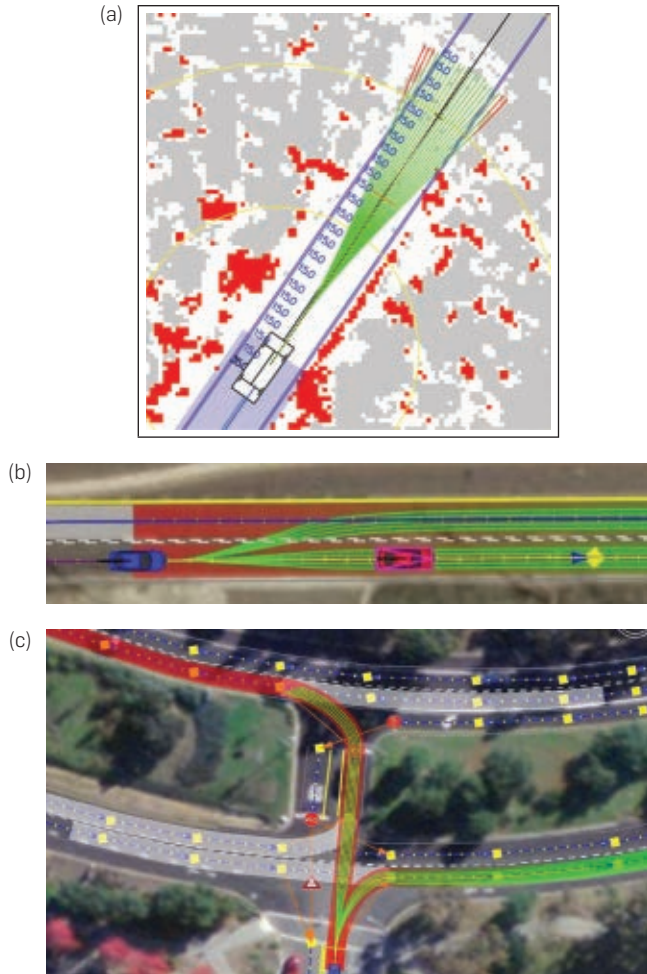


Figure 12. Junior finds paths to any target within milliseconds using a modified version of A*. Shown here are a search tree and the resulting path.

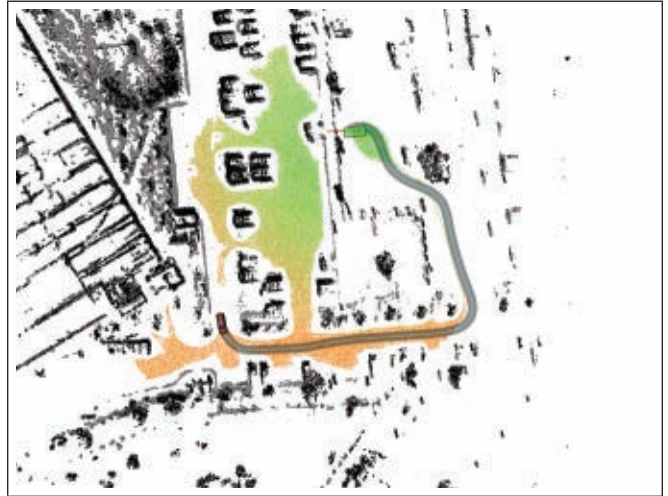


Figure 13. A* generate a U-turn maneuver for Junior, in response to a blocked road.

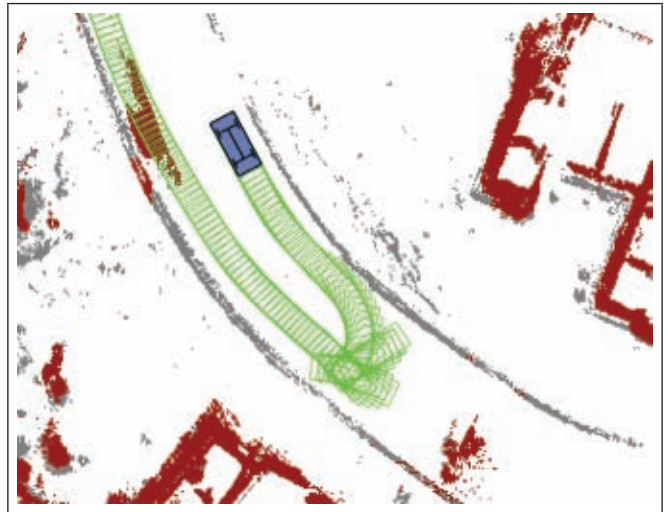


Figure 13 shows the application of this A* planner at a road blockage, where it generates a five-point U-turn.

3.7. Behaviors

Junior employs a behavioral module, which minimizes the risk of getting stuck in unpredictable environments. This module is implemented as a finite state machine, which controls the behavioral mode of the robot; see Figure 14. Under normal driving situations, driving behavior is governed by the appropriate path planner. However, when an impasse occurs, time-out mechanisms trigger to gradually permit increasingly unconstrained driving. Figure 15 illustrates this transition for a simulated traffic jam, where two other vehicles permanently block an intersection. After a time-out period, Junior invokes its unstructured A* path planner to find an unconstrained admissible route to its destination. The ability to gradually relax constraint in the driving process is essential for Junior's ability to succeed in situations as unpredictable as the Urban Challenge.

3.8. Control

The final software component realizes control of the vehicle itself, its throttle, brake, gear shifter, and steering wheel. In the actual race, steering and vehicle velocity were controlled using multiple PID controllers. Simply speaking, steering was adjusted so as to minimize any drift while pointing the front tires along the desired path. Gas and throttle were set so as to not exceed a maximum safe speed, calculated from path curvature, speed limits, and other obstacles (static and moving).

Figure 16 shows a backward side-sliding control maneuver, where the controller simultaneously controls for steering and vehicle speed. This LQR controller uses multiple control modes and vehicle models to transition

Figure 14. Junior’s behavior is governed by a finite state machine, which provides for the possibility that common traffic rules may leave a robot without a legal option as how to proceed. When that happens, the robot will eventually invoke its general-purpose path planner to find a solution, regardless of traffic rules.

