

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

10/2009 VOL.52 NO.10

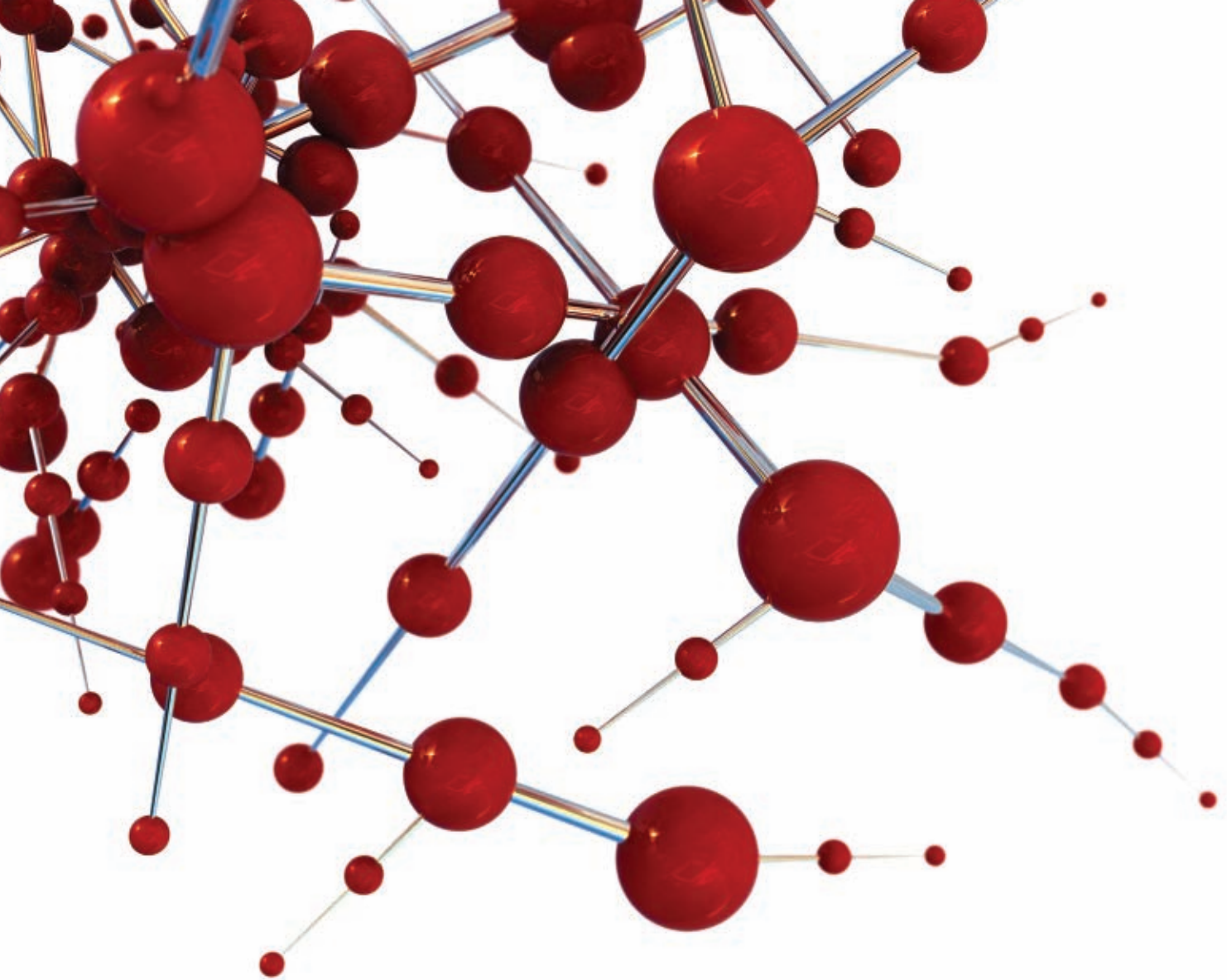
A View of Parallel Computing

A Conversation
with David E. Shaw

Smoothed Analysis

A Retrospective
from C.A.R. Hoare

Debating Net
Neutrality



**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.

Noteworthy Titles



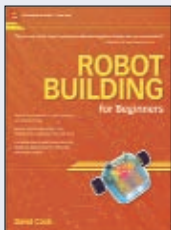
Computational Geometry Algorithms and Applications

M. d. Berg, TU Eindhoven, The Netherlands;
O. Cheong, KAIST, Daejeon, Korea;

M. v. Kreveld, M. Overmars, Utrecht University, Utrecht, The Netherlands

This well-accepted introduction to computational geometry is a textbook for high-level undergraduate and low-level graduate courses. The focus is on algorithms and hence the book is well suited for students in computer science and engineering. The book is largely self-contained and can be used for self-study by anyone with a basic background in algorithms. In this third edition, besides revisions to the second edition, new sections discussing Voronoi diagrams of line segments, farthest-point Voronoi diagrams, and realistic input models have been added.

3rd ed. 2008. XII, 386 p. 370 illus. Hardcover
ISBN 978-3-540-77973-5 ► **\$49.95**



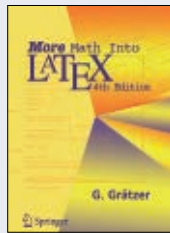
Robot Building for Beginners

D. Cook, Motorola, Whitestown, IN, USA

Learning robotics by yourself isn't easy. This book by an experienced software developer and self-taught mechanic

tells how to build robots from scratch with individual electronic components – in easy-to-understand language with step-by-step instructions.

1st ed. 2002. Corr. 2nd printing 2005. XV, 568 p. Softcover
ISBN 978-1-893115-44-6 ► **\$29.95**



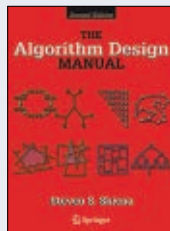
More Math Into LaTeX

G. Grätzer, University of Manitoba, Winnipeg, MB, Canada

For close to two decades, Math into LaTeX has been the standard introduction and

complete reference for writing articles and books containing mathematical formulas. In this fourth edition, the reader is provided with important updates on articles and books. An important new topic is discussed: transparencies (computer projections).

2007. XXXIV, 619 p. 44 illus. Softcover
ISBN 978-0-387-32289-6 ► **\$49.95**



The Algorithm Design Manual

S. S. Skiena, State University of New York, Stony Brook, NY, USA

This expanded and updated second edition of a classic bestseller continues to take the

“mystery” out of designing and analyzing algorithms and their efficacy and efficiency. Expanding on the highly successful formula of the first edition, the book now serves as the primary textbook of choice for any algorithm design course while maintaining its status as the premier practical reference guide to algorithms.

2nd ed. 2008. XVI, 736 p. 115 illus. With online files/update. Hardcover
ISBN 978-1-84800-069-8 ► **\$79.95**



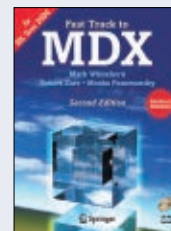
User Interface Design for Programmers

J. Spolsky, Fog Creek Software, New York, NY, USA

The author of one of the most popular indepen-

dent web sites gives you a brilliantly readable book with what programmers need to know (TM) about User Interface Design. Spolsky concentrates especially on the common mistakes that too many programs exhibit. Most programmers dislike user interface programming, but this book makes it quintessentially easy, straightforward, and fun.

2006. 160 p. Softcover
ISBN 978-1-893115-94-1 ► **\$34.99**



Fast Track to MDX

M. Whitehorn, University College Worcester, UK;
R. Zare, M. Pasumansky, Microsoft Corporation, Redmond, WA, USA

Fast Track to MDX

provides all the necessary background needed to write useful, powerful MDX expressions and introduces the most frequently used MDX functions and constructs. No prior knowledge is assumed and examples are used throughout the book to rapidly develop MDX skills to the point where they can solve real business problems. A CD-ROM containing examples from within the book, and a time-limited version of ProClarity, are included.

2nd ed. 2006. XXVI, 310 p. 199 illus. With CD-ROM. Softcover
ISBN 978-1-84628-174-7 ► **\$54.95**

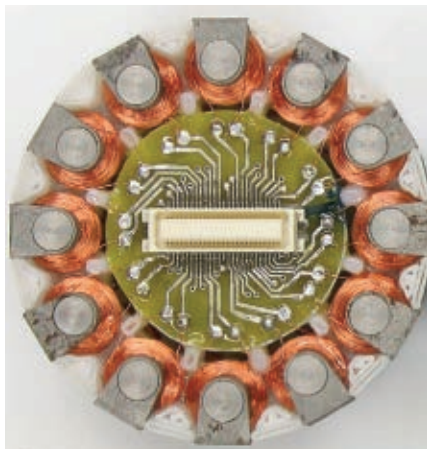
Departments

- 5 **President's Letter**
ACM Europe
By Wendy Hall
-
- 6 **Letters to the Editor**
Time and Computing
-
- 7 **In the Virtual Extension**
-
- 8 **blog@CACM**
The Netflix Prize, Computer Science Outreach, and Japanese Mobile Phones
Greg Linden writes about machine learning and the Netflix Prize, Judy Robertson offers suggestions about getting teenagers interested in computer science, and Michael Conover discusses mobile phone usage and quick response codes in Japan.
-
- 10 **CACM Online**
Following the Leaders
By David Roman
-
- 13 **Calendar**
-
- 107 **Careers**

Last Byte

- 111 **Q&A**
The Networker
Jon Kleinberg talks about algorithms, information flow, and the connections between Web search and social networks.
By Leah Hoffmann

News



- 11 **Managing Data**
Managing terabytes of data is not the monumental task it once was. The difficult part is presenting enormous amounts of information in ways that are most useful to a wide range of users.
By David Lindley
-
- 14 **Debating Net Neutrality**
Advocates seek to protect users from potential business practices, but defenders of the status quo say that concerns are overblown.
By Alan Joch
-
- 16 **Shaping the Future**
To create shape-shifting robotic ensembles, researchers need to teach micro-machines to work together.
By Tom Geller

Viewpoints

- 19 **The Business of Software**
Contagious Crazyness, Spreading Sanity
Some examples of the upward or downward spiral of behaviors in the workplace.
By Phillip G. Armour
-
- 21 **Historical Reflections**
Computing in the Depression Era
Insights from difficult times in the computer industry.
By Martin Campbell-Kelly
-
- 23 **Inside Risks**
Reflections on Conficker
An insider's view of the analysis and implications of the Conficker conundrum.
By Phillip Porras
-
- 25 **Technology Strategy and Management**
Dealing with the Venture Capital Crisis
How New Zealand and other small, isolated markets can act as "natural incubators."
By Michael Cusumano
-
- 28 **Kode Vicious**
Kode Reviews 101
A review of code review do's and don'ts.
By George V. Neville-Neil
-
- 30 **Viewpoint**
Retrospective: An Axiomatic Basis for Computer Programming
C.A.R. Hoare revisits his past *Communications* article on the axiomatic approach to programming and uses it as a touchstone for the future.
By C.A.R. Hoare

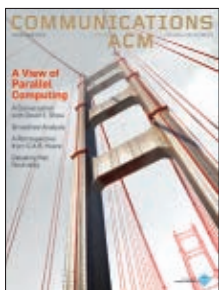


Practice

- 34 **Probing Biomolecular Machines with Graphics Processors**
GPU acceleration and other computer performance increases will offer critical benefits to biomedical science.
By James C. Phillips and John E. Stone
-
- 42 **Unifying Biological Image Formats with HDF5**
The biosciences need an image format capable of high performance and long-term maintenance. Is HDF5 the answer?
By Matthew T. Dougherty, Michael J. Folk, Erez Zadok, Herbert J. Bernstein, Frances C. Bernstein, Kevin W. Eliceiri, Werner Benger, Christoph Best
-
- 48 **A Conversation with David E. Shaw**
Stanford professor Pat Hanrahan sits down with the noted hedge fund founder, computational biochemist, and (above all) computer scientist.

Review Articles

- 76 **Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice**
This Gödel Prize-winning work traces the steps toward modeling real data.
By Daniel A. Spielman and Shang-Hua Teng



About the Cover: Leonello Calvetti brings to life the bridge analogy connecting users to a parallel IT industry as noted by the authors of the cover story beginning on page 56. Calvetti, a photo-realistic illustrator, is a graduate of Benvenuto Cellini College in Florence, Italy, where he studied technical design and developed his artistic applications of 3D software and digital imagery.

applications of 3D software and digital imagery.

Contributed Articles

- 56 **A View of the Parallel Computing Landscape**
Writing programs that scale with increasing numbers of cores should be as easy as writing programs for sequential computers.
By Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John Kubiatawicz, Nelson Morgan, David Patterson, Koushik Sen, John Wawrzynek, David Wessel, and Katherine Yelick
-
- 68 **Automated Support for Managing Feature Requests in Open Forums**
The result is stable, focused, dynamic discussion threads that avoid redundant ideas and engage thousands of stakeholders.
By Jane Cleland-Huang, Horatiu Dumitru, Chuan Duan, and Carlos Castro-Herrera

Research Highlights

- 86 **Technical Perspective**
Relational Query Optimization—Data Management Meets Statistical Estimation
By Surajit Chaudhuri
-
- 87 **Distinct-Value Synopses for Multiset Operations**
By Kevin Beyer, Rainer Gemulla, Peter J. Haas, Berthold Reinwald, and Yannis Sismanis
-
- 96 **Technical Perspective**
Data Stream Processing—When You Only Get One Look
By Johannes Gehrke
-
- 97 **Finding the Frequent Items in Streams of Data**
By Graham Cormode and Marios Hadjieleftheriou

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.

Balancing Four Factors in System Development Projects
Girish H. Subramanian, Gary Klein, James J. Jiang, and Chien-Lung Chan

Attaining Superior Complaint Resolution
Sridhar R. Papagari Sangareddy, Sanjeev Jha, Chen Ye, and Kevin C. Desouza

Making Ubiquitous Computing Available
Vivienne Waller and Robert B. Johnson

De-escalating IT Projects: The DMM Project
Donal Flynn, Gary Pan, Mark Keil, and Magnus Mahrng

Human Interaction for High-Quality Machine Translation
Francisco Casacuberta, Jorge Civera, Elsa Cubel, Antonio L. Lagardia, Guy Lampalme, Elliott Macklovitch, and Enrique Vidal

How Effective is Google's Translation Service in Search?
Jacques Savoy and Ljiljana Dolamic

Overcoming the J-Shaped Distribution of Product Reviews
Nan Hu, Paul A. Pavlou, and Jie Zhang

Technical Opinion
Do SAP Successes Outperform Themselves and Their Competitors?
Richard J. Goeke and Robert H. Faley



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

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
Gul Agha; Michel Beaudouin-Lafon;
Jack Davidson; Nikil Dutt; Carol Hutchins;
Ee-Peng Lim; 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

GROUP PUBLISHER
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; Richard Ladner;
Susan Landau; Carlos Jose Pereira de Lucena;
Helen Nissenbaum; Beng Chin Ooi;
Loren Terveen

PRACTICE
Chair
Stephen Bourne

Board Members
Eric Allman; Charles Beeler;
David J. Brown; Bryan Cantrill;
Terry Coatta; Mark Compton;
Benjamin Fried; Pat Hanrahan;
Marshall Kirk McKusick;
George Neville-Neil
The Practice section of the CACM
Editorial Board also serves as
the Editorial Board of *COMMUNIQUE*.

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; Linda Petzold;
Michael Reiter; Mendel Rosenblum;
Ronitt Rubinfeld; David Salesin;
Lawrence K. Saul; Guy Steele, Jr.;
Gerhard Weikum; Alexander L. Wolf

WEB
Co-chairs
Marti Hearst and James Landay
Board Members
Jason I. Hong; Jeff Johnson;
Greg Linden; Wendy E. MacKay



ACM Copyright Notice
Copyright © 2009 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.

ACM Europe

Increasing ACM's relevance and influence as a membership organization in the global computing community has been a top priority from the outset of my presidency.

For it is only by making concerted efforts to share ACM's vast array of valued resources and services on a global scale, and by discovering the work and welcoming the talent from all corners of the computing arena, that ACM can truly call itself the world's leading computing society.

To this end, ACM has made impressive strides over the last year. Major internationalization initiatives have resulted in multiyear plans for raising the Association's visibility and promoting membership in the major technology hubs of Europe, India, and China. A growing number of ACM's Special Interest Groups (SIGs) are hosting or planning their flagship conferences throughout Asia, Europe, and Latin America. And ACM's Executive Committee, Council, as well as its many Boards and Committees stand solidly behind this cause, fully recognizing what it means—and what it takes—for ACM to be an international society.

As the Association's first non-North American President, and a European, I am especially pleased to announce ACM Europe—a new effort within ACM to recognize and support European members and ACM activities in Europe. Dedicated ACM volunteers and executive staff have been working with industry and academic leaders in Europe to determine ways to better serve ACM's current European members and to draw more into the fold. By strengthening its ties in the region, ACM will be better positioned to appreciate the key issues and challenges within Europe's academic, research, and professional computing communities and to respond effectively.

To celebrate the introduction of ACM Europe, the Association will host a special event in Paris on October 8 in conjunction with the 2009 European Computer Science Summit, sponsored

by Informatics Europe (<http://www.informatics-europe.org/ECSS09/>). The reception will honor the achievements of European computer scientists and recognize European winners of ACM's A.M. Turing Award, as well as other ACM award winners, and ACM Fellows. The event will also serve to share the work and activities planned by ACM Europe.

ACM Europe Council


ACM Europe is spearheaded by the ACM Europe Council—a group of 15 noted European computer scientists from academia and industry who came together last October pledging to help build an ACM presence that would focus on bringing high-quality technical activities, conferences, and services to ACM members and computing professionals throughout the continent. Chairing the ACM Europe Council is Fabrizio Gagliardi, director of external research programs for Microsoft Research Europe. (Council members are listed here.)

The roots of ACM's European community are as old and run as deep as the Association itself, having been in existence over 50 years. Today's ACM Euro-

pean contingent includes over 15,000 members, more than 100 chapters, and an ever-growing number of ACM-sponsored conferences and symposia. The primary goal of ACM Europe is to serve ACM's European members by improving the exchange of ideas, and by raising awareness about ACM with the public and with European decision makers in order to play an active role in critical technical, educational, and social issues related to computing.

The ACM Europe Council has established three subcommittees to launch its initial work in serving its constituency. One subcommittee will review the current landscape of European chapters and help develop strategies for increasing the number of chapters, particularly student chapters. A second group will focus on generating interest in, and nominations for, the many ACM professional awards and distinguished member grades. This group will also work to suggest names for key ACM volunteer positions to add a stronger European presence and perspective to the Association's future plans. The third group will work toward engaging ACM's SIGs to "think Europe" when planning conferences and major events.

ACM Europe is one of many international initiatives currently at the top of ACM's agenda. Planning is already under way for the newly formed ACM India Council to host a launch event in early 2010 and we will be announcing details of our plans for the development of ACM China next year. ACM SIGs continue to hold record numbers of conferences outside North America.

When I agreed to accept the invitation to run for president of ACM in early 2008, I noted in my election statement that I truly believed ACM's future success depends on its ability to increase its diversity and support of academic and research communities around the world. Given the progress made so far through the efforts and ongoing interest of so many devoted volunteers and ACM leaders to this cause, I am heartened to say that we are well on the way to successfully achieving our aims. 

Professor Dame Wendy Hall (wh@ecs.soton.ac.uk) is president of ACM and a professor of computer science at the University of Southampton, U.K.

© 2009 ACM 0001-0782/09/1000 \$10.00

ACM Europe Council

CHAIR:

Fabrizio Gagliardi, Switzerland

MEMBERS:

Andrew McGettrick, Scotland
 Marc Shapiro, France
 Thomas Hofmann, Switzerland
 Gerhard Schimpf, Germany
 Alexander Wolf, U.K.
 Gabriele Kotsis, Austria
 Jan van Leeuwen, The Netherlands
 Avi Mendelson, Israel
 Michel Beaudouin-Lafon, France
 Burkhard Neidecker-Lutz, Germany
 Bertrand Meyer, Switzerland
 Paul Spirakis, Greece
 Wendy Hall, U.K. (and ACM President)
 Mateo Valero, Spain

Time and Computing

EDWARD E. LEE'S "Computing Needs Time" (May 2009) might be the most important, stimulating, and timely article I have read in *Communications* over the past 50 years. It dealt with a major technical topic long neglected by ACM but that is today more important than ever. One sentence in particular resonated: "It is easy to envision new capabilities that are technically within striking distance but that would be extremely difficult to deploy with today's methods." Having been involved in the development of embedded systems in military aircraft from 1968 to 1994, I would say that to succeed, computerized performance systems need the following:

Knowledge. Of the operational objectives and requirements of the system;

Discipline. At all technical and management levels to assure system resources are well defined and carefully budgeted, with timing at the top of the list; and great care

Selection. Of exceptionally self-directed project personnel, including pruning out those unsuited for such system-development work.

You might view them as platitudes, so consider, too, several examples from four programs involving military aircraft:

In both the P-3C maritime patrol aircraft and the carrier-based S-3A antisubmarine aircraft, senior software engineers (in both the contracting agency and the contractor) developed a set of real-time software "requirements" that could not be met with available memory. In each case Navy program managers were steeped in the operational requirements and had exercised their authority to severely limit them.

In the case of the F-117 stealth fighter, an important new operational subsystem was never deployed. The customer and the Lockheed Advanced Development Company jointly invented a set of utopian system "requirements," basically ignoring management directions. The customer's

program manager and Lockheed finally agreed to cancel the project.

It was no surprise 20 or 30 years ago that technical people involved in the development of demanding embedded systems learned primarily on the job, with more trial and error than we care to admit but that we accepted. Are universities today educating the new generation of systems and software engineers so they are able to develop computerized performance systems? In my own recent cynical moments rebooting my Mac, I think such software development is being done by people who would, perhaps, be downright dangerous if assigned to develop critical computerized performance systems.

In any case, Lee's technical vision and thinking were truly impressive, as was his skill in communicating them.

Sherm Mullin, Oxnard, CA

How the Blind Learn Math and Design

Kristen Shinohara and Josh Tenenbergh were absolutely correct in "A Blind Person's Interactions with Technology" (Aug. 2009) saying that replacing text with voice does not automatically provide the blind interaction with or even access to technology. However, they might have added that this says more about how the sighted value text and its characteristics (such as left to right, fixed character sets, and lamina) than it does about the absence of that value in the blind.

In trying to teach computer science to two completely blind students, I have found that two key issues—mathematics and design—are both so dependent on spatial arrangement that the standard approaches are at best unreliable and at worst confusing. Even a cursory glance at the excellent Blindmath notice board (http://www.nfbnet.org/mailman/listinfo/blindmath_nfbnet.org) demonstrates the importance of Abraham Nemeth's Braille coding for math-symbol layout to both experienced practitioners

and as a tool for the young learning skills in the field. However, symbolic and spatial presentation (for the most part the essence of the mathematical teaching method) has little to do with the ability to achieve proficiency in mathematics or computing. Simply linearizing the symbols and spatial arrangement for access by the blind through voice (or Braille) techniques could still miss the point but seems to be the best we can do today.

Similarly, teaching design as hierarchical decomposition and spatial layout redolent in techniques (such as UML) also misses the point; it might thus be replaced by the simple text "generate a hierarchical decomposition and group the resulting units" with examples of what is meant and any of the tricks that might be available. Noting that time arrangement could be replaced with spatial-arrangement restrictions compounds the challenge. (A bundle of tactile knotted strings might be a better exemplar of hierarchy than a neat 2.5D graph, as I believe Donald Knuth once said.)

It may be that seeing the world as the sighted see it is the goal of practitioners and educators alike; the motes, as it were, could indeed be in the wrong eyes.

Bernard M. Diaz, Liverpool, U.K.

Correction

Brad Chen of Google contributed to the development of the news story "Toward Native Web Execution" (July 2009). ■

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.

In the Virtual Extension

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

Balancing Four Factors in System Development Projects

Girish H. Subramanian, Gary Klein, James J. Jiang, and Chien-Lung Chan

It is often the methodology that dictates system development success criteria. However, an organization will benefit most from a perspective that looks at both the quality of the product and an efficient process. This imperative will more likely incorporate the multiple goals of various stakeholders. To this end, a project view of system development should adjust to incorporate controls into a process that stresses flexibility, promote learning in a process that is more rigid, and be certain to use evaluation criteria that stress the quality of the product as well as the efficiency of the process.

Attaining Superior Complaint Resolution

Sridhar R. Papagari Sangareddy, Sanjeev Jha, Chen Ye, and Kevin C. Desouza

Why is customer service more important than ever for consumer technology companies? How can they retain existing customers and prevent them from discontinuing their products? What can they do to provide superior complaint management and resolution? The authors answer these questions and more by proposing and evaluating a model of complaint management and service evaluation. A holistic approach toward complaint management is recommended for retaining customers.

Making Ubiquitous Computing Available

Vivienne Waller and Robert B. Johnson

Computing artifacts that seamlessly support everyday activities must be made both physically and cognitively 'available' to users. Artifacts that are designed using a traditional model of computing tend to get in the way of what we want to do. Drawing on Heidegger, the authors delve deeper into the concept of 'availability' than existing studies in human-computer interaction have done. They find two ways that ubiquitous computing can be truly available are through manipulating the space of possible actions and through

indicating the possibility for action. This article translates the conceptual findings into principles for design.

De-escalating IT Projects: The DMM Project

Donal Flynn, Gary Pan, Mark Keil, and Magnus Mahrng

Taming runaway information technology projects is a continuing challenge for many managers. These are projects that grossly exceed their planned budgets and schedules, often by a factor of 2–3 fold or greater. Many end in failure, not only in the sense of budget or schedule, but in terms of delivered functionality. This article examines three approaches that have been suggested for managing de-escalation. By combining the best elements from the approaches, we provide an integrated framework as well as introducing a de-escalation management maturity (DMM) model that provides a useful roadmap for improving practice.

Human Interaction for High-Quality Machine Translation

Francisco Casacuberta, Jorge Civera, Elsa Cubel, Antonio L. Lagardia, Guy Lampalme, Elliott Macklovitch, and Enrique Vidal

The interactive-predictive approach allows for the construction of machine translation systems that produce high-quality results in a cost-effective manner by placing a human operator at the center of the production process. The human serves as the guarantor of high quality and the role of the automated systems is to ensure increased productivity by proposing well-formed extensions to the current target text, which the operator may then accept, correct, or ignore. Interactivity allows the system to take advantage of the human-validated portion of the text to improve the accuracy of subsequent predictions.

How Effective is Google's Translation Service in Search?

Jacques Savoy and Ljiljana Dolamic

Using freely available translation services, bilingual search is possible

and even effective. Compared to a monolingual search, the automatic query translation hurts the retrieval effectiveness (from 12% to 30% depending on the language pairs). Various translation difficulties as well as linguistic features may explain such degradation. Instead of providing a direct translation for all language pairs, we can select an intermediary language or pivot (for example, English) and such strategy does not always further degrade the search quality.

Overcoming the J-Shaped Distribution of Product Reviews

Nan Hu, Paul A. Pavlou, and Jie Zhang

Product review systems rely on a simple database technology that allows people to rate products. Following the view of information systems as socio-technical systems, product review systems denote the interaction between people and technology. This article provides evidence to support the finding that online product reviews suffer from two kinds of potential biases: purchasing bias and under-reporting bias. Therefore the average of ratings alone may not be representative of product quality, and consumers need to look at the entire distribution of the reviews.

Technical Opinion: Do SAP Successes Outperform Themselves and Their Competitors?

Richard J. Goeke and Robert H. Faley

Managers and researchers have long debated how to measure the business value of IT investments. ERP systems have recently entered this debate, as research measuring the business value of these large IT investments has produced mixed results. Using regression discontinuity analysis, the authors found that successful SAP implementers improved inventory turnover in their post-implementation periods. These SAP successes also significantly improved inventory turnover relative to their competitors. However, profitability improvements were more difficult to find.

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.

DOI:10.1145/1562764.1562769

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

The Netflix Prize, Computer Science Outreach, and Japanese Mobile Phones

Greg Linden writes about machine learning and the Netflix Prize, Judy Robertson offers suggestions about getting teenagers interested in computer science, and Michael Conover discusses mobile phone usage and quick response codes in Japan.



From Greg Linden's "The Biggest Gains Come From Knowing Your Data"

Machine learning is hard. It can be awfully tempting to try to skip the work. Can't we just download a machine learning package? Do we really need to understand what we are doing?

It is true that off-the-shelf algorithms are a fast way to get going and experiment. Just plug in your data and go.

The only issue is if development stops there. By understanding the peculiarities of your data and what people want and need on your site, by experimenting and learning, it is likely you can outperform a generic system.

A great example of how understanding the peculiarities of your data can help came out of the Netflix Prize. Progress on the \$1 million prize largely

stalled until Gavin Potter discovered peculiarities in the data, including that people interpret the rating scale differently.

More recently, Yehuda Koren found additional gains by supplementing the models to allow for temporal effects, such as that people tend to rate older movies higher, that movies rated together in a short time window tend to be more related, and that people over time might start rating all the movies they see higher or lower.

In both cases, looking closely at the data, better understanding how people behave, and then adapting the models yielded substantial gains. Combined with other work, that was enough to win the million-dollar prize.

The Netflix Prize followed a pattern you often see when people try to implement a feature that requires machine learning. Most of the early attempts

threw off-the-shelf algorithms at the data, yielding something that works, but not with particularly impressive results.

Without a clear metric for success and a way to test against that metric, development stops there. But, like Google and Amazon do with ubiquitous A/B testing, the Netflix Prize had a clear metric for success and a way to test against that metric.

There are a lot of lessons that can be taken from the Netflix contest, but a big one should be the importance of constant experimentation and learning. By competing algorithms against each other, by looking carefully at the data, by thinking about what people want and why they do what they do, and by continuous testing and experimentation, you can reap big gains.



From Judy Robertson's "Computer Science Outreach: Meeting the Kids Halfway"

I just spent the afternoon working with teenagers at some of our summer school workshops. As luck would have it, we had two different sessions running on the same afternoon, and while galloping between labs, it occurred to me some interesting things were going on. First, a bit about the workshops; the summer schools were both for 17- and 18-year-olds, both were set up to encourage young people to study computer science, and both involved building virtual worlds. One of the workshops, on making computer games using the *Neverwinter Nights 2*

toolset, lasted for just two hours, and the other was the final presentation session of an eight-week project on *Second Life* programming. Both of them went very well from the point of view of introducing young people to the fun aspects of computer science. Whether they pay off in terms of recruiting people to study our degree courses in CS remains to be seen. But you have to start somewhere, right? Here are some things I noticed that might be useful to others who are interested in schools' outreach and recruitment.

► A relaxed atmosphere prevailed. The young people were joking around and enjoying themselves. Importantly, they were laughing *with* the staff rather than at them. Having some handpicked students who I knew to be friendly and approachable really helped with this.

► The young people were doing stuff rather than listening to me drone on. The games workshop kids spent most of their time exploring the software with minimal time spent in demos. The *Second Life* project groups were presenting their projects and giving demos while their classmates assessed their work. They seemed to be taking the assessment task seriously and responsibly. And I'll tell you what: It really makes them ask sensible questions at the end of each presentation. This is a contrast to the usual setup in class where students sit like turnips when you ask, "Are there any questions?"

► Both the workshops involved creative tasks where the teenagers chose for themselves what to build. This does have the drawback of revealing my ignorance of the latest pop culture fads, but at least I do know what *South Park* is. Seriously, though, this is very important. If you want people to take pride in their work, they need to take some ownership of it. For that to happen, they need to have the choice to work on personally meaningful projects and this often means embracing popular culture in a way which we, as grown-up computer scientists, might find baffling or intensely irritating.

Rather than pushing our agenda of what we think is important and berating young people that they ought to find it interesting, we need to meet them halfway. We need to start from their interests, and then help them to see how computer science knowledge

can help them achieve something that appeals to them. As in "You're interested in alcohol and *The Simpsons*. Ideal. How about you make a 3D Homer Simpson whose arm can move up and down to drink beer?" At that point you can start explaining the necessary programming and math concepts to do the rotation in 3D space. Or even just admire what they have figured out by themselves. Once you have them hooked on programming or signed up on your degree program, you can build on it. I'm not saying we don't need to teach sober, serious, and worthy aspects of computer science. Of course we do. I'm just saying we don't need to push it immediately. It's kind of like when you have a new boyfriend and you know you have to introduce him to your weird family. Do you take him to meet the mad uncle with the scary eyebrows straight off? No, you introduce him to a friendly cousin who will make him feel at home and has something in common with him.

What I'm suggesting is not new—there are pockets of excellent outreach work with kids in various parts of the world. I think it's time we tried more of it, even although it is time consuming. After all, we know we can recruit hardcore computer scientists to our degree programs with our current tactics (you know, the people who are born with silver Linux kernels in their mouths). But given there aren't that many of them, it's well worth the effort to reach out to the normal population. Unleash the inner computer scientist in everyone!



From Michael Conover's "Advertainment"

Mobile phones are a way of life in Japan, and this aspect of the culture manifests itself in many ways. Among the more remarkable are the ubiquitous quick response (QR) codes that adorn a sizable percentage of billboards, magazines, and other printed media. In brief, these two-dimensional bar codes offer camera phones with the appropriate software an opportunity to connect with Web-based resources relating to the product or service featured in an advertisement. Encoding a maximum of 4,296 alphanumeric characters, or

1,817 kanji, QR codes are a forerunner of ubiquitous computing technology and portend great things to come.

What's remarkable to me is, for all our similarities, how widely divergent American and Japanese urban cultures can be. The market penetration numbers aren't that strikingly different; a March 2008 study showed that more than 84% of Americans own a cell phone, where a Wolfram Alpha query shows that 83% of Japanese own one. The differences in practice, however, could not be more pronounced. In terms of mobile phone use, walking the streets of Japan is like being on a college campus all the time. It's not unreasonable to estimate that every fifth person is interacting in some way with a mobile device, and here's the rub on this point—Americans make calls on their phones, the Japanese interact.

Ubiquitous Web access and widespread support for the mobile platform, in addition to the vastly increased data-transfer capabilities, mean Japan is a society in which cell phones are a practical mobile computing platform. QR codes have blossomed in this culture not only because they're immensely useful to both organizations and consumers, but because the cultural soil is ripe for their adoption. QR codes have been met with lukewarm response in the U.S., and I fear it may be yet another mobile technology to which we get hip three to five years behind the curve.

Irrespective of this, the applications of QR codes in Japan are at times astounding. For many high-dollar corporations, such as Louis Vuitton and Coca-Cola, the QR code is the ad (art?) itself. Oftentimes, the QR code is the actual content, made of something unexpected or even a medium for digital activism. Because of its robust digital format, creative marketers have a lot of wiggle room when it comes to creating eye-catching, market-driven applications of this technology and, like ubiquitous translation technology, it's the widespread use of Internet-enabled phones that underlies this technological paradigm shift. ■

Greg Linden is the founder of Geeky Ventures. Michael Conover, a Ph.D. candidate at Indiana University, is a visiting researcher at IBM Tokyo Research Laboratory. Judy Robertson is a lecturer at Heriot-Watt University.



DOI:10.1145/1562764.1562770

David Roman

Following the Leaders

The articles, sections, and services available on *Communications'* Web site all vie for visitor attention. According to our latest Web statistics, the following features are the most popular in pageviews since the site's formal launch in April 2009.

Top Opinion Content

FYI: All but #2 are from the Viewpoints section of the print issue.

- | | |
|--|--|
| 1. Research Evaluation for Computer Science
cacm.acm.org/magazines/2009/4/22954 | 6. Why 'Open Source' Misses the Point of Free Software
cacm.acm.org/magazines/2009/6/28491 |
| 2. Conferences vs. Journals in Computing Research
cacm.acm.org/magazines/2009/5/24632 | 7. Advising Students for Success
cacm.acm.org/magazines/2009/3/21781 |
| 3. CS Education in the U.S.: Heading in the Wrong Direction?
cacm.acm.org/magazines/2009/7/32090 | 8. Teaching Computing to Everyone
cacm.acm.org/magazines/2009/5/24643 |
| 4. Time for Computer Science to Grow Up
cacm.acm.org/magazines/2009/8/34492 | 9. Beyond Computational Thinking
cacm.acm.org/magazines/2009/6/28490 |
| 5. Computing as Social Science
cacm.acm.org/magazines/2009/4/22953 | 10. An Interview With C.A.R. Hoare
cacm.acm.org/magazines/2009/3/21782 |

Top Practice Articles

FYI: High readership is common for material from the print edition's Practice pages.

- | |
|---|
| 1. The Five-Minute Rule 20 Years Later
cacm.acm.org/magazines/2009/7/32091 |
| 2. API Design Matters
cacm.acm.org/magazines/2009/5/24646 |
| 3. Whither Sockets?
cacm.acm.org/magazines/2009/6/28495 |
| 4. Security in the Browser
cacm.acm.org/magazines/2009/5/24645 |
| 5. A Direct Path to Dependable Software
cacm.acm.org/magazines/2009/4/22960 |

Most Popular Browse by Subject Categories

FYI: The AI page gets more pageviews than the BBS landing page.

- | |
|---|
| 1. Artificial Intelligence
cacm.acm.org/browse-by-subject/artificial-intelligence |
| 2. Communications/Networking
cacm.acm.org/browse-by-subject/communications-networking |
| 3. Software
cacm.acm.org/browse-by-subject/software |
| 4. Security
cacm.acm.org/browse-by-subject/security |
| 5. Human-Computer Interaction
cacm.acm.org/browse-by-subject/human-computer-interaction |



Top Services

FYI: These pages deliver a service, not just information.

- | |
|-------------------------|
| 1. Sign In |
| 2. Forgot Your Password |
| 3. Create a Web Account |
| 4. RSS Feeds |
| 5. Alerts and Feeds |

ACM Member News



Susan T. Dumais, a principal researcher in the adaptive systems and interaction group at Microsoft

Research, is the recipient of the 2009 Gerard Salton Award, presented by the Special Interest Group on Information Retrieval (SIGIR). SIGIR recognized Dumais for her “innovative contributions to information indexing and retrieval systems that have widely impacted the quality of search from the desktop to the Web.”

“Obviously I was tremendously honored to be considered in the same category as the previous eight recipients who have shaped the field of information retrieval for almost 50 years,” Dumais said in an email interview. “This was quickly followed by the realization that I had worked on search-related problems for almost 30 years. ☺”

Asked about her role as a mentor, Dumais said: “Throughout my career, I have had pleasure of working with amazing mentors and colleagues (notably Drake Bradley, Richard Shiffirin, Thomas Landauer, and Eric Horvitz) who have challenged me to think critically about research problems. I enjoy collaborating with people from different disciplines and perspectives since it continues to broaden the perspective that I have on problems. By mentoring interns, serving on Ph.D. committees, taking part in doctoral consortia and, more generally, by reviewing and editorial activities, I feel that I can give back to the community and encourage the next generation of researchers to tackle key problems using a wide range of theoretical and experimental methods. In doing research, I believe that it is important to understand past contributions and perspectives, and to pursue new solutions to key problems. In mentoring younger researchers, I encourage them to identify key challenges, to pursue them using a broad range of approaches, and not to give up when the first idea or implementation doesn't work.”

Managing Data

Dealing with terabytes of data is not the monumental task it once was. The difficult part is presenting enormous amounts of information in ways that are most useful to a wide variety of users.

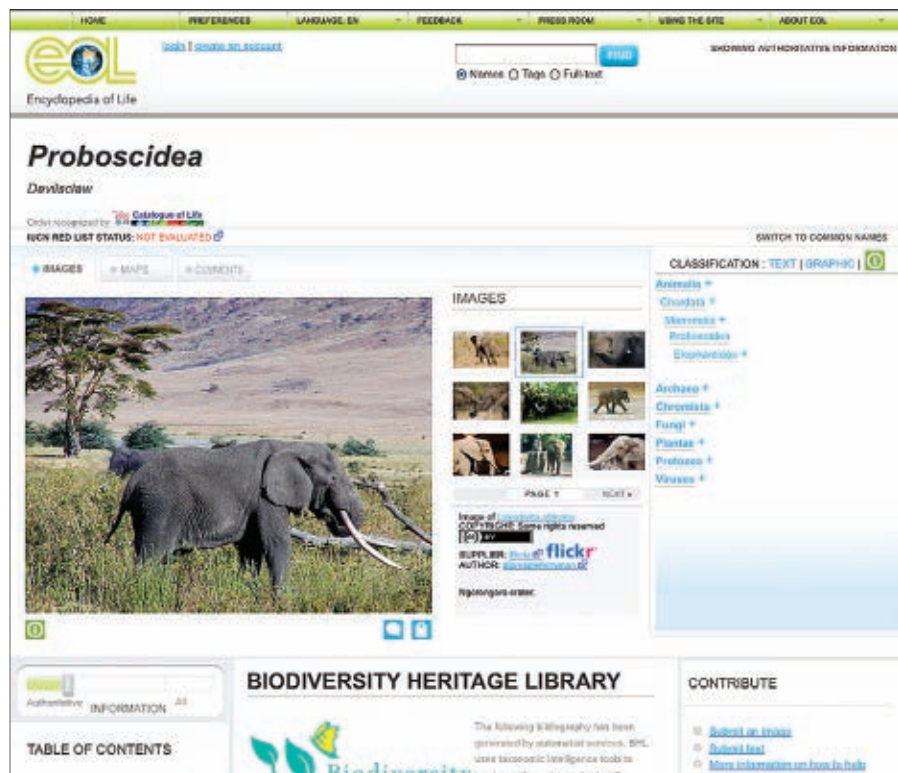
THE HERBARIUM AT the University of Alaska's Museum of the North houses one of the world's largest collections of arctic plant specimens and Russian flora. For scientists studying the ecology of this region or the changing biodiversity caused by human encroachment and climate change, it's an invaluable resource. However, the Herbarium is not ideally located for most scientists. Because of the considerable expense of traveling to Alaska, scientists often have specimens temporarily shipped to their institutions, but that also costs money, and delicate specimens suffer from the attendant wear and tear. As a result, the Herbarium is scanning and digitizing its extensive collection, making the images and text available on the Internet to scientists, not to mention enthusiastic amateurs, everywhere in the world.