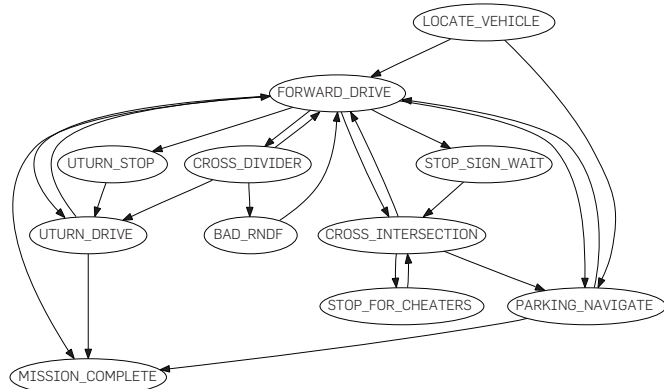
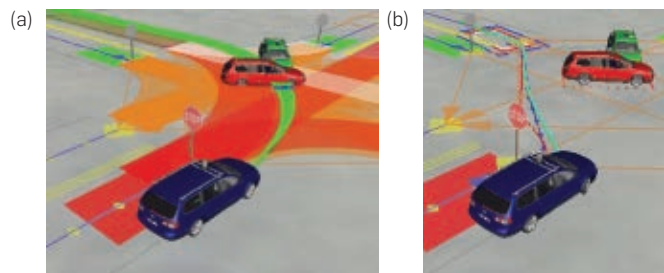


Figure 15. Resolving unexpected problems by invoking a general-purpose path planner after a wait period when facing a traffic jam. The detour is shown in (b).



from conventional driving with full tracking, and sideways sliding.⁶ This experiment illustrates the capabilities of Junior’s present-day low level controls. Clearly, no sideways sliding was required (or desirable) in the DARPA challenges.

4. THE FUTURE

The results of the DARPA Challenges were limited in many ways, yet they point at a future where cars will be safer and a whole lot more convenient.

Future versions of the DARPA Challenge technology may be leveraged into new generation of cars that can “take over” during our daily commutes. Initially, such a take-over may only occur on highways, as limited-access highways are by far the easiest environments for robotic driving. Later, robotic cars may provide door-to-door chauffeur services, very much like a taxi cab without the driver. In doing so, we might exploit some of the advantages of robotic technology over human driving. For example, it seems technically feasible to operate robotic cars at distances of less than 10 m apart at highway speeds. This could double the throughput of highways relative to today, and also decrease energy consumption.⁷ For the customer, such a technology would free

Figure 16. Four snapshots of the car sliding backward into a parking spot.



up significant time—in many cases more than an hour per working day.¹

Robotic technology may also be leveraged to move unoccupied cars. At airports, rental cars may pick up their customers on the curbside, so no more waiting at the rental car counter. But the real potential lies in car sharing. As mentioned in the introduction to this article, cars are only utilized 4% of their lifetime. What if we could, on the click of a button, order a rental car straight to us. And once at our destination, we wasted no time looking for a parking; instead we just let the car drive away to pick up its next customer. Such a vision could drastically reduce the number of cars needed, and also free up important other resources, such as space consumed by parked cars. Perhaps in the future, most of us share cars, enabled through robotic technology.


All these visions require further advances in the technologies discussed. It seems pretty obvious that the main elements for autonomous driving are in place: perception, planning, and control. But there exists a range of challenges that present-day technology fails to fully address.

- One key challenge comes from “low-frequency” change. New roads are built; existing roads are repurposed; lanes are moved; construction zones may block lanes and alter the traffic flow. Any self-driving car needs to react adequately to such rare events. None of the DARPA challenges addressed these issues.
- Further, any robotic car must equal or surpass human reliability. All components of these cars—the sensors, computers, actuators, operating systems, and software—must become orders of magnitude more reliable to meet these goals. The DARPA Challenges established new milestones for robotic autonomy and reliability, but these systems are still far too unreliable for practical use.
- Finally, people need to feel comfortable in a robotic car. There is an urgent need to develop user interfaces and modes of control that make people feel comfortable with this new concept. Research is needed on the type information provided to the human driver and on modes to integrate human and robotic control. There

may be situations that even robotic technology will be unable to handle, which raises questions on how to best leverage human intelligence and driving skills should such situations occur.

Still, the only way to turn this new type of transportation into reality is to invest massively into the vision of smart, robotic cars. The benefits to society will be enormous. We need to overcome the old belief that only people can drive cars, and embrace new modes of transportation that utilize the twenty-first century technology. When this happens, we will free up significant resources that are presently wasted in the inefficiency of today's car-based society.

Acknowledgments

The author thanks the members of the Stanford Racing Team, who were essential in building the vehicles Stanley and Junior. Major sponsorship was provided by Volkswagen of America's Electronics Research Lab, Mohr Davidow Ventures, Android, Red Bull, Android, Google, Intel, NXP, and Applanix—all of which are gratefully acknowledged. The author also thanks DARPA for organizing both challenges, and for providing financial support under its Urban Challenge program. 

References

1. Buehler, M., Iagnemma, K., Singh, S. (eds.). *The 2005 DARPA Grand Challenge: The Great Robot Race*. Springer, Berlin, 2006.
2. Buehler, M., Iagnemma, K., Singh, S. (eds.). *The 2005 DARPA Urban Challenge: Autonomous Vehicles in*

3. Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., Bradski, G. Self-supervised monocular road detection in desert terrain. In *Proceedings of the Robotics Science and Systems Conference*. Sukhatme, G., Schaal, S., Burgard, W., Fox, D. (eds.) (Philadelphia, PA, 2006).
4. Dolgov, D., Montemerlo, M., Thrun, S. Path planning for autonomous driving in unknown environments. In *Proceedings of the International Symposium on Experimental Robotics (ISER)* (Athens, Greece, 2008). Springer Tracts in Advanced Robotics (STAR).
5. The extent of and outlook for congestion: Ministerial meeting on mitigating congestion, 2007. <http://www.internationaltransportforum.org/sofia/sofiadocs.html>.
6. Kolter, G., Plagemann, C., Jackson, D., Ng, A., Thrun, S. Hybrid optimal open-loop probabilistic control, with application to extreme autonomous driving. Submitted for publication, 2009.
7. Michaelian, M., Browand, F. Field experiments demonstrate fuel savings for close-following. Technical Report PATH UCB-ITS-PRR-2000-14, University of California at Berkeley, 2000.
8. Moravec, H.P. Sensor fusion in certainty grids for mobile robots. *AI Mag.* 9, 2 (1988), 61–74.
9. *National Transportation Statistics*. Bureau of Transportation Statistics, Department of Transportation, 2006.
10. *Omnibus Household Survey Information American Commuting Time*. Bureau of Transportation Statistics, Department of Transportation, 2006.
11. Passenger cars, 2006. <http://www.sasi.group.shef.ac.uk/worldmapper/display.php?selected=31>.
12. Singer, P.W. *Wired for War*. Penguin Press, 2009.
13. Thrun, S., Burgard, W., Fox, D. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.
14. Thrun, S., Montemerlo, M., Aron, A. Probabilistic terrain analysis for high-speed desert driving. In *Proceedings of the Robotics Science and Systems Conference*. Sukhatme, G., Schaal, S., Burgard, W., Fox, D. (eds.) (Philadelphia, PA, 2006).
15. *Transportation Energy Data Book (Edition 27)*. US Department of Energy, 2008. <http://cta.ornl.gov/data/index.shtml>.

Sebastian Thrun (thrun@stanford.edu),
Computer Science Department, Stanford
University, Stanford, CA.

© 2010 ACM 0001-0782/10/0400 \$10.00

Announcing ACM's New Career & Job Center!

Are you looking for your next IT job? Do you need Career Advice?

Visit ACM's newest career resource at <http://www.acm.org/careercenter>

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 corporate job postings
- Resume posting – stay connected to the employment market and maintain full control over your confidential information
- An advanced Job Alert system that notifies you of new opportunities matching your criteria
- Live career advice to assist you in resume development, creating cover letters, company research, negotiating an offer and more



The **ACM Career & Job Center** is the perfect place to begin searching for your next employment opportunity!
Visit today at <http://www.acm.org/careercenter>

Aalto University Postdoctoral and Senior Researcher positions in computing research

Helsinki Institute for Information Technology HIIT and the Aalto University Department of Information and Computer Science are inviting applications for postdoctoral and senior researcher positions in several areas of computing research including: machine learning and data analysis; computational methods for networks, interaction and economics; large constraint models; nanoscale self-assembly; enhancement of internet infrastructure; human-centric ubiquitous IT.

The closing date for applications is 15 March 2010. The positions will be filled for three years maximum starting at the earliest 1 August 2010.

Further details of the posts and the application procedure are available at:

<http://www.hiit.fi/jobs> (3 positions)

<http://ics.tkk.fi/en/vacancies> (4 positions)

Aalto University is a newly created research university resulting from the merger of three Finnish universities: Helsinki University of Technology TKK, the Helsinki School of Economics, and the University of Art and Design Helsinki. The new university was launched in January 2010, and opens up a new world of possibilities for multidisciplinary education and research. For further information, see www.aalto.fi/en/.

HIIT is a joint research institute of Aalto University and the University of Helsinki conducting basic and strategic research on information technology.

Columbia University Research Scientist - All Ranks

The Center for Computational Learning Systems is seeking highly qualified researchers who will work with Machine Learning, energy, NLP researchers and software engineers in a Smart Grid pilot funded by the US Department of Energy and other agencies. Research Scientists are expected to have demonstrated innovation and excellence in research have publications in top journals and conferences and for Research Scientists and Sr. Research Scientists a strong record of attracting grant support. Industry experience is desired and that experience should include a strong record of successful research and technology transfer along with management experience. Appointments to the Center will be for fixed renewable terms.

For complete job description, requirements and to apply, applicants should apply to our online RAPS system:

<https://academicjobs.columbia.edu/applicants/Central?quickFind=52620>

Screening of applicants will begin February 1, 2010. The search will remain open no less than 30 days from the date of posting.

Qualified applicants should submit a CV,

statement of research, research abstract and a list of references.

Columbia University is an Equal Opportunity/Affirmative Action employer.

Columbia University Staff Associate and Sr. Staff Associate Positions

The Center for Computational Learning Systems at Columbia University is seeking highly qualified applicants to fill positions of Staff Associate and/or Sr. Staff Associate with specialties in computer software architecture, database engineering and financial engineering. The potential individuals will work with the machine learning, energy, NLP researchers and software engineers in a Smart Grid pilot project funded by the US Department of Energy and other agencies. Appointment to the Center will be for a fixed renewable term.

For Staff Associate position:

Bachelor's degree and 4 years related experience required or a Masters degree and 2 years of experience.

For Senior Staff Associate position:

Bachelor's degree and 8 years related experience required or a Masters degree and 4 years of experience.

General requirements include:

- ▶ Strong analytical and quantitative ability
- ▶ Strong interpersonal, organizational and communication skills
- ▶ Able and willing to work within an academic environment
- ▶ A good combination of education and experience is highly desired.

Qualified applicants should submit a CV, a list of references and a personal statement. A statement of research is optional.

Screening of applicants will begin on 02/01/2010. The search will remain open no less than 30 days from the date of posting.

For complete job description and to apply, applicants should apply through our online RAPS system

<https://academicjobs.columbia.edu/applicants/Central?quickFind=52618>

Columbia University is an Equal Opportunity/Affirmative Action employer.

Columbia University Staff Associate, Computer Science Dept.

Research Associate in computer graphics and software development, Columbia Univ. Computer Science Dept. For full description and to apply, visit: <https://academicjobs.columbia.edu> (Job Title: Staff Associate, Computer Science Dept., Posting ID: 0001056) EEO Statement: Columbia University is an Equal Opportunity/Affirmative Action em-

ployer. NCAA Statement: As a member of the National Collegiate Athletic Association (NCAA) and the Council of Ivy Group Presidents (Ivy League), it is imperative that members of the Columbia University community, in all matters related to the intercollegiate athletics program, exhibit the highest professional standards and ethical behavior with regard to adherence to NCAA, Conference, University, and Department of Intercollegiate Athletics and Physical Education rules and regulations.

Harvard University Tenure-track Faculty Position in Biorobotics

The School of Engineering and Applied Sciences (SEAS) and the Wyss Institute for Biologically Inspired Engineering at Harvard University (Wyss Institute) seek applicants for a tenure-track faculty position. The position will be at the level of assistant professor in SEAS in the field of Biorobotics. Potential subareas include, but are not limited to, animal-inspired robotic systems, bio-inspired adaptive locomotion and control, machine learning and robotics, prosthetics and medical or rehabilitation robotics, swarm and modular robotics, and human-robot interaction.