The amount of data involved—about five terabytes so far—is hardly intimidating by today's standards, but it proved to be overwhelming for the Herbarium's previous database partner. Consequently, the Herbarium teamed up with the University of Texas at Austin's Texas Advanced Computing Center (TACC), gaining access to TACC's Corral, an online 1.2 petabyte

data repository, via the National Science Foundation's TeraGrid cyber-infrastructure. "We take images and they're immediately downloaded to Texas, and in live time we have a link to that file from our database," says Steffi

Ickert-Bond, curator of the Herbarium and an assistant professor of botany at the University of Alaska. The Herbarium's holdings are now accessible via Arctos, an online collaboration of five U.S. university museums.

Nearly 80,000 of the Herbarium's 220,000 specimens have been digitized by hand, with students laboriously keying in detailed information from the specimens' labels, some of which are handwritten and some of which are in Russian. To record the remaining 140,000 specimens, however, the proj-



The ambitious Encyclopedia of Life aims to create a Web page for every known species on Earth.

ect is shifting to automated label scanning using Google's Tesseract optical character recognition engine.

Handling varied data, such as text, numbers, images, and sounds, no longer poses any great technical difficulties, says Chris Jordan, who is in charge of data infrastructure at TACC. Corral uses Sun Microsystems' Lustre file-handling program to present users with a seamless interface to the data. What's trickier is building a presentation layer that gives scientists in a particular discipline access to the resources in an intuitive and user-friendly way. Unlike researchers in physics and engineering, for example, those working in museums or the humanities aren't accustomed to using computers at the command line level, says Jordan, so "a lot of our effort now is on [building] interfaces for people to locate data along with descriptive metadata in a way that's reasonably easy."

Accessible and Useful

Making vast repositories of biological information widely accessible and useful is the aim of the Encyclopedia of Life (EOL), an ambitious project whose long-term goal is to create a Web page for every known species on Earth. "We want to provide all information about all organisms to all audiences," says Cyndy Parr, EOL's species pages group leader, who is based at the Smithsonian's National Museum of Natural History (NMNH) in Washington, D.C. So far, EOL has about 1.4 million species pages, but most of them are little more than placeholders for known organisms, frequently directing visitors to an external link with more information. Some 158,000 pages, however, contain at least one data object that's been scientifically vetted by EOL. The project is chasing a moving target, since the number of species on the planet is unknown. A figure of 1.8 million known species is commonly accepted, says Parr, but new species are being discovered at a rate of 20,000 or more each year, and some extrapolations suggest that there may be as many as 10 million distinct species.

Far from competing with efforts such as those by the University of Alaska's Herbarium, EOL aims to build species pages that deliver essential information in a uniform style and lead

visitors who want to dig deeper to more specialized, detailed resources. Indeed, says EOL Executive Director Jim Edwards, also at NMNH, a principal motive behind EOL was the realization that many research communities are building their own databases—such as AntWeb, FishBase, and others—each with its own design, interface, and search procedures, created to meet the needs of its particular community.

Launched with grants from the MacArthur and Sloan foundations, EOL receives support from several museums and other institutions. EOL invites scientific contributions from amateurs and academics, but uses a network of researchers to decide what information will be included. The site tries to steer a middle course between a pure top-down model, with pages created only by experts, and a self-policed wiki.

EOL resides on servers at the Smithsonian and the Marine Biological Laboratory in Woods Hole, MA, but since it is mainly an index to other information, the data cached on those servers amounts to a few hundred megabytes. EOL's informatics challenges derive in large part from the historical origins of biological information. Even the formal names of species can be treacherous, since some species have been

"discovered" on more than one occasion and received duplicate names, and sometimes molecular or DNA analysis demonstrates that what's long been regarded as a single species is, in fact, two distinct species.

In addition, it's essential to be able to search in less formal ways, such as by habitat type, mating behavior, or leaf shape, characteristics that aren't often described by a standardized set of terms. That's a particular issue for one of EOL's partner efforts, the Biodiversity Heritage Library (BHL), a collaboration of 10 natural history museum libraries, botanical libraries, and research institutions in the U.S. and the U.K. that has put nearly 15 million digitized pages from 37,000 books online. Although optical character recognition software has become reliable, scanning millions of pages is labor-intensive and time consuming, says Chris Freeland, BHL's technical director and manager of the bioinformatics department at the Missouri Botanical Garden in St. Louis. Someone—usually a volunteer student—must turn the pages of every book and correctly position them for the camera. After that phase, though, the digitizing process is automated and efficient. Typewritten or commercially printed material is optically well

Galaxy Zoo uses crowdsourcing to help categorize galaxies as either spiral or elliptical.

recognized, although unusual scripts, such as older typefaces and cursive characters, can be problematic.

To date, BHL's digitized collection totals 50 terabytes. That's not so large, Freeland notes, but replicating it at mirror Web sites and ensuring its reliability and security is a challenge. Once again, though, the larger headache is presenting the data. Searching text for keywords and phrases is easy, but because a great deal of biological information is qualitative and descriptive, it's very difficult to construct a search that will dig out all references to, say, the mating behavior of bearded dragon lizards. As a result, EOL is working with computer scientists on natural language processing software that can intelligently characterize the meaning of digitized texts. But accomplishing that goal automatically and reliably remains elusive, Edwards says.


Crowdsourcing Galaxies

EOL is also investigating crowdsourcing methods in which a Web site visitor is asked to supply keywords for an image or text extract. One example of crowdsourcing paying scientific dividends is the Galaxy Zoo, an offshoot of the Sloan Digital Sky Survey, which uses an automated telescope to scan the sky for galaxies and digitize images of them. (For more about the Sloan Digital Sky Survey, see "Jim Gray, Astronomer" in the November 2008 issue of *Communications*.) The Sloan survey examines so much space that astronomers cannot hope to look at every galaxy image, yet direct visual inspection is how astronomers have traditionally gained their understanding of galaxies, says Bob Nichol, a professor of astrophysics at the University of Portsmouth in England. Based at the University of Oxford, the Galaxy Zoo gets some of that irreplaceable scientific insight from Web site visitors, who are asked to classify a series of galaxy images drawn randomly from the Sloan survey as spiral or elliptical. The task can be addictive, says Nichol, and 250,000 visitors have collectively classified nearly one million galaxies at least 30 times each. One statistically solid result to emerge is that 15% of galaxies don't obey the usual rule that the color of elliptical galaxies tends toward the red end of the spectrum and spiral

The Encyclopedia of Life is working with computer scientists on natural language processing software that can intelligently characterize digitized texts.

galaxies are bluer. Moreover, red spiral galaxies tend to inhabit the outskirts of galaxy clusters, a finding that allows cosmologists to test theories about the galaxies' origins.

One crowdsourcing lesson Nichol has drawn from the success of the Galaxy Zoo is that it is crucial to precisely tell visitors what information is needed and why it is important. Some Galaxy Zoo scientists have been discussing with other researchers, including botanists, how to translate their methods to other disciplines, but for a project as large as EOL, the task is daunting. The required information is not easily reduced to a handful of simple questions, and while pages for popular mammals and pretty flowers receive many hits, pages for undistinguished flies and obscure bacteria languish unseen in the cyberspace equivalent of a dusty museum attic. Crowdsourcing doesn't work without crowds.

Although EOL has to date attained only a tiny fraction of its ultimate goals, it has earned respect and enthusiasm from those in the biological community who were initially skeptical about its utility, Edwards says. And as EOL continues to grow, he thinks its value will be understood by many different communities, including the public and commercial businesses, so that it will become an indispensable resource. 

David Lindley is a science writer and author based in Alexandria, VA.

© 2009 ACM 0001-0782/09/1000 \$10.00

Calendar of Events

October 15-16
2009 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL
Sponsored: SIGSOFT
Contact: Laurie Ann Williams
Email: williams@csc.ncsu.edu

October 17-18
First Asia-Pacific Symposium on Internetworking, Beijing, China
Contact: Hong Mei
Email: meih@pku.edu.cn

October 18-20
4th International Conference on Nano-Networks, Lucerne, Switzerland
Contact: Alexandre Schmid
Email: Alexandre.schmid@epfl.ch

October 19-20
Symposium on Architecture for Networking and Communication Studies, Princeton, NJ
Contact: Peter Z. Onufryk
Email: peter.onufryk@idt.com

October 19-24
ACM Multimedia Conference, Beijing, China
Sponsored: SIGMM
Contact: Wen Gao
Email: wgao@pku.edu.cn

October 20-22
4th International Conference on Performance Evaluation Methodologies and Tools, Pisa, Italy
Contact: Giovanni Stea
Email: g.stea@iet.unipi.it

October 21-23
Asia Information Retrieval Symposium, Sapporo, Japan
Contact: Tetsuya Sakai
Email: tetsuyasakai@xa2.so-net.ne.jp

October 22-24
ACM Special Interest Group for Information Technology Education Conference, Fairfax, VA
Contact: Donald Gantz
Email: dgantz!@gmu.edu

Debating Net Neutrality

Advocates seek to protect users from potential business practices, but defenders of the status quo say that concerns are overblown.

THE CONTROVERSY ABOUT network neutrality—the principle that Internet users should be able to access any Web content or use any applications, without restrictions or limitations from their Internet service provider (ISP)—remains unresolved in the U.S. over who, if anyone, has a legal or commercial right to regulate Internet traffic. Net neutrality proponents advocate for legislation that would keep broadband service providers from controlling Internet content or gaining the ability to impose extra charges for heavy users of the Internet. Opponents argue that existing rules enforced by the Federal Communications Commission (FCC) and others make additional laws unnecessary, or could jeopardize service providers' First Amendment rights.

But now some analysts are warning that the battle may ultimately mean much more than decisions about bits, bandwidth, pricing, and flow control.

"The battle for Net neutrality is also a battle over the shape of American culture," says Tim Wu, a Columbia University law professor specializing in copyrights and communications who wrote a 2003 paper, "Network Neutrality, Broadband Discrimination," which popularized the concept of Net neutrality.

Wu fears that in an under-regulated Internet ruled by ISPs, commercial motives could stifle entertainment, culture, and political diversity. For example, Wu says, the current open Internet is "bad for the business models" of traditional newspapers, many of which don't charge for the news they publish on their Web sites. But what if a news organization, for financial means, aligned itself with an individual ISP to sell premium content?

"If you imagine a non-open Internet, there could be the official newspapers of AT&T's network, let's say," Wu



Musical artist Moby, left, and U.S. Representative Ed Markey at a SavetheInternet.com press conference on net neutrality in Washington, D.C., with a counterdemonstrator's sign blocking the view of the Capitol dome.

explains. Part of an AT&T customer's monthly bill would support the *Wall Street Journal*, for example, but these subscribers "can't get access to *USA Today* or something else. It starts to become a little bit dicey when you can see how that would help [news organization's] business models."

Innovation and experimentation could also suffer, Wu warns. "In the early days of mobile phones, the only way an application appeared on a mobile phone was if it made money for the phone company," he says. But in an Internet context, because some now-familiar features don't have a direct commercial bent, they may never have been introduced. "In an Internet that [ISPs] can control, why would you put Wikipedia on it?" he asks. "It doesn't make sense because it doesn't make money."

Vinton Cerf, chief Internet evangelist at Google, also sees potential accessibility problems if broadband providers wield too much power. "All of us, in the consumer space especially, stand to lose the opportunity to access new products and services if we don't get [the debate] right," Cerf warns.

Cerf cites the well-publicized legal wrangling over broadband provider Comcast's deliberate slowing down of traffic for customers using BitTorrent and other filing-sharing applications on its network in 2007. "As long as we have clumsy or consumer-unfriendly regimes for network management," he says, "we will see problems for some reasonable applications and thus some inhibition of innovative new services."

Cerf envisions a safer Net world where regulations would constrain anti-competitive practices, such as unfair pricing by ISPs that compete with providers of online movie-on-demand services. "Fairly high aggregate caps on total monthly transfers or, preferably, a cap on the minimum assured rate for a given price would be very attractive," Cerf says. The "key point," he adds, is that "the user gets to pay for a guarantee of a certain minimum rate. If the user doesn't use it, others may. That's the whole dynamic of capacity sharing."

Net Neutrality Skeptics

Others dismiss such fears. "The threat level is not red or orange. We are way down in yellow," says Barbara Esbin, director of the Center for Communications and Competition Policy at the Progress and Freedom Foundation, a pro-business Washington, D.C.-based think tank supported in part by broadband providers such as AT&T, Comcast, and Verizon.

Net neutrality skeptics dismiss the notion that the lack of additional Internet controls is endangering society, and argue that the status quo engenders a

competitive market that's adequately watched over by the FCC and Federal Trade Commission. The current environment promotes innovation, says Robert Pepper, vice president of global technology policy for Cisco Systems. He cites the ongoing investments in fiber-optic networks for broadband services as well as the move by wireless carriers to provide high-speed 3G and WiMAX networks. "The whole innovation engine offers numerous examples of new apps being developed, new business models being tried," Pepper says. "Where is the failure to innovate?"

Others go further, saying that new constraints imposed in the name of Net neutrality could make it difficult for service providers to do essential management and maintenance. For example, some Net neutrality advocates worry about ISPs examining not just the headers of the data traffic—information required for routing it through the network—but also peering into the actual content of the packets. They worry that such deep packet inspections could open the door to blocking of content for commercial or political reasons.

But the danger of deep packet inspection should be kept in perspective, say some experts. For one thing, the technique is important for secure network operation, says David Farber, a professor of computer science and public policy at Carnegie Mellon University. "If I can't do deep packet inspections I have no way as a carrier of handling denial-of-service attacks," he says.

Concerns About Privacy

Still, issues surrounding Net neutrality

could impinge on fundamental concerns about personal privacy. David Clark, a senior research scientist at Massachusetts Institute of Technology, says the danger is more subtle than that of ISPs blatantly snooping on what sort of content is being transmitted through their pipes. Using deep-packet inspections to block content "is nonsense in the United States," says Clark. "That's so 20th century." The more relevant danger, according to Clark, is that ISPs will be motivated by profit opportunities to analyze what content subscribers are viewing and then use that data for commercial gain. "What we are going to be fighting about in two years is who has the right to observe everything you do," he predicts. "[ISPs] can completely model you based on your behavior by doing a deep-packet inspection. That's the issue that takes over from the rather simplistic fear that what these ISPs are going to block a packet or degrade access or something."

One attractive commercial option is for ISPs to replicate the revenue niche established by companies like Google. For example, when an Internet user enters a search query for mortgage rates in Arizona, "that guy is targeted" by search engine companies, Clark points out. These search firms can insert relevant ads into the pages that display query results and charge advertisers a premium for delivering their messages to a highly targeted audience. Various types of analytical applications could give broadband providers an efficient way to slice and dice their customers' usage data, and thus gives ISPs an opportunity to argue that they're able to

"What we are going to be fighting about in two years," says David Clark, "is who has the right to observe everything you do."

place ads as precisely tuned to individual users' interests as those inserted by search companies. Clark notes that modeling is already happening thanks to companies such as Phorm, which feed ISPs analyses of consumer behavior on the Web.

But existing regulatory and market forces may already be working to keep abuses in check. "I've heard of a few companies in the past that have bought those services," Farber says. "But they usually backed off rapidly because of the noise they were getting" by congressional watchdogs. Indeed, an ad-targeting company similar to Phorm, called NebuAd, shut down its U.S. operation in mid-2009 after becoming the target of congressional scrutiny. That wasn't the final word, however; the company immediately started doing business in England as Insight Ready Ltd. **■**

Alan Joch is a business and technology writer based in Franconstown, NH.

© 2009 ACM 0001-0782/09/1000 \$10.00

Net Neutrality: A Worldwide Debate

Nations around the world are refereeing similar debates although often in the context of a different competitive environment. For example, industry estimates count more than 200 European network operators that provide Internet services. The U.S., by contrast, has only about a dozen large broadband ISPs.

Nevertheless, the two continents may eventually find themselves influencing

how the other resolves the Net neutrality debate. In May, the European Parliament voted for a package of telecommunications policies, including one that affirms the principle of Net neutrality. According to *The New York Times*, groups of U.S. lobbyists were on site to influence European regulations over the winter when the measure was being debated. The goal: any decisions formalized in

Brussels might shape the views of U.S. regulators.

One analyst isn't surprised by the outreach by U.S. lobbyists. "Net neutrality is a rainmaker issue—you can get a lot of funding for your organization if you can point to a crisis that requires your group's advocacy to resolve it," says Barbara Esbin, director of the Center for Communications and Competition Policy at the

Progress and Freedom Foundation, a pro-market Washington, D.C.-based think tank.

But Esbin believes the U.S. could ultimately play the most influential role of all. "The U.S. is going to be looked at for how we resolve this," she says, "and it would be great if this country could show leadership again on communications policy."

—Alan Joch

Shaping the Future

To create shape-shifting robotic ensembles, researchers need to teach micro-machines to work together.

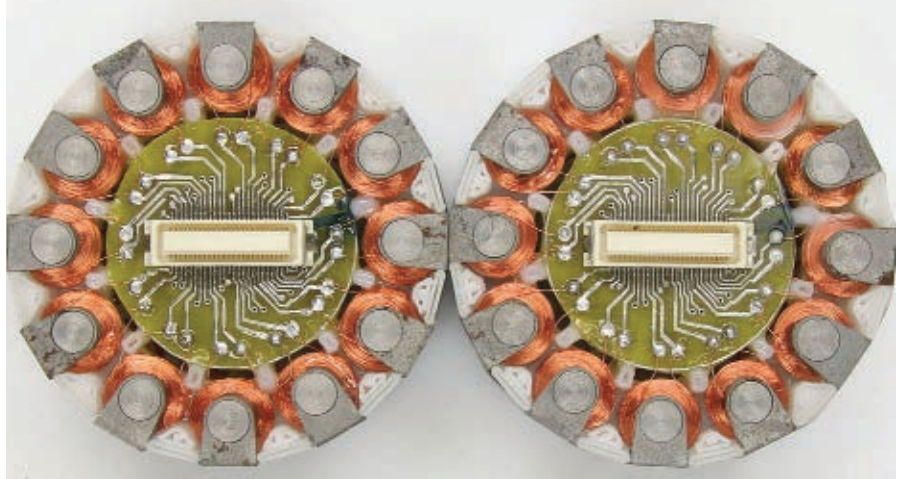
A PROTOTYPE OF YOUR company's latest product sits on the conference table. You suggest sleeker styling. The 3D model reacts, fluidly changing its angles. A colleague recommends different proportions, and again the model reflects those edicts. A pass of the hands changes the model's color; another reveals its interior, while a third folds it back into a flat sheet. Computer-aided design software records all the changes to replay them later, and to easily produce the final version.

That's the vision of researchers at Carnegie Mellon University and elsewhere as they embark in the new field of robotics known as claytronics. In it, a shape-shifting object is comprised of thousands of miniature robots known as claytronic atoms, or catoms. Each catom has the power to move, receive power, compute, and communicate with others, resulting in an ensemble that dynamically envisions three-dimensional space much as the pixels on a computer monitor depict a two-dimensional plane. Functional scenarios for claytronics include a general-purpose computer changing from laptop to cellphone form as needed, or, say, a plumber's tool that changes from a scuttling, insect-like device in crawl spaces to a slithering snake-like shape when inside pipes.

Ultimately, the claytronics dream goes, we may have morphing humanoids like T-1000 in the movie *Terminator 2* or Odo in the TV series "Star Trek: Deep Space Nine." Because their shape-shifting abilities could make them excellent mimics of human forms, such androids might act as reusable stand-ins for surgeons, repair technicians, and other experts who control them remotely.

Collaborative Research

Two joint ventures have taken the lead in this field. One partnership—



A top view of two magnet rings, with individual driver boards, from the self-actuating Planar Catom V7 developed by the Carnegie Mellon University-Intel claytronics research team.

between Carnegie Mellon University's Collaborative Research in Programmable Matter project and Intel Research Pittsburgh's Dynamic Physical Rendering project—focuses on programmable materials' shape-shifting aspects as well as the software needed to drive it. The other major effort eyes military applications; this collaboration is between the Defense Advanced Research Projects Agency (DARPA) and a consortium of colleges including the University of California at Berkeley, Harvard University, the University of Pennsylvania, Massachusetts Institute of Technology, and Cornell University.

To form stable ensembles, catoms must have power, communication, and adherence to one another.

Both projects face a host of challenges. Most immediately, the individual mechanical devices need to be much smaller than the units most recently demonstrated, which were about the size of large salt shakers. Jason Campbell, a senior staff research scientist at Intel's Pittsburgh laboratory, says his group is working with catoms of a tubular structure that are much smaller—1 millimeter in diameter and 10 millimeters in length. He believes that they need only be small enough to disappear to our eyes and touch. "That gives us a size target between a tenth of a millimeter and one millimeter across," he says, with the size requirement depending on application. A shape-changing radio antenna doesn't need to be very small, for example, whereas for a handheld device, "the consumer is the perceptual system" and miniaturization has great value, says Campbell.

Claytronics research flows out of the field of modular robotics, where individual units are typically a few inches across. But as the units get smaller—and more numerous—the physics controlling them changes. Kasper Støy, associate professor of computer systems engineering at the University of South-

ern Denmark, points to one difference between the macro and micro levels. With larger modules, he says, “we have to fight a lot with gravity. All the modules have to be able to lift the others. We’ve long used electromechanical actuators to overcome gravity in large modules. But at the micro level, you don’t want to have a big actuator in every little unit. The question is, How do you go from a bunch of really weak modules and get a big, strong robot?”

Size is not the only challenge. To form stable ensembles, catoms must have power, communication, and adherence to one another. The Carnegie Mellon project is currently attempting to address most of these issues by attaching electrostatic plates to the surface of each catom. Plates given opposite charges would cling to each other; changing the charges in a precisely programmed sequence would force the plates to roll over one another into the desired configuration. Information could be exchanged between adjoining catoms by manipulating their voltage differences.

Claytronics researchers are considering several options for providing power. In early versions, “catoms will get power from the table they sit on, through capacitive coupling,” says Seth Goldstein, an associate professor of computer science who leads the Carnegie Mellon team. But additional technologies are also under consideration, Goldstein says. “We are looking at magnetic resonance coupling, which goes over a longer distance,” he says. “We’re also looking at solar power. Because of

their size, the catoms are translucent. My guess is that we’ll use multiple ways of getting power in.”

Mark Yim, an associate professor of mechanical engineering at the University of Pennsylvania who is involved with the DARPA project, has experimented with drawing power from the environment and letting the catoms’ intelligence and adhesion turn it into useful motion. “I asked, What if we shake the table to get the modules to move around each other, and the modules determine when to grab or let go?” says Yim. “If they get really small, you might be able to move them with sound waves that shake the air.”

As catom production ramps up, these questions assume greater urgency. Goldstein notes that his lab, in collaboration with the U.S. Air Force Research Laboratory, has “refined the process to print the catom’s circuits on a standard silicon die, then post-process it to curl up into a sphere or a tube.” By this December, Goldstein says, he hopes to fabricate a silicon cylinder about one millimeter in diameter that will “move around under its own control, transfer power from a ground plane, and be able to communicate with other units.”

The Shapes to Come

A pile of catoms is useless without coordination. And although the challenge of creating claytronic software is similar to that of any massively parallel computer system, there is one key difference: The catoms’ physical nature makes error-correction more important. “Catoms

Education

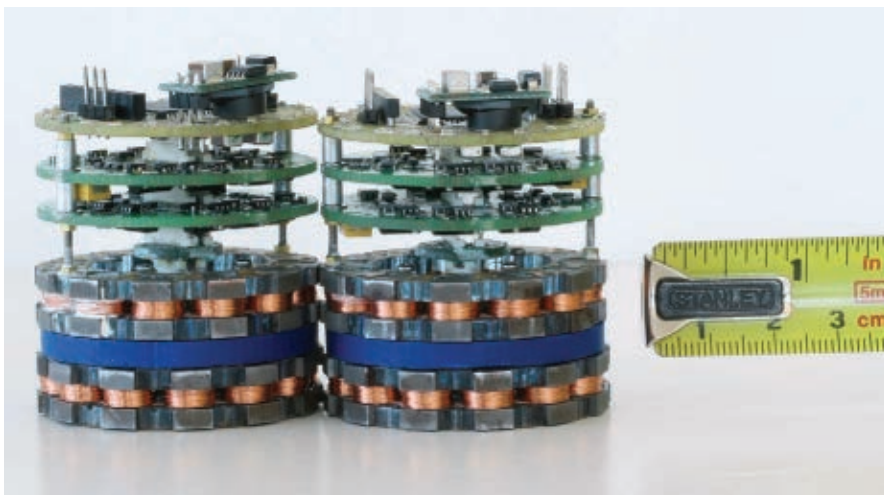
U.S. CS Courses Decline

The number of computer science courses being taught in high schools throughout America is steadily decreasing, according to the 2009 CSTA Secondary Computer Science Survey conducted by the Computer Science Teachers Association. The survey, based on responses from 1,094 high school teachers in spring 2009, found that only 65% of schools offer an introductory CS course, compared with 73% in 2007 and 78% in 2005. Likewise, the number of advanced placement CS courses has also declined, with 27% in 2009, 32% in 2007, and 40% in 2005.

“There are really three big requirements for reversing the trends highlighted in the survey,” said CSTA Director Chris Stephenson in an email interview. “First, we need to ensure that computer science is classified as a rigorous academic subject in K-12 and not as a ‘technical’ or ‘vocational’ subject. Second, as a community, we have to do a much better job of explaining to policymakers, parents, and students what unique skills and knowledge computer science provides and what future opportunities it offers. Finally, we have to fix the mess that is computer science teacher preparation and certification in most states, so that computer science teachers can be certified as CS teachers and not as math, science, business, or tech teachers.

“Our highly decentralized education system in the U.S. makes it very difficult to achieve systematic, long-term change,” said Stephenson. “In each state, the players, the priorities, and the policies are different.”

The survey’s findings were not unexpected. “This is the third time we have done this survey,” said Stephenson, “and so while we were not really surprised by the results, we were more troubled than ever with the continued decline in the number of courses being offered and the fact that it is getting increasingly harder for students to fit elective computer science courses into their timetables.”



A pair of Planar Catom V8s, each of which weighs 100 grams, with their stack of magnet-sensor rings on the bottom and solid-state electronic control rings on the top.

For catoms to be commercially viable, large quantities of the devices must be produced at a low cost.

are mechanical devices, and mechanical devices fail,” says Campbell.

There’s also the question of external control: How much instruction should the aggregate get from an outside source? Boston University electrical and computer engineering professor Tommaso Toffoli expresses skepticism about the ability of a claytronic ensemble to be self-directed. In the 1990s, he says, companies attempting to build massively parallel computers failed because they misread this problem. “People eventually found that instead of having millions of tiny computers the size of ants trying to simulate the brain, they got more work done by having just a few hundred that act as slaves, carrying sand and water to a floating-point processor,” Toffoli says.

Campbell disagrees. He believes that for catoms to be as effective as possible, they will each need to have onboard intelligence and that process-

ing tasks will need to be distributed throughout the ensemble. “The more parts you have, the more it matters that you have processing on each part. There are still things you can centralize, but decisions about what modules to move when have to be done inside the aggregate,” says Campbell.

The Carnegie Mellon team uses simulations to address these questions, and has created two languages that are optimized to program the ensemble. One, called Meld, is a stateless, declarative, logic-based language used to control the behavior of individual catoms. The second, locally distributed predicates (LDP), is used to recognize patterns and evaluate conditions throughout the ensemble as a whole. Meld and LDP are complementary, says language developer Michael Ashley-Rollman, a computer science graduate student at Carnegie Mellon. “If one robot stops working,” he explains, “Meld just throws everything away from that robot and pretends that it was never there.” LDP, on the other hand, “permits programmers to manage states themselves”—at the expense of error correction. LDP, Ashley-Rollman says, “gives Meld rules through which it can determine what the state can be.”

The Reality Challenge

Commercial viability for catoms will require major advances in the ability to produce large quantities of the devices at low cost. Goldstein hopes that

his lab at Carnegie Mellon will build working, submillimeter catoms by the end of 2009. By 2012, he believes, the researchers will have “hundreds of catoms working together intelligently.” More fantastically, Goldstein hopes that in 30 years there will be a claytronic humanoid that looks, feels, and moves in ways almost identical to a person.

But such ambitious goals may obscure a more fundamental issue. “Say, you have something like claytronics,” Yim says. “You have the mechanical part working. You’ve got the computational part working so you can control them in a distributed fashion. The question is, What do you *do* with it? Maybe you want it to go get a beer from the refrigerator. How does it know what is the best shape to be in to open the door?”

Indeed, making claytronics truly useful depends on developments in areas beyond these projects’ current scope, including artificial intelligence and human modeling. But the scenarios described only illustrate possible applications for claytronics, and in fact researchers are quicker to say what the technology will be than what it will do. But that’s the very problem that general-purpose computers faced until the mid-1980s; like PCs, claytronic ensembles may ultimately prove their worth in ways now unimagined. ■

Tom Geller is an Oberlin, Ohio-based science, technology, and business writer.

© 2009 ACM 0001-0782/09/1000 \$10.00

Milestones

Computer Science Awards

The Internet Society and the National Science Foundation (NSF) recently honored members of the computer science community for their innovative research.

POSTEL SERVICE AWARD

The Internet Society awarded the 2009 Jonathan B. Postel Service Award to CSNET, the research networking effort that during the early 1980s provided the critical bridge from the original research undertaken through ARPANET to today’s Internet. Four principal investigators—Peter J. Denning,

David Farber, Anthony C. Hearn, and Lawrence Landweber—and NSF program officer Kent Curtis, who encouraged and funded CSNET, were recognized for their important roles.

CSNET began in 1981 with a five-year NSF grant, and by 1986 CSNET had connected more than 165 academic, government and industry computer research groups, which comprised more than 50,000 researchers, educators and students around the world. Open to computer researchers, CSNET, which was self-governing and self-

supporting, demonstrated that researchers valued the type of informal collaboration it made possible. The success of CSNET encouraged NSF to undertake the NSFNET program, which brought open networking to a larger academic community and presaged the emergence of today’s Internet.

NSF CAREER AWARDS

NSF recently awarded Mathews Jacob, an assistant professor in the department of biomedical engineering at the University of Rochester, and Haijun Su,

an assistant professor in the University of Maryland, Baltimore County’s mechanical engineering department, with Faculty Early Career Development Awards.

Jacob was recognized for his research into the design of computer programs that accelerate the capture of high-resolution images, and that promise to enable early diagnosis of cancer. Su’s virtual reality research involves the creation of a virtual design environment in which conceptual designs can be quickly and inexpensively evaluated.

V viewpoints



DOI:10.1145/1562764.1562774

Phillip G. Armour

The Business of Software Contagious Craziness, Spreading Sanity

Some examples of the upward or downward spiral of behaviors in the workplace.

IN THE LATE 1970s the president of a company I worked with asked the company's economists and financial forecasters to provide their future market predictions for the next decade. This was a capital-intensive business with long lead times for equipment purchases, so a realistic appraisal of the likely future business environment was a pretty important thing. When the predictions arrived, they did not match what the president wanted to see—not at all.

It is not uncommon that people in positions of power are strong-willed, opinionated individuals who are used to getting their own way. The president was no exception and his response to the figures presented was quite dramatic. After some blustering, shouting, and numerous derogatory remarks directed at the professional expertise of the financial forecasting group, he scratched out some numbers on a piece of paper and handed them over to the chief economist. "These are the numbers I want to see," he said, "make these happen."

There is no doubt that the will and

the drive to do things is a very important attribute. Certainly, having a strong conviction that something *cannot* be done is usually a self-fulfilling prophecy. If people are convinced that something is not achievable, then they usually won't achieve it—if we argue for our limitations, we get to keep them. But sometimes things cannot be

accomplished simply through force of will. Just because we really, really want something doesn't mean we will be able to get it—even if we are the president of a major corporation.

Infectious Conduct

Capricious behavior, particularly on the part of powerful or influential peo-



ple, can be infectious. When one person in a system starts acting oddly, nearby people have two choices: to label the behavior as odd or to act like it is normal. If, for whatever reason, people act as if someone's weird actions are OK, they themselves start behaving weirdly. It is almost as if the odd behavior is catching. To compound the problem, we humans have built-in rationalizing capability that kicks in like a reflex when we act in an odd or unethical manner. This rationalization employs a thing called "cognitive dissonance" and allows us to continue to act in a weird way while simultaneously retaining the conviction that we are not acting in a weird way at all.^a

The Project Managers: Can-Do

A friend of mine, a project manager at a large electronics company, described this behavior in the planning meetings he attends. The project managers reporting to the strong-willed and forceful vice president in charge of their division almost compete with each other to promise things to the boss and pretend they and their teams can do things they really don't think they can do at all. The boss is the instigator. He puts enormous pressure on his people and browbeats them if they come up with numbers he doesn't like. When one manager "caves" and agrees to something he (secretly and privately) doesn't think can be done, the others feel they have to do so as well. The compliant managers then get praised by the vice president, which reinforces the behavior.

Privately, the managers bemoan their fate and wring their collective managerial hands over what their boss has forced them to commit to. But, until recently, they didn't change their behavior.

The Construction Manager: First Law of Behavior

Years ago, I was coaching a (non-software) manager working in the construction industry. An affable and customer-centric person, his life was being made very difficult by his primary customer. The construction company built telephone switch centers and no matter what the manager promised and agreed

to do for his customer, the customer always wanted more. In fact, it seemed like the more he gave the customer the more was wanted. The "customer is always right" approach did not seem to be working. After much discussion we decided that:

- ▶ The customer was a very strong-willed and decisive person.

- ▶ Asking for "more" is a perfectly appropriate thing for a customer to do, especially from the customer's perspective.

- ▶ The customer was asking for more *because* more was being provided.

- ▶ If the construction manager agreed to do more, it meant he must be able to do it.

- ▶ As a strong-willed decisive person, the customer was expecting the manager to be equally strong-willed and decisive in deciding what could *not* be done.

- ▶ The customer would continue piling on until the project manager said "no."

This could be called "Newton's First Law of Behavior": *every behavior will continue until acted upon by another behavior*. We could even extend this to Newton's Third Law and infer that the other behavior must be equal and opposite. This meant the project manager had to learn how to apply equal force in saying "no" to the extra work to balance the force the customer was applying in demanding the extra work.

Breaking the Cycle

To stop this behavior, people and organizations must somehow get out of the cycle. For the construction manager it was to learn to be firm and to realize the customer is not best served by trying to do everything. The customer is best served by most effectively doing the most important things.

For the software project managers dealing with the vice president, my friend bravely decided to break the cycle himself. After working a lot on his project's estimation practice, he vigorously defended his estimates to the vice president and simply refused to back down when pressured to reduce the projections. At one point, he even challenged the vice president to fire him if necessary. The vice president wisely chose not to do this and privately commented that it "took guts" to stand up

and hold your ground like that.

Then an interesting thing started happening. Since there were dependencies operating between the division's projects, other project managers started intentionally "hooking" their project estimates and plans to my friend's project plan. Their reasoning was that since the boss doesn't mess with *that* project if my project has dependencies on it, he won't mess with my project either. As each project stabilized by being strongly coupled to more realistic project deadlines, everyone started calming down. Inexorably, sanity spread across the organization as people started committing only to things they really believed they could do.

Infectious Insanity

The opposite can be true too, as evidenced in the case of the financial forecasting scenario mentioned earlier. Confronted with the president's demand to "...make these numbers happen..." the Ph.D. economists returned to their financial forecasting groups and had to figure out how to ignore factors or promote factors to make it happen. One can imagine the chief economist's assistant saying "...but why would we ignore this factor, boss? We'd be crazy to do that! That's not how it works!" To which the chief economist might reply "Well, that's how it works if you want to keep your job..."

Then cognitive dissonance kicks in and people start rationalizing: "...well maybe the factor really isn't that important..." and, starting from the top, everyone starts separating from reality.

In the business of software, this cycle results in significant and perennial overcommitments. Lacking well-defined estimation practices, these commitments are simply the wishes of the strongest-willed people with the highest authority—unless the organizations and people that work for them can provide the appropriate counter-response.

It seems that sensible behavior or weird behavior will grow within organizations rather like a disease propagates. It is an upward or a downward spiral. But we can choose the direction. ■

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

Copyright held by author.

a Tavis, C. and Aronson, E. *Mistakes Were Made (But Not by Me)*. Harvest Books, 2003.



DOI:10.1145/1562764.1562775

Martin Campbell-Kelly

Historical Reflections Computing in the Depression Era

Insights from difficult times in the computer industry.

SINCE THE BEGINNING of the computer industry, it has been through several major recessions. The first big one was in 1969; there was a major shakeout in the mid-1980s; and most recently there was the dot-com bust in 2001. A common pattern observed in these downturns was that they occurred approximately five years after the establishment of a new computing paradigm—the launch of the IBM System/360 platform in 1964, the personal computer in the late 1970s, and the commercial Internet in 1995. These new computing modes created massive opportunities that the entrepreneurial economy rapidly supplied and then oversupplied. It then took only a small downturn in the wider economy to cause a major recession in the computing industry.

The current recession appears to be quite different when compared to these earlier downturns. Unlike in earlier recessions, computing is not a victim of its own excess, but is suffering from the general economic malaise. Computing is not much different than any other industry in the current recession—it has no unique immunity. However, it has no unique vulnerability either, which offers a small amount of comfort.

Lessons from the Great Depression

To get some insight into what is happening today we have to look back to the Great Depression. Electronic computers did not exist at the time of the Wall Street crash of 1929, of course.



Despite the Great Depression, IBM increases its employment, trains more salesmen, and increases engineering efforts.

But there was an office machine industry whose products—typewriters, adding machines, and accounting equipment—performed many of the tasks now done with computers. Indeed, the most successful of the early computer firms sprang from the office machine industry—IBM, NCR, Remington Rand (later Univac), and Burroughs among them.

During the depression years protectionism was seen as one of the policy options. Because this option still surfaces from time to time, it is instructive to see what happened in the 1930s. The U.S. was a net exporter of office machines, so it was not interested in protectionism in this sector, of course. Most of the drive for protection came from nations that imported office machinery—but these policies were often the result of a tit-for-tat elsewhere in the economy. The world's two biggest

exporters of office machinery in the 1930s were the U.S. and Germany. Most other advanced countries, such as Britain and France, had their own office machine industries, but they found it difficult to compete at the best of times. Their response in the depression was to impose “ad valorem” import duties of 25% or 50%. In these countries import duties combined with a general lack of business investment made office machines formidably costly, and the domestic products were not always an adequate substitute. Although the import duties on office machinery may have briefly helped domestic manufacturers, retaliatory protectionist measures in other industries simply led to a downward spiral of economic activity. At the height of the depression in 1932, office machine consumption worldwide was down a staggering 60%. It is a difficult lesson, but selective pro-

tectionism did not help in the Great Depression and it won't help today.

IBM and NCR

The cases of IBM and NCR make interesting contrasts in weathering the economic storm.^a NCR fared badly—it didn't go out of business, but it was not until World War II that it fully recovered. In 1928, the year before the crash, NCR was ranked the world's second largest office machine firm (after Remington Rand) with annual sales of \$49 million; IBM was ranked the fourth largest with sales of less than \$20 million. A decade later, and well into the recovery, NCR sales were only \$37 million, whereas IBM had sales approaching \$40 million and it was the largest office machine supplier in the world.

The depression years 1929–1932 were a desperate time for NCR. Before the crash it had been a wonderful firm to work for. Headquartered in Dayton, Ohio, it had built a “daylight factory” set in urban parkland in the 1890s and it had pioneered in employee welfare, with all kinds of health, recreational, and cultural benefits. During the depression years NCR's sales fell catastrophically. Overseas sales, which had formerly amounted to 45% of total sales, were badly affected by protectionism. According to the then-CEO Edward Deeds “commercial treaties, tariff barriers, trades restrictions, and money complications” took “productivity from the Dayton factory.” It had to cut its work force by more than half in order to survive—from 8,600 down to 3,500 workers. At the worst of times, all that could be done was to sponsor a relief kitchen, run by the NCR Women's Club, to feed the unemployed and their families. Mirroring the fall in business, NCR's shares fell from a peak of \$165 in 1928 to \$6.79 in 1932. Recovery was very slow, and was only fully achieved with the coming of World War II when NCR was put to work making armaments, analog computer bombsights, and code-breaking machinery.

The story of IBM in the Great Depression could hardly be more differ-

ent than NCR's. IBM's main product line between the wars was punched card accounting equipment, which was the most “high-tech” office machinery. There were machines for punching and verifying cards, others for sorting and rearranging them, and the most complex machine—the tabulator—could perform sophisticated accounting procedures and report generation. Although orders for new machines fell drastically, IBM's president Thomas J. Watson, Sr. decided to maintain the manufacturing operation. Watson reasoned that rather than disband IBM's skilled design and manufacturing work force it would be more economical, as well as socially desirable, to stockpile machines for when the upturn came. For IBM, the upturn came in 1935 when President Franklin D. Roosevelt launched the New Deal and the Social Security Administration. The new administration was hailed as the “world's biggest bookkeeping operation.” IBM turned out to be the only office machine firm left that had an adequate inventory of machines to service the operation and it supplied 400 accounting machines to the government. It was a turning point for IBM and its profits soared for the remainder of the 1930s. Watson was justly celebrated for his faith in the future. He became a confidant of Roosevelt and chairman of the International Chamber of Commerce.

Heroic as Watson's strategy was, it would not have been possible for NCR, Remington Rand, or Burroughs to do the same. IBM had an income that was practically recession-proof. First, IBM's machines were not sold but leased. The manufacturing cost of an accounting machine was recovered in the first one or two year's rental, and after that, apart from the cost of maintenance, the machine revenue was pure profit. The accounting machines had an average life of at least 10 years, so it was an extremely profitable business. During the depression, although new orders stalled, very few firms gave up their accounting machines—not only were they dependent on the equipment, but they needed them more than ever to improve efficiency. During the depression years, while IBM did not lease many new machines, it was kept afloat by the revenues from those already in the field.

IBM's second big revenue source was the sale of punched cards. IBM enforced a rule—and got away with it until an antitrust suit in 1936—that only IBM cards could be used on IBM machines. Cards were sold for \$1.40 a thousand, far more than they cost to produce. Card revenues accounted for an astounding 30% of IBM's total revenues and an even higher percentage of its profits. Because cards were a consumable, firms had to continually purchase them so long as they were still in business.

The key difference between NCR and IBM was that NCR made almost all its money from the sale of new machines, whereas IBM made its money from two sources: leasing and the sale of punched card consumables. Looking back at the NCR and IBM experiences with the benefit of hindsight, we can see it was an early incarnation of the product-versus-services business models. When they start out, product firms have the advantage that they get a very rapid return on investment. In a single sale, the firm gets all the returns it will ever get from a customer. This helps a firm to grow organically in its early years. In contrast, when a services firm takes a new order it gets a modest annual income extending over many years. This slower rate of return makes it difficult for a firm to retain profits to achieve organic growth and it may need access to capital until it starts to generate a positive income. But the slower growth and steady income makes for much less volatility. As *Communications* columnist Michael Cusumano noted in 2003—writing in the aftermath of the dot-com bust—the trick for survival of all firms is getting the right mix of products and services.^b Products generate a rapid return on investment, but services provide a steady income that gives some immunity against recessions. It's not easy to get the balance right, but IBM did it in the 1930s. ■

^b Michael A. Cusumano, “Finding Your Balance in the Products and Services Debate,” *Commun. ACM* 46, 3 (Mar. 2003), 15–17.

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.

^a An excellent economic and business history of these firms is: James W. Cortada, *Before the Computer: IBM, NCR, Burroughs, and Remington Rand and the Industry They Created 1865–1965*, Princeton University Press, 1993.

Inside Risks

Reflections on Conficker

An insider's view of the analysis and implications of the Conficker conundrum.

Conficker is the name applied to a sequence of malicious software. It initially exploited a flaw in Microsoft software, but has undergone significant evolution since then (versions A through E thus far). This column summarizes some of the most interesting aspects of Conficker from the viewpoint of an insider who has been analyzing it. The lessons for the future are particularly relevant.

IN MID-FEBRUARY 2009, something unusual, but not unprecedented, occurred in the malware defense community. Microsoft posted its fifth bounty on the heads of those responsible for one of the latest multimillion-node malware outbreaks.⁴ Previous bounties have included Sasser (awarded), Mydoom, MSBlaster, and Sobig. Conficker's alarming growth rate in early 2009 along with the apparent mystery surrounding its ultimate purpose had raised enough concern among whitehat security researchers that reports were being distributed to the general public and raising concerns in Washington, D.C.

Was it all hype and of relative small importance among an ever-increasing stream of new and sophisticated malware families? What weaknesses in the ways of the Internet had this botnet brought into focus? Why was it here and when would it end? More broadly, why do some malware outbreaks garner wide attention while other multimillion-victim epidemics (such as Seneka) receive little notice? All are fair questions, and to some degree still remain open.

In several ways, Conficker was not fundamentally novel. The primary infiltration method used by Conficker to



infect PCs around the world was known well before Conficker began to spread in late November 2008. The earliest accounts of the Microsoft Windows buffer overflow used by Conficker arose in early September 2008, and a patch to this vulnerability had been distributed nearly a month before Conficker was released. Neither was Conficker the first to introduce dynamic domain generation as a method for selecting the daily Internet rendezvous points used to coordinate its infected population. Prior malware such as Bobax, Kraken, and more recently Torpig and a few other malware families have used dynamic domain generation as well. Conficker's most recent introduction of an encrypted peer-to-peer (P2P) channel to upgrade its ability to rapidly disseminate malware binaries is also preceded by other well-established kin, Storm worm being perhaps the most well-known example.

Nevertheless, among the long history of malware epidemics, very few can claim sustained worldwide infiltration of multiple millions of infected drones. The rapidly evolving set of Conficker variants do represent the state of the

art in advanced commercial Internet malware, and provide several valuable insights for those willing to look close.

What Have We Learned?

Nearly from its inception, Conficker demonstrated just how effective a random scanning worm can take advantage of the huge worldwide pool of poorly managed and unpatched Internet-accessible computers. Even on those occasions when patches are diligently produced, widely publicized, and auto-disseminated by operating system (OS) and application manufacturers, Conficker demonstrates that millions of Internet-accessible machines may remain permanently vulnerable. In some cases, even security-conscious environments may elect to forgo automated software patching, choosing to trade off vulnerability exposure for some perceived notion of platform stability.⁷

Another lesson of Conficker is the ability of malware to manipulate the current facilities through which Internet name space is governed. Dynamic domain generation algorithms (DGAs), along with fast flux (domain name lookups that translate to hundreds or thousands of potential IP addresses), are increasingly adopted by malware perpetrators as a retort to the growing efficiency with which whitehats were able to behead whole botnets by quickly identifying and removing their command and control sites and redirecting all bot client links. While not an original concept, Conficker's DGA produced a new and unique struggle between Conficker's authors and the whitehat community, who fought for control of the daily sets of domains used as Conficker's Internet

rendezvous points.²

Yet another lesson from the study of Conficker is the ominous sophistication with which modern malware is able to terminate, disable, reconfigure, or blackhole native OS and third-party security services.⁶ Today's malware truly poses a comprehensive challenge to our legacy host-based security products, including Microsoft's own anti-malware and host recovery technologies. Conficker offers a nice illustration of the degree to which security vendors are challenged to not just hunt for malicious logic, but to defend their own availability, integrity, and the network connectivity vital to providing them a continual flow of the latest malware threat intelligence. To address this concern, we may eventually need new OS services (perhaps even hardware support) specifically designed to help third-party security applications maintain their foothold within the host.

What Is Conficker's Purpose?

Perhaps one of the main reasons why Conficker had gained so much early attention was our initial lack of understanding of why it was there. From analyzing its internal binary logic, there is little mystery to Conficker. It is, in fact, a robust and secure distribution utility for disseminating malicious binary applications to millions of computers across the Internet. This utility incorporates a potent arsenal of methods to defend itself from security products, updates, and diagnosis tools. Its authors maintain a rapid development pace and defend their current foothold on their large pool of Internet-connected victim hosts.

Nevertheless, knowing what it can do does not tell us why it is there. What did Conficker's authors plan to send to these infected drones and for what purpose? Early speculation included everything from typical malware business models (spam, rogueAV, host trading, data theft, phishing), to building the next 'Dark Google',⁵ to fears of full-fledged nation-state information warfare. In some sense, we are fortunate that it now appears that Conficker is currently being used as a platform for conducting wide-scale fraud, spam, and general Internet misuse (rather traditional uses with well-understood motives). As recently as April 2009,

Conficker has been somewhat of a catalyst to help unify a large group of professional and academic whitehats.

Conficker-infected hosts have been observed downloading Waledec from Waledec server sites, which are known to distribute spam. Conficker has also been observed installing rogue antivirus fraudware, which has proven a lucrative business for malware developers.³

Is Conficker Over?

From October to April 2009, Conficker's authors had produced five major variants, lettered A through E: a development pace that would rival most Silicon Valley startups. With the exception of Conficker's variant E, which appeared in April and committed suicide on May 5th, Conficker is here to stay, barring some significant organized eradication campaign that goes well beyond security patches. Unlike traditional botnets that lay dormant until instructed, Conficker hosts operate with autonomy, independently and permanently scanning for new victims, and constantly seeking out new coordination points (new Internet rendezvous points and peers for its P2P protocol). However, despite their constant hunt for new victims, our Conficker A and B daily census (C is an overlay on prior-infected hosts) appears to have stabilized at between approximately 5 and 5.5 million unique IP addresses (as of July 2009).¹ Nevertheless, any new exploit (a new propagation method) that Conficker's authors decide to distribute is but one peer exchange away from every current Conficker victim. It is most probable that Conficker will remain a profitable criminal tool and relevant threat to the Internet for the foreseeable future.

Is There Any Good News?

Yes. Perhaps in ways the Conficker developers have not foreseen and certainly

to their detriment, Conficker has been somewhat of a catalyst to help unify a large group of professional and academic whitehats. Organized groups of whitehats on invitation-only forums have certainly previously self-organized to discuss and share insights. But Conficker brought together a focused group of individuals on a much larger scale with a clearly defined target, now called the Conficker Working Group (CWG).¹

The details of the CWG and its structure are outside the scope of this column, but the output from this group provides some insight into their capabilities. Perhaps its most visible action has been the CWG's efforts to work with top-level domain managers to block Internet rendezvous point domains from use by Conficker's authors. Additionally, group members have posted numerous detailed analyses of the Conficker variants, and have used this information to rapidly develop diagnosis and remediation utilities that are widely available to the general public. They actively track the infected population, and have worked with organizations and governments to help identify and remove infected machines. They continue to provide government policymakers, the Internet governance community, and Internet data providers with information to better reduce the impact of future malware epidemics. Whether such organized efforts can be sustained and applied to new Internet threats has yet to be demonstrated. ■

References

1. Conficker Working Group Web site (June 2009); <http://www.confickerworkinggroup.org>
2. Giles, J. The inside story of the Conficker worm. *New Scientist Journal* (June 12, 2009); <http://www.newscientist.com/article/mg20227121.500-the-inside-story-of-the-conficker-worm.html?full=true>
3. Krebs, B. Massive profits fueling rogue antivirus market. *The Washington Post* (Mar. 16, 2009); http://voices.washingtonpost.com/securityfix/2009/03/obsene_profits_fuel_rogue_ant.html
4. Lemos, R. Cabal forms to fight Conficker, offers bounty. *Security Focus* (Feb. 13, 2009); <http://www.securityfocus.com/news/11546>
5. Markoff, J. The Conficker worm: April Fool's joke or unthinkable disaster? *Bits: The New York Times* (Mar. 19, 2009); <http://bits.blogs.nytimes.com/2009/03/19/the-conficker-worm-april-fools-joke-or-unthinkable-disaster/>
6. Porras, P.A., Saidi, H., and Yegneswaran, V. Conficker C analysis. SRI International Technical Report (Apr. 4, 2009); <http://mtc.sri.com/Conficker/addendumC/#SecurityProductDisablement>
7. Williams, C. Conficker seizes city's hospital network. *The Register (U.K.)* (Jan. 20, 2009); http://www.theregister.co.uk/2009/01/20/sheffield_conficker/

Phillip Porras (porras@csl.sri.com) leads SRI International's Cyber Threat Analytics effort that includes the Malware Threat Center and BotHunter.

Copyright held by author.



Technology Strategy and Management

Dealing with the Venture Capital Crisis

How New Zealand and other small, isolated markets can act as “natural incubators.”

THE VENTURE CAPITAL industry, like financial services in general, has fallen on hard times. Venture funds historically have returned about 20% annually to investors, twice the average of U.S. stocks. Like stocks, though, returns over the past year have been sharply negative and investing has fallen dramatically. U.S. investments have dropped from the 2000 peak of \$105 billion to a low of \$20 billion in 2003 and recovered only to \$28 billion in 2008.¹ The 2009 numbers are running at half the 2008 level. Other countries see similar trends, including Israel, which usually has the highest level of venture capital investment given the size of its economy.³ Investment there was merely \$800 million in 2008, down from \$2.7 billion in 2000.^a

Part of the problem is that large payoffs have become increasingly scarce: Average valuations of venture-backed startups in 2008 were half those of 2007, while public offerings (IPOs) have dwindled—only six in the U.S. in 2008, compared to 86 in 2007.^b But creativity may be another problem for the industry.

Venture capital firms have been in-

a Israel Venture Capital Research Center; <http://www.ivic-online.com/>

b See VC Industry Statistics Archive from the National Venture Capital Association at <http://www.nvca.org/>



vesting recently in energy and the environment as well as more traditional areas, such as software, health care, biotechnology, medical devices, multimedia, and communications. But perhaps the biggest future challenge will be not the sector but the geography: VC firms put most of their money in home markets to keep a close watch on their entrepreneurs. U.S.-based VC firms also can argue that their home market offers the biggest public offerings and asset sales. But a recent survey of 700 VC firms found that 52% are now investing overseas. Nearly this same number planned to invest more in China, India, and other parts of Asia, while others want to invest in South America.² It makes

sense to explore these big, growing regions. But there is still a lot of competition. What really might jump-start the industry is more creative globalization, with an eye toward using some overseas markets as “natural incubators.”

Governments and universities have long supported entrepreneurs with incubators that offer seed money, office space, financial and business advice, and introductions to potential executives and customers. But most incubators I know of have done poorly (I was an advisor to four firms in the late 1990s and worked for several venture funds). In the U.S., entrepreneurs with the best ideas usually do not need to be incubated and get funding directly from VC



New Zealand-based special-effects company Weta Workshop, whose work includes *King Kong* shown here, has been awarded NZ\$5.8 million in government funding for a research partnership with TechNZ—the business investment program of the Foundation for Research, Science and Technology.

firms, private individuals serving as “angel” investors, or corporations. During the Internet boom, incubators also funded weak ideas and weaker teams. Nonetheless, in markets with limited venture capital or angel funding, incubators are still useful to help entrepreneurs get started.

It also occurs to me that potentially even more useful are markets that serve as natural incubators: small countries usually overlooked by international VC firms, either because of their size or isolation, but which have advanced economies, sophisticated customers, good universities, strong intellectual property rights, favorable tax laws, and vibrant entrepreneurial cultures. They might spawn ventures that become important new sources of wealth, social welfare, and employment—for the hosts and the world.

Israel (population seven million) used to be one of these natural incubators. It is the source of numerous important technologies (for example, instant messaging, security software, software testing tools, and SAP’s NetWeaver) and was discovered in the 1980s and 1990s by the VC community as well as American and European multinationals. Also, the U.S. provides nearly \$3 billion a year in aid to Israel. This, along with large defense and security spending, helps fuel demand for high technology. Another small, advanced market is Finland (population five million). But

this country is an integral part of Europe, and Nokia’s huge international success has increased the level of international attention and investment. Ireland (population four million) may fall into this category (see my October 2005 *Communications* column, “Software in Ireland: A Balance of Entrepreneurship and...Lifestyle Management?”), though its entrepreneurs generally prefer to remain small and benefit from proximity to the U.K., with 60 million customers, and Europe, with a half-billion people. There may well be other interesting markets to explore in Southeast Asia, Africa, and Latin America.

But a truly remote country I have come to know well is New Zealand (population four million). Over the past two years I have visited there twice, sponsored by the Foundation for Research, Science, and Technology (FRST; <http://www.frst.govt.nz/>), a government agency that invests in R&D

Perhaps the biggest future challenge will not be the sector but the geography.

projects at early-stage companies. This experience has encouraged me to think more about how small, isolated markets (though not small in land area—New Zealand is as big as California or Japan) can add some new life to the venture capital industry.

Economically, New Zealand is an advanced nation, though per-capita income used to rank with Western Europe and has fallen below Spain. The majority of the economy is services, like other industrialized nations, but it also relies heavily on tourism as well as large-scale exports of agricultural commodities. The desire for more variety in the economy is one reason why the government has promoted software, biotech, and other technology-based entrepreneurship. FRST, established in 1990, annually invests about 500 million New Zealand dollars (US\$300 million) in local company R&D and other promotional or educational efforts. The government works with local venture capitalists, who in 2008 invested NZ\$66 million (US\$40 million) in 52 deals (down from NZ\$82 million in 60 deals in 2007).⁴ Americans have not ignored New Zealand entirely. Motorola has invested in Opencloud (mobile software), Vinod Khosla in LanzaTech (bio-fuel), and Sequoia Capital and others in Right Hemisphere (visual collaboration software). In general, though, most of the world’s VC firms view New Zealand as too small and remote to merit much attention; this is shortsighted. The following brief descriptions of software-related firms I visited suggest a wide variety of technologies and ideas are being commercialized in just this one sector:

► Endace (<http://www.endace.com>) sells products that probe and monitor networks by putting “time stamps” on Internet packets. Financial institutions, governments, and other organizations are using the technology to improve security and optimize network performance. The company is public on London’s AIM exchange and reported revenues in the last fiscal year of about US\$30 million, with customers in 30 countries.

► Framecad (<http://www.framecad.com>) sells machinery and design software as well as consulting services that enable construction companies to create small or medium-sized steel-framed buildings. These are relatively

inexpensive and do not require much skilled labor to construct. In addition to New Zealand, the company has found customers in developing economies such as China, the Middle East, and Southeast Asia.

► GFG Group (<http://www.gfg-group.com>) provides off-the-shelf electronic payment solutions (credit and debit cards, mobile phone applications, ATM and POS applications) and related services to 115 million customers in 40 countries. Most of its business outside New Zealand is in Australia, Singapore, the Philippines, and the United Arab Emirates.

► Inro Technologies (<http://www.inro.co.nz>) sells robotics technology to retrofit manual forklifts and turn them into automated vehicles. Fonterra, New Zealand's largest firm and a major exporter of dairy products, invested because finding forklift drivers is difficult and expensive. It is even more expensive to build new automated warehouses from scratch.

► Methodware (<http://www.methodware.com>) provides customized risk management and internal audit software to financial services, energy, and utilities companies as well as the public sector. It originally targeted small and medium-size firms but through partnerships has been able to sell to 1,800 corporate clients in 80 countries.

► NextWindow (<http://www.nextwindow.com>) sells touch-screen computer displays and overlay touch-screen devices and software, initially for large screens and kiosks. It has grown very rapidly through international distribution partnerships and alliances with PC manufacturers around the world, especially in the U.S.

► OpenCloud (<http://www.opencloud.com>) sells a suite of real-time Java application servers (Rhino) as well as provides development tools and consulting for companies interested in building multimedia products and services, especially for mobile phones. It moved its headquarters to the U.K. to gain better access to customers.

► Smallworlds (<http://www.smallworlds.com>) is a 3D "virtual world" and social networking site that enables users to set up their own room spaces and then do instant messaging as well as share audio and video content or play games with their friends. Outside com-

With the proper level of ambition, talent, and opportunity, even a small, isolated company can turn the world into its market.

panies can use the site and tools as a development platform. The applications run inside a browser and do not require large (and slow) software downloads.

► Weta Workshop (<http://www.weta-workshop.com>) is one of the largest video-effects design and production companies in the world serving the movie industry and now branching out into other markets, such as animation for children's TV and technology for video game producers. It is best known for video effects in the *Lord of the Rings* movies (which were shot in New Zealand). It has won five Academy Awards for visual effects, costumes, and makeup.

► Wherescape (<http://www.wherescape.com>) combines unique in-house tools with an agile development methodology to build data warehouses quickly and cheaply for a variety of industries. It has hundreds of customers, mainly in New Zealand and Australia, but also works with partners around the world.

► Virtual Infrastructure Professionals (<http://www.vipl.co.nz>) provides custom-built virtualization, hosting, and disaster recovery solutions (using mostly VMWare and Citrix). It partners with most of the major software and hardware producers, but does most of its business in New Zealand.

These firms seemed very interested in exports, though many lack capital and experienced difficulties growing beyond a certain size. Some companies succeeded only because there was little international competition. The critical decision for government as well as private investors is to determine—before they put in too much time and money—which firms can export in volume. "Horizontal products" that can be sold to almost anyone in any market because

they represent common needs, are highly standardized, and do not need customization or specialized knowledge are the easiest to scale and export. But these businesses attract the most competition (see my July 2003 *Communications* column, "Beware the Lure of the Horizontal"). The easiest markets to gain a foothold in are "vertical services," such as custom-built applications or specialized services for a particular industry. But labor-intensive or skill-dependent work is difficult to scale and more difficult to export—which is why having an incubator and time to experiment can be important.

The population, physical characteristics, or other unique local requirements of a country can also inspire entrepreneurial creativity. For example, New Zealand has a severe shortage of people, so we see firms use software and other technologies to foster automation (such as retrofit forklifts) and devise inexpensive, fast solutions to common problems (building construction, data warehouse design, virtual hosting, electronic payments). Other firms take advantage of New Zealand's breathtaking scenery and creative art communities. But perhaps most important is for VC firms as well as entrepreneurs investing in small markets to think big—and follow the lead of Nokia or one of the 70 or so Israeli companies that have been listed on the NASDAQ stock exchange (more than any other foreign country). With the proper level of ambition, talent, and opportunity, even a small, isolated company can turn the world into its market. ■

References

1. Cain Miller, C. What will fix the venture capital crisis? *The New York Times* (May 4, 2009); <http://bits.blogs.nytimes.com/2009/05/04/what-will-fix-the-venture-capital-crisis/> and Sales of startups plummet, along with prices, *The New York Times* (Apr. 1, 2009); <http://bits.blogs.nytimes.com/2009/04/01/sales-of-startups-plummet-along-with-prices/>
2. *Global Economic Downturn Driving Evolution of Venture Capital Industry*, National Venture Capital Association Press Release (June 10, 2009); www.nvca.org
3. Megginson, W.L. Towards a global model of venture capital? *Journal of Applied Corporate Finance* 16 (2004), 89–107.
4. New Zealand Private Equity and Venture Capital Association and Ernst & Young, *The New Zealand Private Equity and Venture Capital Monitor 2008*; <http://www.nzvca.co.nz/Shared/Content/Documents/>

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.



DOI:10.1145/1562764.1562778

George V. Neville-Neil

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

Kode Vicious Kode Reviews 101

A review of code review do's and don'ts.

Dear KV,

My company recently went through a round of layoffs, and at the same time a lot of people left the company voluntarily. While trying to pick up the pieces, we've discovered that there are just some components of our system that no one understands. Now management is considering hiring back some folks as "consultants" to try to fix this mess. It's not like the code is un-commented or spaghetti; it's just that there are bits of it that no one remaining with the company understands.

It all seems pretty stupid and wasteful, but perhaps I'm just a bit grumpy because I didn't get a nice farewell package and instead have to clean up the mess.

Holding the Bag

Dear Holding,

Maybe you should quit and see if you get hired back as a consultant; I hear they get really good rates! Maybe that's not the right advice here. I meant to say, "Welcome to the latest round of recession," wherein companies that grew bloated in good times try to grow lean in bad times, but realize they can't shed all the useless pounds they thought they could. In my career this is round three of this wheel of fortune, and I am sure it will not be the last.

The best way to make sure that everyone on a team or that enough people in a group or company are able to maintain a significant piece of software is to institute system code reviews and to beat senseless anyone who does not at-



tend. Right now, that advice is a bit like closing the barn door after the train has left the station, or...whatever. It does bring me to a few things I would like to say about how to do a proper code review, which is something I don't think most people ever learn how to do—and certainly most programmers would not learn to do this if it were a choice.

There are three phases to any code review: preparation, the review, and afterward. One of the things most people miss when they call for—or in the case of managers, *order*—a code review is that it is unproductive just to shove a bunch of people in a room and show them unfamiliar code. When I say unproductive, what I mean is that it is a teeth-grinding, head-banging-

on-the-desk, vein-pulsing-in-the-head kind of experience. I've stopped such code reviews after 10 minutes when I realized no one in the room had read a single line of the code before they showed up in the meeting room. Please, to preserve life and sanity, prepare for a code review.

Preparations for a code review include selecting a constrained section of the code to look at, informing everyone which piece of code you have picked, and telling them that if they don't read the code before the appointed meeting time, you will be very displeased. When I say constrained, I do not normally mean a single page of code, nor do I mean a set of 30 files. Balance, as in all things KV talks about, is impor-

tant to the process. If you're reviewing a particularly nasty bit of code—for example, a job scheduler written by someone who is no longer with the company—then you're going to want to take smaller chunks for each review. The reason for the smaller chunks is that the person who wrote it is not present to explain it to you. A code review without a guide takes about two to three times as long as one conducted with the author of the code.

The next step is to schedule a time to review the code. Do *not* review code directly after lunch. KV once worked for someone who would schedule meetings at 2 P.M., when lunch was normally from noon to 1 P.M. I never failed to snore in his meetings. Code reviews should be done when people have plenty of brainpower and energy because reading someone else's code normally takes more energy than reading or writing your own: it's extremely difficult to get into someone else's mind-set. Trying to adjust your way of thinking to that of some other programmer takes quite an effort, if only to keep yourself from beating on that other programmer for being so apparently foolish. When I call a code review, it is either early in the day or two to three hours after lunch. Do not perform a code review for more than two hours. There is no such thing as a productive four-hour meeting, except for management types who equate the number of hours they've blathered on with the amount of work they've done.

Now for the review itself. Providing coffee and food is probably a good idea. Food has the effect of making sure people show up, and coffee has the effect of keeping them awake. The person lead-

One of the most difficult challenges in a code review is to avoid getting distracted by minutiae.

ing the code review, who may or may not be the author of the code, should give a short (no more than 10- to 15-minute) introduction to the code to be reviewed. Make sure to keep the person focused on the code being reviewed. Letting a programmer wax poetic leads to poor poetry and to wishing that, like Ulysses, you had wax in your ears. Once the introduction is complete, you can walk through any header files or places where data structures, base classes, and other elements are defined. With the basics of the code covered, you can move on to the functions or methods and review those next.

One of the most difficult challenges in a code review is to avoid getting distracted by minutiae. Many people like to point out every spelling and grammatical error, as if they're scoring points on a test. Such people are simply distracting the group from understanding the overall structure of the code and possibly digging for deeper problems. The same is true for language fascists, who feel they need to quote the language standard, chapter and verse, as if an ANSI standard is a holy book. Both of these types of issues should be noted, quickly, and then you should move on. Do not dwell here, for it is here that you will lose your way and be dashed upon the rocks by the Scylla of spelling and Charybdis of syntax.

As in any other engineering endeavor, someone will need to take notes on what problems or issues were found in the review. It is infuriating in the extreme to review the same piece of code twice in six months and to find the same issues, all because everyone was too lazy to write down the issues. A whiteboard or flip chart is fine for this, and in a pinch you might be able to trust the author of the code to do this for you. I would follow the latter path only if I trusted the author of the code, because programmers are generally lazy and will subconsciously avoid work. It's not even that they'll know they left off something to fix, but three months later you'll say to them, "Wait, we told you to fix this. Why isn't this fixed?!" To which you will receive curious looks accompanied by "What? This? Oh, you meant this?" If it's an issue, write it down while the group is thinking about it, and go over the list at the end of the review.

A compiler reads your code, but it doesn't understand its purpose or design; for the moment, only a person can do that.

When the review is over, there is still work to be done. Of course, someone has to fix all the spelling, grammatical, and language conformance issues, as well as the genuine bugs, but that's not all. Copies of the notes should be distributed to all participants, just in case something happens to the author of the code, and the marked-up copies of the code should be kept somewhere for future reference. There are now some tools that will handle this electronically. Google has a code-review tool called Rietveld for code kept in the subversion source-code control system. Although an electronic system for recording and acting on code-review issues is an excellent tool, it is not a substitute for formal code reviews where you discuss the design, as well as the implementation, of a piece of code. A compiler reads your code, but it doesn't understand its purpose or design; for the moment, only a person can do that.

KV

Related articles on queue.acm.org

A conversation with Steve Bourne, Eric Allman, and Bryan Cantrill

<http://queue.acm.org/detail.cfm?id=1454460>

Kode Vicious: The Return

<http://queue.acm.org/detail.cfm?id=1039521>

The Yin and Yang of Software Development

<http://queue.acm.org/detail.cfm?id=1388787>

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.

Viewpoint

Retrospective: An Axiomatic Basis for Computer Programming

C.A.R. Hoare revisits his past Communications article on the axiomatic approach to programming and uses it as a touchstone for the future.

THIS MONTH MARKS the 40th anniversary of the publication of the first article I wrote as an academic.^a I have been invited to give my personal view of the advances that have been made in the subject since then, and the further advances that remain to be made. Which of them did I expect, and which of them surprised me?

Retrospective (1969–1999)

My first job (1960–1968) was in the computer industry; and my first major project was to lead a team that implemented an early compiler for ALGOL 60. Our compiler was directly structured on the syntax of the language, so elegantly and so rigorously formalized as a context-free language. But the semantics of the language was even more important, and that was left informal in the language definition. It occurred to me that an elegant formalization might consist of a collection of axioms, similar to those introduced by Euclid to formalize the science of land measurement. My hope was to find axioms that would be strong enough to enable programmers to discharge their responsibility to write correct and efficient programs. Yet I wanted them to be weak enough to permit a variety of efficient implementation strategies, suited to the particular characteristics

of the widely varying hardware architectures prevalent at the time.

I expected that research into the axiomatic method would occupy me for my entire working life; and I expected that its results would not find widespread practical application in industry until after I reached retirement age. These ex-

pectations led me in 1968 to move from an industrial to an academic career. And when I retired in 1999, both the positive and the negative expectations had been entirely fulfilled.

The main attraction of the axiomatic method was its potential provision of an objective criterion of the quality of



C.A.R. Hoare attending the NATO Software Engineering Techniques Conference in 1969.

^a Hoare, C.A.R. An axiomatic basis for computer programming. *Commun. ACM* 12, 10 (Oct. 1969), 576–580.

a programming language, and the ease with which programmers could use it. For this reason, I appealed to academic researchers engaged in programming language design to help me in the research. The latest response comes from hardware designers, who are using axioms in anger (and for the same reasons as given above) to define the properties of modern multicore chips with weak memory consistency.

One thing I got spectacularly wrong. I could see that programs were getting larger, and I thought that testing would be an increasingly ineffective way of removing errors from them. I did not realize that the success of tests is that they test the programmer, not the program. Rigorous testing regimes rapidly persuade error-prone programmers (like me) to remove themselves from the profession. Failure in test immediately punishes any lapse in programming concentration, and (just as important) the failure count enables implementers to resist management pressure for premature delivery of unreliable code. The experience, judgment, and intuition of programmers who have survived the rigors of testing are what make programs of the present day useful, efficient, and (nearly) correct. Formal methods for achieving correctness must support the intuitive judgment of programmers, not replace it.

My basic mistake was to set up proof in opposition to testing, where in fact both of them are valuable and mutually supportive ways of accumulating evidence of the correctness and serviceability of programs. As in other branches of engineering, it is the responsibility of the individual software engineer to use all available and practicable methods, in a combination adapted to the needs of a particular project, product, client, or environment. The best contribution of the scientific researcher is to extend and improve the methods available to the engineer, and to provide convincing evidence of their range of applicability. Any more direct advocacy of personal research results actually excites resistance from the engineer.

Progress (1999–2009)

On retirement from University, I accepted a job offer from Microsoft Research in Cambridge (England). I was surprised to discover that assertions,

I did not realize that the success of tests is that they test the programmer, not the program.

sprinkled more or less liberally in the program text, were used in development practice, not to prove correctness of programs, but rather to help detect and diagnose programming errors. They are evaluated at runtime during overnight tests, and indicate the occurrence of any error as close as possible to the place in the program where it actually occurred. The more expensive assertions were removed from customer code before delivery. More recently, the use of assertions as contracts between one module of program and another has been incorporated in Microsoft implementations of standard programming languages. This is just one example of the use of formal methods in debugging, long before it becomes possible to use them in proof of correctness.

In 1969, my proof rules for programs were devised to extract easily from a well-asserted program the mathematical ‘verification conditions’, the proof of which is required to establish program correctness. I expected that these conditions would be proved by the reasoning methods of standard logic, on the basis of standard axioms and theories of discrete mathematics. What has happened in recent years is exactly the opposite of this, and even more interesting. New branches of applied discrete mathematics have been developed to formalize the programming concepts that have been introduced since 1969 into standard programming languages (for example, objects, classes, heaps, pointers). New forms of algebra have been discovered for application to distributed, concurrent, and communicating processes. New forms of modal logic and abstract domains, with carefully restricted expressive power, have been invented to simplify human and mechanical rea-

soning about programs. They include the dynamic logic of actions, temporal logic, linear logic, and separation logic. Some of these theories are now being reused in the study of computational biology, genetics, and sociology.

Equally spectacular (and to me unexpected) progress has been made in the automation of logical and mathematical proof. Part of this is due to Moore’s Law. Since 1969, we have seen steady exponential improvements in computer capacity, speed, and cost, from megabytes to gigabytes, and from megahertz to gigahertz, and from megabucks to kilobucks. There has been also at least a thousand-fold increase in the efficiency of algorithms for proof discovery and counterexample (test case) generation. Crudely multiplying these factors, a trillion-fold improvement has brought us over a tipping point, at which it has become easier (and certainly more reliable) for a researcher in verification to use the available proof tools than not to do so. There is a prospect that the activities of a scientific user community will give back to the tool-builders a wealth of experience, together with realistic experimental and competition material, leading to yet further improvements of the tools.

For many years I used to speculate about the eventual way in which the results of research into verification might reach practical application. A general belief was that some accident or series of accidents involving loss of life, perhaps followed by an expensive suit for damages, would persuade software managers to consider the merits of program verification.

This never happened. When a bug occurred, like the one that crashed the maiden flight of the Ariane V spacecraft in 1996, the first response of the manager was to intensify the test regimes, on the reasonable grounds that if the erroneous code had been exercised on test, it would have been easily corrected before launch. And if the issue ever came to court, the defense of ‘state-of-the-art’ practice would always prevail. It was clearly a mistake to try to frighten people into changing their ways. Far more effective is the incentive of reduction in cost. A recent report from the U.S. Department of Commerce has suggested that the cost of programming error to the world economy is measured in tens

of billions of dollars per year, most of it falling (in small but frequent doses) on the users of software rather than on the producers.

The phenomenon that triggered interest in software verification from the software industry was totally unpredicted and unpredictable. It was the attack of the hacker, leading to an occasional shutdown of worldwide commercial activity, costing an estimated \$4 billion on each occasion. A hacker exploits vulnerabilities in code that no reasonable test strategy could ever remove (perhaps by provoking race conditions, or even bringing dead code cunningly to life). The only way to reach these vulnerabilities is by automatic analysis of the text of the program itself. And it is much cheaper, whenever possible, to base the analysis on mathematical proof, rather than to deal individually with a flood of false alarms. In the interests of security and safety, other industries (automobile, electronics, aerospace) are also pioneering the use of formal tools for programming. There is now ample scope for employment of formal methods researchers in applied industrial research.

Prospective (2009–)

In 1969, I was afraid industrial research would dispose such vastly superior resources that the academic researcher would be well advised to withdraw from competition and move to a new area of research. But again, I was wrong. Pure academic research and applied industrial research are complementary, and should be pursued concurrently and in collaboration. The goal of industrial research

The phenomenon that triggered interest in software verification from the software industry was totally unpredicted and unpredictable.

is (and should always be) to pluck the 'low-hanging fruit'; that is, to solve the easiest parts of the most prevalent problems, in the particular circumstances of here and now. But the goal of the pure research scientist is exactly the opposite: it is to construct the most general theories, covering the widest possible range of phenomena, and to seek certainty of knowledge that will endure for future generations. It is to avoid the compromises so essential to engineering, and to seek ideals like accuracy of measurement, purity of materials, and correctness of programs, far beyond the current perceived needs of industry or popularity in the marketplace. For this reason, it is only scientific research that can prepare mankind for the unknown unknowns of the forever uncertain future.

So I believe there is now a better scope than ever for pure research in computer science. The research must be motivated by curiosity about the fundamental principles of computer programming, and the desire to answer the basic questions common to all branches of science: what does this program do; how does it work; why does it work; and what is the evidence for believing the answers to all these questions? We know in principle how to answer them. It is the specifications that describes what a program does; it is assertions and other internal interface contracts between component modules that explain how it works; it is programming language semantics that explains why it works; and it is mathematical and logical proof, nowadays constructed and checked by computer, that ensures mutual consistency of specifications, interfaces, programs, and their implementations.

There are grounds for hope that progress in basic research will be much faster than in the early days. I have already described the vastly broader theories that have been proposed to understand the concepts of modern programming. I have welcomed the enormous increase in the power of automated tools for proof. The remaining opportunity and obligation for the scientist is to conduct convincing experiments, to check whether the tools, and the theories on which they are based, are adequate to cover the vast range of programs, design patterns, languages, and applications of today's comput-

ers. Such experiments will often be the rational reengineering of existing realistic applications. Experience gained in the experiments is expected to lead to revisions and improvements in the tools, and in the theories on which the tools were based. Scientific rivalry between experimenters and between tool builders can thereby lead to an exponential growth in the capabilities of the tools and their fitness to purpose. The knowledge and understanding gained in worldwide long-term research will guide the evolution of sophisticated design automation tools for software, to match the design automation tools routinely available to engineers of other disciplines.

The End

No exponential growth can continue forever. I hope progress in verification will not slow down until our programming theories and tools are adequate for all existing applications of computers, and for supporting the continuing stream of innovations that computers make possible in all aspects of modern life. By that time, I hope the phenomenon of programming error will be reduced to insignificance: computer programming will be recognized as the most reliable of engineering disciplines, and computer programs will be considered the most reliable components in any system that includes them.

Even then, verification will not be a panacea. Verification technology can only work against errors that have been accurately specified, with as much accuracy and attention to detail as all other aspects of the programming task. There will always be a limit at which the engineer judges that the cost of such specification is greater than the benefit that could be obtained from it; and that testing will be adequate for the purpose, and cheaper. Finally, verification cannot protect against errors in the specification itself. All these limits can be freely acknowledged by the scientist, with no reduction in enthusiasm for pushing back the limits as far as they will go. □

C.A.R. Hoare (thoare@microsoft.com) is a principal researcher at Microsoft Research in Cambridge, U.K., and Emeritus Professor of Computing at Oxford University.

Copyright held by author.



Browse the ACM Digital Library and Guide to Computing Literature Index

Journals ♦ Magazines ♦ Conference Proceedings ♦ ACM Web sites ♦ Newsletters ♦ ACM Oral History Interviews ♦ Publications by Affiliated Organizations

The Most Complete Global Resource for Computer Scientists and Information Technology Professionals

ACM Membership:

Professionals and students may purchase both an ACM Membership and the *ACM Digital Library* benefitting from a wide variety of resources, including access to over 40 high-impact publications; free online books and courses for professional development; a searchable Digital Library; a growing online community of electronic forums; and more. For pricing information or to subscribe visit us online at <https://campus.acm.org/public/quickJoin/interim.cfm> or contact **ACM Member Services Department** at 1.800.342.6626 (U.S. and Canada), 212.626.0500 (Global) or email acmhelp@acm.org.



Association for Computing Machinery

- ♦ Unlimited full-text access to all scholarly content ever published by ACM since 1954 to the present
- ♦ Includes 256,000 Articles, 270+ Conference Proceedings Titles, 37 High-Impact Journals, 9 Specialized Magazines, and 43 Special Interest Groups contributing content
- ♦ Full-text access to the complete contents of ACM's award-winning flagship publication *Communications of the ACM*, including access to all premium content on the magazine's Web site <http://cacm.acm.org>
- ♦ Access to *ACM TechNews* technology news briefing service, and all video and multimedia content published by ACM, including 800+ multimedia files
- ♦ Contains *The Guide to Computing Literature Index* — the largest and most comprehensive bibliographic database in the field of computing with over 1.3 million records and 7 million references to the computing and IT literature
- ♦ Access to *The Online Computing Reviews Service*, providing timely commentary and critiques by leading computing experts of the most essential books and articles.
- ♦ Advanced technology includes cutting-edge search functionality and guided navigation
- ♦ COUNTER Compliant Usage Statistics
- ♦ Advancing the field of computing and technology with the highest-quality content at affordable rates

ACM Digital Library access is available by annual subscription to institutions and individuals worldwide through ACM Membership, Academic Consortia and Corporate Libraries.

Consortia and Corporate libraries are eligible to subscribe to special Digital Library Packages, which include the *DL Core Package*, *DL Master SIG Package*, and *The Guide to Computing Literature*. For pricing information or to subscribe, contact Nolen S. Harris at 212.626.0676 or email nolen.harris@hq.acm.org.

**FOR MORE INFORMATION VISIT
WWW.ACM.ORG/DL**

Article development led by ACMqueue
queue.acm.org

GPU acceleration and other computer performance increases will offer critical benefits to biomedical science.