In addition to having a faculty appointment in SEAS, the successful candidate will also become a core faculty member of the Wyss Institute, which is composed of engineers, scientists, clinicians and theoreticians from Harvard, its affiliated hospitals, and other leading academic institutions in the Boston/Cambridge region. The Wyss Institute focuses on fundamental science-driven technology development in the field of Biologically Inspired Engineering.

For additional information, visit the following Websites:

SEAS: <http://www.seas.harvard.edu/>

Wyss Institute: <http://wyss.harvard.edu/>

Candidates must have the ability to develop a leading research program with a focus on technology development and translation. An enthusiasm for teaching is essential, and responsibilities will include both core undergraduate engineering courses as well as graduate-level courses.

An application, assembled as a single PDF file, should include a curriculum vitae, separate two-page statements of research and teaching interests, up to three scientific papers, and names and contact information for at least three writers of letters of recommendation. Applications should be sent to:

biorobotics_search@seas.harvard.edu.

Applications will be reviewed beginning February 2010 and will be accepted until the position is filled.

Harvard University is an Equal Opportunity/Affirmative Action Employer.

Applications from women and minority candidates are strongly encouraged.

Icelandic Institute for Intelligent Machines Researcher/Software Engineer

The Icelandic Institute for Intelligent Machines (IIIM) is a newly founded research institute in Reykjavik, with a focus on artificial intelligence and simulation. Over the next years IIIM aims to build a world-class research and development team. Through collaboration with world-leading partners on both sides of the Atlantic, IIIM seeks to bridge between industry and academia and increase the speed of innovation in its fields and related areas.

We have several positions open at senior and junior levels, including research scientists, software engineers, post-docs and technical specialists. Qualified applicants may advise students and teach advanced courses at Reykjavik University's School of Computer Science, the country's largest CS department.

We are particularly interested in individuals with experience in at least one of the following areas:

- ▶ Machine learning, machine vision/hearing
- ▶ Realtime/dynamical systems
- ▶ Multi-agent/high-performance systems
- ▶ Robotics, sensor fusion, signal processing

If you consider yourself capable of developing complex integrated systems and want to participate in defining the future of advanced AI and simulation, please submit your application or questions by email to: jobs@iiim.is. Applications will be reviewed four times in 2010, first week in March, May, August and November. For more information about us, see <http://www.iiim.is>

Kansas State University Department of Computing and Information Sciences Assistant/Associate Professor

The department of Computing and Information Sciences at Kansas State University invites applications for a tenure track position beginning in Fall 2010 at the level of assistant or associate professor from candidates working in the areas of health information technology and/or security. For the area of health care, we seek candidates working on designing improved health care systems which ensure patient safety, preserve privacy of data, establish high-assurance information infrastructure, and provide automated decision support capabilities.

The department offers a stimulating environment for research and teaching, and has several ongoing collaborative projects involving researchers in different areas of computer science as well as other engineering and science departments. The department has a faculty of nineteen, more than 100 graduate students, and 250 undergraduate students and offers BS, MS, MSE, and PhD degrees. Computing facilities include a large network of servers, workstations and PCs with more than 300 machines and a Beowulf cluster with 1000+ processors. Details of the CIS Department can be found at the URL <http://www.cis.ksu.edu/>.

Applicants must be committed to both teaching and research. Applicants should have a PhD degree in computer science or related discipline; salary will be commensurate with qualifications. Applications must include descriptions of teaching and research interests along with copies of

representative publications.

Preference will be given to candidates who will complement the existing areas of strengths of the department which include enterprise system security, medical data privacy, language-based security, high assurance systems, medical device plug-n-play interoperability, medical device integration frameworks, and health information management.

Please send applications to Chair of the Recruiting Committee, Department of Computing and Information Sciences, 234 Nichols Hall, Kansas State University, Manhattan, KS 66506 (email: Recruiting@cis.ksu.edu). Review of applications will commence March 1 and continue until the position is filled.

Kansas State University is an Equal Opportunity Employer and actively seeks diversity among its employees. Paid for by Kansas State University. Background Check Required.

Leica Geosystems, Inc.

Sr SW Engineer - High Definition Scanning

Prototype & create commercial grade implementations of model/feature/mesh extractopm tools. Must be hands-on, with ability to rapidly prototype ideas & develop robust software implementations. Must have advanced degree in CS related to 3D scan data processing and experience in C++/C#. Skills include: work with large scanner data sets, modern engineering practices and sense of algorithmic efficiency.

Apply URL: <http://www.leica-geosystems.com>



IN SEARCH OF EXCELLENCE!

ICCAS – Innovation Center Computer Assisted Surgery – University of Leipzig

ICCAS was founded in 2005 as one of six Centres for Innovation Competence (German: Zentren für Innovationskompetenz, ZIK) at the second-oldest university in Germany. Since then, the centre has successfully established itself as an international interdisciplinary research institution. In 2010 two new research groups will be established within the centre, supported by the German Federal Ministry of Education and Research (BMBF). ICCAS, in collaboration with the BMBF, is now seeking applications from highly motivated and outstanding junior scientists for two

Junior Research Group Leader positions

The key initiative of ICCAS is to carry out research and development work on applying methods and tools for system software for the modern surgical workplace (Therapy Imaging and Model Management System – TIMMS). Specific applications of computer-assisted surgery include, but are not limited to, neuro-, ENT- and cardiovascular surgery. This work assumes the realisation of an integrated digital operating room (OR), a surgical planning unit and other aspects of the treatment path. In order to achieve these strategic aims, two research groups will be established:

Junior Research Group: Digital Patient and Process Model

- Mathematical modelling and informatics-based structuring of patient data sets
- Specification and modelling of surgical workflows including standard operating procedures (SOP)
- Software-engineering-based prototyping of patient and process modelling systems as well as semantic-aware tools and services in the context of surgical assist systems

Junior Research Group: System Tools for Surgical Cockpit

- Integration of the patient and process models in a near-real set-up
- Knowledge and decision management
- Projects with strong relation to existing commercial products and prototypes
- Cooperation with projects focusing on OR architecture, medical technology, medical devices, human-machine-interaction

Candidates for both positions should be familiar with the broad areas of software and systems engineering, health informatics, computer-assisted radiology and surgery, or related fields. A strong background in computer science is desirable with evidence of the ability to pursue and lead a research programme. Most important, however, is the ability to build up and manage a large cross-disciplinary collaborative team.