BY JAMES C. PHILLIPS AND JOHN E. STONE

Probing Biomolecular Machines with Graphics Processors

COMPUTER SIMULATION HAS become an integral part of the study of the structure and function of biological molecules. For years, parallel computers have been used to conduct these computationally demanding simulations and to analyze their results. These simulations function as a “computational microscope,” allowing the scientist to observe details of molecular processes too small, fast, or delicate to capture with traditional instruments. Over time, commodity GPUs (graphics processing units) have evolved into massively parallel computing devices, and more recently it has become possible to program them

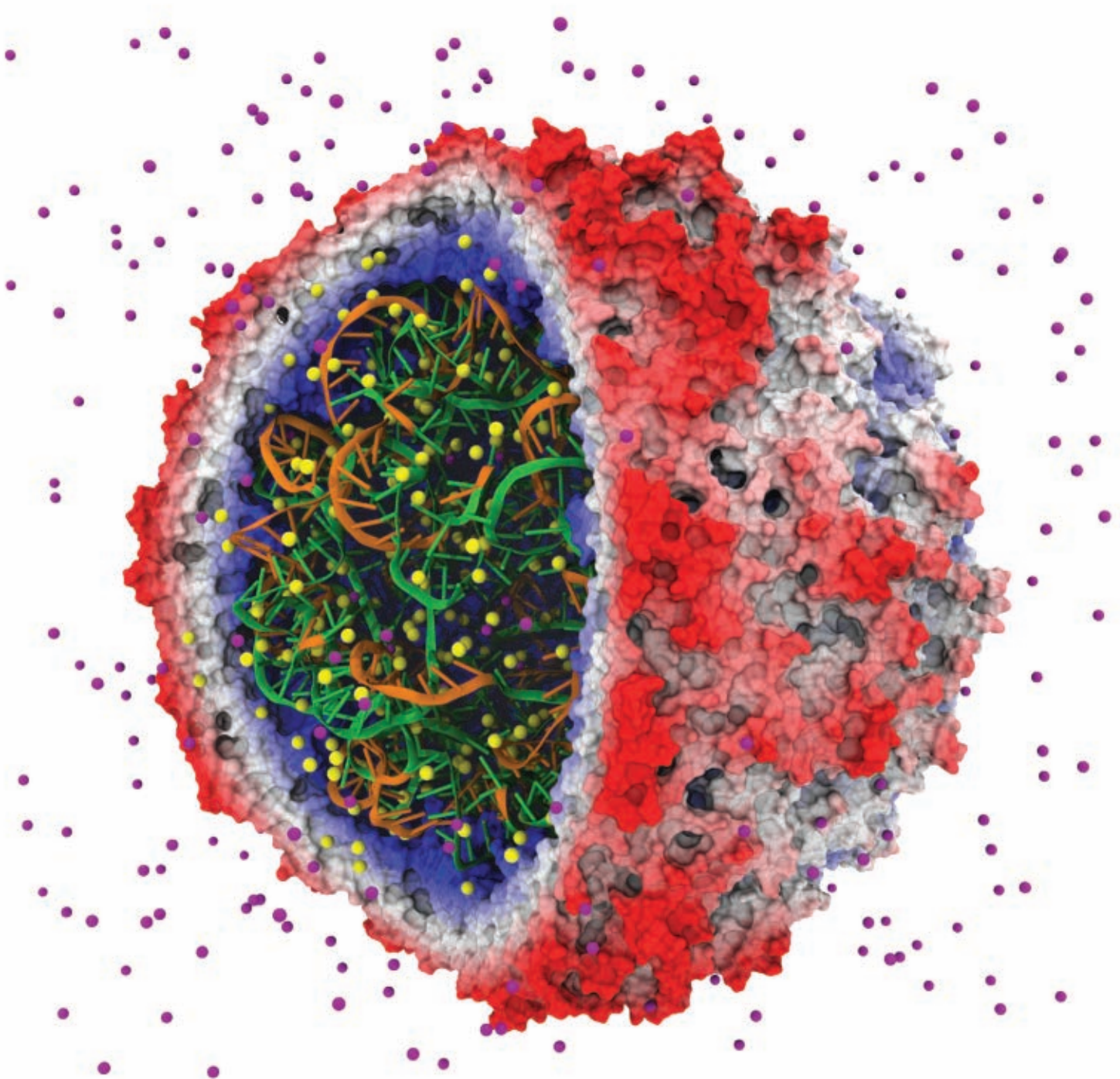
in dialects of the popular C/C++ programming languages.

This has created a tremendous opportunity to employ new simulation and analysis techniques that were previously too computationally demanding to use. In other cases, the computational power provided by GPUs can bring analysis techniques that previously required computation on high-performance computing (HPC) clusters down to desktop computers, making them accessible to application scientists lacking experience with clustering, queuing systems, and the like.

This article is based on our experiences developing software for use by and in cooperation with scientists, often graduate students, with backgrounds in physics, chemistry, and biology. Our programs, NAMD¹⁸ and VMD¹⁰ (Visual Molecular Dynamics), run on computer systems ranging from laptops to supercomputers and are used to model proteins, nucleic acids, and lipids at the atomic level in order to understand how protein structure enables cellular functions such as catalyzing reactions, harvesting sunlight, generating forces, and sculpting membranes (for additional scientific applications, see <http://www.ks.uiuc.edu/>). In 2007 we began working with the Nvidia CUDA (Compute Unified Device Architecture) system for general-purpose graphics processor programming to bring the power of many-core computing to practical scientific applications.²²

Bottom-up Biology

If one were to design a system to safeguard critical data for thousands of years, it would require massive redundancy, self-replication, easily replaceable components, and easily interpreted formats. These are the same challenges faced by our genes, which build around themselves cells, organisms, populations, and entire species for the sole purpose of continuing their own survival. The DNA of every cell contains both data (the amino acid sequences of every protein required for life) and metadata (large stretches of “junk” DNA that inter-



An early success in applying GPUs to biomolecular modeling involved rapid calculation of electrostatic fields used to place ions in simulated structures. The satellite tobacco mosaic virus model contains hundreds of ions (individual atoms shown in yellow and purple) that must be correctly placed so that subsequent simulations yield correct results.²²

act with hormones to control whether a sequence is exposed to the cell's protein expression machinery or hidden deep inside the coils of the chromosome).

The protein sequences of life, once expressed as a one-dimensional chain of amino acids by the ribosome, then fold largely unaided into the unique three-dimensional structures required for their functions. The same protein from different species may have similar structures despite greatly differing sequences. Protein sequences have been selected for the capacity to fold, as random chains of amino acids do

not self-assemble into a unique structure in a reasonable time. Determining the folded structure of a protein based only on its sequence is one of the great challenges in biology, for while DNA sequences are known for entire organisms, protein structures are available only through the painstaking work of crystallographers.

Simply knowing the folded structure of a protein is not enough to understand its function. Many proteins serve a mechanical role of generating, transferring, or diffusing forces and torques. Others control and catalyze chemical

reactions, efficiently harnessing and expending energy obtained from respiration or photosynthesis. While the amino acid chain is woven into a scaffold of helices and sheets, and some components are easily recognized, there are no rigid shafts, hinges, or gears to simplify the analysis.

To observe the dynamic behavior of proteins and larger biomolecular aggregates, we turn to a *computational microscope* in the form of a molecular dynamics simulation. As all proteins are built from a fixed set of amino acids, a model of the forces acting on every atom can


be constructed for any given protein, including bonded, electrostatic, and van der Waals components. Newton's laws of motion then describe the dynamics of the protein over time. When experimental observation is insufficient in resolving power, with the computer we have a perfect view of this simple and limited model.

Is it necessary to simulate every atom in a protein to understand its function? Answering no would require a complete knowledge of the mechanisms involved, in which case the simulation could produce little new insight. Proteins are not designed cleanly from distinct components but are in a sense hacked together from available materials. Rising above the level of atoms necessarily abandons some detail, so it is best to reserve this for the study of aggregate-level phenomena that are otherwise too large or slow to simulate.


Tracking the motions of atoms requires advancing positions and velocities forward through millions or billions of femtosecond (10^{-15} second) time steps to simulate nanoseconds or microseconds of simulated time. Simulation sizes range from a single protein in water with fewer than 100,000 atoms to large multicomponent structures of 1–10 million atoms. Although every atom interacts with every other atom, numerical methods have been developed to calculate long-range interactions for N atoms with order N or $N \log N$ rather than N^2 complexity.

Before a molecular dynamics simulation can begin, a model of the biomolecular system must be assembled in as close to a typical state as possible. First, a crystallographic structure of any proteins must be obtained (pdb.org provides a public archive), missing details filled in by comparison with other structures, and the proteins embedded in a lipid membrane or bound to DNA molecules as appropriate. The entire complex is then surrounded by water molecules and an appropriate concentration of ions, located to minimize their electrostatic energy. The simulation must then be equilibrated at the proper temperature and pressure until the configuration stabilizes.

Processes at the atomic scale are stochastic, driven by random thermal fluctuations across barriers in the energy landscape. Simulations starting from



If one were to design a system to safeguard critical data for thousands of years, it would require massive redundancy, self-replication, easily replaceable components, and easily interpreted formats. These are the same challenges faced by our genes.



nearly identical initial conditions will quickly diverge, but over time their average properties will converge if the possible conformations of the system are sufficiently well sampled. To guarantee that an expected transition is observed, it is often necessary to apply steering forces to the calculation. Analysis is performed, both during the calculation and later, on periodically saved snapshots to measure average properties of the system and to determine which transitions occur with what frequency.

As the late American mathematician R. W. Hamming said, "The purpose of computing is insight, not numbers." Simulations would spark little insight if scientists could not see the biomolecular model in three dimensions on the computer screen, rotate it in space, cut away obstructions, simplify representations, incorporate other data, and observe its behavior to generate hypotheses. Once a mechanism of operation for a protein is proposed, it can be tested by both simulation and experiment, and the details refined. Excellent visual representation is then needed to an equal extent to publicize and explain the discovery to the biomedical community.

Biomedical Users

We have more than a decade of experience guiding the development of the NAMD (<http://www.ks.uiuc.edu/Research/namd/>) and VMD (<http://www.ks.uiuc.edu/Research/vmd/>) programs for the simulation, analysis, and visualization of large biomolecular systems. The community of scientists that we serve numbers in the tens of thousands and circles the globe, ranging from students with only a laptop to leaders of their fields with access to the most powerful supercomputers and graphics workstations. Some are highly experienced in the art of simulation, while many are primarily experimental researchers turning to simulation to help explain their results and guide future work.

The education of the computational scientist is quite different from that of the scientifically oriented computer scientist. Most start out in physics or another mathematically oriented field and learn scientific computing informally from their fellow lab mates and advisors, originally in Fortran 77 and today in Matlab. Although skilled at

solving complex problems, they are seldom taught any software design process or the reasons to prefer one solution to another. Some go for years in this environment before being introduced to revision-control systems, much less automated unit tests.

As software users, scientists are similar to programmers in that they are comfortable adapting examples to suit their needs and working from documentation. The need to record and repeat computations makes graphical interfaces usable primarily for interactive exploration, while batch-oriented input and output files become the primary artifacts of the research process.

One of the great innovations in scientific software has been the incorporation of scripting capabilities, at first rudimentary but eventually in the form of general-purpose languages such as Tcl and Python. The inclusion of scripting in NAMD and VMD has blurred the line between user and developer, exposing a safe and supportive programming language that allows the typical scientist to automate complex tasks and even develop new methods. Since no recompilation is required, the user need not worry about breaking the tested, performance-critical routines implemented in C++. Much new functionality in VMD has been developed by users in the form of script-based plug-ins, and C-based plug-in interfaces have simplified the development of complex molecular structure analysis tools and readers for dozens of molecular file formats.

Scientists are quite capable of developing new scientific and computational approaches to their problems, but it is unreasonable to expect the biomedical community to extend their interest and attention so far as to master the ever-changing landscape of high-performance computing. We seek to provide users of NAMD and VMD with the experience of *practical supercomputing*, in which the skills learned with toy systems running on a laptop remain of use on both the departmental cluster and national supercomputer, and the complexities of the underlying parallel decomposition are hidden. Rather than a fearful and complex instrument, the supercomputer now becomes just another tool to be called upon as the user's work requires.

Given the expense and limited avail-

ability of high-performance computing hardware, we have long sought better options for bringing larger and faster simulations to the scientific masses. The last great advance in this regard was the evolution of commodity-based Linux clusters from cheap PCs on shelves into the dominant platform today. The next advance, *practical acceleration*, will require a commodity technology with strong commercial support, a sustainable performance advantage over several generations, and a programming model that is accessible to the skilled scientific programmer. We believe that this next advance is to be found in 3D graphics accelerators inspired by public demand for visual realism in computer games.

GPU Computing

Biomolecular modelers have always had a need for sophisticated graphics to elucidate the complexities of the large molecular structures commonly studied in structural biology. In 1995, 3D visualization of such molecular structures required desk-side workstations costing tens of thousands of dollars. Gradually, the commodity graphics hardware available for personal computers began to incorporate fixed-function hardware for accelerating 3D rendering. This led to widespread development of 3D games and funded a fast-paced cycle of continuous hardware evolution that has ultimately resulted in the GPUs that

have become ubiquitous in modern computers.

GPU hardware design. Modern GPUs have evolved to a high state of sophistication necessitated by the complex interactive rendering algorithms used by contemporary games and various engineering and scientific visualization software. GPUs are now fully programmable massively parallel computing devices that support standard integer and floating-point arithmetic operations.¹¹ State-of-the-art GPU devices contain more than 240 processing units and are capable of performing up to 1TFLOPS. High-end devices contain multiple gigabytes of high-bandwidth on-board memory complemented by several small on-chip memory systems that can be used as program-managed caches, further amplifying effective memory bandwidth.

GPUs are designed as throughput-oriented devices. Rather than optimizing the performance of a single thread or a small number of threads of execution, GPUs are designed to provide high aggregate performance for tens of thousands of independent computations. This key design choice allows GPUs to spend the vast majority of chip die area (and thus transistors) on arithmetic units rather than on caches. Similarly, GPUs sacrifice the use of independent instruction decoders in favor of SIMD (single-instruction multiple-data) hardware designs wherein groups of



Figure 1. The recently constructed Lincoln GPU cluster at the National Center for Supercomputing Applications contains 1,536 CPU cores, 384 GPUs, 3TB of memory, and achieves an aggregate peak floating-point performance of 47.5TFLOPS.

processing units share an instruction decoder. This design choice maximizes the number of arithmetic units per mm² of chip die area, at the cost of reduced performance whenever branch divergence occurs among threads on the same SIMD unit.

The lack of large caches on GPUs means that a different technique must be used to hide the *hundreds* of clock cycles of latency to off-chip GPU or host memory. This is accomplished by multiplexing many threads of execution onto each physical processing unit, managed by a hardware scheduler that can exchange groups of active and inactive threads as queued memory operations are serviced. In this way, the memory operations of one thread are overlapped with the arithmetic operations of others. Recent GPUs can simultaneously schedule as many as 30,720 threads on an entire GPU. Although it is not necessary to saturate a GPU with the maximum number of independent threads, this provides the best opportunity for latency hiding. The requirement that the GPU be supplied with large quantities of fine-grained data-parallel work is the key factor that determines whether or not an application or algorithm is well

suitable for GPU acceleration.

As a direct result of the large number of processing units, high-bandwidth main memory, and fast on-chip memory systems, GPUs have the potential to significantly outperform traditional CPU architectures significantly on highly data-parallel problems that are well matched to the architectural features of the GPU.

GPU programming. Until recently, the main barrier to using GPUs for scientific computing had been the availability of general-purpose programming tools. Early research efforts such as Brook² and Sh¹³ demonstrated the feasibility of using GPUs for nongraphical calculations. In mid-2007 Nvidia released CUDA,¹⁶ a new GPU programming toolkit that addressed many of the shortcomings of previous efforts and took full advantage of a new generation of compute-capable GPUs. In late 2008 the Khronos Group announced the standardization of OpenCL,¹⁴ a vendor-neutral GPU and accelerator programming interface. AMD, Nvidia, and many other vendors have announced plans to provide OpenCL implementations for their GPUs and CPUs. Some vendors also provide low-level proprietary interfaces that allow

third-party compiler vendors such as RapidMind¹² and the Portland Group to target GPUs more easily. With the major GPU vendors providing officially supported toolkits for GPU computing, the most significant barrier to widespread use has been eliminated.

Although we focus on CUDA, many of the concepts we describe have analogs in OpenCL. A full overview of the CUDA programming model is beyond the scope of this article, but John Nickolls et al. provide an excellent introduction in their article, “Scalable parallel programming with CUDA,” in the March/April 2008 issue of *ACM Queue*.¹⁵ CUDA code is written in C/C++ with extensions to identify functions that are to be compiled for the host, the GPU *device*, or both. Functions intended for execution on the device, known as *kernels*, are written in a dialect of C/C++ matched to the capabilities of the GPU hardware. The key programming interfaces CUDA provides for interacting with a *device* include routines that do the following:

- ▶ Enumerate available devices and their properties
- ▶ Attach to and detach from a device
- ▶ Allocate and deallocate device memory
- ▶ Copy data between host and device memory
- ▶ Launch kernels on the device
- ▶ Check for errors

When launched on the device, the kernel function is instantiated thousands of times in separate *threads* according to a kernel *configuration* that determines the dimensions and number of threads per *block* and blocks per *grid*. The kernel configuration maps the parallel calculations to the device hardware and can be selected at runtime for the specific combination of input data and CUDA device capabilities. During execution, a kernel uses its thread and block indices to select desired calculations and input and output data. Kernels can contain all of the usual control structures such as loops and if/else branches, and they can read and write data to shared device memory or global memory as needed. Thread synchronization primitives provide the means to coordinate memory accesses among threads in the same block, allowing them to operate cooperatively on shared data.

The key challenges involved in de-

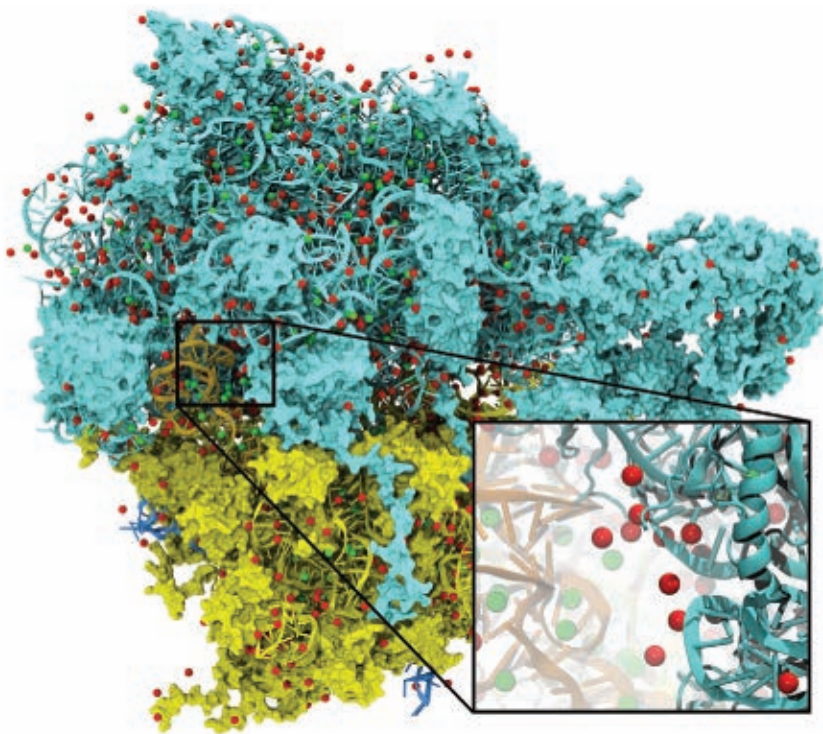



Figure 2. GPU-accelerated electrostatics algorithms enable researchers to place ions appropriately during early stages of model construction. Placement of ions in large structures such as the ribosome shown here previously required the use of HPC clusters for calculation but can now be performed on a GPU-accelerated desktop computer in just a few minutes.

veloping high-performance CUDA kernels revolve around efficient use of several memory systems and exploiting all available data parallelism. Although the GPU provides tremendous computational resources, this capability comes at the cost of limitations in the number of per-thread registers, the size of per-block shared memory, and the size of constant memory. With hundreds of processing units, it is impractical for GPUs to provide a thread-local stack. Local variables that would normally be placed on the stack are instead allocated from the thread's registers, so recursive kernel functions are not supported.


Analyzing applications for GPU acceleration potential. The first step in analyzing an application to determine its suitability for any acceleration technology is to profile the CPU time consumed by its constituent routines on representative test cases. With profiling results in hand, one can determine to what extent Amdahl's Law limits the benefit obtainable by accelerating only a handful of functions in an application. Applications that focus their runtime into a few key algorithms or functions are usually the best candidates for acceleration.

As an example, if profiling shows that an application spends 10% of its runtime in its most time-consuming function, and the remaining runtime is scattered among several tens of unrelated functions of no more than 2% each, such an application would be a difficult target for an acceleration effort, since the best performance increase achievable with moderate effort would be a mere 10%. A much more attractive case would be an application that spends 90% of its execution time running a single algorithm implemented in one or two functions.

Once profiling analysis has identified the subroutines that are worth accelerating, one must evaluate whether they can be reimplemented with data-parallel algorithms. The scale of parallelism required for peak execution efficiency on the GPU is usually on the order of 100,000 independent computations. The GPU provides extremely fine-grain parallelism with hardware support for multiplexing and scheduling massive numbers of threads onto the pool of processing units. This makes it possible for CUDA to extract parallelism at a level of granularity that is orders of magnitude finer than is usually practical with other



One of the most compelling and successful applications for GPU acceleration has been molecular dynamics simulation, which is dominated by N-body atomic force calculation.



parallel-programming paradigms.

GPU-accelerated clusters for HPC. Given the potential for significant acceleration provided by GPUs, there has been a growing interest in incorporating GPUs into large HPC clusters.^{3,6,8,19,21,24} As a result of this interest, Nvidia now makes high-density rack-mounted GPU accelerators specifically designed for use in such clusters. By housing the GPUs in an external case with its own independent power supply, they can be attached to blade or 1U rackmount servers that lack the required power and cooling capacity for GPUs to be installed internally. In addition to increasing performance, GPU accelerated clusters also have the potential to provide better power efficiency than traditional CPU clusters.

In a recent test on the AC GPU cluster (<http://iacat.uiuc.edu/resources/cluster/>) at the National Center for Supercomputing Applications (NCSA, <http://www.ncsa.uiuc.edu/>), a NAMM simulation of STMV (satellite tobacco mosaic virus) measured the increase in performance provided by GPUs, as well as the increase in performance per watt. In a small-scale test on a single node with four CPU cores and four GPUs (HP xw9400 workstation with a Tesla S1070 attached), the four Tesla GPUs provided a factor of 7.1 speedup over four CPU cores by themselves. The GPUs provided a factor of 2.71 increase in the performance per watt relative to computing only on CPU cores. The increases in performance, space efficiency, power, and cooling have led to the construction of large GPU clusters at supercomputer centers such as NCSA and the Tokyo Institute of Technology. The NCSA Lincoln cluster (<http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/Intel64TeslaCluster/TechSummary/>) containing 384 GPUs and 1,536 CPU cores is shown in Figure 1.

GPU Applications

Despite the relatively recent introduction of general-purpose GPU programming toolkits, a variety of biomolecular modeling applications have begun to take advantage of GPUs.


Molecular Dynamics. One of the most compelling and successful applications for GPU acceleration has been molecular dynamics simulation, which is dominated by N-body atomic force calculation. One of the early successes with

the use of GPUs for molecular dynamics was the Folding@Home project^{5,7} where continuing efforts on development of highly optimized GPU algorithms have demonstrated speedups of more than a factor of 100 for a particular class of simulations (for example, protein folding) of very small molecules (5,000 atoms and less). Folding@Home is a distributed computing application deployed on thousands of computers worldwide. GPU acceleration has helped make it the most powerful distributed computing cluster in the world, with GPU-based clients providing the dominant computational power (<http://fah-web.stanford.edu/cgi-bin/main.py?type=osstats>).


HOOMD (Highly Optimized Object-oriented Molecular Dynamics), a recently developed package specializing in molecular dynamics simulations of polymer systems, is unique in that it was designed from the ground up for execution on GPUs.¹ Though in its infancy, HOOMD is being used for a variety of coarse-grain particle simulations and achieves speedups of up to a factor of 30 through the use of GPU-specific algorithms and approaches.

NAMD¹⁸ is another early success in the use of GPUs for molecular dynamics.^{17,19,22} It is a highly scalable parallel program that targets all-atom simulations of large biomolecular systems containing hundreds of thousands to many millions of atoms. Because of the large number of processor-hours consumed by NAMD users on supercomputers around the world, we investigated a variety of acceleration options and have used CUDA to accelerate the calculation of nonbonded forces using GPUs. CUDA acceleration mixes well with task-based parallelism, allowing NAMD to run on clusters with multiple GPUs per node. Using the CUDA streaming API for asynchronous memory transfers and kernel invocations to overlap GPU computation with communication and other work done by the CPU yields speedups of up to a factor of nine times faster than CPU-only runs.¹⁹

At every iteration NAMD must calculate the short-range interaction forces between all pairs of atoms within a cutoff distance. By partitioning space into patches slightly larger than the cutoff distance, we can ensure that all of an atom's interactions are with atoms in the same or neighboring cubes. Each block



We expect GPUs to maintain their current factor-of-10 advantage in peak performance relative to CPUs, while their obtained performance advantage for well-suited problems continues to grow.



in our GPU implementation is responsible for the forces on the atoms in a single patch due to the atoms in either the same or a neighboring patch. The kernel copies the atoms from the first patch in the assigned pair to shared memory and keeps the atoms from the second patch in registers. All threads iterate in unison over the atoms in shared memory, accumulating forces for the atoms in registers only. The accumulated forces for each atom are then written to global memory. Since the forces between a pair of atoms are equal and opposite, the number of force calculations could be cut in half, but the extra coordination required to sum forces on the atoms in shared memory outweighs any savings.

NAMD uses constant memory to store a compressed lookup table of bonded atom pairs for which the standard short-range interaction is not valid. This is efficient because the table fits entirely in the constant cache and is referenced for only a small fraction of pairs. The texture unit, a specialized feature of GPU hardware designed for rapidly mapping images onto surfaces, is used to interpolate the short-range interaction from an array of values that fits entirely in the texture cache. The dedicated hardware of the texture unit can return a separate interpolated value for every thread that requires it faster than the potential function could be evaluated analytically.

Building, visualizing, and analyzing molecular models. Another area where GPUs show great promise is in accelerating many of the most computationally intensive tasks involved in preparing models for simulation, visualizing them, and analyzing simulation results (<http://www.ks.uiuc.edu/Research/gpu/>).

One of the critical tasks in the simulation of viruses and other structures containing nucleic acids is the placement of ions to reproduce natural biological conditions. The correct placement of ions (see Figure 2) requires knowledge of the electrostatic field in the volume of space occupied by the simulated system. Ions are placed by evaluating the electrostatic potential on a regularly spaced lattice and inserting ions at the minima in the electrostatic field, updating the field with the potential contribution of the newly added ion, and repeating the insertion process as necessary. Of these steps, the initial electrostatic

field calculation dominates runtime and is therefore the part best suited for GPU acceleration.


A simple quadratic-time direct Coulomb summation algorithm computes the electrostatic field at each lattice point by summing the potential contributions for all atoms. When implemented optimally, taking advantage of fast reciprocal square-root instructions and making extensive use of near-register-speed on-chip memories, a GPU direct summation algorithm can outperform a CPU core by a factor of 44 or more.^{17,22} By employing a so-called “short-range cutoff” distance beyond which contributions are ignored, the algorithm can achieve linear time complexity while still outperforming a CPU core by a factor of 26 or more.²⁰ To take into account the long-range electrostatic contributions from distant atoms, the short-range cutoff algorithm must be combined with a long-range contribution. A GPU implementation of the linear-time multilevel summation method, combining both the short-range and long-range contributions, has achieved speedups in excess of a factor of 20 compared with a CPU core.⁹

GPU acceleration techniques have proven successful for an increasingly diverse range of other biomolecular applications, including quantum chemistry simulation (<http://mtzweb.stanford.edu/research/gpu/>) and visualization,^{23,25} calculation of solvent-accessible surface area,⁴ and others (<http://www.hicomb.org/proceedings.html>). It seems likely that GPUs and other many-core processors will find even greater applicability in the future.

Looking Forward

Both CPU and GPU manufacturers now exploit fabrication technology improvements by adding cores to their chips as feature sizes shrink. This trend is anticipated in GPU programming systems, for which many-core computing is the norm, whereas CPU programming is still largely based on a model of serial execution with limited support for tightly coupled on-die parallelism. We therefore expect GPUs to maintain their current factor-of-10 advantage in peak performance relative to CPUs, while their obtained performance advantage for well-suited problems continues to grow. We further note that GPUs have

maintained this performance lead despite historically lagging CPUs by a generation in fabrication technology, a handicap that may fade with growing demand.

The great benefits of GPU acceleration and other computer performance increases for biomedical science will come in three areas. The first is doing the same calculations as today, but faster and more conveniently, providing results over lunch rather than overnight to allow hypotheses to be tested while they are fresh in the mind. The second is in enabling new types of calculations that are prohibitively slow or expensive today, such as evaluating properties throughout an entire simulation rather than for a few static structures. The third and greatest is in greatly expanding the user community for high-end biomedical computation to include all experimental researchers around the world, for there is much work to be done and we are just now beginning to uncover the wonders of life at the atomic scale. 

Related articles on queue.acm.org

GPUs: A Closer Look

Kayvon Fatahalian and Mike Houston

<http://queue.acm.org/detail.cfm?id=1365498>

Scalable Parallel Programming with CUDA

John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron

<http://queue.acm.org/detail.cfm?id=1365500>

Future Graphics Architectures

William Mark

<http://queue.acm.org/detail.cfm?id=1365501>

References

- Anderson, J.A., Lorenz, C.D., and Travesset, A. General-purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Chemical Physics* 227, 10 (2008), 5342–5359.
- Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., and Hanrahan, P. Brook for GPUs: Stream computing on graphics hardware. In *Proceedings of 2004 ACM SIGGRAPH*. ACM, NY, 777–786.
- Davis, D., Lucas, R., Wagenbreth, G., Tran, J., and Moore, J. A GPU-enhanced cluster for accelerated FMS. In *Proceedings of the 2007 DoD High-performance Computing Modernization Program Users Group Conference*. IEEE Computer Society, 305–309.
- Dynerman, D., Butzlaff, E., and Mitchell, J.C. CUSA and CUDE: GPU-accelerated methods for estimating solvent accessible surface area and desolvation. *Journal of Computational Biology* 16, 4 (2009), 523–537.
- Elsen, E., Vishal, V., Houston, M., Pande, V., Hanrahan, P., and Darve, E. N-body simulations on GPUs. Technical Report, Stanford University (June 2007); <http://arxiv.org/abs/0706.3060>.
- Fan, Z., Qiu, F., Kaufman, A., and Yoakum-Stover, S. GPU cluster for high-performance computing. In *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*. IEEE Computer Society, 47.
- Friedrichs, M.S., Eastman, P., Vaidyanathan, V., Houston, M., Legrand, S., Beberg, A.L., Ensign, D.L., Bruns, C.M., and Pande, V.S. Accelerating molecular dynamic

- simulation on graphics processing units. *Journal of Computational Chemistry* 30, 6 (2009), 864–872.
- Göddeke, D., Strzodka, R., Mohd-Yusof, J., McCormick, P., Buijssen, S.H.M., Grajewski, M., and Turek, S. Exploring weak scalability for FEM calculations on a GPU-enhanced cluster. *Parallel Computing* 33, 10–11 (2007), 685–699.
- Hardy, D.J., Stone, J.E., and Schulten, K. Multilevel summation of electrostatic potentials using graphics processing units. *Parallel Computing* 35 (2009), 164–177.
- Humphrey, W., Dalke, A., and Schulten, K. VMD: Visual molecular dynamics. *Journal of Molecular Graphics* 14 (1996), 33–38.
- Lindholm, E., Nickolls, J., Oberman, S., and Montrym, J. Nvidia Tesla: A unified graphics and computing architecture. *IEEE Micro* 28, 2 (2008), 39–55.
- McCool, M. Data-parallel programming on the Cell BE and the GPU using the RapidMind development platform. In *Proceedings of GSPx Multicore Applications Conference* (Oct.–Nov. 2006).
- McCool, M., Du Toit, S., Popa, T., Chan, B., and Moule, K. Shader algebra. *ACM Transactions on Graphics* 23, 3 (2004), 787–795.
- Munshi, A. OpenCL specification version 1.0 (2008); <http://www.khronos.org/registry/cl/>.
- Nickolls, J., Buck, I., Garland, M., and Skadron, K. Scalable parallel programming with CUDA. *ACM Queue* 6, 2 (2008): 40–53.
- Nvidia CUDA (Compute Unified Device Architecture) Programming Guide. Nvidia, Santa Clara, CA 2007.
- Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., and Phillips, J.C. GPU computing. In *Proceedings of IEEE 96* (2008), 879–899.
- Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kale, L., and Schulten, K. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry* 26 (2005), 1781–1802.
- Phillips, J.C., Stone, J.E., and Schulten, K. Adapting a message-driven parallel application to GPU-accelerated clusters. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. IEEE Press, 2008.
- Rodrigues, C.I., Hardy, D.J., Stone, J.E., Schulten, K., and Hwu, W.W. GPU acceleration of cutoff pair potentials for molecular modeling applications. In *Proceedings of the 2008 ACM Conference on Computing Frontiers* (2008), 273–282.
- Showerman, M., Enos, J., Pant, A., Kindratenko, V., Steffen, C., Pennington, R., and Hwu, W. QP: A heterogeneous multi-accelerator cluster. In *Proceedings of the 10th LCI International Conference on High-performance Clustered Computing* (Mar. 2009).
- Stone, J.E., Phillips, J.C., Freddolino, P.L., Hardy, D.J., Trabuco, L.G., and Schulten, K. Accelerating molecular modeling applications with graphics processors. *Journal of Computational Chemistry* 28 (2007), 2618–2640.
- Stone, J.E., Saam, J., Hardy, D.J., Vandivort, K.L., Hwu, W.W., and Schulten, K. High-performance computation and interactive display of molecular orbitals on GPUs and multicore CPUs. In *Proceedings of the 2nd Workshop on General-purpose Processing on the Graphics Processing Units*. ACM International Conference Proceeding Series 383 (2009), 9–18.
- Takizawa, H. and Kobayashi, H. Hierarchical parallel processing of large-scale data clustering on a PC cluster with GPU coprocessing. *Journal of Supercomputing* 36, 3 (2006), 219–234.
- Ufimtsev, I.S. and Martinez, T.J. Quantum chemistry on graphical processing units. Strategies for two-electron integral evaluation. *Journal of Chemical Theory and Computation* 4, 2 (2008), 222–231.

James Phillips (jim@ks.uiuc.edu) is a senior research programmer in the Theoretical and Computational Biophysics Group at the Beckman Institute for Advanced Science and Technology at the University of Illinois at Urbana-Champaign. For the last decade Phillips has been the lead developer of NAMD, the highly scalable parallel molecular dynamics program for which he received a Gordon Bell Award in 2002.

John Stone (johns@ks.uiuc.edu) is a senior research programmer in the Theoretical and Computational Biophysics Group at the Beckman Institute for Advanced Science and Technology at the University of Illinois at Urbana-Champaign. He is the lead developer of the VMD molecular visualization and analysis program.

© 2009 ACM 0001-0782/09/1000 \$10.00

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

The biosciences need an image format capable of high performance and long-term maintenance. Is HDF5 the answer?

BY MATTHEW T. DOUGHERTY, MICHAEL J. FOLK, EREZ ZADOK,
HERBERT J. BERNSTEIN, FRANCES C. BERNSTEIN,
KEVIN W. ELICEIRI, WERNER BENGER, CHRISTOPH BEST

Unifying Biological Image Formats with HDF5

THE BIOLOGICAL SCIENCES need a generic image format suitable for long-term storage and capable of handling very large images. Images convey profound ideas in biology, bridging across disciplines. Digital imagery began 50 years ago as an obscure technical phenomenon. Now it is an indispensable computational tool. It has produced a variety of incompatible image file formats, most of which are already obsolete.

Several factors are forcing the obsolescence: rapid increases in the number of pixels per image;

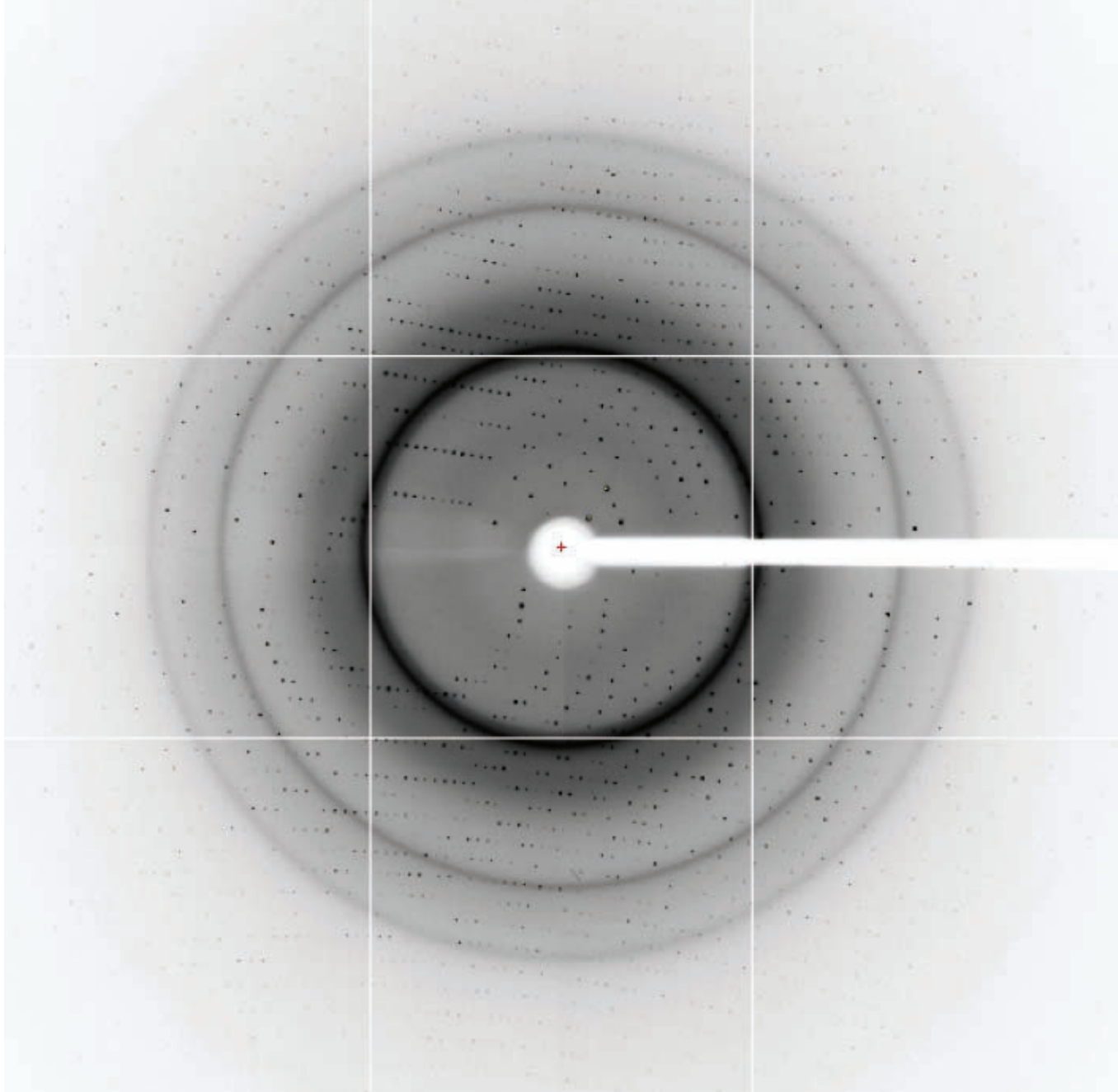
acceleration in the rate at which images are produced; changes in image designs to cope with new scientific instrumentation and concepts; collaborative requirements for interoperability of images collected in different labs on different instruments; and research metadata dictionaries that must support frequent and rapid extensions. These problems are not unique to the biosciences. Lack of image standardization is a source of delay, confusion, and errors for many scientific disciplines.

There is a need to bridge biological and scientific disciplines with an image framework capable of high computational performance and interoperability. Suitable for archiving, such a framework must be able to maintain images far into the future. Some frameworks represent partial solutions: a few, such as XML, are primarily suited for interchanging metadata; others, such as CIF (Crystallographic Information Framework),² are primarily suited for the database structures needed for crystallographic data mining; still others, such as DICOM (Digital Imaging and Communications in Medicine),³ are primarily suited for the domain of clinical medical imaging.

What is needed is a common image framework able to interoperate with all of these disciplines, while providing high computational performance. HDF (Hierarchical Data Format)⁶ is such a framework, presenting a historic opportunity to establish a coin of the realm by coordinating the imagery of many biological communities. Overcoming the digital confusion of incoherent bio-imaging formats will result in better science and wider accessibility to knowledge.

Semantics: Formats, Frameworks, and Images

Digital imagery and computer technology serve a number of diverse biological communities with terminology differences that can result in very different perspectives. Consider the word *format*. To the data-storage community the hard-drive format will play a ma-



An x-ray diffraction image taken by Michael Soltis of LSAC on SSRL BL9-2 using an ADSC Q315 detector (SN901).

major role in the computer performance of a community's image format, and to some extent, they are inseparable. A format can describe a standard, a framework, or a software tool; and formats can exist within other formats.

Image is also a term with several uses. It may refer to transient electrical signals in a CCD (charge-coupled device), a passive dataset on a storage device, a location in RAM, or a data structure written in source code. Another example is *framework*. An image framework might implement an image standard, resulting in image files created by a software-imaging tool. The framework, the standard, the files, and the tool, as in the case of HDF,⁶ may be so interrelated that they represent dif-

ferent facets of the same specification. Because these terms are so ubiquitous and varied due to perspective, we shall use them interchangeably, with the emphasis on the storage and management of pixels throughout their lifetime, from acquisition through archiving.

Hierarchical Data Format Version 5

HDF5 is a generic scientific data format with supporting software. Introduced in 1998, it is the successor to the 1988 version, HDF4. NCSA (National Center for Supercomputing Applications) developed both formats for high-performance management of large heterogeneous scientific data. Designed to move data efficiently between secondary storage and memory,

HDF5 translates across a variety of computing architectures. Through support from NASA (National Aeronautics and Space Administration), NSF (National Science Foundation), DOE (Department of Energy), and others, HDF5 continues to support international research. The HDF Group, a nonprofit spin-off from the University of Illinois, manages HDF5, reinforcing the long-term business commitment to maintain the format for purposes of archiving and performance.

Because an HDF5 file can contain almost any collection of data entities in a single file, it has become the format of choice for organizing heterogeneous collections consisting of very large and complex datasets. HDF5 is

used for some of the largest scientific data collections, such as the NASA Earth Observation System's petabyte repository of earth science data. In 2008, netCDF (network Common Data Form)¹⁰ began using HDF5, bringing in the atmospheric and climate communities. HDF5 also supports the neutron and X-ray communities for instrument data acquisition. Recently, MATLAB implemented HDF5 as its primary storage format. Soon HDF5 will formally be adopted by the International Organization for Standardization (ISO), as part of specification 10303 (STEP, Standard for the Exchange of Product model data). Also of note is the creation of BioHDF¹ for organizing rapidly growing genomics data volumes.

The HDF Group's digital preservation efforts make HDF5 well suited for archival tasks. Specifically their involvement with NARA (National Archives and Records Administration), their familiarity with the ISO standard *Reference Model for an Open Archival Information System (OAIS)*,¹³ and the HDF5 implementation of the *Metadata Encoding and Transmission Standard (METS)*⁸ developed by the Digital Library Federation and maintained by the Library of Congress.

Technical Features of HDF5

An HDF5 file is a data container, similar to a file system. Within it, user communities or software applications define their organization of data objects. The basic HDF5 data model is simple, yet extremely versatile in terms of the scope of data that it can store. It contains two primary objects: *groups*, which provide the organizing structures, and *datasets*, which are the basic storage structures. HDF5 groups and datasets may also have *attributes* attached, a third type of data object consisting of small textual or numeric metadata defined by user applications.

An HDF5 dataset is a uniform multidimensional array of elements. The elements might be common data types (for example, integers, floating-point numbers, text strings), *n*-dimensional memory chunks, or user-defined compound data structures consisting of floating-point vectors or an arbitrary bit-length encoding (for example, 97-

bit floating-point number). An HDF5 group is similar to a directory, or folder, in a computer file system. An HDF5 group contains links to groups or datasets, together with supporting meta-data. The organization of an HDF5 file is a directed graph structure in which groups and datasets are nodes, and links are edges. Although the term *HDF* implies a hierarchical structuring, its topology allows for other arrangements such as meshes or rings.

HDF5 is a completely portable file format with no limit on the number or size of data objects in the collection. During I/O operations, HDF5 automatically takes care of data-type differences, such as byte ordering and data-type size. Its software library runs on Linux, Windows, Mac, and most other operating systems and architectures, from laptops to massively parallel systems. HDF5 implements a high-level API with C, C++, Fortran 90, Python, and Java interfaces. It includes many tools for manipulating and viewing HDF5 data, and a wide variety of third-party applications and tools are available.

The design of the HDF5 software provides a rich set of integrated performance features that allow for access-time and storage-space optimizations. For example, it supports efficient extraction of subsets of data, multiscale representation of images, generic dimensionality of datasets, parallel I/O, tiling (2D), bricking (3D), chunking (nD), regional compression, and the flexible management of user metadata that is interoperable with XML. HDF5 transparently manages byte ordering in its detection of hardware. Its software extensibility allows users to insert custom software "filters" between secondary storage and memory; such filters allow for encryption, compression, or image processing. The HDF5 data model, file format, API, library, and tools are open source and distributed without charge.

MEDSBIO

X-ray crystallographers formed MEDSBIO (Consortium for Management of Experimental Data in Structural Biology)⁷ in 2005 to coordinate various research interests. Later the electron⁴ and optical¹⁴ microscopy communities began attending. During the past 10 years, each community considered

HDF5 as a framework to create their independent next-generation image file formats. In the case of the NeXus,¹¹ the format developed by the neutron and synchrotron facilities, HDF5 has been the operational infrastructure in its design since 1998.

Ongoing discussions by MEDSBIO have led to the realization that common computational storage algorithms and formats for managing images would tremendously benefit the X-ray, neutron, electron, and optical acquisition communities. Significantly, the entire biological community would benefit from coherent imagery and better-integrated data models. With four bio-imaging communities concluding that HDF5 is essential to their future image strategy, this is a rare opportunity to establish comprehensive agreements on a common scientific image standard across biological disciplines.

Concerns Identified

The following deficiencies impede the immediate and long-term usefulness of digital images:

- ▶ *The increase in pixels caused by improving digital acquisition resolutions, faster acquisition speeds, and expanding user expectations for "more and faster" is unmanageable.* The solution requires technical analysis of the computational infrastructure. The image designer must analyze the context of computer hardware, application software, and the operating-system interactions. This is a moving target monitored over a period of decades. For example, today's biologists use computers having 2GB–16GB of RAM. What method should be used to access a four-dimensional, 1TB image having 30 hyperspectral values per pixel? Virtually all of the current biological image formats organize pixels as 2D XY image planes. A visualization program may require the entire set of pixels read into RAM or virtual memory. This, coupled with poor performance of the mass storage relating to random disk seeks, paging, and memory swaps, effectively makes the image unusable. For a very large image, it is desirable to store it in multiple resolutions (multiscale) allowing interactive access to regions of interest. Visualization software may intensively compute these intermediate data resolutions, later discarded upon

exit from the software.

► *The inflexibility of current biological image file designs prevents them from adapting to future modalities and dimensionality.* Rapid advances in biological instrumentation and computational analysis are leading to complex imagery involving novel physical and statistical pixel specifications.

► *The inability to assemble different communities' imagery into an overarching image model allows for ambiguity in the analysis.* The integration of various coordinate systems can be an impassable obstacle if not properly organized. There is an increasing need to correlate images of different modalities in order to observe spatial continuity from millimeter to angstrom resolutions.

► *The non-archival quality of images undermines their long-term value.* The current designs usually do not provide basic archival features recommended by the Digital Library Federation, nor do they address issues of provenance. Frequently, the documentation of a community image format is incomplete, outdated, or unavailable, thus eroding the ability to interpret the digital artifact properly.

Consensus

It would be desirable to adopt an existing scientific, medical, or computer image format, and simply benefit from the consequences. All image formats have their strengths and weaknesses. They tend to fall into two categories: generic and specialized formats. Generic image formats usually have fixed dimensionality or pixel design. For example, MPEG2⁹ is suitable for many applications as long as it is 2D spatial plus 1D temporal using red-green-blue modality that is lossy compressed for the physiological response of the eye. Alternatively, the specialized image formats suffer the difficulties of the image formats we are already using. For example, DICOM³ (medical imaging standard) and FITS⁵ (astronomical imaging standard,) store their pixels as 2D slices, although DICOM does incorporate MPEG2 for video-based imagery.

The ability to tile (2D), brick (3D), or chunk (nD) is required to access very large images. Although this is conceptually simple, the software is not, and must be tested carefully or risk that

subsequent datasets be corrupted. That risk would be unacceptable for operational software used in data repositories and research. This function and its certification testing are critical features of HDF software that are not readily available in any other format.

Common Objectives

The objectives of these acquisition communities are identical, requiring performance, interoperability, and archiving. There is a real need for the different bio-imaging communities to coordinate within the same HDF5 data file by using identical high-performance methods to manage pixels; avoiding namespace collisions between the biological communities; and adopting the same archival best practices. All of these would benefit downstream communities such as visualization developers and global repositories.

Performance. The design of an image file format and the subsequent organization of stored pixels determine the performance of computation because of various hardware and software datapath bottlenecks. For example, many specialized biological image formats use simple 2D pixel organizations, frequently without the benefit of compression. These 2D pixel organizations are ill suited for very large 3D images such as electron tomograms or 5D optical images. Those bio-imaging files have sizes that are orders of magnitude larger than the RAM of computers. Worse, widening gaps have formed between CPU/memory speeds, persistent storage speeds, and network speeds. These gaps lead to significant delays in processing massive data sets. Any file format for massive data has to account for the complex behavior of software layers, all the way from the application, through middleware, down to operating-systems device drivers. A generic *n*-dimensional multimodal image format will require new instantiation and infrastructure to implement new types of data buffers and caches to scale large datasets into much smaller RAM; much of this has been resolved within HDF5.

Interoperability. Historically the acquisition communities have defined custom image formats. Downstream communities, such as visualization

and modeling, attempt to implement these formats, forcing the communities to confront design deficiencies. Basic image metadata definitions such as rank, dimension, and modality must be explicitly defined so the downstream communities can easily participate. Different research communities must be able to append new types of metadata to the image, enhancing the imagery as it progresses through the pipeline. Ongoing advances in the acquisition communities will continue to produce new and significant image modalities that feed this image pipeline. Enabling downstream users easily to access pixels and append their community metadata supports interoperability, ultimately leading to fundamental breakthroughs in biology. This is not to suggest that different communities' metadata can be or should be uniformly defined as a single biological metadata schema and ontology in order to achieve an effective image format.

Archiving. Scientific images have a general lack of archival design features. As the sophistication of bio-imagery improves, the demand for the placement of this imagery into long-term global repositories will be greater. This is being done by the Electron Microscopy Databank⁴ in joint development by the National Center for Macromolecular Imaging, the RCSB (Research Collaboratory for Structural Bioinformatics) at Rutgers University, and the European Bioinformatics Institute. Efforts such as the Open Microscopy Environment¹⁴ are also developing bio-image informatics tools for lab-based data sharing and data mining of biological images that also are requiring practical image formats for long-term storage and retrieval. Because of the evolving complexity of bio-imagery and the need to subscribe to archival best practices, an archive-ready image format must be self-describing. That is, there must be sufficient infrastructure within the image file design to properly document its content, context, and structure of the pixels and related community metadata, thereby minimizing the reliance on external documentation for interpretation.

The Inertia of Legacy Software

Implementing a new unified image

format supporting legacy software across the biological disciplines is a Gordian knot. Convincing software developers to make this a high priority is a difficult proposition. Implementation occurring across hundreds of legacy packages and flawlessly fielded in thousands of laboratories is not a trivial task. Ideally, presenting images simultaneously in their legacy formats and in a new advanced format would mitigate the technical, social, and logistical obstacles. However, this must be accomplished without duplicating the pixels in secondary storage.

One proposal is to mount an HDF5 file as a VFS (virtual file system) so that HDF5 groups become directories and HDF5 datasets become regular files. Such a VFS using FUSE (Filesystem-in-User-Space) would execute simultaneously across the user-process space and the operating system space. This hyperspace would manage all HDF-VFS file activity by interpreting, intercepting, and dynamically rearranging legacy image files. A single virtual file presented by the VFS could be composed of several concatenated HDF5 datasets, such as a metadata header dataset and a pixel dataset. Such a VFS file could have multiple simultaneous filenames and legacy formats depending on the virtual folder name that contains it, or the software application attempting to open it.

The design and function of an HDF-VFS has several possibilities. First, non-HDF5 application software could interact transparently with HDF5 files. PDF files, spreadsheets, and MPEGs would be written and read as routine file-system byte streams. Second, this VFS, when combined with transparent on-the-fly compression, would act as an operationally usable compressed tarball. Third, design the VFS with unique features such as interpreting incoming files as image files. Community-based legacy image format filters would rearrange legacy image files. For example, the pixels would be stored as HDF5 datasets in the appropriate dimensionality and modality, and the related metadata would be stored as a separate HDF5 1D byte dataset. When legacy application software opens the legacy image file, the virtual file is dynamically recombined and presented by the VFS to the legacy software in the

same byte order as defined by the legacy image format. The fourth possibility is to endow the VFS with archival and performance analysis tools that could transparently provide those services to legacy application software.

Recommendations

To achieve the goal of an exemplary image design having wide, long-term support, we offer the following recommendations to be considered through a formal standards process:

1. Permit and encourage scientific communities to continually to evolve their own image designs. They know the demands of their disciplines best. Implementing community image formats through HDF5 provides these communities flexible routes to a common image model.

2. Adopt the archival community's recommendations on archive-ready datasets. Engaging the digital preservation community from the onset, rather than as an afterthought, will produce better long-term image designs.

3. Establish a common image model. The specification must be conceptually simple and should merely distinguish the image's pixels from the various metadata. The storage of pixels should be in an appropriate dimensional dataset. The encapsulation of community metadata should be in 1D byte datasets or attributes.

4. The majority of the metadata is uniquely specific to the biological community that designs it. The use of binary or XML is an internal concern of the community creating the image design; however, universal image metadata will overlap across disciplines, such as rank, dimensionality, and pixel modality. Common image nomenclature should be defined to bridge metadata namespace conversions to legacy formats.

5. Use RDF (Resource Description Framework)¹⁵ as the primary mechanism to manage the association of pixel datasets and the community metadata. A Subject-Predicate-Object-Time tuple stored as a dataset can benefit from HDF5's B-tree search features. Such an arrangement provides useful time stamps for provenance and generic logging for administration and performance testing. The definition of RDF predicates and objects should

follow the extensible design strategy used in the organization of NFS (Network File System) version 4 protocol metadata.¹²

6. In some circumstances it will be desirable to define adjuncts to the common image model. An example is MPEG video, where the standardized compression is the overriding reason to store the data as a 1D byte stream rather than decompressing it into the standard image model as a 3D YCbCr pixel dataset. Proprietary image format is another type of adjunct requiring 1D byte encapsulation rather than translating it into the common image model. In this scenario, images are merely flagged as such and routine archiving methods applied.

7. Provide a comprehensively tested software API in lockstep with the image model. Lack of a common API requires each scientific group to develop and test the software tools from scratch or borrow them from others, resulting in not only increased cost for each group, but also increased likelihood of errors and inconsistencies among implementations.

8. Implement HDF5 as a virtual file system. HDF-VFS could interpret incoming legacy image file formats by storing them as pixel datasets and encapsulated metadata. HDF-VFS could also present such a combination of HDF datasets as a single legacy-format image file, byte-stream identical. Such a file system could allow user legacy applications to access and interact with the images through standard file I/O calls, obviating the requirement and burden of legacy software to include, compile, and link HDF5 API libraries in order to access images. The duality of presenting an image as a file and an HDF5 dataset offers a number of intriguing possibilities for managing images and non-image datasets such as spreadsheets or PDF files, or managing provenance without changes to legacy application software.

9. Make the image specification and software API freely accessible and available without charge. Preferably, such software should be available under an open source license that allows a community of software developers to contribute to its development. Charging the individual biological imaging communities and laboratories adds

financial complexity to the pursuit of scientific efforts that are frequently underfunded.

10. Establish methods for verification and performance testing. A critical requirement is the ability to determine compliance. Not having compliance testing significantly weakens the archival value by undermining the reliability and integrity of the image data. Performance testing using prototypical test cases assists in the design process by flagging proposed community image design that will have severe performance problems. Defining baseline test cases will quickly identify software problems in the API.

11. Establish ongoing administrative support. Formal design processes can take considerable time to complete, but some needs—such as technical support, consultation, publishing technical documentation, and managing registration of community image designs—require immediate attention. Establishing a mechanism for imaging communities to register their HDF5 root level groups as community specific data domains will provide an essential cornerstone for image design and avoid namespace collisions with other imaging communities.

12. Examine how other formal standards have evolved. Employ the successful strategies and avoid the pitfalls. Developing strategies and alliances with these standards groups will further strengthen the design and adoption of a scientific image standard.

13. Establishing the correct forum is crucial and will require the guidance of a professional standards organization—or organizations—that perceives the development of such an image standard as part of its mission to serve the public and its membership. Broad consensus and commitment by the scientific, governmental, business, and professional communities is the best and perhaps only way to accomplish this.

Summary


Out of necessity, bioscientists are independently assessing and implementing HDF5, but no overarching group is responsible for establishing a comprehensive bio-imaging format, and there are few best practices to rely on. Thus, there is a real possibility that biolo-


gists will continue with incompatible methods for solving similar problems, such as not having a common image model.

The failure to establish a scalable n -dimensional scientific image standard that is efficient, interoperable, and archival will result in a less-than-optimal research environment and a less-certain future capability for image repositories. The strategic danger of not having a comprehensive scientific image storage framework is the massive generation of unsustainable bio-images. Subsequently, the long-term risks and costs of comfortable inaction will likely be enormous and irreversible.

The challenge for the biosciences is to establish a world-class imaging specification that will endow these indispensable and nonreproducible observations with long-term maintenance and high-performance computational access. The issue is not whether the biosciences will adopt HDF5 as a useful imaging framework—that is already happening—but whether it is time to gather the many separate pieces of the currently highly fragmented patchwork of biological image formats and place them under HDF5 as a common framework. This is the time to unify the imagery of biology, and we encourage readers to contact the authors with their views.

Acknowledgments

This work was funded by the National Center for Research Resources (P41-RR-02250), National Institute of General Medical Sciences (5R01GM079429, Department of Energy (ER64212-1027708-0011962), National Science Foundation (DBI-0610407, CCF-0621463), National Institutes of Health (1R13RR023192-01A1, R03EB008516), The HDF Group R&D Fund, Center for Computation and Technology at Louisiana State University, Louisiana Information Technology Initiative, and NSF/EPS-CoR (EPS-0701491, CyberTools). 

 Related articles
on queue.acm.org

[Catching disk latency in the act](http://queue.acm.org/detail.cfm?id=1483106)

<http://queue.acm.org/detail.cfm?id=1483106>

Better Scripts, Better Games

<http://queue.acm.org/detail.cfm?id=1483106>

Concurrency's Shysters

http://blogs.sun.com/bmc/entry/concurrency_s_shysters

References

1. BioHDF; <http://www.geospiza.com/research/biohdf/>.
2. Crystallographic Information Framework. International Union of Crystallography; <http://www.iucr.org/resources/cif/>.
3. DICOM (Digital Imaging and Communications in Medicine); <http://medical.nema.org>.
4. EMDB (Electron Microscopy Data Bank); <http://emdatbank.org/>.
5. FITS (Flexible Image Transport System); <http://fits.gsfc.nasa.gov/>.
6. HDF (Hierarchical Data Format); <http://www.hdfgroup.org>.
7. MEDSPIO (Consortium for Management of Experimental Data in Structural Biology); <http://www.medsbio.org>.
8. METS (Metadata Encoding and Transmission Standard); <http://www.loc.gov/standards/mets/>.
9. MPEG (Moving Picture Experts Group); <http://www.chiariglione.org/mpeg/>.
10. netCDF (network Common Data Form); <http://www.unidata.ucar.edu/software/netcdf/>.
11. NeXus (neutron, x-ray and muon science); <http://www.nexusformat.org>.
12. NFS (Network File System); <http://www.ietf.org/rfc/rfc3530.txt>.
13. OAI (Open Archival Information System); <http://nsl.gsfc.nasa.gov/isoas/overview.html>.
14. OME (Open Microscopy Environment); <http://www.openmicroscopy.org/>.
15. RDF (Resource Description Framework); <http://www.w3.org/RDF/>.

Matthew T. Dougherty (matthewd@bcm.edu) is at the National Center for Macromolecular Imaging, specializing in cryo-electron microscopy, visualization, and animation.

Michael J. Folk (mfolk@hdfgroup.org) is president of The HDF Group.

Erez Zadok (ezk@cs.sunysb.edu) is associate professor at Stony Brook University, specializing in computer storage systems performance and design.

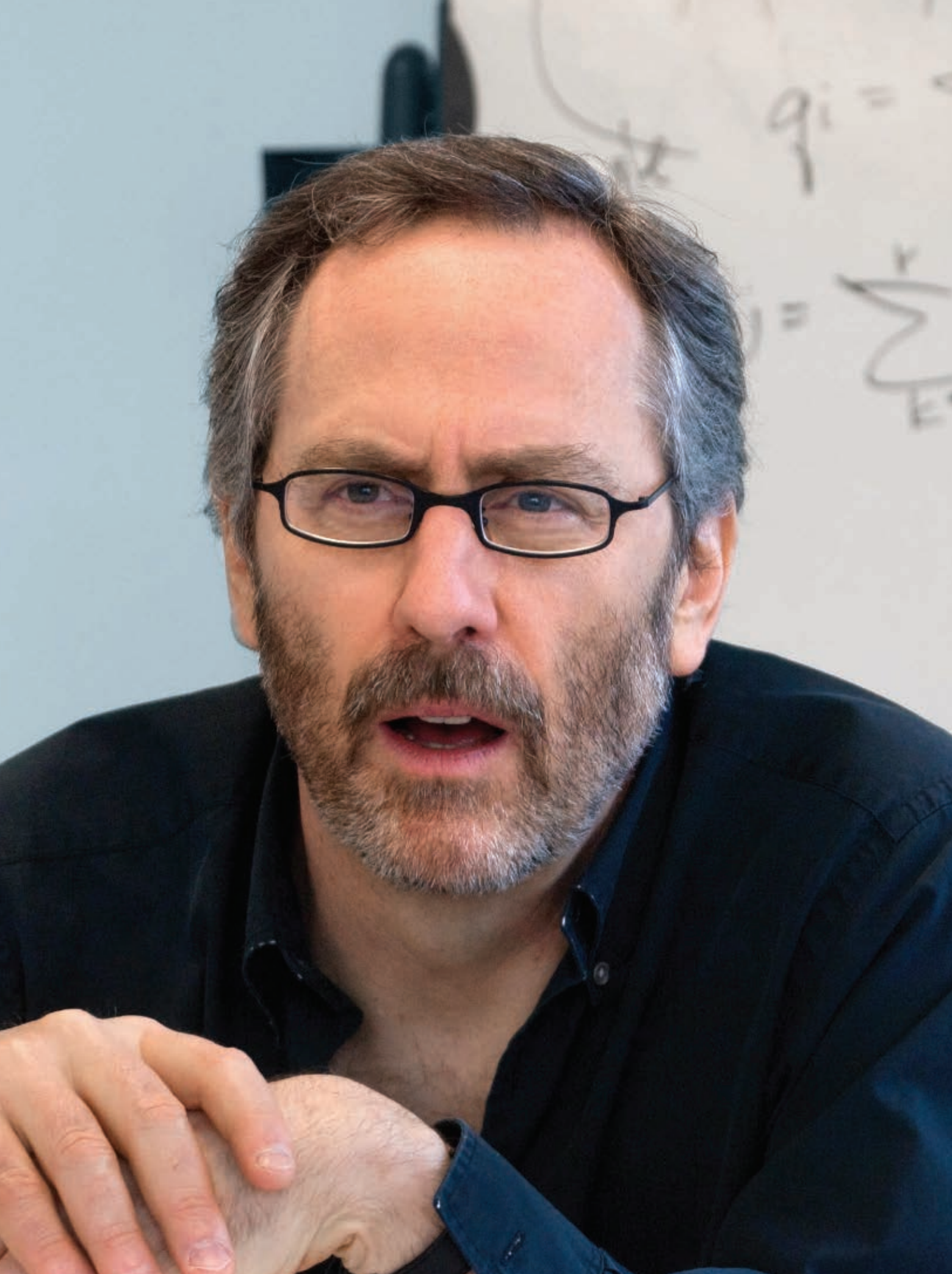
Herbert J. Bernstein (yaya@dowling.edu) is professor of computer science at Dowling College, active in the development of IUCr standards.

Frances C. Bernstein (fcb@bernstein-plus-sons.com) is retired from Brookhaven National Laboratory after 24 years at the Protein Data Bank, active in macromolecular data representation and validation.

Kevin W. Eliceiri (eliceiri@wisc.edu) is director at the Laboratory for Optical and Computational Instrumentation, University of Wisconsin-Madison, active in the development of tools for bio-image informatics.

Werner Benger (werner@cct.lsu.edu) is visualization research scientist at Louisiana State University, specializing in astrophysics and computational fluid dynamics.

Christoph Best (best@ebi.ac.uk) is project leader at the European Bioinformatics Institute, specializing in electron microscopy image informatics.



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

Stanford professor Pat Hanrahan sits down with the noted hedge fund founder, computational biochemist, and (above all) computer scientist.

A Conversation with David E. Shaw

DAVID SHAW CONSIDERS himself first and foremost a computer scientist. It's a fact that's sometimes overshadowed by the activities of his two highly successful, yet very different, ventures: the hedge fund D. E. Shaw & Co., which he founded 20 years ago, and the research lab D. E. Shaw Research, where he now conducts hands-on research in the field of computational biochemistry. The former makes money through rigorous quantitative and qualitative investment techniques, while the latter spends money

simulating complex biochemical processes. But a key element to both organizations' success has been Shaw's background in computer science. Serving as interviewer, computer graphics researcher and Stanford professor Pat Hanrahan^a points out that one of Shaw's unique gifts is his ability to effectively apply computer science techniques to diverse problem domains.

In this interview, Hanrahan and Shaw discuss Shaw's latest project at D. E. Shaw Research—Anton, a special-purpose supercomputer designed to speed up molecular dynamics simula-

tions by several orders of magnitude. Four 512-processor machines are now active and already helping scientists to understand how proteins interact with each other and with other molecules at an atomic level of detail. Shaw's hope is that these "molecular microscopes" will help unravel some biochemical mysteries that could lead to the development of more effective drugs for cancer and other diseases. If his track record is any indication, the world has a lot to be hopeful for.

PAT HANRAHAN: What led you to form D. E. Shaw Research?

DAVID SHAW: Before starting the lab, I'd spent a number of years in the fi-

^a <http://graphics.stanford.edu/~hanrahan/>

nancial world applying quantitative and computational techniques to the process of investment management. During the early years of D. E. Shaw & Co., the financial firm, I'd been personally involved in research aimed at understanding various financial markets and phenomena from a mathematical viewpoint. But as the years went by and the company grew, I had to spend more time on general management, and I could feel myself getting stupider with each passing year. I didn't like that, so I started solving little theoretical problems at night just for fun—things I could tackle on my own, since I no longer had a research group like I did when I was on the faculty at Columbia. As time went by, I realized that I was enjoying that more and more, and that I missed doing research on a full-time basis.

I had a friend, Rich Friesner, who was a chemistry professor at Columbia, and he was working on problems like protein folding and protein dynamics, among other things. Rich is a computational chemist, so a lot of what he did involved algorithms, but his training was in chemistry rather than computer science, so when we got together socially, we often talked about some of the intersections between our fields. He'd say, "You know, we have this problem: the inner loop in this code does such-and-such, and we can't do the studies we want to do because it's too slow. Do you have any ideas?"

Although I didn't understand much at that point about the chemistry and biology involved, I'd sometimes take the problem home, work on it a little bit, and try to come up with a solution that would speed up his code. In some cases, the problem would turn out to be something that any computer scientist with a decent background in algorithms would have been able to solve. After thinking about it for a little while, you'd say, "Oh, that's really just a special case of this known problem, with the following wrinkle."

One time I managed to speed up the whole computation by about a factor of 100, which was very satisfying to me. It didn't require any brilliant insight; it was just a matter of bringing a bit of computer science into a research area where there hadn't yet been all that much of it.

At a certain point, when I was approaching my 50th birthday, I felt like it was a natural time to think about what I wanted to do over the coming years. Since my graduate work at Stanford and my research at Columbia were focused in part on parallel architectures and algorithms, one of the things I spent some time thinking about was whether there might be some way to apply those sorts of technologies to one of the areas Rich had been teaching me about. After a fair amount of reading and talking to people, I found one application—the simulation of molecular dynamics—where it seemed like a massive increase in speed could, in principle, have a major impact on our understanding of biological processes at the molecular level.

It wasn't immediately clear to me whether it would actually be possible to get that sort of speed up, but it smelled like the sort of problem where there might be some nonobvious way to make it happen. The time seemed ripe from a technological viewpoint, and I just couldn't resist the impulse to see if it could be done. At that point, I started working seriously on the problem, and I found that I loved being involved again in hands-on research. That was eight years ago, and I still feel the same way.

HANRAHAN: In terms of your goals at D. E. Shaw Research, are they particularly oriented toward computational chemistry or is there a broader mission?

SHAW: The problems I'm most interested in have a biochemical or biophysical focus. There are lots of other aspects of computational chemistry that are interesting and important—nanostructures and materials science and things like that—but the applications that really drive me are biological, especially things that might lead not only to fundamental insights into molecular biological processes, but also to tools that someone might use at some point to develop lifesaving drugs more effectively.

Our particular focus is on the structure and function of biological molecules at an atomic level of detail, and not so much at the level of systems biology, where you try to identify and understand networks of interacting proteins, figure out how genetic variations

affect an individual's susceptibility to various human diseases, and so forth. There are a lot of computer scientists working in the area that's commonly referred to as bioinformatics, but not nearly as many who work on problems involving the three-dimensional structures and structural changes that underlie the physical behavior of individual biological molecules. I think there's still a lot of juicy, low-hanging fruit in this area, and maybe even some important unifying principles that haven't yet been discovered.

HANRAHAN: When you mention drug discovery, do you see certain applications like that in the near-term or are you mostly trying to do pure research at this point?

SHAW: Although my long-term hope is that at least some of the things we discover might someday play a role in curing people, that's not something I expect to happen overnight. Important work is being done at all stages in the drug development pipeline, but our own focus is on basic scientific research with a relatively long time horizon, but a large potential payoff. To put this in perspective, many of the medications we use today were discovered more or less by accident, or through a brute-force process that's not based on a detailed understanding of what's going on at the molecular level. In many areas, these approaches seem to be running out of steam, which is leading researchers to focus more on targeting drugs toward specific proteins and other biological macromolecules based on an atomic-level understanding of the structure and behavior of those targets.

The techniques and technologies we've been working on are providing new tools for understanding the biology and chemistry of pharmaceutically relevant molecular systems. Although developing a new drug can take as long as 15 years, our scientific progress is occurring over a much shorter timescale, and we're already discovering things that we hope might someday be useful in the process of drug design. But I also enjoy being involved in the unraveling of biological mysteries, some of which have puzzled researchers for 40 or 50 years.

HANRAHAN: This machine you've built, Anton, is now operational. Can

you tell us a little bit about that machine and the key ideas behind it?

SHAW: Anton was designed to run molecular dynamics (MD) simulations a couple orders of magnitude faster than the world's fastest supercomputers. MD simulations are conceptually pretty simple. In biological applications like the ones that interest us, the idea is to simulate the motions of large biomolecules, such as proteins or DNA, at an atomic level of detail over a time period during which biologically interesting things happen in real life. The trajectories followed by these molecules are typically calculated by numerically integrating Newton's laws of motion, using a model based on classical mechanics that approximates the effects of the various types of forces that act on the atoms. During each "time step" in the integration process, the atoms move a very short distance, after which the forces are recomputed, the atoms are moved again, and so on.

HANRAHAN: And you're doing this on a femtosecond (10^{-15} seconds) time scale, correct?


SHAW: That's right. Each of those time steps can only cover something on the order of a couple femtoseconds of biological time, which means it takes a really long time to observe anything of biological interest.

HANRAHAN: If you were to do that simulation with a conventional computer, such as a normal core or dual or quad core, how much biological time could you simulate in a day?


SHAW: Depending on the number of cores and other characteristics of the processor chip, you'd expect performance on the order of a few nanoseconds per day, as measured using a standard molecular system called the Joint AMBER-CHARMM Benchmark System, which represents a particular protein surrounded by water.

HANRAHAN: And what's the comparable figure for Anton?

SHAW: Our latest benchmark measurement was 16,400 nanoseconds per day on a 512-node Anton configuration, so Anton would run three or four orders of magnitude faster, and roughly two orders of magnitude faster than the fastest that can be achieved under practical conditions on supercomputers or massively parallel clusters.



The techniques and technologies we've been working on are providing new tools for understanding the biology and chemistry of pharmaceutically relevant molecular systems.



HANRAHAN: When I read the Anton paper in *Communications*,^b it reminded me a lot of what I worked on, which were graphics chips—just in the way you're choreographing communication and keeping everything busy and all these really important tricks. Can you summarize some of the key innovations in Anton that make it so fast?

SHAW: Each node of the Anton machine is based on a specialized ASIC that contains, among other things, 32 very long, arithmetically dense, non-programmable pipelines designed specifically to calculate certain forces between pairs of interacting particles. Those pipelines are the main source of Anton's speed. In addition, Anton uses an algorithm that I developed in conjunction with the machine's top-level architecture to reduce the amount of data that would have to pass from one Anton ASIC to another in the course of exchanging information about the current positions of the atoms and the forces that act on them. In essence, it's a technique for parallelizing the range-limited version of the classic *N*-body problem of physics, but the key thing for our purposes is that it's highly efficient within the range of parameter values we typically encounter in simulating biological systems.

HANRAHAN: Is this the midpoint algorithm or the NT algorithm?

SHAW: The one I'm referring to is the NT algorithm, which is what runs on Anton. The midpoint algorithm is used in Desmond, an MD code that was designed to run on ordinary commodity clusters. They're both examples of what I've referred to as "neutral territory" methods. The NT method has better asymptotic properties, but the constants tend to favor the midpoint method on a typical cluster and the NT method at a higher degree of parallelism.

HANRAHAN: When I first saw your NT algorithm I was stunned by it, just because people have been thinking about this problem for so long. Force calculations between particles are such an important part of computational science, and to have made the observation that

^b "Anton, A Special-Purpose Machine for Molecular Dynamics Simulation" by D.E. Shaw et al. was published in the July 2008 issue of *Communications of the ACM*, 91–97.


the best way to compute the interactions of two particles involves sending them both somewhere else is just amazing. I understand your proof but it still boggles me because it seems so counterintuitive.

SHAW: It is kind of weird, but if you look back through the literature, you can find various pieces of the puzzle in a number of different contexts. Although I wasn't aware of this at the time, it turns out that the idea of "meeting on neutral territory" can be found in different forms in publications dating back as far as the early 1990s, although these early approaches didn't offer an asymptotic advantage over traditional spatial decomposition methods. Later, Marc Snir independently came up with an algorithm that achieved the same asymptotic properties as mine in a different way, and he described his method in a very nice paper that appeared shortly before my own. His paper included a proof that this is the best you can do from an asymptotic viewpoint. Although the constant factors were such that his method wouldn't have performed well in the sorts of applications we're interested in, it's clear with the benefit of hindsight that the straightforward addition of certain features from the NT method would have made his algorithm work nearly as well as NT itself for that class of applications.


But the important thing from my perspective was not that the NT algorithm had certain asymptotic advantages, but that with the right kind of machine architecture it removed a key bottleneck that would otherwise have kept me from exploiting the enormous amount of application-specific arithmetic horsepower I'd wanted to place on each chip.

HANRAHAN: I think it's a great example of computer science thinking because it's a combination of a new algorithm, which is a new way of organizing the computation, as well as new hardware. Some people think all advances in computing are due to software or hardware, but I think some of the most interesting ones are where those things coevolve in some sense.

SHAW: I agree completely. The history of special-purpose machines has been marked by more failures than successes, but I suspect that the cre-



Our latest benchmark measurement was 16,400 nanoseconds per day on a 512-node Anton configuration, so Anton would run three or four orders of magnitude faster, and roughly two orders of magnitude faster than the fastest that can be achieved under practical conditions on supercomputers or massively parallel clusters.



ative codesign of new architectural and algorithmic approaches could have increased, at least to some extent, the number of applications in which a sufficient speedup was achieved that outweighed the very real economies of scale associated with the use of general-purpose commodity hardware. In our case, I don't think we could have reached our performance goals either by implementing standard computational chemistry algorithms on new hardware or by using standard high-performance architectures to run new algorithms.

HANRAHAN: I think a lot of people in computer science were very surprised by your Anton paper because they might have thought that normal computing paradigms are efficient and that there's not a lot to be gained. You read these papers where people get 5% improvements, and then you sort of blow them out of the water with this 100-fold improvement. Do you think this is just a freak thing or are there chances that we could come up with other revolutionary new ways of solving problems?

SHAW: I've been asked that before, but I'm really not sure what the answer is. I'd be surprised if this turned out to be the only computationally demanding application that could be addressed using the general approach we've taken to the design of a special-purpose machine. On the other hand, there are a lot of scientific problems for which our approach would clearly not be effective. For one thing, some problems are subject to unavoidable communication bottlenecks that would dominate any speedup that might be achieved using an application-specific chip. And in some other applications, flexibility may be important enough that a cluster-based solution, or maybe one based on the use of GPUs, would be a better choice.

One of the things I found attractive about our application from the start was that although it was nowhere close to "embarrassingly parallel," it had several characteristics that seemed to beg for the use of special-purpose hardware. First, the inner loop that accounted for a substantial majority of the time required for a biologically oriented molecular dynamics simulation was highly regular, and could be



Anton, a massively parallel machine designed to perform molecular dynamic simulations.

mapped onto silicon in an extremely area- and power-efficient way. It also turned out that these inner-loop calculations could be structured in such a way that the same data was used a number of times, and with well-localized data transfers that minimized the need to send data across the chip, much less to and from off-chip memory.

There were some parts of our application where any function-specific hardware would have been grossly underutilized or unable to take advantage of certain types of new algorithms or biophysical models that might later be discovered. Fortunately, most of those calculations aren't executed all that frequently in a typical biomolecular simulation. That made it feasible to incorporate a set of programmable on-chip processors that could be used for various calculations that fell outside the inner loop. We were also able to make use of problem-specific knowledge to provide hardware support for

a specific type of inter-chip communication that was especially important in our application.

Since my own interest is in the application of molecular dynamics simulations to biological problems, I haven't been forced to think very hard about what aspects of the approach we've followed might be applicable to the design of special-purpose machines and algorithms for other applications. If I had to guess, I'd say that at least some aspects of our general approach might wind up being relevant to researchers who are looking for an insane increase in speed for certain other applications, but that hunch isn't based on anything very solid.

HANRAHAN: In the *Communications* paper, Anton was described as a computational microscope. I really liked that phrase and that the name Anton came from van Leeuwenhoek, who was one of the first microscopists.

SHAW: Part of the reason I like the

metaphor of a computational microscope is that it emphasizes one of the key things that Anton isn't. Although we sometimes describe the machine as a "special-purpose supercomputer," its range of applicability is in practice so narrow that thinking of Anton as a computer is a bit like thinking of a microscope as a general-purpose laboratory instrument. Like the optical microscope, Anton is really no more than a specialized tool for looking at a particular class of objects that couldn't be seen before.

HANRAHAN: So now that you have this microscope, what do you want to point it at? I know you must be collaborating, and you have computational chemists and biologists at D. E. Shaw Research. Do you have some problems that you want to go after with it?

SHAW: There's a wide range of specific biological phenomena we'd like to know more about, but at this point, there's a lot we can learn by simply

putting things under the microscope and seeing what's there. When Anton van Leeuwenhoek first started examining pond water and various bodily fluids, there's no way he could have predicted that he'd see what we now know to be bacteria, protozoa, and blood and sperm cells, none of which had ever been seen before. Although we have no illusions about our machine having an impact comparable to the optical microscope, the fact is that nobody has ever seen a protein move over a period of time even remotely close to what we're seeing now, so in some ways, we're just as clueless as van Leeuwenhoek was when he started looking.

All that being said, there are some biological systems and processes that we've been interested in for a while, and we're beginning to learn more about some of them now that we're able to run extremely long simulations. The one that's probably most well known is the process of protein folding, which is when a string of amino acids folds up into a three-dimensional protein. We've already started to learn some interesting things related to folding that we wouldn't have known if it hadn't been for Anton, and we're hoping to learn more over time. We've also conducted studies of a class of molecules called kinases, which play a central role in the development and treatment of cancer. And we're looking at several proteins that transfer either ions or signals through membranes in the cell. We're also using Anton to develop new algorithms and methods, and to test and improve the quality of the physical models that are used in the simulation process itself.

HANRAHAN: It seems like you're almost at a tipping point. I worked at Pixar, and one of the singular events was when computer graphics were used in *Jurassic Park*. Even bigger was *Toy Story*. Once the graphics software reached a certain maturity, and once you showed that it could be used to make a blockbuster, then it wasn't that long afterward that almost every movie included computer-generated effects. How close are we to that tipping point in structural biology? If you're able to solve some important problems in structural biology, then people might begin considering molecular dynamics simulations

as part of standard practice, and then routinely deploy this approach to solving biological problems.