The two junior research group leaders have the opportunity to form their own research group with four scientists each. The groups are provided with secured funding for five years, and are endowed with an above-average budget and excellent infrastructure of state-of-the-art demonstration OR and computer facilities. They will be supported by a centre manager and the administrative and scientific staff of ICCAS, and will be embedded in an active academic and student environment.

The Medical Faculty offers the possibility of a "Junior Professor" (W1) position in the above-mentioned scientific field. The successful group leader can be offered a tenured position after positive evaluation.

Please send your application by 30 April 2010 to:

Universität Leipzig
Medizinische Fakultät
ICCAS
Professor J. Meixensberger
Semmelweisstrasse 14
04103 Leipzig, Germany

and

Project Management Jülich
Division Technological and Regional Innovations (TRI)
Forschungszentrum Jülich GmbH
Zimmerstrasse 26-27
10969 Berlin, Germany
Email: k.-d.husemann@fz-juelich.de

For further information, please contact:

Professor Jürgen Meixensberger (meix@medizin.uni-leipzig.de);
<http://www.iccas.de>

For all details on the application requirements, please go to:
<http://www.unternehmen-region.de>

Louisiana Tech University
Postdoctoral Positions in the
Center for Secure Cyberspace

Applications are invited for multiple postdoctoral positions at Louisiana Tech University in the Center for Secure Cyberspace. A Ph.D. required in computer science, statistics, mathematics, or closely related field. The Center has funding to pursue applied cutting edge technologies. Research focus is on data mining, Internet security, and social networks. Salary around \$60,000. Send vita to Professor Vir Phoha at phoha@latech.edu

NIIT University, INDIA
Department of Computer Science & Engg
and Department of Information and
Communication Technology
Faculty Positions in Computer Science &
Engg and Information and Communication
Technology

NIIT University invites applications at Assistant Professor, Associate Professor and Professor levels.

Candidates must have earned (or expect shortly) a Ph.D for Assistant Professor, Ph.D with 5 years of experience for Associate Professor, Ph.D with 10 years of experience for Professor in the above disciplines. Candidates must have a good publication record; proven ability to establish an independent research program; demonstrate flair for innovation; be open to participate in developing new interdisciplinary programs of study; have the commitment to excel both in research and teaching and establish linkage with industry. Industry experience and/or post-doctoral experience will be considered an asset.

NIIT University is located about 120 Kms from New Delhi. The University provides an intellectual environment and is committed to academic excellence. The four core principles of NIIT University, namely, Industry-linked, Technology-based, Research-driven and Seamless define the DNA of the University.

For more information, please visit the university website www.niituniversity.in.

Interested applicants are invited to submit their curriculum vitae including employment history, a statement outlining research and teaching interests, list of consultancies and projects undertaken and names of at least three referees.

Applications may be sent electronically in PDF format to careers@niituniversity.in

NEC Laboratories America, Inc.
Research Staff Member -
Large-Scale Distributed Systems

NEC Laboratories America, Inc. (<http://www.nec-labs.com>) conducts research in support of NEC U.S. and global businesses. The research program covers many areas - reflecting the breadth of NEC business - and maintains a balanced mix of fundamental and applied research.

The Large-Scale Distributed Systems group conducts advanced research in the area of design, analysis, modeling and evaluation of distributed systems. Our current focus is to create innovative technologies to build next generation large-scale computing platforms and to simplify and automate the management of complex IT systems and services. Our researchers have expertise in

networking, statistics, modeling, distributed systems, and operating systems. Our group has many ongoing projects, especially in the emerging Cloud Computing area. We strongly believe in publishing our research and advancing the state-of-the-art. We also build technologies that solve real world problems and ultimately help industry business needs. Many of our research results have been /will be transferred into industry products.

The group is seeking a member to work in the area of system modeling and analysis. The candidate must have a PhD in CS/CE with strong publication records in related areas. He/she is expected to develop advanced system management solutions by utilizing the mathematical skills, especially in data mining and machine learning, as well as the knowledge about system architecture and design. Specifically the candidate will analyze the vast amount of system measurement data to model and infer the system operational behavior. The qualified person must have research experiences that cover the following topics:

- ▶ Distributed systems and data center
- ▶ Statistical data mining, optimization, and information theory
- ▶ Virtualization and resource provisioning
- ▶ Performance, reliability, dependability and security
- ▶ Autonomic computing

For consideration, please forward your resume and a research statement to recruit@nec-labs.com and reference "ASDS-RSM" in the subject line.

EOE/AA/MF/DV.

Sandia National Laboratories
Math & CS Research
Discrete Math and Computer Science R&D for
Social and Computer Network Analysis

Sandia National Laboratories seeks new PhD researchers for long-term positions in mathematics and computer science for understanding large-scale, complex, social and engineered networks. Of particular interest are experts in computational topology, graph-feature identification, community detection, statistics, machine learning, computational linguistics, and uncertainty.

PhD in CS, math, statistics, or equivalent is required. Publications, software, and application experience is desired. The position involves national security applications; the ability to obtain and maintain a U.S. security clearance is required.

Apply at <http://www.sandia.gov/careers> to Job ID 64380 before 3 April 2010.

See <http://www.cs.sandia.gov/hpc-informatics/careers/index.htm> or contact Brett Bader (bwbader@sandia.gov) for details.

Sandia National Laboratories is an Equal Opportunity Employer M/F/D/V.

Santa Clara University
Dept. of Computer Engineering
Tenure-Track Assistant or
Associate Professor opening

The Department of Computer Engineering (<http://www.scu.edu/engineering/cse/>) at Santa Clara University (www.scu.edu) invites applications for a ten-

**ASSISTANT, ASSOCIATE,
FULL PROFESSOR
Bioinformatics**

Hunter College of The City University of New York invites applications for a tenure-track faculty position in bioinformatics and high performance computing with expertise in the development, adaptation and implementation of HPC algorithms applied to such data-intensive biological areas as (but not limited to) next-generation sequencing, biomedical ontologies, and systems biology.

The successful candidate will be appointed to the doctoral faculty at the CUNY Graduate Center and will participate actively in Hunter's NIH-funded Quantitative Biology (QuBi) Project.

The anticipated start date is Fall 2010. The appointment will be made in the Department of Computer Science or Biological Sciences depending on qualifications. Preference will be given to candidates with well-established strong records of peer-reviewed publications and external funding, but exceptional junior candidates will also be considered. Evaluation is ongoing and will continue until the position is filled.