SHAW: That's a great analogy. Although the evidence is still what I'd characterize as preliminary, I think there's enough of it at this point to predict that MD simulations will probably be playing a much larger role within the field of structural biology a few years from now. It's hard to tell in advance whether there's a *Toy Story* around the corner, since the biggest breakthroughs in biology often turn out to be ones that would have been difficult to even characterize before they happened. It may be that there are some deep principles out there that are waiting to be discovered in silico—ones that can tell us something fundamental about some of the mechanisms nature has evolved to do what it wants to get done. It's hard to plan for a discovery like that; you just have to be in the right territory with the tools and skills you think you might need in order to recognize it when you see it.

HANRAHAN: There's no place that's more fun to be when you have a new microscope. van Leeuwenhoek must have had a good time. I've read a bunch about Robert Hooke, too, who was a contemporary of van Leeuwenhoek. He was part of the Royal Society. Every week or two, they would get together and make all these discoveries because they were looking at the world in a different way.


SHAW: I've always thought it would be great to live during a period when a lot of fundamental discoveries were being made, and a single scientist could stay on top of most of the important advances being made across the full breadth of a given discipline. It's a bit sad that we can't do that anymore—victims of our success.

HANRAHAN: But one thing that amazes me about you is the number of fields you've had an impact on. I'm trying to figure out your secret sauce. You seem to be able to bring computation to bear in problem areas that other people haven't been as successful in. Obviously you've had a huge impact on the financial industry with the work you did on modeling stocks and portfolios. And now you're doing biochemistry. How do you approach these new areas? How do you bring computers to bear

on these new problems?

SHAW: I'm not sure I deserve those very kind words, but for what it's worth, I tend to generate a plentiful supply of ideas, the vast majority of which turn out to be bad ones. In some cases, they involve transplanting computational techniques from one application to another, and there's usually a good reason why the destination field isn't already using that technique. I also have a remarkable capacity to delude myself into thinking that each idea has a higher probability of working than it really does, which provides me with the motivation I need to keep working on it. And, every once in a while, I stumble on an idea that actually works.

HANRAHAN: It sounds like you have this “gene” for computing. You know algorithms, you know architecture, but yet you still are fascinated with applying them to new problems. That's what's often missing in our field. People learn the techniques of the field but they don't know how to apply them in a new problem domain.

SHAW: I love learning about new fields, but in some ways I feel like a tourist whose citizenship is computer science. I think to myself, “I'm doing computational finance, but I am a computer scientist. I'm doing computational biology, but I am a computer scientist.” When we computer scientists start infiltrating a new discipline, what we bring to the table is often more than just a bag of tricks. What's sometimes referred to as “computational thinking” is leaving its mark on one field after another—and the night is still young. 

Related articles on queue.acm.org

A Conversation with Kurt Akeley and Pat Hanrahan

<http://queue.acm.org/detail.cfm?id=1365496>

Beyond Beowulf Clusters

Philip Papadopoulos, Greg Bruno, Mason Katz
<http://queue.acm.org/detail.cfm?id=1242501>

Databases of Discovery

James Ostell
<http://queue.acm.org/detail.cfm?id=1059806>



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:10.1145/1562764.1562783

Writing programs that scale with increasing numbers of cores should be as easy as writing programs for sequential computers.

BY KRSTE ASANOVIC, RASTISLAV BODIK, JAMES DEMMEL, TONY KEAVENY, KURT KEUTZER, JOHN KUBIATOWICZ, NELSON MORGAN, DAVID PATTERSON, KOUSHIK SEN, JOHN WAWRZYNEK, DAVID WESSEL, AND KATHERINE YELICK

A View of the Parallel Computing Landscape

INDUSTRY NEEDS HELP from the research community to succeed in its recent dramatic shift to parallel computing. Failure could jeopardize both the IT industry and the portions of the economy that depend on rapidly improving information technology. Here, we review the issues and, as an example, describe an integrated approach we're developing at the Parallel Computing Laboratory, or Par Lab, to tackle the parallel challenge.

Over the past 60 years, the IT industry has improved the cost-performance of sequential computing by about 100 billion times overall.²⁰ For most of the past 20 years, architects have used the rapidly increasing transistor speed and budget made possible by silicon

technology advances to double performance every 18 months. The implicit hardware/software contract was that increased transistor count and power dissipation were OK as long as architects maintained the existing sequential programming model. This contract led to innovations that were inefficient in terms of transistors and power (such as multiple instruction issue, deep pipelines, out-of-order execution, speculative execution, and prefetching) but that increased performance while preserving the sequential programming model.

The contract worked fine until we hit the power limit a chip is able to dissipate. Figure 1 reflects this abrupt change, plotting the projected microprocessor clock rates of the International Technology Roadmap for Semiconductors in 2005 and then again just two years later.¹⁶ The 2005 prediction was that clock rates should have exceeded 10GHz in 2008, topping 15GHz in 2010. Note that Intel products are today far below even the conservative 2007 prediction.

After crashing into the power wall, architects were forced to find a new paradigm to sustain ever-increasing performance. The industry decided the only viable option was to replace the single power-inefficient processor with many efficient processors on the same chip. The whole microprocessor industry thus declared that its future was in parallel computing, with increasing numbers of processors, or cores, each technology generation every two years. This style of chip was labeled a multicore microprocessor. Hence, the leap to multicore is not based on a breakthrough in programming or architecture and is actually a retreat from the more difficult task of building power-efficient, high-clock-rate, single-core chips.⁵

Many startups have sold parallel computers over the years, but all failed, as programmers accustomed to continuous improvement in sequential performance saw little need to explore parallelism. Convex, Encore, Floating Point Systems, Immos, Kendall Square



Research, MasPar, nCUBE, Sequent, Silicon Graphics, and Thinking Machines are just the best-known members of the Dead Parallel Computer Society. Given this sad history, multicore pessimism abounds. Quoting computing pioneer John Hennessy, President of Stanford University:

*“...when we start talking about parallelism and ease of use of truly parallel computers, we’re talking about a problem that’s as hard as any that computer science has faced. ...I would be panicked if I were in industry.”*¹⁹

Jeopardy for the IT industry means opportunity for the research community. If researchers meet the parallel challenge, the future of IT is rosy. If they don’t, it’s not. Hence, there are few restrictions on potential solutions. Given an excuse to reinvent the whole software/hardware stack, this opportunity is also a once-in-a-career chance to fix other weaknesses in computing that have accumulated over the decades like barnacles on the hull of an old ship.

Here, we lay out one view of the opportunities, then, as an example, describe in more depth the approach of the Berkeley Parallel Computing Lab, or Par Lab, updating two long technical reports^{4,5} that include more detail. Our goal is to recruit more parallel revolutionaries.

Parallel Bridge

The bridge in Figure 2 represents an analogy connecting computer users on the right to the IT industry on the left. The left tower is hardware, the right tower is applications, and the

long span in between is software. We use the bridge analogy throughout this article. The aggressive goal of the parallel revolution is to make it as easy to write programs that are as efficient, portable, and correct (and that scale as the number of cores per microprocessor increases biennially) as it has been to write programs for sequential computers. Moreover, we can fail overall if we fail to deliver even one of these “parallel virtues.” For example, if parallel programming is unproductive, this weakness will delay and reduce the number of programs that are able to exploit new multicore architectures.

Hardware tower. The power wall forces the change in the traditional programming model, but the question for parallel researchers is what kind of computing architecture should take its place. There is a technology sweet spot around a pipelined processor of five-to-eight stages that is most efficient in terms of performance per joule and silicon area.⁵ Using simple cores means there is room for hundreds of them on the same chip. Moreover, having many such simple cores on a chip simplifies hardware design and verification, since each core is simple, and replication of cores is nearly trivial. Just as it’s easy to add spares to mask manufacturing defects, “manycore” computers can also have higher yield.

One example of a manycore computer is from the world of network processors, which has seen a great deal of innovation recently due to the growth of the networking market. The best-designed network processor is arguably the Cisco Silicon Packet Processor, also known as

Metro, which has 188 five-stage RISC cores, plus four spares to help yield and dissipate just 35 watts.

It may be reasonable to assume that manycore computers will be homogeneous, like the Metro, but there is an argument for heterogeneous manycores as well. For example, suppose 10% of the time a program gets no speedup on a 100-core computer. To run this sequential piece twice as fast, assume a single fat core would need 10 times as many resources as a thin core due to larger caches, a vector unit, and other features. Applying Amdahl’s Law, here are the speedups (relative to one thin core) of 100 thin cores and 90 thin cores for the parallel code plus one fat core for the sequential code:

$$\text{Speedup}_{100} = 1 / (0.1 + 0.9/100) = 9.2 \text{ times faster}$$

$$\text{Speedup}_{91} = 1 / (0.1/2 + 0.9/90) = 16.7 \text{ times faster}$$

In this example of manycore processor speedup, a fat core needing 10 times as many resources would be more effective than the 10 thin cores it replaces.^{5,15}

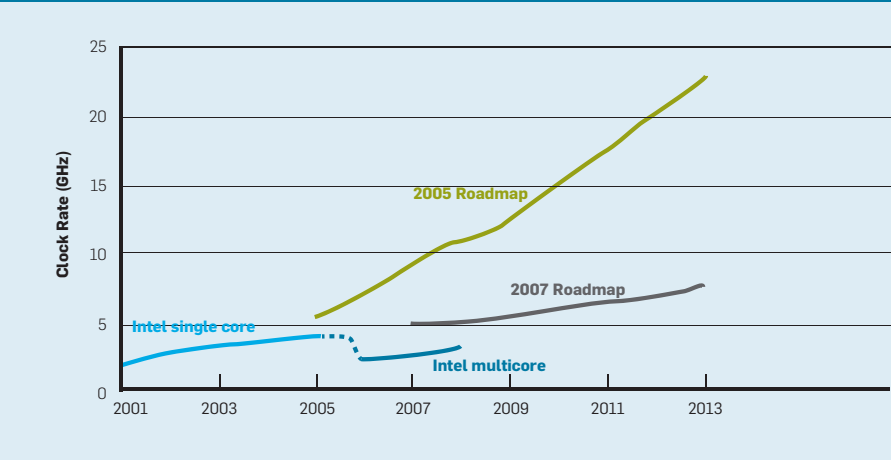
One notable challenge for the hardware tower is that it takes four to five years to design and build chips and port software to evaluate them. Given this lengthy cycle, how could researchers innovate more quickly?

Software span. Software is the main problem in bridging the gap between users and the parallel IT industry. Hence, the long distance of the span in Figure 2 reflects the daunting magnitude of the software challenge.

One especially vexing challenge for the parallel software span is that sequential programming accommodates the wide range of skills of today’s programmers. Our experience teaching parallelism suggests that not every programmer is able to understand the nitty gritty of concurrent software and parallel hardware; difficult steps include locks, barriers, deadlocks, load balancing, scheduling, and memory consistency. How can researchers develop technology so all programmers benefit from the parallel revolution?

A second challenge is that two critical pieces of system software—compilers and operating systems—have grown large and unwieldy and hence

Figure 1. Microprocessor clock rates of Intel products vs. projects from the International Roadmap for Semiconductors in 2005 and 2007.¹⁶



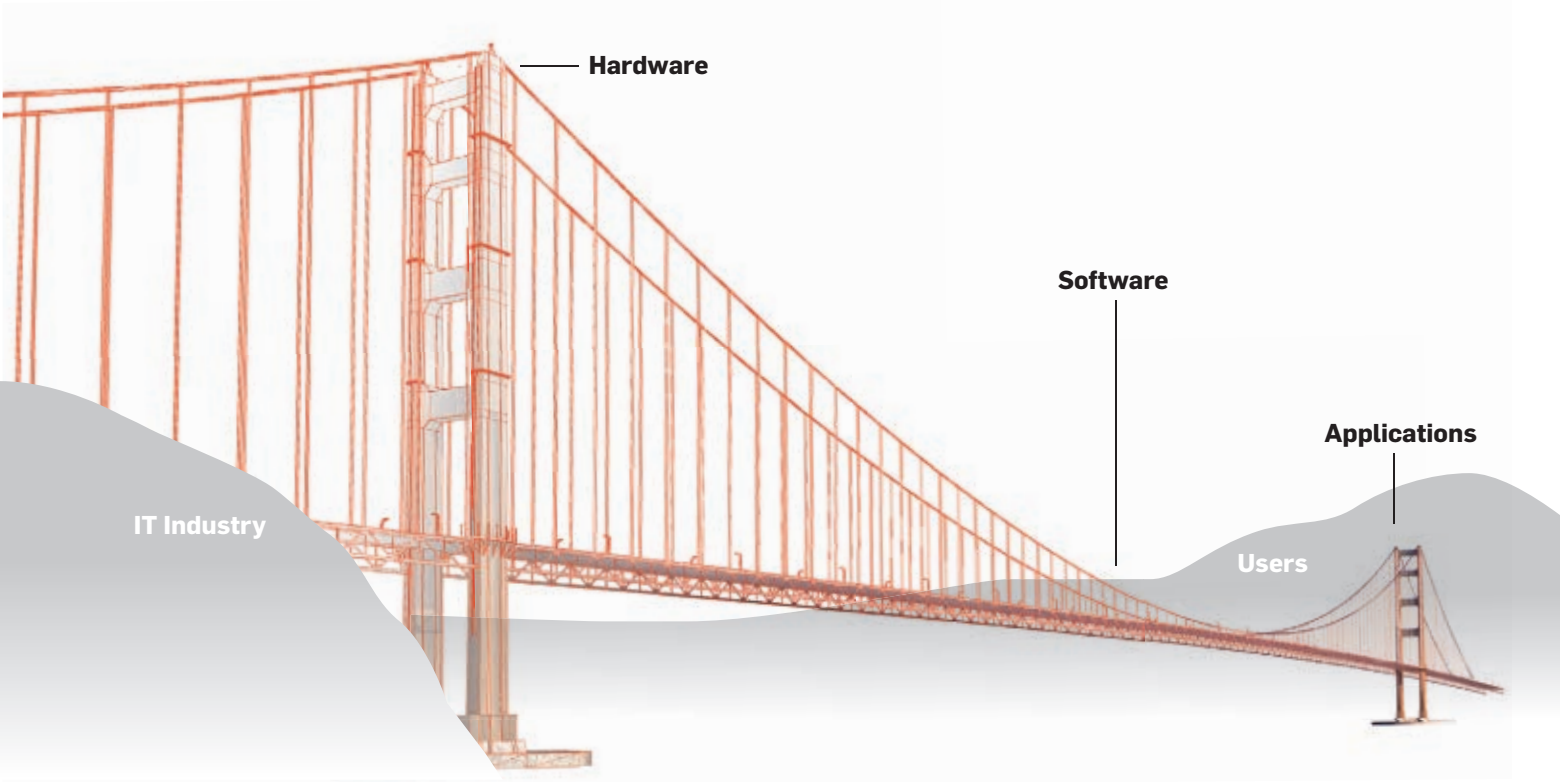


Figure 2. Bridge analogy connecting users to a parallel IT industry, inspired by the view of the Golden Gate Bridge from Berkeley, CA.

resistant to change. One estimate is that it takes a decade for a new compiler optimization to become part of production compilers. How can researchers innovate rapidly if compilers and operating systems evolve so glacially?

A final challenge is how to measure improvement in parallel programming languages. The history of these languages largely reflects researchers deciding what they think would be better and then building it for others to try. As humans write programs, we wonder whether human psychology and human-subject experiments shouldn't be allowed to play a larger role in this revolution.¹⁷

Applications tower. The goal of research into parallel computing should be to find compelling applications that thirst for more computing than is currently available and absorb biennially increasing number of cores for the next decade or two. Success does not require improvement in the performance of all legacy software. Rather, we need to create compelling applications that effectively utilize the growing number of cores while providing software environments that ensure that legacy code still works with acceptable performance.

Note that the notion of “better”

is not defined by only average performance; advances could be in, say, worst-case response time, battery life, reliability, or security. To save the IT industry, researchers must demonstrate greater end-user value from an increasing number of cores.

Par Lab

As a concrete example of the parallel landscape, we describe Berkeley's Par Lab project,^a exploring one of many potential approaches, though we won't know for years which of our ideas will bear fruit. We hope it inspires more researchers to participate, increasing the chance of finding a solution before it's too late for the IT industry.

Given a five-year project, we project the state of the field in five to 10 years, anticipating that IT will be driven to extremes in size due to the increasing popularity of software as a service, or SaaS:

The datacenter is the server. Amazon, Google, Microsoft, and other major IT vendors are racing to construct build-

ings with 50,000 or more servers to run SaaS, inspiring the new catchphrase “cloud computing.”^b They have also begun renting thousands of machines by the hour to enable smaller companies to benefit from cloud computing. We expect these trends to accelerate; and

The mobile device (laptops and handhelds) is the client. In 2007, Hewlett-Packard, the largest maker of PCs, shipped more laptops than desktops. Millions of cellphones are shipped each day with ever-increasing functionality, a trend we expect to accelerate as well.

Surprisingly, these extremes in computing share many characteristics. Both concern power and energy—the datacenter due to the cost of power and cooling and the mobile client due to battery life. Both concern cost—the datacenter because server cost is replicated 50,000 times and mobile clients because of a lower unit-price target. Finally, the software stacks are becoming similar, with more layers for mobile clients and increasing concern about protection and security.

^a In March 2007, Intel and Microsoft invited 25 universities to propose five-year centers for parallel computing research; the Berkeley and Illinois efforts were ranked first and second.

^b See Ambrust, M. et al. *Above the Clouds: A Berkeley View of Cloud Computing*. University of California, Berkeley, Technical Report EECS-2009-28.

Many datacenter applications have ample parallelism across independent users, so the Par Lab focuses on parallelizing applications for clients. The multicore and manycore chips in the datacenter stand to benefit from the same tools and techniques developed for similar chips in mobile clients.

Given this projection, we decided to take a fresh approach: the Par Lab will be driven top-down, applications first, then software, and finally hardware.

Par Lab application tower. An unfortunate computer science tradition is we build research prototypes, then wonder why applications people don't use them. In the Par Lab, we instead selected applications up-front to drive research and provide concrete goals and metrics to evaluate progress. We selected each application based on five criteria: compelling in terms of likely market or social impact, with short-term feasibility and longer-term potential; requiring significant speedup or smaller, more efficient platform to work as intended; covering the possible platforms and markets likely to dominate usage; enabling technology for other applications; and involvement of a local committed expert application partner to help design, use, and evaluate our technology.

Here are the five initial applications we're developing:

Music/hearing. High-performance signal processing will permit: concert-quality sound-delivery systems for home sound systems and conference calls; composition and gesture-driven live-performance systems; and much improved hearing aids;

Speech understanding. Dramatically improved automatic speech recognition in moderately noisy and reverberant environments would greatly improve existing applications and enable new ones, like, say, a real-time meeting transcriber with rewind and search. Depending on acoustic conditions, current transcribers can generate many errors;

Content-based image retrieval. Consumer-image databases are growing so dramatically they require automated search instead of manual labeling. Low error rates require processing very high dimensional feature spaces. Current image classifiers are too slow to deliver adequate response times;

Intraoperative risk assessment for

stroke patients. Advanced physiological blood-flow modeling based on computational analysis of 3D medical images of a patient's cerebral vasculature enables "virtual stress testing" to risk-stratify stroke victims intraoperatively. Patients thus identified at low risk of complications can then be treated to mitigate the effects of the stroke. This technology will ultimately lower complication rates in treating stroke victims, improve quality of life, reduce medical care expenditures, and save lives; and

Parallel browser. The browser will be the largest and most important application on many mobile devices. We will first parallelize sequential browser bottlenecks. Rather than parallelizing JavaScript programs, we are pursuing an actor language with implicit parallelism. Such a language may be accessible to Web programmers while allowing them to extract the parallelism in the browser's JIT compiler, thereby turning all Web-site developers unknowingly into parallel programmers.

Application-domain experts are first-class members of the Par Lab project. Rather than try to answer design questions abstractly, we ask our experts what they prefer in each case. Project success is judged by the user experience with the collective applications on our hardware-software prototypes. If successful, we imagine building on these five applications to create other applications that are even more compelling, as in the following two examples:

Name Whisperer. Imagine that your mobile client peeking out of your shirt pocket is able to recognize the person walking toward you to shake your hand. It would search a personal image database, then whisper in your ear, "This man is John Smith. He got an A- from you in CS152 in 1993"; and

Health Coach. As your mobile client is always with you, you could take pictures and weigh your dishes (assuming it has a built-in scale) before and after each meal. It would also record how much you exercise. Given calories consumed and burned and an image of your body, it could visualize what you're likely to look like in six months at this rate and what you'd look like if you ate less or exercised more.

Par Lab software span. Software is the major effort of the project, and we're taking a different path from pre-

vious parallel projects, emphasizing software architecture, autotuning, and separate support for productivity vs. performance programming.

Architecting parallel software with design patterns, not just parallel programming languages. Our situation is similar to that found in other engineering disciplines where a new challenge emerges that requires a top-to-bottom rethinking of the entire engineering process; for example, in civil architecture, Filippo Brunelleschi's solution in 1418 for how to construct the dome for the Cathedral of Florence required innovations in tools and building techniques, as well as rethinking the whole process of developing an architecture. All computer science faces a similar challenge; parallel programming is overdue for a fundamental rethinking of the process of designing software.

Programmers have been trying to craft parallel code for decades and learned a great deal about what works and what doesn't work. Automatic parallelism doesn't work. Compilers are great at low-level scheduling decisions but can't discover new algorithms to exploit concurrency. Programmers in high-performance computing have shown that explicit technologies (such as MPI and OpenMP) can be made to work but too often require heroic effort untenable for most commercial software vendors.

To engineer high-quality parallel software, we plan to rearchitect the software through a "design pattern language." As explored in his 1977 book, civil architect Christopher Alexander wrote that "design patterns" describe time-tested solutions to recurring problems within a well-defined context.³ An example is Alexander's "family of entrances" pattern, addressing how to simplify comprehension of multiple entrances for a first-time visitor to a site. He defined a "pattern language" as a collection of related and interlocking patterns, constructed such that the patterns flow into each other as the designer solves a design problem.


Computer scientists are trained to think in well-defined formalisms. Pattern languages encourage a less formal, more associative way of thinking about a problem. A pattern language does not impose a rigid methodology; rather, it fosters creative problem

solving by providing a common vocabulary to capture the problems encountered during design and identify potential solutions from among families of proven designs.


The observation that design patterns and pattern languages might be useful for software design is not new. An example is Gamma et al.'s 1994 book *Design Patterns*, which outlined patterns useful for object-oriented programming.¹² In building our own pattern language, we found Shaw's and Garlan's report,²³ which described a variety of architectural styles useful for organizing software, to be very effective. That these architectural styles may also be viewed as design patterns was noted earlier by Buschmann in his 1996 book *Pattern-Oriented Software Architecture*.⁷ In particular, we adopted Pipe-and-Filter, Agent-and-Repository, Process Control, and Event-Based architectural styles as structural patterns within our pattern language. To this list, we add MapReduce and Iterator as structural design patterns.

These patterns define the structure of a program but do not indicate what is actually computed. To address this blind spot, another key part of our pattern language is the set of "dwarfs" of the Berkeley View reports^{4,5} (see Figure 3). Dwarfs are best understood as computational patterns providing the computational interior of the structural patterns discussed earlier. By analogy, the structural patterns describe a factory's physical structure and general workflow. The computational patterns describe the factory's machinery, flow of resources, and work products. Structural and computational patterns can be combined to architect arbitrarily complex parallel software systems.

Convention holds that truly useful patterns are not invented but mined from successful software applications. To arrive at our list of useful computational patterns we began with those compiled by Phillip Collela of Lawrence Berkeley National Laboratory of the "seven dwarfs of high-performance computing." Then, in 2006 and 2007 we worked with domain experts to broadly survey other application areas, including embedded systems, general-purpose computing (SPEC benchmarks), databases, games, artificial intelligence/machine learning, computer-aided design of integrated



If researchers meet the parallel challenge, the future of IT is rosy. If they don't, it's not.



circuits, and high-performance computing. We then focused in depth on the patterns in the applications we described earlier. Figure 3 shows the results of our pattern mining.

Computational and structural patterns can be hierarchically composed to define an application's high-level software architecture, but a complete pattern language for application design must at least span the full range, from high-level architecture to detailed software implementation and tuning. Mattson et al's 2004 book *Patterns for Parallel Programming*¹⁸ was the first such attempt to systematize parallel programming using a complete pattern language. We combine the structural and computational patterns mentioned earlier in our pattern language to literally sit on top of the algorithmic structures and implementation structures in the pattern language in Mattson's book. The resulting pattern language is still under development but is already employed by the Par Lab to develop the software architectures and parallel implementations of such diverse applications as content-based image retrieval, large-vocabulary continuous speech recognition, and timing analysis for integrated circuit design.

Patterns are conceptual tools that help a programmer reason about a software project and develop an architecture but are not themselves implementation mechanisms for producing code.

Split productivity and efficiency layers, not just a single general-purpose layer. A key Par Lab research objective is to enable programmers to easily write programs that run as efficiently on manycore systems as on sequential ones. Productivity, efficiency, and correctness are inextricably linked and must be addressed together. These objectives cannot be accomplished with a single-point solution (such as a universal language). In our approach, productivity is addressed in a productivity layer that uses a common composition and coordination language to glue together the libraries and programming frameworks produced by the efficiency-layer programmer. Efficiency is principally handled through an efficiency layer that is targeted for use by expert parallel programmers.

The key to generating a successful

multicore software developer community is to maximally leverage the efforts of parallel programming experts by encapsulating their software for use by the programming masses. We use the term “programming framework” to mean a software environment that supports implementation of the solution proposed by the associated design pattern. The difference between a programming framework and a general programming model or language is that in a programming framework the customization is performed only at specified points that are harmonious with the style embodied in the original design pattern. An example of a successful sequential programming framework is the Ruby on Rails framework, which is based on the Model-View-Controller pattern.²⁶ Users have ample opportunity to customize the framework but only in harmony with the core Model-View-Controller pattern.

Frameworks include libraries, code generators, and runtime systems that assist programmers with implementation by abstracting difficult portions of the computation and incorporating them into the framework itself. Historically successful parallel frameworks encode the collective experience of the programming community’s solutions

to recurring problems. Basing frameworks on pervasive design patterns will help make parallel frameworks broadly applicable.

Productivity-layer programmers will compose libraries and programming frameworks into applications with the help of a composition and coordination language.¹³ The language will be implicitly parallel; that is, its composition will have serial semantics, meaning the composed programs will be safe (such as race-free) and virtualized with respect to processor resources. It will document and check interface restrictions to avoid concurrency bugs resulting from incorrect composition, as in, say, instantiating a framework with a stateful function when a stateless one is required. Finally, it will support definition of domain-specific abstractions for constructing frameworks for specific applications, offering a programming experience similar to MATLAB and SQL.

Parallel programs in the efficiency layer are written very close to the machine, with the goal of allowing the best possible algorithm to be written in the primitives of the layer. Unfortunately, existing multicore systems do not offer a common low-level programming model for parallel code. We are thus

defining a thin portability layer that runs efficiently across single-socket platforms and includes features for parallel job creation, synchronization, memory allocation, and bulk-memory access. To provide a common model of memory across machines with coherent caches, local stores, and relatively slow off-chip memory, we are defining an API based on the idea of logically partitioned shared memory, inspired by our experience with Unified Parallel C,²⁷ which partitions memory among processors but not (currently) between on- and off-chip.

We may implement this efficiency language either as a set of runtime primitives or as a language extension of C. It will be extensible with libraries to experiment with various architectural features (such as transactions, dynamic multithreading, active messages, and collective communication). The API will be implemented on some existing multicore and manycore platforms and on our own emulated manycore design.

To engineer parallel software, programmers must be able to start with effective software architectures, and the software engineer would describe the solution to a problem in terms of a design pattern language. Based on this language, the Par Lab is creating a fam-

Figure 3. The color of a cell (for 12 computational patterns in seven general application areas and five Par Lab applications) indicates the presence of that computational pattern in that application; red/high; orange/moderate; green/low; blue/rare.

	Embed	SPEC	DB	Games	ML	CAD	HPC	Health	Image	Speech	Music	Browser
1. Finite State Mach.	Red	Red	Red	Red	Red	Red	Blue	Blue	Blue	Blue	Blue	Red
2. Circuits	Red	Blue	Green	Blue	Green	Blue	Blue	Blue	Blue	Blue	Blue	Red
3. Graph Algorithms	Red	Orange	Orange	Orange	Red	Red	Blue	Red	Blue	Red	Green	Green
4. Structured Grid	Red	Red	Blue	Orange	Blue	Blue	Red	Blue	Red	Blue	Blue	Blue
5. Dense Matrix	Red	Red	Orange	Red	Red	Red	Red	Blue	Red	Red	Red	Blue
6. Sparse Matrix	Orange	Orange	Blue	Red	Red	Red	Red	Red	Blue	Blue	Red	Blue
7. Spectral (FFT)	Orange	Blue	Blue	Orange	Orange	Orange	Red	Blue	Green	Red	Red	Red
8. Dynamic Prog	Orange	Blue	Red	Blue	Red	Red	Red	Blue	Blue	Orange	Blue	Red
9. Particle Methods	Blue	Orange	Blue	Orange	Blue	Blue	Red	Blue	Blue	Blue	Blue	Blue
10. Backtrack/B&B	Blue	Blue	Orange	Blue	Red	Red	Blue	Blue	Blue	Blue	Orange	Blue
11. Graphical Models	Blue	Blue	Blue	Blue	Red	Blue	Blue	Blue	Blue	Blue	Red	Blue
12. Unstructured Grid	Blue	Blue	Blue	Orange	Orange	Orange	Red	Red	Blue	Blue	Red	Blue

ily of frameworks to help turn a design into working code. The general-purpose programmer will work largely with the frameworks and stay within what we call the productivity layer. Specialist programmers trained in the details of parallel programming technology will work within the efficiency layer to implement the frameworks and map them onto specific hardware platforms. This approach will help general-purpose programmers create parallel software without having to master the low-level details of parallel programming.

Generating code with search-based autotuners, not compilers. Compilers that automatically parallelize sequential code may have great commercial value as computers go from one to two to four cores, though as described earlier, history suggests they will be unable to scale from 32 to 64 to 128 cores. Compiling will be even more difficult, as the switch to multicore means microprocessors are becoming more diverse, since conventional wisdom is not yet established for multicore architectures. For example, the table here shows the diversity in designs of x86 and SPARC multicore computers. In addition, as the number of cores increase, manufacturers will likely offer products with differing numbers of cores per chip to cover multiple price-performance points. They will also allow each core to vary its clock frequency to save power. Such diversity will make the goals of efficiency, scaling, and portability even more difficult for conventional compilers, at a time when innovation is desperately needed.

In recent years, autotuners have become popular for producing high-quality, portable scientific code for serial microprocessors,¹⁰ optimizing a set of library kernels by generating many variants of a kernel and measuring each variant by running on the target platform. The search process effectively tries many or all optimization switches; hence, searching may take hours to complete on the target platform. However, search is performed only once, when the library is installed. The resulting code is often several times faster than naive implementations. A single autotuner can be used to generate high-quality code for a variety of machines. In many cases, the autotuned code is faster than vendor libraries that were specifically hand-tuned for the target

machine. This surprising result is partly explained by the way the autotuner tirelessly tries many unusual variants of a particular routine. Unlike libraries, autotuners also allow tuning to the particular problem size. Autotuners also preserve clarity and support portability by reducing the temptation to mangle the source code to improve performance for a particular computer.

Autotuning also helps with production of parallel code. However, parallel architectures introduce many new optimization parameters; so far, there are few successful autotuners for parallel codes. For any given problem, there may be several parallel algorithms, each with alternative parallel data layouts. The optimal choice may depend not only on the processor architecture but also on the parallelism of the computer and memory bandwidth. Consequently, in a parallel setting, the search space will be much larger than for traditional serial hardware.

The table lists the results of autotuning on three multicores for three kernels related to the dwarfs' sparse matrix, stencil for PDEs, and structured grids^{9,30,31} mentioned earlier. This autotuned code is the fastest known for these kernels for all three computers. Performance increased by factors of two to four over standard code, much better than you would expect from an optimizing compiler.

Efficiency-layer programmers will be able to build autotuners for use by domain experts and other efficiency-layer programmers to help deliver on the goals of efficiency, portability, and scalability.

Synthesis with sketching. One challenge for autotuning is how to produce the high-performance implementations explored by the search. One approach is to synthesize these complex programs. In doing so, we rely on the search for performance tuning, as well as for programmer productivity. To address the main challenge of traditional synthesis—the need for experts to communicate their insight with a formal domain theory—we allow that insight to be communicated directly by programmers who write an incomplete program, or “sketch.” In it, they provide an algorithmic skeleton, and the synthesizer supplies the low-level mechanics by filling in the holes in the sketch.

The synthesized mechanics could be barrier synchronization expressions or tricky loop bounds in stencil loops. Our sketching-based synthesis is to traditional, deductive synthesis what model checking is to theorem proving; rather than interactively deriving a program, our system searches a space of candidate programs with constraint solving. Efficiency is achieved by reducing the problem to one solved with two communicating SAT solvers. In future work, we hope to synthesize parallel sparse matrix codes and data-parallel algorithms for additional problems (such as parsing).

Verification and testing, not one or the other. Correctness is addressed differently at the two layers. The productivity layer is free from concurrency problems because the parallelism models are restricted, and the restrictions are enforced. The efficiency-layer code is checked automatically for subtle concurrency errors.

A key challenge in verification is obtaining specifications for programs to verify. Modular verification and automated unit-test generation require the specification of high-level semantic constraints on the behavior of the individual modules (such as parallel frameworks and parallel libraries). To simplify specification, we use executable sequential programs with the same behavior as a parallel component, augmented with atomicity constraints on a task,²¹ predicate abstractions of the interface of a module,¹⁴ or multiple ownership types.⁸

Programmers often find it difficult to specify such high-level contracts involving large modules; however, most find it convenient to specify local properties of programs using assert statements and type annotations. Local assertions and type annotations are often generated from a program's implicit correctness requirements (such as data race, deadlock freedom, and memory safety). The system propagates implications of these local assertions to the module boundaries through a combination of static verification and directed automated unit testing. These implications create serial contracts that specify how the modules (such as frameworks) are used correctly. When the contracts for the parallel modules are in place, programmers use static program verification to

Autotuned performance in GFLOPS/s on three kernels for dual-socket systems.

MPU Type	Intel e5345 Xeon 4 out-of-order cores, 2.3GH			AMD 2356 Opteron X4 4 out-of-order cores, 2.3GHz			Sun 5140 UltraSPARC T2 8 multithreaded cores, 1.2GHz		
	SpMV	Stencil	LBMHD	SpMV	Stencil	LBMHD	SpMV	Stencil	LBMHD
Kernel Optimization									
Standard	1.0	1.3	3.5	1.4	1.5	3.0	2.1	0.5	3.4
NUMA	1.0	—	3.5	2.4	2.6	3.7	3.5	0.5	3.8
Padding	—	1.3	4.5	—	3.1	5.8	—	0.5	3.8
Vectorization	—	—	4.6	—	—	7.7	—	—	9.7
Unrolling	—	1.7	4.6	—	3.6	8.0	—	0.5	9.7
Prefetching	1.1	1.7	4.6	2.9	3.8	8.1	3.6	0.5	10.5
Compression	1.5	—	—	3.6	—	—	4.1	—	—
\$/TLB block	—	2.2	—	—	4.9	—	—	5.1	—
Collab Thread	—	—	—	—	—	—	—	6.7	—
SIMD	—	2.5	5.6	—	8.0	14.1	—	—	—
Final	1.5	2.5	5.6	3.6	8.0	14.1	4.1	6.7	10.5

check if the client code composed with the contracts is correct.

Static program analysis in the presence of pointers and heap memory falsely reports many errors that cannot really occur. For restricted parallelism models with global synchronization, this analysis becomes more tractable, and a recently introduced technique called “directed automated testing,” or concolic unit testing, has shown promise for improving software quality through automated test generation using a combination of static and dynamic analyses.²¹ The Par Lab combines directed testing with model-checking algorithms to unit-test parallel frameworks and libraries composed with serial contracts. Such techniques enable programmers to quickly test executions for data races and deadlocks directly, since a combination of directed test input generation and model checking hijacks the underlying scheduler and controls the synchronization primitives. Our testing techniques will provide deterministic replay and debugging capabilities at low cost. We will also develop randomized extensions of our directed testing techniques to build a probabilistic model of path cover-

age. The probabilistic models will give a more realistic estimate of coverage of race and other concurrency errors in parallel programs.

Parallelism for energy efficiency. While the earlier computer classes—desktops and laptops—reused the software of their own earlier ancestors, the energy efficiency for handheld operation may need to come from data parallelism in tasks that are currently executed sequentially, possibly from three sources:

Efficiency. Completing a task on slow parallel cores will be more efficient than completing it in the same time sequentially on one fast core;

Energy amortization. Preferring data-parallel algorithms over other styles of parallelism, as SIMD and vector computers amortize the energy expended on instruction delivery; and

Energy savings. Message-passing programs may be able to save the energy used by cache coherence.

We apply these principles in our work on parallel Web browsers. In algorithm design, we observe that to save energy with parallelization, parallel algorithms must be close to “work efficient,” that is, they should perform no more total work than a sequential algorithm, or

else parallelization is counterproductive. The same argument applies to optimistic parallelization. Work efficiency is a demanding requirement, since, for some “inherently sequential” problems, like finite-state machines, only work-*inefficient* algorithms are known. In this context, we developed a nearly work-efficient algorithm for lexical analysis. We are also working on data-parallel algorithms for Web-page layout and identifying parallelism in future Web-browser applications, attempting to implement them with efficient message passing.

Space-time partitioning for deconstructed operating systems. Space-time partitioning is crucial for manycore client operating systems. A spatial partition (partition for short) is an isolated unit containing a subset of physical machine resources (such as cores, cache partitions, guaranteed fractions of memory or network bandwidth, and energy budget). Space-time partitioning virtualizes spatial partitions by time-multiplexing whole partitions onto available hardware but at a coarse-enough granularity to allow efficient programmer-level scheduling in a partition.

The presence of space-time partitioning leads to restructuring systems


services as a set of interacting distributed components. We propose a new “deconstructed OS” called Tessellation structured around space-time partitioning and two-level scheduling between the operating system and application runtimes. Tessellation implements scheduling and resource management at the partition granularity. Applications and OS services (such as file systems) run within their own partitions. Partitions are lightweight and can be resized or suspended with similar overheads to a process-context swap.

A key tenet of our approach is that resources given to a partition are either exclusive (such as cores or private caches) or guaranteed via a quality-of-service contract (such as a minimum fraction of network or memory bandwidth). During a scheduling quantum, the application runtime within a partition is given unrestricted “bare metal” access to its resources and may schedule tasks onto them in some way. Within a partition, our approach has much in common with the Exokernel.¹¹ In the common case, we expect many application runtimes to be written as libraries (similar to libOS). Our Tessellation kernel is a thin layer responsible for only the coarse-grain scheduling and assignment of resources to partitions and implementation of secure restricted communications among partitions. The Tessellation kernel is much thinner than traditional kernels or even hypervisors. It avoids many of the performance issues associated with traditional microkernels by providing OS services through secure messaging to spatially co-resident service partitions, rather than context-switching to time-multiplexed service processes.


Par Lab hardware tower. Past parallel projects were often driven by the hardware determining the application and software environment. The Par Lab is driven top down from the applications, so the question this time is what should architects do to help with the goals of productivity, efficiency, correctness, portability, and scalability?

Here are four examples of this kind of help that illustrate our approach:

Supporting OS partitioning. Our hardware architecture enforces partitioning of not only the cores and on-chip/off-chip memory but also the communication bandwidth among these com-



To save the IT industry, researchers must demonstrate greater end-user value from an increasing number of cores.



ponents, providing quality-of-service guarantees. The resulting performance predictability improves parallel program performance, simplifies code autotuning and dynamic load balancing, supports real-time applications, and simplifies scheduling.

Optional explicit control of the memory hierarchy. Caches were invented so hardware could manage a memory hierarchy without troubling the programmer. When it takes hundreds of clock cycles to go to memory, programmers and compilers try to reverse-engineer the hardware controllers to make better use of the hierarchy. This backward situation is especially apparent for hardware prefetchers when programmers try to create a particular pattern that will invoke good prefetching. Our approach aims to allow programmers to quickly turn a cache into an explicitly managed local store and the prefetch engines into explicitly controlled Direct Memory Access engines. To make it easy for programmers to port software to our architecture, we also support a traditional memory hierarchy. The low-overhead mechanism we use allows programs to be composed of methods that rely on local stores and methods that rely on memory hierarchies.

Accurate, complete counters of performance and energy. Sadly, performance counters on current single-core computers often miss important measurements (such as prefetched data) or are unique to a computer and only understandable by the machine’s designers. We will include performance enhancements in the Par Lab architecture only if they have counters to measure them accurately and coherently. Since energy is as important as performance, we also include energy counters so software can improve both. Moreover, these counters must be integrated with the software stack to provide insightful measurements to the efficiency-layer and productivity-layer programmers. Ideally, this research will lead to a standard for performance counters so schedulers and software development kits can count on them on any multicore.

Intuitive performance model. The multicore diversity mentioned earlier exacerbates the already difficult jobs performed by programmers, compiler writers, and architects. Hence, we developed an easy-to-understand visual

model with built-in performance guidelines to identify bottlenecks in the dozen dwarfs in Figure 3.²⁹ The Roofline model plots computational and memory-bandwidth limits, then determines the best possible performance of a kernel by examining the average number of operations per memory access. It also plots ceilings below the “roofline” to suggest the optimizations that might be useful for improving performance. One goal of the performances counters should be to provide everything needed to automatically create Roofline models.

A notable challenge from our earlier description of the hardware tower is how to rapidly innovate at the hardware/software interface, when it can take four to five years to build chips and run programs needed to evaluate them. Given the capacity of field-programmable gate arrays (FPGAs), researchers can prototype full hardware and software systems that run fast enough to investigate architectural innovations. This flexibility means researchers can “tape out” every day, rather than over years. We will leverage the Research Accelerator for Multiple Processors (RAMP) Project (<http://ramp.eecs.berkeley.edu/>) to build flexible prototypes fast enough to run full software stacks—including new operating systems and our five compelling applications—to enable rapid architecture innovation using future prototype software, rather than past benchmarks.²⁸

Reasons for Optimism

Given the history of parallel computing, it’s easy to be pessimistic about our chances. The good news is that there are plausible reasons researchers could succeed this time:

No killer microprocessor. Unlike in the past, no one is building the faster serial microprocessor; programmers needing more performance have no option other than parallel hardware;

New measures of success. Rather than the traditional goal of linear speedup for all software as the number of processors increases, success can reflect improved responsiveness or MIPS/Joule for a few new parallel killer apps;

All the wood behind one arrow. As there is no alternative, the whole IT industry is committed, meaning many more people and companies are working on the problem;

Manycore synergy with cloud computing. SaaS applications in data centers with millions of users are naturally parallel and thus aligned with manycore, even if clients apps are not;

Vitality of open source software. The OSS community is a meritocracy, so it’s likely to embrace technical advances rather than be limited by legacy code. Though OSS has existed for years, it is more important commercially today than it was;

Single-chip multiprocessors enable innovation. Having all processors on the same chip enables inventions that were impractical or uneconomical when spread across many chips; and

FPGA prototypes shorten the hardware/software cycle. Systems like RAMP help researchers explore designs of easy-to-program manycore architectures and build prototypes more quickly than they ever could with conventional hardware prototypes.

Given the importance of the challenges to our shared future in the IT industry, pessimism is not a sufficient excuse to sit on the sidelines. The sin is not lack of success but lack of effort.

Related Projects

Computer science hasn’t solved the parallel challenge though not because it hasn’t tried. There could be a dozen conferences dedicated to parallelism, including Principles and Practice of Parallel Programming, Parallel Algorithms and Architectures, Parallel and Distributed Processing, and Supercomputing. All traditionally focus on high-performance computing; the target hardware is usually large-scale computers with thousands of microprocessors. Similarly, there are many high-performance computing research centers. Rather than review this material, here we highlight four centers focused on multicore computers and their approaches to the parallel challenge in academia:

Illinois. The Universal Parallel Computing Research Center (<http://www.upcr.illinois.edu/>) at the University of Illinois focuses on making it easy for domain experts to take advantage of parallelism, so the emphasis is more on productivity in specific domains than on generality or performance.¹ It relies on advancing compiler technology to find opportunities for parallelism, whereas the Par Lab focuses on autotuning. The

Center is pursuing deterministic models that allow programmers to reason with sequential semantics for testing while naturally exposing a parallel performance model for WYSIWYG performance. For reactive programs where parallelism is part of the problem, it is pursuing a shared-nothing approach that leverages actor-like models used in distributed systems. For application domains that allow greater specialization, it is developing a framework to generate domain-specific environments that either hide concurrency or expose only specialized forms of concurrency to the end user while exploiting domain-specific optimizations and performance measures. Initial applications and domains include teleimmersion via “virtual teleportation” (multimedia), dynamic real-time virtual environments (computer graphics), learning by reading, and authoring assistance (natural language processing).

Stanford. The Pervasive Parallelism Laboratory (http://ppl.stanford.edu/wiki/index.php/Pervasive_Parallelism_Laboratory) at Stanford University takes an application-driven approach toward parallel computing that extends from programming models down to hardware architecture. The key technical concepts are domain-specific languages for increasing programmer productivity and a common parallel runtime environment combining dynamic and static approaches for concurrency and locality management. There are domain-specific languages for artificial intelligence and robotics, business data analysis, and virtual worlds and gaming. The experimental platform is the Flexible Architecture Research Machine, or FARM, system, combining commercial processors with FPGAs in the memory fabric.

Georgia Tech. The Sony, Toshiba, IBM Center of Competence for the Cell Broadband Engine Processor (<http://sti.cc.gatech.edu/>) at Georgia Tech focuses on a single multicore computer, as its name suggests. Researchers explore versions of programs on Cell, including image compression⁶ and financial modeling.² The Center also sponsors workshops and provides remote access to Cell hardware.

Rice University. The Habanero Multicore Software Project (http://habanero.rice.edu/Habanero_Home.html) at

Rice University is developing languages, compilers, managed runtimes, concurrency libraries, and tools that support portable parallel abstractions with high productivity and high performance for multicores; examples include parallel language extensions²⁵ and optimized synchronization primitives.²⁴

Conclusion

We've provided a general view of the parallel landscape, suggesting that the goal of computer science should be making parallel computing productive, efficient, correct, portable, and scalable. We highlighted the importance of finding new compelling applications and the advantages of manycore and heterogeneous hardware. We also described the research of the Berkeley Par Lab. While it will take years to learn which of our ideas work well, we share it here as a concrete example of a coordinated attack on the problem.

Unlike the traditional approach of making hardware king, the Par Lab is application-driven, working with domain experts to create compelling applications in music, image- and speech-recognition, health, and parallel browsers.

The software span connecting applications to hardware relies more on parallel software architectures than on parallel programming languages. Instead of traditional optimizing compilers, we depend on autotuners, using a combination of empirical search and performance modeling to create highly optimized libraries tailored to specific machines. By splitting the software stack into a productivity layer and an efficiency layer and targeting them at domain experts and programming experts respectively, we hope to bring parallel computing to all programmers while keeping domain experts productive and allowing expert programmers to achieve maximum efficiency. Our approach to correctness relies on verification where possible, then uses the same tools to reduce the amount of testing where verification is not possible.

The hardware tower of the Par Lab serves the software span and application tower. Examples of such service include support for OS partitioning, explicit control for the memory hierarchy, accurate measurement for performance and energy, and an intuitive, multicore

performance model. We also plan to try to scrape off the barnacles that have accumulated on the hardware/software stack over the years.

This parallel challenge offers the worldwide research community an opportunity to help IT remain a growth industry, sustain the parts of the worldwide economy that depend on the continuous improvement in IT cost-performance, and take a once-in-a-career chance to reinvent the whole software/hardware stack. Though there are reasons for optimism, the difficulty of the challenge is reflected in the numerous parallel failures of the past.

Combining upside and downside, this research challenge represents the most significant of all IT challenges over the past 50 years. We hope many more innovators will join this quest to build a parallel bridge.

Acknowledgments

This research is sponsored by the Universal Parallel Computing Research Center, which is funded by Intel and Microsoft (Award # 20080469) and by matching funds from U.C. Discovery (Award #DIG07-10227). Additional support comes from the Par Lab Affiliate companies: National Instruments, NEC, Nokia, NVIDIA, and Samsung. We wish to thank our colleagues in the Par Lab and the Lawrence Berkeley National Laboratory collaborations who shaped these ideas. □

References

1. Adve, S. et al. *Parallel Computing Research at Illinois: The UPCRC Agenda*. White Paper. University of Illinois, Urbana-Champaign, IL, Nov. 2008.
2. Agarwal, V., Liu, L.-K., and Bader, D. Financial modeling on the Cell broadband engine. In *Proceedings of 22nd IEEE International Parallel and Distributed Processing Symposium* (Miami, FL, Apr. 14–18, 2008).
3. Alexander, C. et al. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1997.
4. Asanovic, K. et al. *The Parallel Computing Laboratory at U.C. Berkeley: A Research Agenda Based on the Berkeley View*. UCB/ECS-2008-23, University of California, Berkeley, Mar. 21, 2008.
5. Asanovic, K. et al. *The Landscape of Parallel Computing Research: A View from Berkeley*. UCB/ECS-2006-183, University of California, Berkeley, Dec. 18, 2006.
6. Bader, D.A. and Patel, S. High-performance MPEG-2 software decoder on the Cell broadband engine. In *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium* (Miami, FL, Apr. 14–18, 2008).
7. Buschmann, F. et al. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, Inc., New York, 1996.
8. Clarke, D.G. et al. Ownership types for flexible alias protection. In *Proceedings of the OOPSLA Conference* (Vancouver, BC, Canada, 1998), 48–64.
9. Datta, K. et al. Stencil computation optimization and autotuning on state-of-the-art multicore architectures. In *Proceedings of the ACM/IEEE Supercomputing (SC) 2008 Conference* (Austin, TX, Nov. 15–21). IEEE Press, Piscataway, NJ, 2008.

10. Demmel, J., Dongarra, J., Eijkhout, V., Fuentes, E., Petitet, A., Vuduc, R., Whaley, R., and Yelick, K. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE, Special Issue on Program Generation, Optimization, and Adaptation* 93, 2 (Feb. 2005), 293–312.
11. Engler, D.R. Exokernel: An operating system architecture for application-level resource management. In *Proceedings of the 15th Symposium on Operating Systems Principles* (Cooper Mountain, CO, Dec. 3–6, 1995), 251–266.
12. Gamma, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Reading, MA, 1994.
13. Gelernter, D. and Carriero, N. Coordination languages and their significance. *Commun. ACM* 35, 2 (Feb. 1992), 97–107.
14. Henzinger, T.A. et al. Permissive interfaces. In *Proceedings of the 10th European Software Engineering Conference* (Lisbon, Portugal, Sept. 5–9). ACM Press, New York, 2005, 31–40.
15. Hill, M. and Marty, M. Amdahl's Law in the multicore era. *IEEE Computer* 41, 7 (2008), 33–38.
16. International Technology Roadmap for Semiconductors. Executive Summary, 2005 and 2007; <http://public.itrs.net/>.
17. Kantowitz, B. and Sorkin, R. *Human Factors: Understanding People-System Relationships*. John Wiley & Sons, Inc., New York, 1983.
18. Mattson, T., Sanders, B., and Massingill, B. *Patterns for Parallel Programming*. Addison-Wesley Professional, Reading, MA, 2004.
19. O'Hanlon, C. A conversation with John Hennessy and David Patterson. *Queue* 4, 10 (Dec. 2005/Jan. 2006), 14–22.
20. Patterson, D. and Hennessy, J. *Computer Organization and Design: The Hardware/Software Interface, Fourth Edition*. Morgan Kaufmann Publishers, Boston, MA, Nov. 2008.
21. Sen, K. and Viswanathan, M. Model checking multithreaded programs with asynchronous atomic methods. In *Proceedings of the 18th International Conference on Computer-Aided Verification* (Seattle, WA, Aug. 17–20, 2006).
22. Sen, K. et al. CUTE: A concolic unit testing engine for C. In *Proceedings of the Fifth Joint Meeting European Software Engineering Conference* (Lisbon, Portugal, Sept. 5–9). ACM Press, New York, 2005, 263–272.
23. Shaw, M. and Garland, D. *An Introduction to Software Architecture*. Technical Report CMU/SEI-94-TR-21, ESC-TR-94-21. CMU Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1994.
24. Shirako, J., Peixotto, D., Sarkar, V., and Scherer, W. Phasers: A unified deadlock-free construct for collective and point-to-point synchronization. In *Proceedings of the 22nd ACM International Conference on Supercomputing* (Island of Kos, Greece, June 7–12). ACM Press, New York, 2008, 277–288.
25. Shirako, J., Kasahara, H., and Sarkar, V. Language extensions in support of compiler parallelization. In *Proceedings of the 20th Workshop on Languages and Compilers for Parallel Computing* (Urbana, IL, Oct. 11–13). Springer-Verlag, Berlin, 2007, 78–94.
26. Thomas, D. et al. *Agile Web Development with Rails, Second Edition*. The Pragmatic Bookshelf, Raleigh, NC, 2008.
27. UPC Language Specifications, Version 1.2. Technical Report LBNL-59208. Lawrence Berkeley National Laboratory, Berkeley, CA, 2005.
28. Wawrzynek, J. et al. RAMP: Research Accelerator for Multiple Processors. *IEEE Micro* 27, 2 (Mar. 2007), 46–57.
29. Williams, S., Waterman, A., and Patterson, D. Roofline: An insightful visual performance model for floating-point programs and multicore architectures. *Commun. ACM* 52, 4 (Apr. 2009), 65–76.
30. Williams, S. et al. Lattice Boltzmann simulation optimization on leading multicore platforms. In *Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium* (Miami, FL, Apr. 14–18, 2008).
31. Williams, S. et al. Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In *Proceedings of the Supercomputing (SC07) Conference* (Reno, NV, Nov. 10–16). ACM Press, New York, 2007.

The authors are all affiliated with the Par Lab (<http://parlab.eecs.berkeley.edu/>) at the University of California, Berkeley.

DOI:10.1145/1562764.1562784

The result is stable, focused, dynamic discussion threads that avoid redundant ideas and engage thousands of stakeholders.

BY JANE CLELAND-HUANG, HORATIU DUMITRU, CHUAN DUAN, AND CARLOS CASTRO-HERRERA

Automated Support for Managing Feature Requests in Open Forums

AS SOFTWARE PROJECTS grow larger and more complex and involve more stakeholders across geographical and organizational boundaries, project managers increasingly rely on open discussion forums to elicit requirements and otherwise communicate with other stakeholders. Unfortunately, open forums generally don't support all aspects of the requirements-elicitation process; for example, in most forums, stakeholders create their own discussion threads, introducing significant redundancy of ideas and possibly causing them to miss important discussions.

Here, we describe an automated forum management (AFM) system we designed to organize discussion threads in open forums, exploring it through a series of case studies and experiments using feature requests mined from open source forums.

Using Web-based forums to gather requirements from stakeholders is common in open source forums and increasingly common across a number of enterprise-level projects. They can grow quite large, with many thousands of stakeholders. Most use discussion threads to help focus the conversation; however, user-defined threads tend to result in redundant ideas, while predefined categories might be overly rigid, possibly leading to coarse-grain topics that fail to facilitate focused discussions.

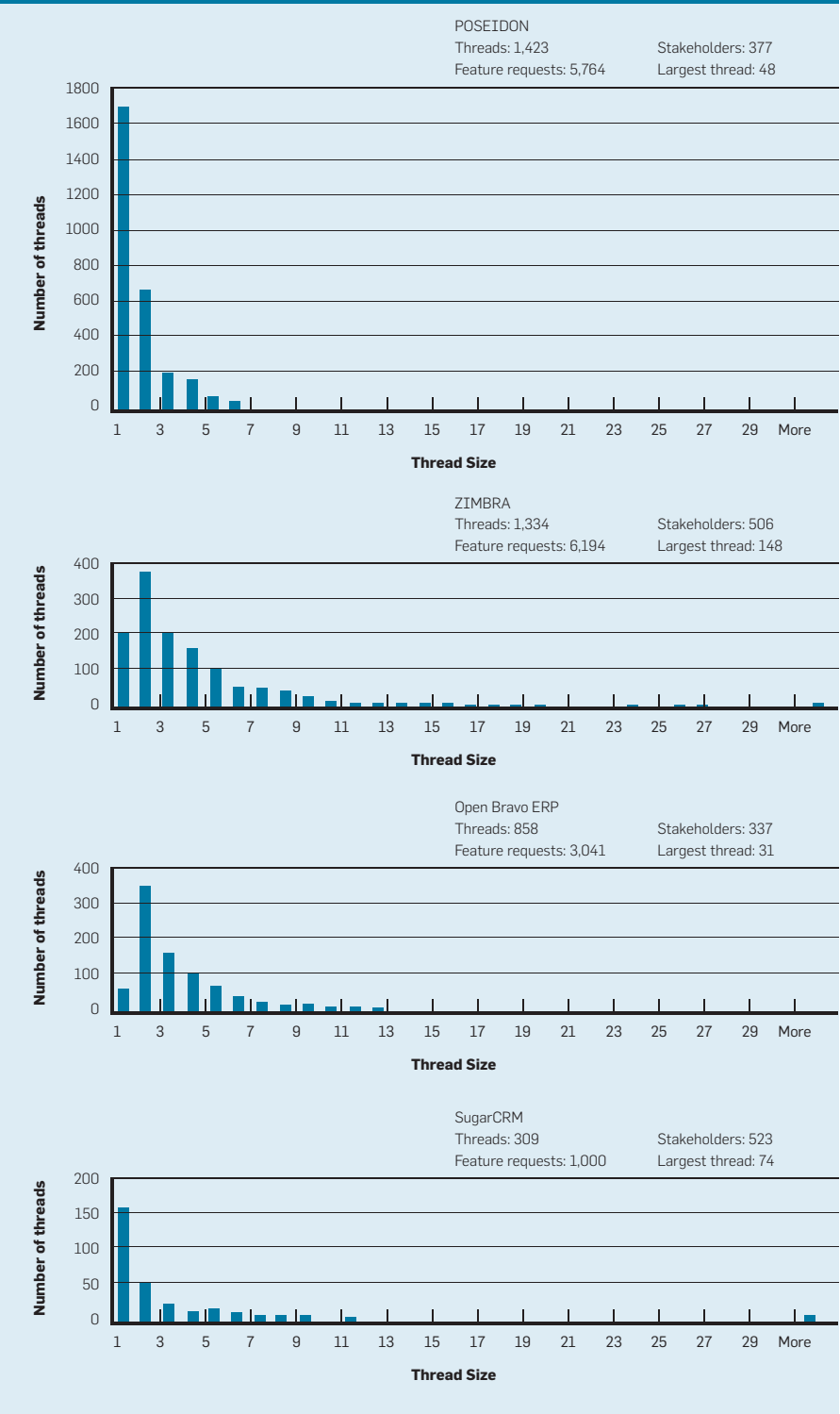
To better understand the problems and challenges of capturing requirements in open forums, we surveyed several popular open source projects, evaluating how the structure of their forums and organization of their feature requests help stakeholders work collaboratively toward their goals. The survey covered a number of tools and frameworks: a customer relationship management system called SugarCRM¹¹; a UML modeling tool called Poseidon¹⁰; an enterprise resource planning tool called Open Bravo¹⁰; a groupware tool called ZIMBRA¹⁰; a Web tool for administering the MySQL server called PHP-MyAdmin¹⁰; an open .NET architecture for Linux called Mono¹⁰; and the large Web-based immersive game world *Second Life*.⁹ All exhibited a significantly high percentage of discussion threads consisting of only one or two feature requests. For example, as shown in Figure 1, 59% of Poseidon threads, 70% of SugarCRM threads, 48% of Open Bravo threads, and 42% of Zimbra threads included only one or two requests. The presence of so many small threads suggests either a significant number of distinct discussion topics or that users had created unnecessary new threads. An initial analysis found the second case—unnecessary new threads—held for all forums we surveyed; for example in

the SugarCRM forum, stakeholders' requests related to email lists were found across 20 different clusters, 13 of which included only one or two comments.

We designed an experiment to determine quantitatively if user feature requests and comments (placed in new threads) should instead have been placed in larger preexisting threads. We conducted it using the data from SugarCRM, an open source customer relationship management system supporting campaign management, email marketing, lead management, marketing analysis, forecasting, quote management, and case management. We mined 1,000 feature requests contributed by 523 different stakeholders over a two-year period (2006–2008) distributed across 309 threads from the open discussion forum. Of the requests, 272 had been placed (by users) into discussion threads with no more than two feature requests, 309 had been placed in groups with nine or more requests, and the rest were placed in intermediate-size groups. We clustered all the feature requests using a consensus Spherical K-Means algorithm (described in the following sections), then estimated the degree to which each feature request belonged to a topic by computing the proximity of the request to the centroid, or center, of its assigned cluster.

We hypothesized that feature requests representing unique topics would have low proximity scores, while those fitting well into a topic would have higher scores. Figure 2 shows the results for feature requests assigned to small groups with only one or two feature requests vs. those assigned to larger groups of nine or more requests. We performed a T-Test that returned a p-value of 0.0005, showing the distributions exhibited a significant difference; nevertheless, there was a high degree of overlap between the two distributions, suggesting that many of the features assigned to individual or small threads could have been placed with other feature requests in larger threads. It was also reasonable to assume that once stakeholders found a relevant thread and added a feature request to it, their choice of words would have been influenced by words in the existing thread, thereby increasing the similarity of feature requests placed in shared threads. We observed that users

Figure 1. Forums are characterized by numerous small threads, many with only one or two feature requests.



often responded directly to earlier entries in the thread, possibly accounting for some of the differences in proximity scores between the two groupings. The results from the experiment and our subjective observations generally suggest that in many cases users incorrectly decided to create new threads.

Automated Approach

Many such problems can be alleviated through AFM tools that employ data clustering techniques to create distinct, highly focused discussion threads. However, clustering is hindered by significant background noise in feature requests (such as spelling errors, poor

Figure 2. The extent to which feature requests assigned to individual vs. larger threads fit into global topics.

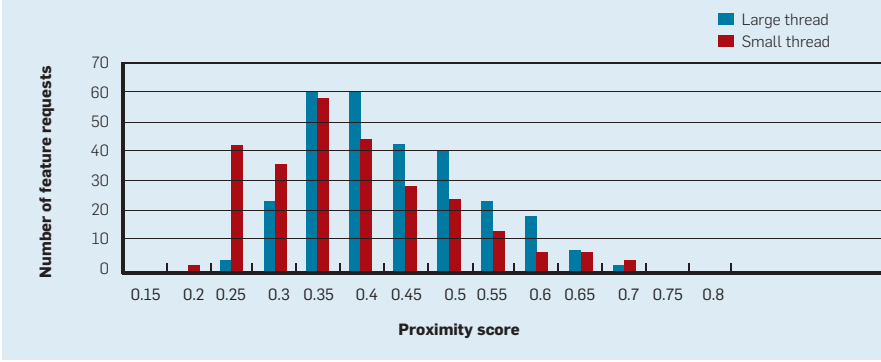


Figure 3. Two-stage spherical K-means.

Algorithm: Two-stage spherical K-means clustering

Input: unlabeled instances $\mathcal{X} = \{x_i\}_{i=1}^N$, number of clusters K , initial centroids I , convergence condition

Output: crisp K-partition $P = [c_i]_{i=1}^K$.

Steps:

1. Initialization: initialize centroids using $I: \{\mu_i^0\}_{i=1}^K = I; t = 0$.
2. Assign instances to centroids and update centroids until convergence
 - 2a. assign each instance x to the nearest cluster
 - 2b. update each centroid to the average of its assigned instances:

$$\mu_i^{t+1} = \frac{\sum_{x \in c_i^{t+1}} x}{|c_i^{t+1}|}$$

2c. $t = t + 1$

3. Incremental optimization until convergence:
 - 3a. randomly select an instance x , move it to another cluster that maximizes the objective function

$$J_{sp} = \sum_{i=1}^K \sum_{x \in c_i^t} x^T \mu_i^t \therefore$$

- 3b. update each centroid as in step 2b.
- 3c. $t = t + 1$

grammar, slang, long-winded comments, nonstandard abbreviations, redundancies, inconsistent use of terms, and off-topic discussions). To be effective, an AFM must deliver high-quality clusters representing a focused theme and be distinct from other clusters in order to minimize the redundancy of ideas across discussions. Clustering algorithms must also execute quickly, so clustering occurs inconspicuously in the background. Finally, as open-discussion forums are characterized by a steady influx of feature requests, clusters must be relatively stable, so requests are not constantly moved from cluster to cluster.

We designed the AFM process to meet these goals. Once an initial set of feature requests is collected, the project manager places the forum under the management of the AFM system, and existing requests are analyzed to identify themes and generate discussion threads. Beyond this point, arriving feature requests are classified into existing threads, unless a new topic is detected, in which case a new thread is created. Here, we describe these processes and the underlying algorithms we developed as the result of extensive experimentation in clustering requirements techniques.^{6,7}

In preparation for clustering, fea-

ture requests are preprocessed to remove common words (such as “this” and “because”) not useful for identifying underlying themes. The remaining terms are then stemmed to their grammatical roots. Each feature request x is represented as a vector of terms $(t_{x,1}, t_{x,2}, \dots, t_{x,w})$ that is then normalized through a technique called term frequency, inverse document frequency (tf-idf), where tf represents the original term frequency of term t in the feature request, and idf represents the inverse document frequency; idf is often computed as $\log_2(N/df_t)$, where N represents the number of feature requests in the entire forum, and df_t represents the number of feature requests containing t . The similarity between each normalized requests vector $a=(a_1, a_2, \dots, a_w)$ and each centroid $b=(b_1, b_2, \dots, b_w)$ is then computed as

$$sim(a, b) = \frac{\sum_{i=1}^w a_i \cdot b_i}{\sqrt{\sum_{i=1}^w a_i^2 \cdot \sum_{i=1}^w b_i^2}},$$

where W represents the total number of terms in the entire set of feature requests, and a_i is computed as the number of times term t_i occurs in feature request a , weighted according to the idf value. Intuitively, this formula returns higher similarity scores between two feature requests if they share a significant number of relatively rare terms. We then determine an appropriate cluster granularity through a technique devised by F. Can and E.A. Ozkarahan² in 1990 that predicts the ideal number of clusters by considering the degree each feature request differentiates itself from other such requests. The ideal number of clusters K is computed as

$$K = \sum_{x \in D} \sum_{i=1}^w \frac{f_{x,i}}{|x|} * \frac{f_{x,i}}{N_i} = \sum_{x \in D} \frac{1}{|x|} * \sum_{i=1}^w \frac{f_{x,i}^2}{N_i}$$

where $f_{x,i}$ is the frequency of term t_i in artifact x , $|x|$ is the length of the artifact, and N_i is the total occurrence of term t_i . This approach has been shown to be effective across multiple data sets^{2,6} and can be calculated quickly so the ideal number of clusters is recomputed frequently.

Our approach uses a clustering algorithm called Spherical K-Means (SPK)⁵ that exhibits fast running times and returns relatively high-quality results.⁶ As described more formally in Figure 3, a set of K centroids, or seeds, is initially

selected by the algorithm, with the objective that they are as dissimilar from each other as possible. The distance from each feature request to each centroid is computed, and each feature request is placed in the cluster associated with its closest centroid. Once all feature requests are assigned to clusters, the centroids are repositioned so as to increase their average proximity to all feature requests in the cluster. This is followed by a series of incremental optimizations in which an attempt is made to move a randomly selected feature request to the cluster for which it maximizes the overall cohesion gain. The process continues until no further optimization is possible. This simple post-processing step has been shown to significantly improve the quality of clustering results.⁶

Clustering results are strongly influenced by initial centroid selection, meaning poor selection can lead to low-quality clusterings. However, this problem can be alleviated through a consensus technique for performing the initial clustering, an approach that generates multiple individual clusterings, then uses a voting mechanism to create a final result.⁸ Though it has a much longer running time than SPKMeans, consensus clustering has been shown to consistently deliver clusterings that are of higher-than-average quality compared to the standalone SPK clusterings.⁶ In the AFM system, consensus clustering is used only for the initial clustering in order to create the best possible set of start-up threads.

Following the arrival of each new feature request, the algorithm recomputes the ideal granularity to determine if a new cluster should be added. To add a new cluster in a way that preserves the stability of existing clusters and minimizes clustering time, the AFM approach identifies the least-cohesive cluster, then bisects it using SPK, with $K = 2$. Feature requests from neighboring clusters are then reevaluated to determine if they exhibit closer proximity to one of the two new centroids than they do to their own currently assigned centroids. If this closer proximity occurs they are reassigned to the relevant cluster. To ensure continued cluster quality, the entire data set is re-clustered periodically following the arrival of a fixed number of new

feature requests. Re-clustering is performed through a modified SPKMeans algorithm (we call Stable SPKMeans) designed to minimize the movement of feature requests between clusters through reuse of the current set of centroids as seeds for the new clustering.

Cluster quality is also improved through user feedback specifying whether a pair of feature requests belong together. For example, users not happy with the quality of a cluster can specify that a given feature request does not fit the cluster. They might also provide additional tags to help place the feature request in a more appropriate cluster. These user constraints, along with the tag information, are then incorporated into the SPK algorithm of future clusterings. This reassignment maximizes the quality of the individual clusters and optimizes conformance to user constraints. Our prior work in this area demonstrated significant improvement in cluster quality when constraints are considered.¹³

Evaluating AFM

We conducted a series of experiments designed to evaluate the AFM's ability to quickly deliver cohesive, distinct, and stable clusters. They utilized three data sets: the first was the SugarCRM data set discussed earlier; the second

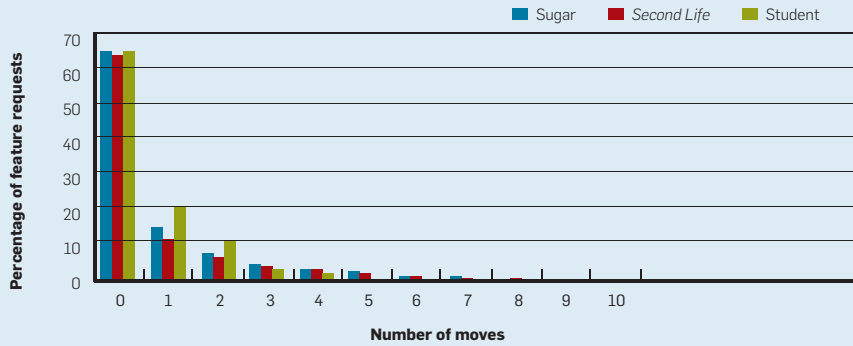
included 4,205 feature requests we mined from *Second Life*, an Internet-based virtual world video game in which stakeholders are represented by interacting avatars; and the third described the features of an online Amazon-like portal designed specifically for students. In spring 2008 we asked 36 master's-level students enrolled in two different advanced software-engineering classes at DePaul University to consider the needs of a typical student, textbook reseller, textbook author, textbook publisher, instructor, portal administrator, architect, project manager, and developer and create relevant feature requests. The result was 366 feature requests.

To evaluate cluster quality, we constructed an "ideal" clustering for the SugarCRM data by reviewing and modifying the natural discussion threads created by SugarCRM users. Modifications included merging closely related singleton threads, decomposing large megathreads into smaller more cohesive ones, and reassigning misfits to new clusters. The answer set enabled us to compare the quality of the generated clusters using a metric called Normalized Mutual Information (NMI) to measure the extent to which the knowledge of one cluster reduces uncertainty of other clusters.⁹ On a scale of 0 (no

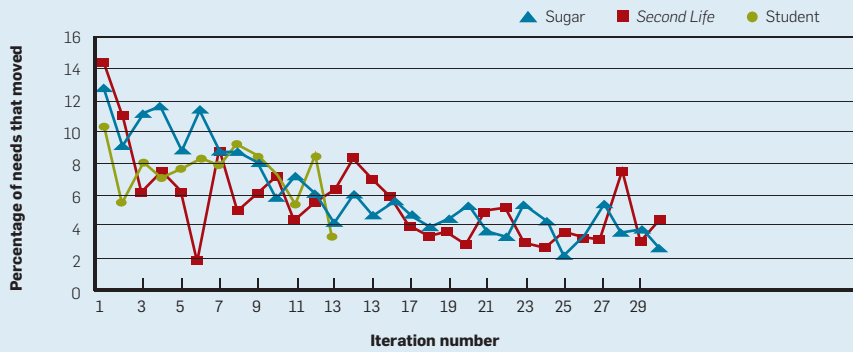
Figure 4. A partial cluster with security-related requirements generated after gathering 1,000 constraints from the Student data set.

- (1) The system must protect stored confidential information.
- (2) The system must encrypt purchase/transaction information.
- (3) A privacy policy must describe in detail to the users how their information is stored and used.
- (4) Transmission of personal information must be encrypted.
- (5) Transmission of financial transactions must be encrypted.
- (6) The system must use both encrypt and decrypt in some fields.
- (7) The system must allow users to view their previous transactions.
- (8) Databases must use the TripleDES encryption standard for database security. AES is still new and has had compatibility issues with certain types of databases, including SQL Server express edition.
- (9) The site must ensure that payment information is confidential and credit card transactions are encrypted to prevent hackers from retrieving information.
- (10) Because the system will be used to buy books, we must focus on security and consider transaction control in the architecture used to build it.
- (11) Correct use of cryptography techniques must be applied in the Amazon portal system to protect student information from outsiders and staff who might potentially acquire the information if left unprotected.
- (12) Sessions that handle payment transactions must be encrypted.

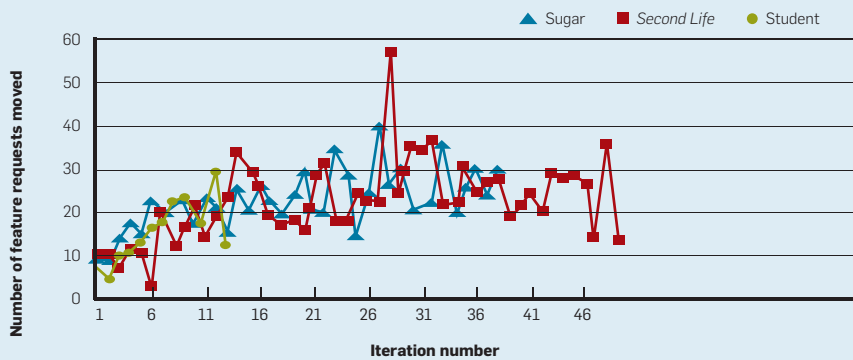
Figure 5. Stability of threads in the AFM clustering process.



a. Number of moves per feature request from beginning to end of the incremental clustering process; movements of < 0.1% are not shown.



b. Percentage of feature requests moved during the incremental clustering process.



c. Number of feature requests moved during the incremental clustering process.

similarity) to 1 (identical clusterings), the SugarCRM clusterings scored 0.57, indicating some degree of similarity between the generated cluster and the answer set. In experiments using the answer set to simulate the gathering of nonconflicting user feedback, NMI scores increased to 0.62 after 1,000 pairwise constraints were randomly selected from the pairs of requests exhibiting borderline (neither very strong nor very weak) proximity scores⁶ We also created an answer set for the Student Portal data set; initial NMI cluster-

ing scores of 0.58 increased to 0.7 following 1,000 pairwise constraints. Figure 4 outlines a cohesive cluster of feature requests for the Student data sets generated using 1,000 system-wide constraints. Note that 1,000 constraints represent only 1.5% of possible constraints for the Student Portal data set and only 0.2% for SugarCRM.

These results demonstrate that many of the themes we identified in our two answer sets were also detected by the unsupervised clustering algorithms. However, because more than

one clustering is possible for a given data set, these metrics alone lack sufficient insight to judge whether the cluster quality is better or worse than the user-defined threads. NMI metrics could not be used to evaluate user-defined threads due to the significant disparity between the number of user-defined threads and the number of threads in the answer set.

We thus conducted a case study using the SugarCRM data set to compare the treatment of four topics in the original SugarCRM user-managed forum versus the AFM approach. We selected the topics by examining an unstructured list of feature requests, including distribution lists, appointment scheduling, language support, and document attachments. For each topic, we first identified the set of related feature requests, then compared the way they had been allocated to both user-defined and AFM-generated threads (see Table 1). We did not solicit user feedback for these experiments.

We identified 14 feature requests for the first topic, distribution lists. Human users placed them in a thread called “email management,” including a total of 74 feature requests. The AFM system similarly placed all requests in the thread called “send email to sugar contacts.” In this case, both approaches were relatively successful.

For the second topic, appointment scheduling, we identified 32 feature requests. Human stakeholders placed them across six different threads, with 15 in a thread called “scheduling,” 12 in a thread called “how to improve calendaring [sic] and activities in sugar sales,” two in a thread called “calendar,” and one each in threads related to mass updates, printing, and email management. The AFM placed 18 feature requests in a discussion related to daily appointment scheduling, another 10 in a discussion related to meeting scheduling, and the remaining four in two clusters related to email, Web-enabled calendars, and integration with other tools. In this case—appointment scheduling—the AFM performed marginally better than the user-defined threads, as the feature requests were slightly less dispersed.

For the third topic, language support, we identified 11 feature requests. Human stakeholders placed them across

seven relatively small threads epitomizing the kinds of problems we found in the open source forums we studied. The AFM created a focused discussion forum on the topic of languages in which it placed nine of the 11 requests. The AFM approach excelled here.

The fourth and final topic, attach documents, represents a case in which a fairly dominant concern was dispersed by human users across 13 different threads, while the AFM placed all related requests in a single highly focused discussion thread.

It was evident that in each of the four cases the AFM approach either matched or improved on the results of the user-created threads.

AFM was designed to minimize the movement of feature requests among clusters in order to preserve stability and quality. In a series of experiments against the SugarCRM, *Second Life*, and

Student data sets, we found that the Stable SPKMeans clustering algorithm had no significant negative effect on the quality of the final clusters. In fact, the NMI scores for the two data sets—SugarCRM and Student for which “target clusterings” were available—showed a slight improvement when we used the stable algorithm.

To evaluate the stability of the modified SPKMeans algorithm, we tracked the movement of feature requests between clusters for re-clustering intervals of 25 feature requests. Figure 5a shows the number of moves per feature request, reported in terms of percentage. Approximately 62%–65% of feature requests were never moved, 20%–30% of feature requests were moved once or twice, and only about 5%–8% were moved more frequently. A significant amount of this movement is accounted for by the arrival of new topics and the

subsequent creation of new threads, causing reclassification of some existing feature requests. Figure 5b shows that the percentage of feature requests moved across clusters gradually decreases across subsequent iterations. However, as shown in Figure 5c, the actual number of movements increases in early iterations, then becomes relatively stable. Though not discussed here, we also conducted extensive experiments that found the traditional unmodified SPKMeans algorithm resulted in approximately 1.6–2.6 times more volatility of feature requests than our modified version.

We analyzed the total execution time of the Stable SPKMeans algorithm for each of the three data sets using MATLAB on a 2.33GHz machine; Table 2 outlines the total time needed to perform the clusterings at various increments. We excluded initial consensus

Table 1. Four topics in human vs. automated thread management.

Topic	User-defined threads	# FRs	Cluster Size	AFM-defined threads	# FRs	Cluster Size
Distribution lists	Email management	14	74	Send email to sugar contacts	14	62
	Scheduling	15	37	View daily appointments in calendar	18	58
Scheduling appointments	How to improve calendaring and activities in sugar sales	12	42	Schedule meetings	10	39
	Calendar	2	32	Calendar support features	3	27
	Mass updates	1	14	Send email to sugar contacts	1	62
	Printing	1	5			
	Email management	1	74			
	Language support	International salutations using one language pack	2	3	Contact languages	9
	Customer salutation in email template	2	2	Users roles and defaults	1	32
	Outlook category synchronization	2	7	Contact categories and fields	1	13
	Project: Translation system	2	4			
	Fallback option in language files	1	1			
	Language Preference option	1	1			
	A simplified workflow for small business	1	7			
Document attachments	Attaching documents to cases, projects	6	7	Attach and manage documents	25	32
	Attachments for bug tracker	1	1			
	Display attachment	3	3			
	Display size of attachments and sugar documents	1	1			
	Documents to a customer	2	3			
	Email management	2	74			
	File upload field	2	5			
	Koral document management integration	1	1			
	Link documents with project	2	2			
	Module builder	1	6			
	New feature request: Attach file to an opportunities	1	1			
	View all in all lists	2	3			
	WebDAV access to "documents"	1	3			

Note: AFM-generated threads are initially labeled according to the most common terms and then renamed by stakeholders. For example, the thread “Send email to sugar contacts” was first generated as “email, send, list, sugar, contact”

The threads here have been renamed.

Spelling and grammar is maintained from the original user forums.

clustering times that, with typical parameters of 50–100 requests, could take up to two minutes. As depicted in this example, the SugarCRM data set took a total of 57.83 seconds to complete all 29 clusterings at increments of 50 feature requests. Individual clusterings are notably fast; for example, the complete *Second Life* data set, consisting of 4,205 requests, was clustered in 75.18 seconds using standard SPKMeans and in only 1.02 seconds using our stable approach.

The Stable SPKMeans algorithm significantly improves the performance of the AFM, mainly because it takes less time to converge on a solution when quality seeds are passed forward from the previous clustering. Smaller increments require more clusterings, so the overall clustering time increases as the increment size decreases. However, our experiments found that increasing the increment size to 25 or even 50 feature requests has negligible effect on the quality and stability of the clusters.

Conclusion

We have identified some of the problems experienced in organizing discussion threads in open forums. The survey we conducted in summer 2008 of several open source forums suggests that expecting users to manually create and manage threads may not be the most effective approach. In contrast, we described an automated technique involving our own AFM for creating stable, high-quality clusters to anchor related discussion groups. Though no automated technique always delivers clusters that are cohesive and distinct from other clusters, our reported experiments and case studies demonstrate the advantages of using

data-mining techniques to help manage discussion threads in open discussion forums. Our ongoing work aims to improve techniques for incorporating user feedback into the clustering process so clusters that appear ad hoc to users or contain multiple themes can be improved.

These findings are applicable across a range of applications, including those designed to gather comments from a product’s user base, support activities (such as event planning), and capture requirements in large projects when stakeholders are dispersed geographically. Our ongoing work focuses on the use of forums to support the gathering and prioritizing of requirements where automated forum managers improve the allocation of feature requests to threads and use recommender systems to help include stakeholders in relevant discussion groups.³ They also improve the precision of forum search and enhance browsing capabilities by predicting and displaying stakeholders’ interest in a given discussion thread.