Hunter's computing and biomedical research community, is supported by extensive funding from NIH and NSF. Hunter College is strategically located in Manhattan near other major biomedical research centers and universities and attracts an engaging and diverse student body.

For information on how to apply, visit:

<http://www.hunter.cuny.edu/qubi/>

Questions may be emailed to: bioinformatics@hunter.cuny.edu

The City University of New York
An Equal Employment Opportunity/Affirmative
Action/Immigration Reform and Control Act/
Americans with Disabilities Act Employer



ure-track Assistant or Associate Professor opening. The Department has a particular preference for individuals with teaching and research interests in Web Science, Computer Networks, Information Assurance, or Media Technology, but strong candidates will be considered in all computing areas.

For more information and application procedure, please visit <http://scu.edu/hr/careers/faculty.cfm?id=2504>.

Applicants should submit detailed CVs, statements of research interests, statements of teaching interests, and names and contact information for three references. Electronic submission of applications is preferred, with PDF formatted documents mailed to: coensearch@scu.edu.

Santa Clara University is an Equal Opportunity/Affirmative Action employer, committed to excellence through diversity and inclusion, and, in this spirit, particularly welcomes applications from women, persons of color, and members of historically underrepresented groups. The University will provide reasonable accommodations to individuals with a disability.

United States Naval Academy Director, Center for Cyber Security Studies

The Center for Cyber Security Studies encompasses support for all programs that contribute to knowledge, study, research and practice of cyber warfare and information dominance at the U.S. Naval Academy. The Director of the Center for Cyber Security Studies will develop, manage and nurture a comprehensive institution-wide program in cyber warfare

by coordinating curriculum development, facilitating the sharing of expertise and perspectives in cyber warfare from across the Academy and enhancing inter-disciplinary research and practical exercises in cyber warfare. The Director will take a leadership role in defining the future directions of the Center through the development of new resources, the coordination of joint interdisciplinary research efforts between the Naval Academy and other institutions and agencies, and through the development of goals and strategic annual and long-range plans for the Center. Complete details can be found at www.usna.edu/cyber. Apply URL: <http://www.usna.edu/cyber>

University Of Detroit Mercy Computer Science / Software Engineering Tenure Track Faculty Position

University of Detroit Mercy seeking a faculty member in Computer Science or Software Engineering. Send e-mail to: Dr. Kevin Daimi at daimikj@udmercy.edu. The deadline for submissions is April 16, 2010.

More information is available at: <http://eng-sci.udmercy.edu/opportunities/index.htm>

University of Mary Hardin Baylor Department of Computer Science and Engineering Faculty Position

The University of Mary Hardin Baylor's College of Sciences seeks applicants for a full-time, tenure-

track faculty position in our Department of Computer Science and Engineering beginning August 2010. Doctorate in Computer Science, Engineering or a closely related field and university-level teaching experience are required. UMHB seeks active, committed Christians who are dedicated to teaching within the context of a Christ-centered institution. To apply, visit www.umhb.edu (Resources and Services, Employment).

University of Minnesota Post-Doctoral Associate

Post-Doc will work in the Institute for Health Informatics at the University of Minnesota. Job duties include: Manager research projects, Write and manage grants, Collect and analyze data, Write publications, Support curriculum development. Apply URL: <http://employment.umn.edu/applicants/Central?quickFind=84882>

University of South Carolina Upstate Assistant or Associate Professor of Computer Science Computer Science - Requisition #002233

Assistant or Associate Professor in Computer Science beginning August 2010. Ph.D. in CS or a closely related field required. Review of applications begins March 15, 2010. Contact: Dr. Jerome Lewis, Chair, jlewis@uscupstate.edu, 864-503-5305 For complete information and online application submission process go to <http://www.uscupstate.edu/jobs>. **The University of South Carolina is an equal opportunity institution.**

University of Tartu Professor of Information Security The Institute of Computer Science of the University of Tartu invites applications for the position of: Full Professor of Information Security

This position is part of the institute's strategic goal to develop a strong international profile by establishing world-class research groups, by developing international degree programmes, and by engaging in high-impact applied research and technology transfer.

The successful candidate is expected to establish a research group in information security. Seed funding is provided for recruiting PhD students and/or postdocs. The successful candidate is also expected to lead research projects in the context of the Estonian Centre of Excellence in Computer Science and the Software Technology & Applications Competence Center, of which the institute is a partner. The successful candidate will also enhance the existing Masters curricula, particularly the Masters of Cyber-Security. The working language of the institute is English.

Employment conditions include a salary of approximately €5000 gross per month and an annual leave of 55 calendar days. The position is initially for 5 years. Tenure at University of Tartu is achieved after two successful periods. This position is co-financed by the European Social Fund's DoRa program.

The deadline for applications is 30 April 2010. Details of the application procedure will be posted at: <http://www.ut.ee/en/employment>

For further information contact Peeter Laud (Peeter.Laud@ut.ee)



كاوست
KAUST Through Inspiration, Discovery
King Abdullah University of Science and Technology

Postdoctoral Fellowships and Research Scientists

This is an ideal time to consider joining the King Abdullah University of Science and Technology (KAUST) family, as the University embarks on a period of unparalleled growth and opportunities abound for career enrichment.

The Division of Mathematical & Computer Sciences & Engineering at KAUST invites applications for Postdoctoral Candidates and Research Scientists for multiple positions in several areas of computer systems research, including hardware and software systems, operating systems, parallel and concurrent programming, virtualization, energy-efficiency, and cloud computing technologies.

The successful candidates will conduct world-class research and develop computer systems, as well as be involved in dissemination of research results through conference and journal publications and conference presentations. Candidates may also have the opportunity to teach graduate-level seminar courses subject to approval by the Dean.

Qualifications: Candidates should have recently obtained a PhD in Computer Science.

Application Requirements: CV (include contact e-mail, postal address, phone number); Thesis summary (one-page maximum); Research proposal (no more than two pages); Reprint of applicant's most significant research publications; Names and contact information for three references.

Initial appointment will be for one year with potential for renewal for an additional 1-2 years.

KAUST's globally competitive pay structure is the core of its Total Rewards package. Salaries are offered on a graded system that includes all positions within the University.

Applications for the positions are accepted on a continuing basis until the position is filled.

The King Abdullah University of Science and Technology (KAUST) in Saudi Arabia is an international, graduate-level research university dedicated to inspiring a new age of scientific achievement in the Kingdom, the region, and beyond. As an independent, merit-based institution, KAUST will enable top researchers from around the globe and across all cultures to work together to solve challenging scientific and technological issues. This environmentally sensitive campus, located on 3,200 acres on the Red Sea at Thuwal, opened in September 2009 has already attracted some of the finest minds around.

To apply visit: <http://aptrkr.com/138114>



[CONTINUED FROM P. 112] but it puts a lot of obstacles in the way of somebody trying to get into the system.

How do you think microkernel systems can penetrate the broader consumer marketplace?

I don't think it's going to be easy, but I can think of a couple routes by which it could happen. Some people in the European Union have talked about changing regulations to require software to fall under the same liability laws as everything else. If you make a tire, and one in 10 million explodes, you can't say, "Well, tires explode sometimes." Why isn't software like any other product? Imagine if there were some liability associated with its not working. Manufacturers would suddenly be very interested in making things reliable.

You've also worked on security-related projects, like an e-voting system.

The trouble with voting machines is knowing that the software can be trusted. And cryptographers who work in this area have complicated schemes

"I think microkernels are a more reliable way of doing things."

to address that, but the schemes are so complicated that virtually nobody except a professional cryptographer could understand them. We've designed a scheme which, though it uses cryptography, is much simpler and easier to use.

We assume that the software is open source. At the time you go to vote, you could come with some handheld device and query the voting machine, "Give me a cryptographic checksum of the software currently in your memory." Then you could check that and know the software running on the machine is the software that's supposed to be there. We have a lot of other design issues dealing with how the keys

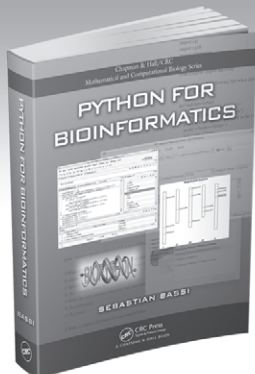
are distributed, and not trusting any single party. But basically we're trying to design a system that's more secure than current ones and less subject to hacking.

It sounds like a very comprehensive approach.

I've always tried to look at the systems aspect of a problem. I've also done work on sensor networks—people have proposed dropping sensors along the national border to prevent people from sneaking in. The only thing they worry about is, "Suppose someone captures a sensor and steals all the keys?" What they hadn't considered is, "What is the range of a sensor? How does it detect the difference between an illegal immigrant and a rabbit?" Any system will be attacked at the weakest link, so you have to pay attention to where the weakest link is, and not which is the most mathematically interesting problem. □

Leah Hoffmann is a Brooklyn, NY-based technology writer.

© 2010 ACM 0001-0782/10/0400 \$10.00



Python for Bioinformatics

Sebastian Bassi

Universidad Nacional de Quilmes, Bernal, Argentina

Requiring no prior knowledge of programming-related concepts, this volume focuses on the easy-to-use, yet powerful, Python computer language. It contains working code that solves real-world biological problems and includes a DVD with a ready-to-run virtual machine (VM) based in DNALinux to test the code.

Catalog no. C9292

January 2010, 587 pp., Soft Cover

ISBN: 978-1-58488-929-8, \$69.95 / £44.99

After Savings: \$55.96 / £35.99

Save 20% on these new titles in data mining, bioinformatics, and machine learning from CRC Press!



CRC Press
Taylor & Francis Group

Coming Soon!

Patterns of Data Modeling

Michael Blaha

OMT Associates, Inc. Placida, Florida, USA

"... The breadth of coverage is enormous, ranging from basic data structures through star schema, archetypes for representing commonly found concepts, and canonical models for tough problems."

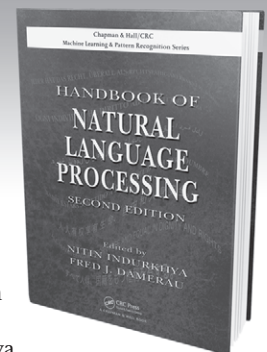
—Paul C. Brown, Principal Software Architect, TIBCO Software Inc.

Catalog no. K11048

April 2010, c. 264 pp., Soft Cover

ISBN: 978-1-4398-1989-0, \$49.95 / £31.99

After Savings: \$39.96 / £25.59



Handbook of Natural Language Processing

Second Edition

Edited by

Nitin Indurkha

University of New South Wales, Sydney, Australia

Fred J. Damerau

Goshen, Connecticut, USA

This comprehensive, modern handbook presents practical tools and techniques for implementing natural language processing in computer systems. It includes a new applications section, a broader multilingual scope to include Asian and European languages, and an actively maintained wiki (<http://handbookofnlp.cse.unsw.edu.au>) that provides supplementary information.

Catalog no. C5921

February 2010, c. 704 pp.

ISBN: 978-1-4200-8592-1, \$99.95 / £63.99

After Savings: \$79.96 / £51.19

Enter promo code 694AM at www.crcpress.com to receive 20% discount.

Offer expires May 31, 2010.

Q&A

Systematic Thinking

Andrew S. Tanenbaum talks about MINIX, microkernels, and electronic voting systems.

ANDREW S. TANENBAUM, a professor of computer science at the Vrije Universiteit in Amsterdam, has been at the forefront of operating systems design for more than 20 years. For an appreciation of Tanenbaum's sense of humor, academic musings, and philosophy of life, see his homepage's FAQ (<http://www.cs.vu.nl/~ast/home/faq.html>).

You are best known for MINIX, the Unix-like operating system that you created in 1987.

Version 6 of UNIX was widely used in universities, and it was a popular tool in operating systems courses. Then some bean counter at AT&T said, "Gee, if we keep this secret, we can make more money," so they put a clause in the version 7 contract saying you couldn't teach it anymore. At that point I decided I would write something that looked quite a bit like UNIX but was my own code and wasn't tied to their license.

What's going on with MINIX now? I understand you continued to develop and refine it over the years.

Yes, and in 2004 I decided to pick it up again and really make the point that I think microkernels are a more reliable way of doing things.

This is the idea that we can build more reliable operating systems by breaking up the components.

I find it very peculiar that anybody believes you can take anything as complicated as an operating system and have it run as one gigantic program. It's just too complicated. I mean, there are



Andrew S. Tanenbaum giving a keynote presentation at linux.conf.au 2007.

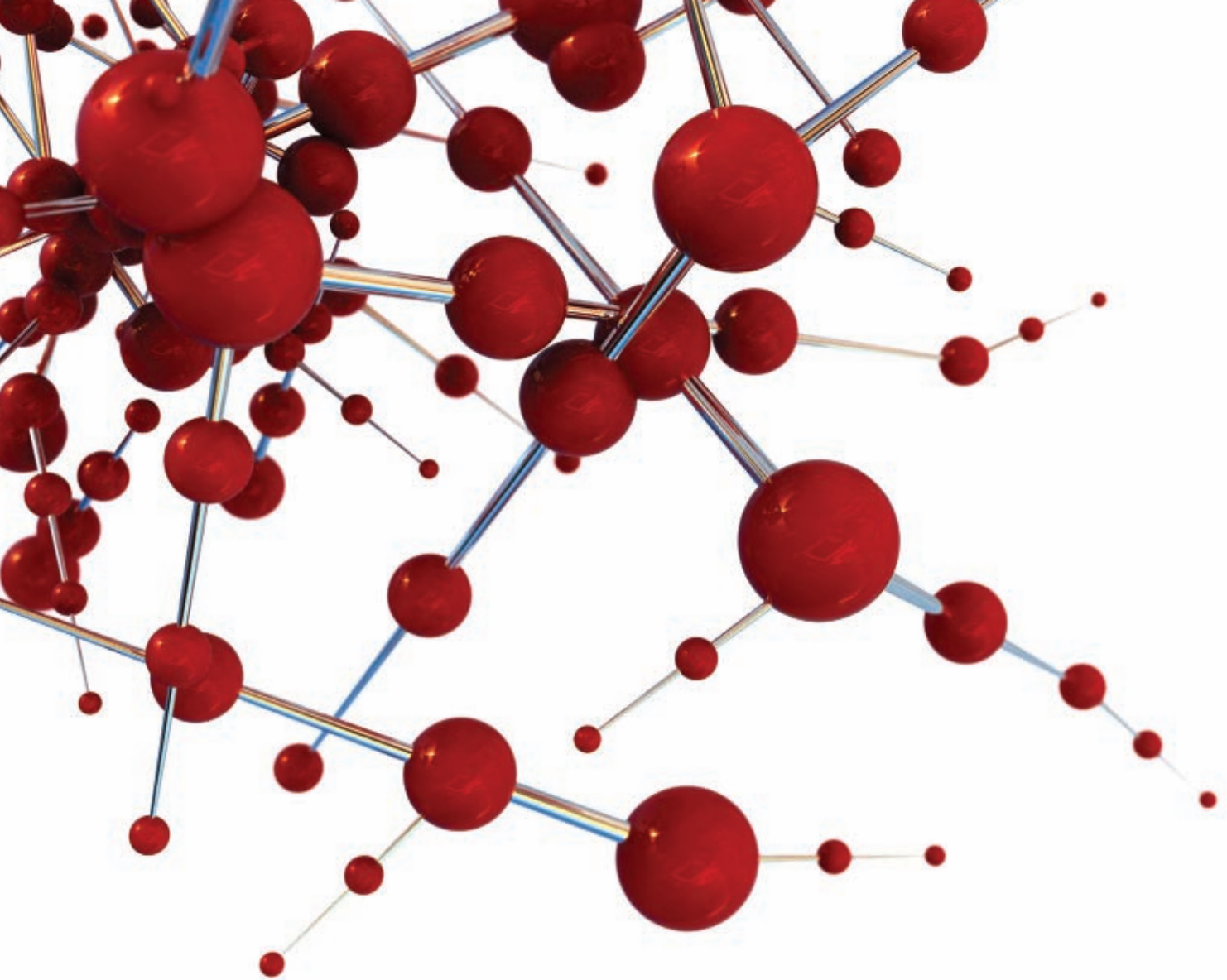
other complicated things in the world, but they're generally modular.

And that's why you decided to base MINIX on a microkernel rather than a monolithic kernel.

Yes, a microkernel is a very small piece of code running in kernel mode that provides the basis for the system, and doesn't do very much itself. Then the operating system runs as a collection of other processes in user mode. So, instead of one big program that does everything, you've got one program that drives the disk, one program that handles the audio, and one program that manages memory, each doing one, specific task and communicating with the others by well-defined protocols.

So by breaking up the operating system into well-defined pieces, you make it more reliable.

It also means you can replace pieces independently, which has a variety of consequences, such as security. MINIX has all these components—these servers and device drivers that run the user processes. Each piece has certain powers. And there's a data structure within the kernel that tells, for every process, exactly what it's allowed to do. So if a hacker found a vulnerability in the audio driver, took it over, and wanted to fork it, the kernel would say, "Sorry, you don't have permission to fork, there's no reason for audio drivers to do that." It's not perfect, of course, [CONTINUED ON P. 111]



**CONNECT WITH OUR
COMMUNITY OF EXPERTS.**

www.reviews.com



Association for
Computing Machinery

Reviews.com

They'll help you find the best new books
and articles in computing.

Computing Reviews is a collaboration between the ACM and Reviews.com.

25TH ANNUAL ACM
CONFERENCE ON
SYSTEMS,
PROGRAMMING,
LANGUAGES,
APPLICATIONS:
SOFTWARE FOR
HUMANITY

MARCH 25, 2010

Submission deadline for
OOPSLA Research Papers,
Practitioner Reports,
Educators' and Trainers' Symposium,
and proposals for Tutorials,
Workshops, Panels

APRIL 23, 2010

Onward! Papers and Essays

JUNE 24, 2010

Submission deadline for Posters,
Demonstrations, Doctoral Symposium,
Onward! Films, and Student Research
Competition and Volunteers

LOCATION

John Ascuaga's Nugget Hotel
Reno/Tahoe Nevada USA

COLOCATED CONFERENCES

Onward!
Dynamic Language Symposium (DLS)
Pattern Languages of Programs (PLoP)
and more

CONFERENCE CHAIR

William R. Cook, UT Austin
chair@splashcon.org

OOPSLA PROGRAM CHAIR

Martin Rinard, MIT
program@splashcon.org

For information, please contact
ACM Member Services Department
1-800-342-6626 (US & Canada)
+1-212-626-0500 (global)
info@splashcon.org



SPLASH/OOPSLA is sponsored by
ACM SIGPLAN and SIGSOFT

WWW.SPLASHCON.ORG

ACM SIGPLAN/SIGSOFT

announce

OOPSLA

is now part of

SPLASH



SPLASH

RENO 2010
OCTOBER 17-21