From the user’s perspective, AFM facilitates the process of entering feature requests. Enhanced search features help users decide where to place new feature requests more accurately. Underlying data-mining functions then test the validity of the choice and (when placement is deemed incorrect) recommend moving the feature request to another existing discussion group or sometimes to an entirely new thread.

All techniques described here are being implemented in the prototype AFM tool we are developing to test and evaluate the AFM as an integral component of large-scale, distributed-requirements processes.

Acknowledgments

This work was partially funded by National Science Foundation grant CCR-0447594, including a Research Experiences for Undergraduates summer supplement to support the work of Horatiu Dumitru. We would also like to acknowledge Brenton Bade, Phik Shan Foo, and Adam Czauderna for their work developing the prototype. □

References

1. Basu, C., Hirsh, H., and Cohen, W. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence* (Madison, WI, July 26–30). MIT Press, Cambridge, MA, 1998, 714–720.
2. Can, F. and Ozkarahan, E.A. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems* 15, 4 (Dec. 1990), 483–517.
3. Castro-Herrera, C., Duan, C., Cleland-Huang, J., and Mobasher, B. A recommender system for requirements elicitation in large-scale software projects. In *Proceedings of the 2009 ACM Symposium on Applied Computing* (Honolulu, HI, Mar. 9–12). ACM Press, New York, 2008, 1419–1426.
4. Davis, A., Dieste, O., Hickey, A., Juristo, N., and Moreno, A. Effectiveness of requirements elicitation techniques. In *Proceedings of the 14th IEEE International Requirements Engineering Conference* (Minneapolis, MN, Sept.). IEEE Computer Society, 2006, 179–188.
5. Dhillon, I.S. and Modha, D.S. Concept decompositions for large sparse text data using clustering. *Machine Learning* 42, 1–2 (Jan. 2001), 143–175.
6. Duan, C., Cleland-Huang, J., and Mobasher, B. A consensus-based approach to constrained clustering of software requirements. In *Proceedings of the 17th ACM International Conference on Information and Knowledge Management* (Napa, CA, Oct. 26–30). ACM Press, New York, 2008, 1073–1082.
7. Frakes, W.B. and Baeza-Yates, R. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1992.
8. Fred, A.L. and Jain, A.K. Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 6 (June 2005), 835–850.
9. *Second Life* virtual 3D world; <http://secondlife.com>, feature requests downloaded from the *Second Life* issue tracker <https://jira.secondlife.com/secure/Dashboard.jspa>.
10. SourceForge. Repository of Open Source Code and Applications; feature requests for Open Bravo, ZIMBRA, PHPMyAdmin, and Mono downloaded from SourceForge forums <http://sourceforge.net/>.
11. Sugar CRM. Commercial open source customer relationship management software; <http://www.sugarcrm.com/crm/>; feature requests mined from feature requests at <http://www.sugarcrm.com/forums/>.
12. Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S. Constrained K-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning* (June 28–July 1). Morgan Kaufman Publishers, Inc., San Francisco, 2001, 577–584.

Jane Cleland-Huang (jhuang@cs.depaul.edu) is an associate professor in the School of Computing at DePaul University, Chicago, IL.

Horatiu Dumitru (dumitru.horatiu@gmail.com) is an undergraduate student studying computer science at the University of Chicago, Chicago, IL.

Chuan Duan (macduan@gmail.com) is a post-doctoral researcher in the School of Computing at DePaul University, Chicago, IL.

Carlos Castro-Herrera (ccaastroh@cdm.depaul.edu) is a Ph.D. student in the School of Computing at DePaul University, Chicago, IL.

Table 2. Performance measured by total time spent clustering (in seconds).

Data set	Increment Size				Time to cluster entire set of feature requests one time
	1	10	25	50	
Student (366 feature requests)					
Stable SPK Means	7.49	0.98	0.62	0.41	0.04
Standard SPK Means	101.82	10.78	4.74	2.92	0.73
Sugar (1,000 feature requests)					
Stable SPK Means	84.54	13.24	6.66	4.27	0.22
Standard SPK Means	2,374.31	249.92	108.62	57.83	6.69
Second Life (4,205 feature requests)					
Stable SPK Means	1,880.69	268.15	146.75	96.11	1.02
Standard SPK Means	11,409.57	12,748.63	5,917.94	2,619.90	75.18



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.

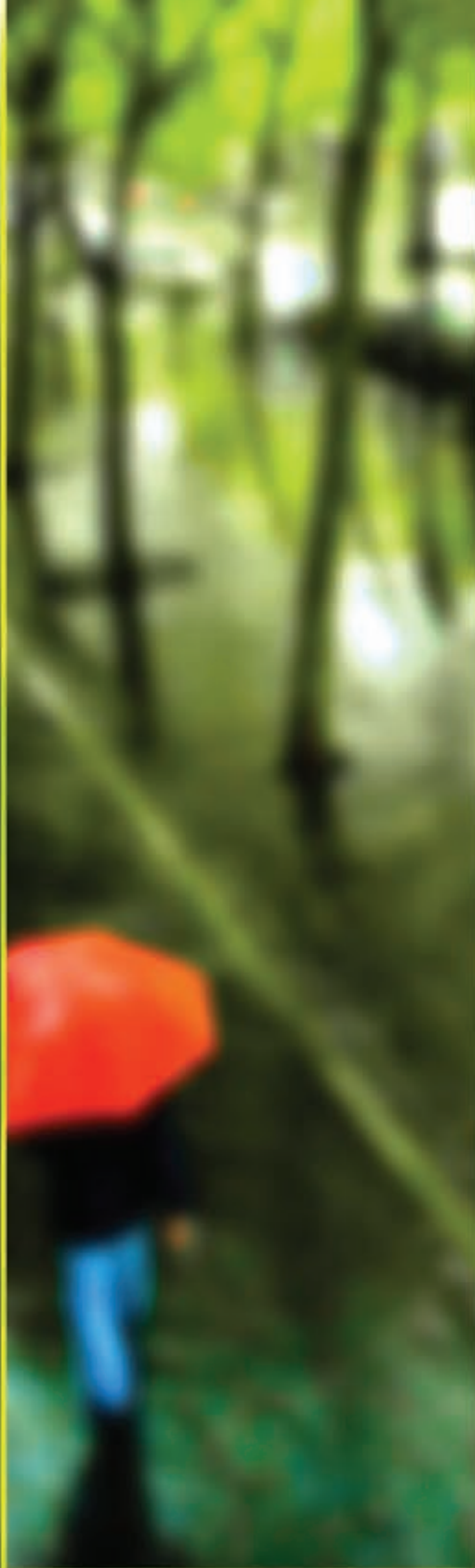


Make a difference to a student in your field.

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.



DOI:10.1145/1562764.1562785

This Gödel Prize-winning work traces the steps toward modeling real data.

BY DANIEL A. SPIELMAN AND SHANG-HUA TENG

Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice

“My experiences also strongly confirmed my previous opinion that the best theory is inspired by practice and the best practice is inspired by theory.”

Donald E. Knuth, “Theory and Practice,”
Theoretical Computer Science, 1991.

ALGORITHMS ARE HIGH-LEVEL descriptions of how computational tasks are performed. Engineers and experimentalists design and implement algorithms, and generally consider them a success if they work in practice. However, an algorithm that works well in one practical domain might perform poorly in another. Theorists also design and analyze algorithms, with the goal of providing provable guarantees about their performance. The traditional goal of theoretical computer science is to prove that an algorithm

performs well in the worst case: if one can prove that an algorithm performs well in the worst case, then one can be confident that it will work well in every domain. However, there are many algorithms that work well in practice that do not work well in the worst case. Smoothed analysis provides a theoretical framework for explaining why some of these algorithms do work well in practice.

The performance of an algorithm is usually measured by its running time, expressed as a function of the input size of the problem it solves. The performance profiles of algorithms across the landscape of input instances can differ greatly and can be quite irregular. Some algorithms run in time linear in the input size on all instances, some take quadratic or higher order polynomial time, while some may take an exponential amount of time on some instances.

Traditionally, the complexity of an algorithm is measured by its *worst-case* performance. If a single input instance triggers an exponential runtime, the algorithm is called an exponential-time algorithm. A polynomial-time algorithm is one that takes polynomial time on all instances. While polynomial-time algorithms are usually viewed as being efficient, we clearly prefer those whose runtime is a polynomial of low degree, especially those that run in nearly linear time.

It would be wonderful if every algorithm that ran quickly in practice was a polynomial-time algorithm. As this is not always the case, the worst-case framework is often the source of discrepancy between the theoretical evaluation of an algorithm and its practical performance.

It is commonly believed that practical inputs are usually more favorable than worst-case instances. For example, it is known that the special case of the Knapsack problem in which one must determine whether a set of n numbers can be divided into two groups of equal sum does not have a polynomial-time algorithm, unless NP

is equal to \mathbf{P} . Shortly before he passed away, Tim Russert of NBC's "Meet the Press," commented that the 2008 election could end in a tie between the Democratic and the Republican candidates. In other words, he solved a 51-item Knapsack problem^a by hand within a reasonable amount of time, and most likely without using the pseudo-polynomial-time dynamic-programming algorithm for Knapsack!

In our field, the simplex algorithm is the classic example of an algorithm that is known to perform well in practice but has poor worst-case complexity. The simplex algorithm solves a linear program, for example, of the form,

$$\max \mathbf{c}^T \mathbf{x} \quad \text{subject to} \quad \mathbf{A} \mathbf{x} \leq \mathbf{b}, \quad (1)$$

where \mathbf{A} is an $m \times n$ matrix, \mathbf{b} is an m -place vector, and \mathbf{c} is an n -place vector. In the worst case, the simplex algorithm takes exponential time.²⁵ Developing rigorous mathematical theories that explain the observed performance of practical algorithms and heuristics has become an increasingly important task in *Theoretical Computer Science*. However, modeling observed data and practical problem instances is a challenging task as insightfully pointed out in the 1999 "Challenges for Theory of Computing" Report for an NSF-Sponsored Workshop on Research in Theoretical Computer Science.^b

While theoretical work on models of computation and methods for analyzing algorithms has had enormous payoff, we are not done. In many situations, simple algorithms do well. Take for example the Simplex algorithm for linear programming, or the success of simulated annealing of certain supposedly intractable

problems. We don't understand why! It is apparent that worst-case analysis does not provide useful insights on the performance of algorithms and heuristics and our models of computation need to be further developed and refined. Theoreticians are investing increasingly in careful experimental work leading to identification of important new questions in algorithms area. Developing means for predicting the performance of algorithms and heuristics on real data and on real computers is a grand challenge in algorithms.

Needless to say, there are a multitude of algorithms beyond simplex and simulated annealing whose performance in practice is not well explained by worst-case analysis. We hope that theoretical explanations will be found for the success in practice of many of these algorithms, and that these theories will catalyze better algorithm design.

The Behavior of Algorithms

When A is an algorithm for solving problem P we let $T_A[\mathbf{x}]$ denote the running time of algorithm A on an input instance \mathbf{x} . If the input domain Ω has only one input instance \mathbf{x} , then we can use the instance-based measure $T_{A_1}[\mathbf{x}]$ and $T_{A_2}[\mathbf{x}]$ to decide which of the two algorithms A_1 and A_2 more efficiently solves P . If Ω has two instances \mathbf{x} and \mathbf{y} , then the instance-based measure of an algorithm A defines a two-dimensional vector $(T_A[\mathbf{x}], T_A[\mathbf{y}])$. It could be the case that $T_{A_1}[\mathbf{x}] < T_{A_2}[\mathbf{x}]$ but $T_{A_1}[\mathbf{y}] > T_{A_2}[\mathbf{y}]$. Then, strictly speaking, these two algorithms are not comparable. Usually, the input domain is much more complex, both in theory and in practice. The *instance-based complexity measure* $T_A[\cdot]$ defines an $|\Omega|$ dimensional vector when Ω is finite. In general, it can be viewed as a function from Ω to \mathbb{R}_+^1 but it is unwieldy. To compare two algorithms, we require a more concise complexity measure.

An input domain Ω is usually viewed as the union of a family of subdomains $\{\Omega_1, \dots, \Omega_n, \dots\}$, where Ω_n represents all instances in Ω of size n . For example, in sorting, Ω_n is the set of all tuples of n elements; in graph algorithms, Ω_n

is the set of all graphs with n vertices; and in computational geometry, we often have $\Omega_n \in \mathbb{R}^n$. In order to succinctly express the performance of an algorithm A , for each Ω_n one defines a scalar $T_A(n)$ that summarizes the instance-based complexity measure, $T_A[\cdot]$, of A over Ω_n . One often further simplifies this expression by using big-O or big- Θ notation to express $T_A(n)$ asymptotically.

Traditional Analyses. It is understandable that different approaches to summarizing the performance of an algorithm over Ω_n can lead to very different evaluations of that algorithm. In *Theoretical Computer Science*, the most commonly used measures are the worst-case measure and the average-case measures.

The *worst-case measure* is defined as

$$\text{WC}_A(n) = \max_{\mathbf{x} \in \Omega_n} T_A[\mathbf{x}]$$

The *average-case measures* have more parameters. In each average-case measure, one first determines a distribution of inputs and then measures the expected performance of an algorithm assuming inputs are drawn from this distribution. Supposing \mathcal{S} provides a distribution over each Ω_n , the average-case measure according to \mathcal{S} is

$$\text{Ave}_A^{\mathcal{S}}(n) = \mathbb{E}_{\mathbf{x} \in_{\mathcal{S}} \Omega_n} [T_A[\mathbf{x}]],$$

where we use $\mathbf{x} \in_{\mathcal{S}} \Omega_n$ to indicate that \mathbf{x} is randomly chosen from Ω_n according to distribution \mathcal{S} .

Critique of Traditional Analyses. Low worst-case complexity is the gold standard for an algorithm. When low, the worst-case complexity provides an absolute guarantee on the performance of an algorithm no matter which input it is given. Algorithms with good worst-case performance have been developed for a great number of problems.

However, there are many problems that need to be solved in practice for which we do not know algorithms with good worst-case performance. Instead, scientists and engineers typically use heuristic algorithms to solve these problems. Many of these algorithms work well in practice, in spite of having a poor, sometimes exponential, worst-

^a In presidential elections in the United States, each of the 50 states and the District of Columbia is allocated a number of electors. All but the states of Maine and Nebraska use a winner-take-all system, with the candidate winning the majority votes in each state being awarded all of that states electors. The winner of the election is the candidate who is awarded the most electors.

^b Available at <http://sigact.acm.org/>

case running time. Practitioners justify the use of these heuristics by observing that worst-case instances are usually not “typical” and rarely occur in practice. The worst-case analysis can be too pessimistic. This theory-practice gap is not limited to heuristics with exponential complexity. Many polynomial-time algorithms, such as interior-point methods for linear programming and the conjugate gradient algorithm for solving linear equations, are often much faster than their worst-case bounds would suggest. In addition, heuristics are often used to speed up the practical performance of implementations that are based on algorithms with polynomial worst-case complexity. These heuristics might in fact worsen the worst-case performance, or make the worst-case complexity difficult to analyze.

Average-case analysis was introduced to overcome this difficulty. In average-case analysis, one measures the expected running time of an algorithm on some distribution of inputs. While one would ideally choose the distribution of inputs that occurs in practice, this is difficult as it is rare that one can determine or cleanly express these distributions, and the distributions can vary greatly between one application and another. Instead, average-case analyses have employed distributions with concise mathematical descriptions, such as Gaussian random vectors, uniform $\{0, 1\}$ vectors, and Erdős–Rényi random graphs.

The drawback of using such distributions is that the inputs actually encountered in practice may bear very little resemblance to the inputs that are likely to be generated by such distributions. For example, one can see what a random image looks like by disconnecting most TV sets from their antennas, at which point they display “static.” These random images do not resemble actual television shows. More abstractly, Erdős–Rényi random graph models are often used in average-case analyses of graph algorithms. The Erdős–Rényi distribution $G(n, p)$, produces a random graph by including every possible edge in the graph independently with probability p . While the average degree of a graph chosen from $G(n, 6/(n - 1))$ is approximately six such a graph will be very different

from the graph of a triangulation of a point set in two dimensions, which will also have average degree approximately six.

In fact, random objects such as random graphs and random matrices have special properties with exponentially high probability, and these special properties might dominate the average-case analysis. Edelman¹⁴ writes of random matrices:

What is a mistake is to psychologically link a random matrix with the intuitive notion of a “typical” matrix or the vague concept of “any old matrix.”

In contrast, we argue that “random matrices” are *very* special matrices.

Smoothed Analysis: A Step Toward Modeling Real Data. Because of the intrinsic difficulty in defining practical distributions, we consider an alternative approach to modeling real data. The basic idea is to identify typical properties of practical data, define an input model that captures these properties, and then rigorously analyze the performance of algorithms assuming their inputs have these properties.

Smoothed analysis is a step in this direction. It is motivated by the observation that practical data is often subject to some small degree of random noise. For example,

- In industrial optimization and economic prediction, the input parameters could be obtained by physical measurements, and measurements usually have some of low magnitude uncertainty.

- In the social sciences, data often comes from surveys in which subjects provide integer scores in a small range (say between 1 and 5) and select their scores with some arbitrariness.

- Even in applications where inputs are discrete, there might be randomness in the formation of inputs. For instance, the network structure of the Internet may very well be governed by some “blueprints” of the government and industrial giants, but it is still “perturbed” by the involvements of smaller Internet service providers.

In these examples, the inputs usually are neither completely random

nor completely arbitrary. At a high level, each input is generated from a two-stage model: In the first stage, an instance is generated and in the second stage, the instance from the first stage is slightly perturbed. The perturbed instance is the input to the algorithm.

In smoothed analysis, we assume that an input to an algorithm is subject to a slight random perturbation. The *smoothed measure* of an algorithm on an input instance is its expected performance over the perturbations of that instance. We define the *smoothed complexity* of an algorithm to be the maximum smoothed measure over input instances.

For concreteness, consider the case $\Omega_n = \mathbb{R}^n$, which is a common input domain in computational geometry, scientific computing, and optimization. For these continuous inputs and applications, the family of Gaussian distributions provides a natural model of noise or perturbation.

Recall that a univariate Gaussian distribution with mean 0 and standard deviation σ has density

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}.$$

The standard deviation measures the magnitude of the perturbation. A *Gaussian random vector* of variance σ^2 centered at the origin in $\Omega_n = \mathbb{R}^n$ is a vector in which each entry is an independent Gaussian random variable of standard deviation σ and mean 0. For a vector $\bar{x} \in \mathbb{R}^n$, a σ -Gaussian perturbation of \bar{x} is a random vector $x = \bar{x} + g$, where g is a Gaussian random vector of variance σ^2 . The standard deviation of the perturbation we apply should be related to the norm of the vector it perturbs. For the purposes of this article, we relate the two by restricting the unperturbed inputs to lie in $[-1, 1]^n$. Other reasonable approaches are taken elsewhere.

DEFINITION 1. (SMOOTHED COMPLEXITY). Suppose A is an algorithm with $\Omega_n = \mathbb{R}^n$. Then, the smoothed complexity of A with σ -Gaussian perturbations is given by

$$\text{Smoothed}_A^\sigma(n) = \max_{\bar{x} \in [-1, 1]^n} \mathbb{E}_g [T_A(\bar{x} + g)],$$

where \mathbf{g} is a Gaussian random vector of variance σ^2 .

In this definition, the “original” input $\bar{\mathbf{x}}$ is perturbed to obtain the input $\bar{\mathbf{x}} + \mathbf{g}$, which is then fed to the algorithm. For each original input, this measures the expected running time of algorithm A on random perturbations of that input. The maximum out front tells us to measure the smoothed analysis by the expectation under the worst possible original input.

The smoothed complexity of an algorithm measures the performance of the algorithm both in terms of the input size n and in terms of the magnitude σ of the perturbation. By varying σ between zero and infinity, one can use smoothed analysis to interpolate between worst-case and average-case analysis. When $\sigma = 0$, one recovers the ordinary worst-case analysis. As σ grows large, the random perturbation \mathbf{g} dominates the original $\bar{\mathbf{x}}$, and one obtains an average-case analysis. We are most interested in the situation in which σ is small relative to $\|\bar{\mathbf{x}}\|$, in which case $\bar{\mathbf{x}} + \mathbf{g}$ may be interpreted as a slight perturbation of $\bar{\mathbf{x}}$. The dependence on the magnitude σ is essential and much of the work in smoothed analysis demonstrates that noise often makes a problem easier to solve.

DEFINITION 2. *A has **polynomial smoothed complexity** if there exist positive constants $n_0, \sigma_0, c, k_1,$ and k_2 such that for all $n \geq n_0$ and $0 \leq \sigma \leq \sigma_0$,*

$$\text{Smoothed}_A^\sigma(n) \leq c \cdot \sigma^{-k_2} \cdot n^{k_1}, \quad (2)$$

From Markov’s inequality, we know that if an algorithm A has smoothed complexity $T(n, \sigma)$, then

$$\max_{\bar{\mathbf{x}} \in [-1, 1]^n} \Pr[T_A(\bar{\mathbf{x}} + \mathbf{g}) \leq \delta^{-1}T(n, \sigma)] \geq 1 - \delta. \quad (3)$$

Thus, if A has polynomial smoothed complexity, then for any $\bar{\mathbf{x}}$, with probability at least $(1 - \delta)$, A can solve a random perturbation of $\bar{\mathbf{x}}$ in time polynomial in $n, 1/\sigma$, and $1/\delta$. However, the probabilistic upper bound given in (3) does not necessarily imply that the smoothed complexity of A is $O(T(n, \sigma))$. Blum and Dunagan⁷ and subsequently Beier and Vöcking⁶ introduced a

relaxation of polynomial smoothed complexity.

DEFINITION 3. *A has **probably polynomial smoothed complexity** if there exist constants $n_0, \sigma_0, c,$ and $\alpha,$ such that for all $n \geq n_0$ and $0 \leq \sigma \leq \sigma_0$,*

$$\max_{\bar{\mathbf{x}} \in [-1, 1]^n} \mathbb{E}[T_A(\bar{\mathbf{x}} + \mathbf{g})^\alpha] \leq c \cdot \sigma^{-1} \cdot n. \quad (4)$$

They show that some algorithms have probably polynomial smoothed complexity, in spite of the fact that their smoothed complexity according to Definition 1 is unbounded.

Examples of Smoothed Analysis

In this section, we give a few examples of smoothed analysis. We organize them in five categories: mathematical programming, machine learning, numerical analysis, discrete mathematics, and combinatorial optimization. For each example, we will give the definition of the problem, state the worst-case complexity, explain the perturbation model, and state the smoothed complexity under the perturbation model.

Mathematical Programming. The typical problem in mathematical programming is the optimization of an objective function subject to a set of constraints. Because of its importance to economics, management science, industry, and military planning, many optimization algorithms and heuristics have been developed, implemented and applied to practical problems. Thus, this field provides a great collection of algorithms for smoothed analysis.

Linear Programming. Linear programming is the most fundamental optimization problem. A typical linear program is given in Equation 1. The most commonly used linear programming algorithms are the simplex algorithm¹² and the interior-point algorithms.

The simplex algorithm, first developed by Dantzig in 1951,¹² is a family of iterative algorithms. Most of them are two-phase algorithms: Phase I determines whether a given linear program is infeasible, unbounded in the objective direction, or feasible with a bounded solution, in which case, a vertex \mathbf{v}_0 of the feasible region is also computed.

Phase II is iterative: in the i th iteration, the algorithm finds a neighboring vertex \mathbf{v}_i of \mathbf{v}_{i-1} with better objective value or terminates by returning \mathbf{v}_{i-1} when no such neighboring vertex exists. The simplex algorithms differ in their pivot rules, which determine which vertex \mathbf{v}_i to choose when there are multiple choices. Several pivot rules have been proposed; however, almost all existing pivot rules are known to have exponential worst-case complexity.²⁵

Spielman and Teng³⁶ considered the smoothed complexity of the simplex algorithm with the shadow-vertex pivot rule, developed by Gass and Saaty.¹⁸ They used Gaussian perturbations to model noise in the input data and proved that the smoothed complexity of this algorithm is polynomial. Vershynin³⁸ improved their result to obtain a smoothed complexity of

$$O(\max(n^5 \log^2 m, n^9 \log^4 n, n^3 \sigma^{-4})).$$

See Blum and Dunagan, and Dunagan et al.^{7, 13} for smoothed analyses of other linear programming algorithms such as the interior-point algorithms and the perceptron algorithm.

Quasi-Concave Minimization. Another fundamental optimization problem is quasi-concave minimization. Recall that a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is quasi-concave if all of its upper level sets $L_\gamma = \{x \mid f(x) \geq \gamma\}$ are convex. In quasi-concave minimization, one is asked to find the minimum of a quasi-concave function subject to a set of linear constraints. Even when restricted to concave quadratic functions over the hypercube, concave minimization is NP-hard.

In applications such as stochastic and multiobjective optimization, one often deals with data from low-dimensional subspaces. In other words, one needs to solve a quasi-concave minimization problem with a low-rank quasi-concave function.²³ Recall that a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ has rank k if it can be written in the form

$$f(x) = g(a_1^T x, a_2^T x, \dots, a_k^T x),$$

for a function $g: \mathbb{R}^k \rightarrow \mathbb{R}$ and linearly independent vectors a_1, a_2, \dots, a_k .

Kelner and Nikolova²³ proved that, under some mild assumptions on the feasible convex region, if k is a constant

then the smoothed complexity of quasi-concave minimization is polynomial when f is perturbed by noise. Key to their analysis is a smoothed bound on the size of the k -dimensional shadow of the high-dimensional polytope that defines the feasible convex region. Their result is a nontrivial extension of the analysis of two-dimensional shadows of Kelner and Spielman, and Spielman and Teng.^{24,36}

Machine Learning. Machine learning provides many natural problems for smoothed analysis. The field has many heuristics that work in practice, but not in the worst case, and the data defining most machine learning problems is inherently noisy.

k -means. One of the fundamental problems in machine learning is that of k -means clustering: the partitioning of a set of d -dimensional vectors $Q = \{q_1, \dots, q_n\}$ into k clusters $\{Q_1, \dots, Q_k\}$ so that the intracluster variance

$$V = \sum_{i=1}^k \sum_{q_j \in Q_i} \|q_j - \mu(Q_i)\|^2,$$

is minimized, where $\mu(Q_i) = (\sum_{q_j \in Q_i} q_j) / |Q_i|$ is the centroid of Q_i .

One of the most widely used clustering algorithms is Lloyd's algorithm (*IEEE Transaction on Information Theory*, 1982). It first chooses an arbitrary set of k centers and then uses the Voronoi diagram of these centers to partition Q into k clusters. It then repeats the following process until it stabilizes: use the centroids of the current clusters as the new centers, and then repartition Q accordingly.

Two important questions about Lloyd's algorithm are how many iterations it takes to converge, and how close to optimal is the solution it finds? Smoothed analysis has been used to address the first question. Arthur and Vassilvitskii proved that in the worst case, Lloyd's algorithm requires $2^{\Omega(\sqrt{n})}$ iterations to converge,³ but that it has smoothed complexity polynomial in n^k and σ^{-1} .⁴ Manthey and Röglin²⁸ recently reduced this bound to polynomial in $n^{\sqrt{k}}$ and σ^{-1} .

Perceptrons, Margins, and Support Vector Machines. Blum and Dunagan's analysis of the perceptron algorithm⁷ for linear programming implicitly contains results of interest in machine

learning. The ordinary perceptron algorithm solves a fundamental problem: given a collection of points $x_1, \dots, x_n \in \mathbb{R}^d$ and labels $b_1, \dots, b_n \in \{\pm 1\}^n$, find a hyperplane separating the positively labeled examples from the negatively labeled ones, or determine that no such plane exists. Under a smoothed model in which the points x_1, \dots, x_n are subject to a σ -Gaussian perturbation, Blum and Dunagan show that the perceptron algorithm has probably polynomial smoothed complexity, with exponent $\alpha = 1$. Their proof follows from a demonstration that if the positive points can be separated from the negative points, then they can probably be separated by a large margin. That is, there probably exists a plane separating the points for which no point is too close to that separating plane.

It is known that the perceptron algorithm converges quickly in this case. Moreover, this margin is exactly what is maximized by support vector machines.

PAC Learning. Probably approximately correct learning (PAC learning) is a framework in machine learning introduced by Valiant in which a learner is provided with a polynomial number of labeled examples from a given distribution, and must produce a classifier that is usually correct with reasonably high probability. In standard PAC learning, the distribution from which the examples are drawn is fixed. Recently, Kalai and Teng²¹ applied smoothed analysis to this problem by perturbing the input distribution. They prove that polynomial-sized decision trees are PAC-learnable in polynomial time under perturbed product distributions. In contrast, under the uniform distribution even super-constant size decision trees are not known to be PAC-learnable in polynomial time.

Numerical Analysis. One of the foci of numerical analysis is the determination of how much precision is required by numerical methods. For example, consider the most fundamental problem in computational science—that of solving systems of linear equations. Because of the round-off errors in computation, it is crucial to know how many bits of precision a linear solver should maintain so that its solution is meaningful.

For example, Wilkinson (*JACM* 1961) demonstrated a family of linear systems^c of n variables and $\{0, -1, 1\}$ coefficients for which Gaussian elimination with partial pivoting—the most popular variant in practice—requires n -bits of precision.

Precision Requirements of Gaussian Elimination. In practice one almost always obtains accurate answers using much less precision. High-precision solvers are rarely used or needed. For example, Matlab uses 64 bits.

Building on the smoothed analysis of condition numbers (discussed below), Sankar et al.^{33,34} proved that it is sufficient to use $O(\log^2(n/\sigma))$ bits of precision to run Gaussian elimination with partial pivoting when the matrices of the linear systems are subject to σ -Gaussian perturbations.

The Condition Number. The smoothed analysis of the condition number of a matrix is a key step toward understanding numerical precision required in practice. For a square matrix \mathbf{A} , its condition number $\kappa(\mathbf{A})$ is given by $\kappa(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2$ where $\|\mathbf{A}\|_2 = \max_x \|\mathbf{A}\mathbf{x}\|_2 / \|\mathbf{x}\|_2$. The condition number of \mathbf{A} measures how much the solution to a system $\mathbf{A}\mathbf{x} = \mathbf{b}$ changes as one makes slight changes to \mathbf{A} and \mathbf{b} : If one solves the linear system using fewer than $\log(\kappa(\mathbf{A}))$ bits of precision, then one is likely to obtain a result far from a solution.

The quantity $1/\|\mathbf{A}^{-1}\|_2 = \min_x \|\mathbf{A}\mathbf{x}\|_2 / \|\mathbf{x}\|_2$ is known as the smallest singular value of \mathbf{A} . Sankar et al.³⁴ proved the following statement: For any square matrix $\bar{\mathbf{A}}$ in $\mathbb{R}^{n \times n}$ satisfying $\|\bar{\mathbf{A}}\|_2 \leq \sqrt{n}$, and for any $x > 1$,

$$\Pr_{\mathbf{A}}[\|\mathbf{A}^{-1}\|_2 \geq x] \leq 2.35 \frac{\sqrt{n}}{x\sigma},$$

where \mathbf{A} is a σ -Gaussian perturbation of $\bar{\mathbf{A}}$. Wschebor⁴⁰ improved this bound to show that for $\sigma \leq 1$ and $\|\bar{\mathbf{A}}\|_2 \leq 1$,

$$\Pr[\kappa(\mathbf{A}) \geq x] \leq O\left(\frac{\sqrt{n}}{x\sigma}\right).$$

See Bürgisser et al. and Dunagan et al.^{10,13} for smoothed analysis of the condition numbers of other problems.

Discrete Mathematics. For problems in discrete mathematics, it is

^c See the second line of the Matlab code at the end of the Discussion section for an example.

more natural to use Boolean perturbations: Let $\bar{x} = (\bar{x}_1, \dots, \bar{x}_n) \in \{0, 1\}^n$ or $\{-1, 1\}^n$. The σ -Boolean perturbation of \bar{x} is a random string $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ or $\{-1, 1\}^n$, where $x_i = \bar{x}_i$ with probability $1 - \sigma$ and $x_i \neq \bar{x}_i$ with probability σ . That is, each bit is flipped independently with probability σ .

Believing that σ -perturbations of Boolean matrices should behave like Gaussian perturbations of real matrices, Spielman and Teng³⁵ made the following conjecture:

Let \mathbf{A} be an n by n matrix of independently and uniformly chosen ± 1 entries. Then

$$\Pr_{\mathbf{A}}[\|\mathbf{A}^{-1}\|_2 \geq x] \leq \frac{\sqrt{n}}{x} + \alpha^n,$$

for some constant α .

Statements close to this conjecture and its generalizations were recently proved by Vu and Tao³⁹ and Rudelson and Vershynin.³²

The σ -Boolean perturbation of a graph can be viewed as a smoothed extension of the classic Erdős-Rényi random graph model. The σ -perturbation of a graph G , which we denote by $G_{\bar{G}}(n, \sigma)$, is a distribution of random graphs in which every edge is removed with probability σ and every nonedge is included with probability σ . Clearly for $p \in [0, 1]$, $G(n, p) = G_{\emptyset}(n, p)$, i.e., the p -Boolean perturbation of the empty graph. One can define a smoothed extension of other random graph models. For example, for any m and $G = (V, E)$, Bohman et al.⁹ define $G(\bar{G}, m)$ to be the distribution of the random graphs $(V, E \cup T)$ where T is a set of m edges chosen uniformly at random from the complement of E , i.e., chosen from $\bar{E} = \{(i, j) \notin E\}$.

A popular subject of study in the traditional Erdős-Rényi model is the phenomenon of phase transition: for many properties such as being connected or being Hamiltonian, there is a critical p below which a graph is unlikely to have each property and above which it probably does have the property. Related phase transitions have also been found in the smoothed Erdős-Rényi models $G_{\bar{G}}(n, \sigma)$.^{17,26}

Smoothed analysis based on Boolean perturbations can be applied

to other discrete problems. For example, Feige¹⁶ used the following smoothed model for 3CNF formulas. First, an adversary picks an arbitrary formula with n variables and m clauses. Then, the formula is perturbed at random by flipping the polarity of each occurrence of each variable independently with probability σ . Feige gave a randomized polynomial-time refutation algorithm for this problem.

Combinatorial Optimization. Beier and Vöcking⁶ and Röglin and Vöcking³¹ considered the smoothed complexity of integer linear programming. They studied programs of the form

$$\max \mathbf{c}^T \mathbf{x} \quad \text{subject to} \quad \mathbf{A} \mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \in \mathcal{D}^n, \tag{5}$$

where \mathbf{A} is an $m \times n$ real matrix, $\mathbf{b} \in \mathbb{R}^m$, and $\mathcal{D} \subset \mathbb{Z}$.

Recall that ZPP denotes the class of decision problems solvable by a randomized algorithm that always returns the correct answer, and whose expected running time (on every input) is polynomial. Beier, Röglin, and Vöcking^{6,31} proved the following statement: For any constant c , let Π be a class of integer linear programs of form 5 with $|D| = O(n^c)$. Then, Π has an algorithm of probably polynomial smoothed complexity if and only if $\Pi_u \in \text{ZPP}$, where Π_u is the “unary” representation of Π . Consequently, the 0/1-knapsack problem, the constrained shortest path problem, the constrained minimum spanning tree problem, and the constrained minimum weighted matching problem have probably polynomial smoothed complexity.

Smoothed analysis has been applied to several other optimization problems such as local search and TSP,¹⁵ scheduling,⁵ motion planning,¹¹ superstring approximation,²⁷ multiobjective optimization,³¹ string alignment,² and multidimensional packing.²²

Discussion

A Theme in Smoothed Analyses. One idea is present in many of these smoothed analyses: perturbed inputs are rarely ill-conditioned. Informally speaking, the condition number of a problem measures how much its answer can be made to change by slight modifications of its input. Many

numerical algorithms are known to run faster when their inputs are well conditioned. Some smoothed analyses, such as that of Blum and Dunagan,⁷ exploit this connection explicitly. Others rely on similar ideas.

For many problems, the condition number of an input is approximately the inverse of the distance of that input to the set of degenerate or ill-posed problems. As these degenerate sets typically have measure zero, it is intuitively reasonable that a perturbed instance should be unlikely to be too close to a degenerate one.

Other Performance Measures.

Although we normally evaluate the performance of an algorithm by its running time, other performance parameters are often important. These performance parameters include the amount of space required, the number of bits of precision required to achieve a given output accuracy, the number of cache misses, the error probability of a decision algorithm, the number of random bits needed in a randomized algorithm, the number of calls to a particular subroutine, and the number of examples needed in a learning algorithm. The quality of an approximation algorithm could be its approximation ratio; the quality of an online algorithm could be its competitive ratio; and the parameter of a game could be its price of anarchy or the rate of convergence of its best-response dynamics. We anticipate future results on the smoothed analysis of these performance measures.

Precursors to Smoothed Complexity.

Several previous probabilistic models have also combined features of worst-case and average-case analyses.

Haimovich¹⁹ and Adler¹ considered the following probabilistic analysis: Given a linear program $\mathcal{L} = (\mathbf{A}, \mathbf{b}, \mathbf{c})$ of form (1), they defined the *expected complexity* of \mathcal{L} to be the expected complexity of the simple algorithm when the inequality sign of each constraint is uniformly flipped. They proved that the expected complexity of the worst possible \mathcal{L} is polynomial.

Blum and Spencer⁸ studied the design of polynomial-time algorithms for the semi-random model, which combines the features of the semi-random source with the random graph

model that has a “planted solution.” This model can be illustrated with the *k*-Coloring Problem: An adversary plants a solution by partitioning the set V of n vertices into k subsets V_1, \dots, V_k . Let

$$F = \{(u, v) \mid u \text{ and } v \text{ are in different subsets}\}$$

be the set of potential inter-subset edges. A graph is then constructed by the following semi-random process that perturbs the decisions of the adversary: In a sequential order, the adversary decides whether to include each edge of F in the graph, and then a random process reverses the decision with probability σ . Note that every graph generated by this semi-random process has the planted coloring: $c(v) = i$ for all $v \in V_i$, as both the adversary and the random process preserve this solution by only considering edges from F .

As with the smoothed model, one can work with the semi-random model by varying σ between 0 and 1 to interpolate between worst-case and average-case complexity for *k*-coloring.

Algorithm Design and Analysis for Special Families of Inputs. Probabilistic approaches are not the only means of characterizing practical inputs. Much work has been spent on designing and analyzing inputs that satisfy certain deterministic but practical input conditions. We mention a few examples that excite us.

In parallel scientific computing, one may often assume that the input graph is a well-shaped finite element mesh. In VLSI layout, one often only considers graphs that are planar or nearly planar. In geometric modeling, one may assume that there is an upper bound on the ratio among the distances between points. In Web analysis, one may assume that the input graph satisfies some powerlaw degree distribution or some small-world properties. When analyzing hash functions, one may assume that the data being hashed has some non-negligible entropy.³⁰

Limits of Smoothed Analysis. The goal of smoothed analysis is to explain why some algorithms have much better performance in practice than predicted by the traditional worst-case analysis. However, for many problems, there may be better explanations.

For example, the worst-case complexity and the smoothed complexity of the problem of computing a market equilibrium are essentially the same.²⁰ So far, no polynomial-time pricing algorithm is known for general markets. On the other hand, pricing seems to be a practically solvable problem, as Kamal Jain put it “If a Turing machine can’t compute then an economic system can’t compute either.”

A key step to understanding the behaviors of algorithms in practice is the construction of analyzable models that are able to capture some essential aspects of practical input instances. For practical inputs, there may often be multiple parameters that govern the process of their formation.

One way to strengthen the smoothed analysis framework is to improve the model of the formation of input instances. For example, if the input instances to an algorithm A come from the output of another algorithm B , then algorithm B , together with a model of B ’s input instances, provide a description of A ’s inputs. For example, in finite-element calculations, the inputs to the linear solver A are stiffness matrices that are produced by a meshing algorithm B . The meshing algorithm B , which could be a randomized algorithm, generates a stiffness matrix from a geometric domain Ω and a partial differential equation F . So, the distribution of the stiffness matrices input to algorithm A is determined by the distribution \mathcal{D} of the geometric domains Ω and the set F of partial differential equations, and the randomness in algorithm B . If, for example, $\bar{\Omega}$ is the design of an advanced rocket from a set \mathcal{R} of “blueprints” and F is from a set \mathcal{F} of PDEs describing physical parameters such as pressure, speed, and temperature, and Ω is generated by a perturbation model \mathcal{P} of the blueprints, then one may further measure the performance of A by the smoothed value:

$$\max_{F \in \mathcal{F}, \bar{\Omega} \in \mathcal{R}} \mathbb{E}_{\Omega \leftarrow \mathcal{P}(\bar{\Omega})} \left[\mathbb{E}_{X \leftarrow B(\Omega, F)} [Q(A, X)] \right],$$

where $\Omega \leftarrow \mathcal{P}(\bar{\Omega})$ indicates that Ω is obtained from a perturbation of $\bar{\Omega}$ and $X \leftarrow B(\Omega, F)$ indicates that X is the output of the randomized algorithm B .

A simpler way to strengthen smoothed analysis is to restrict the family of perturbations considered. For example, one could employ *zero-preserving perturbations*, which only apply to nonzero entries. Or, one could use *relative perturbations*, which perturb every real number individually by a small multiplicative factor.

Algorithm Design Based on Perturbations and Smoothed Analysis.

Finally, we hope insights gained from smoothed analysis will lead to new ideas in algorithm design. On a theoretical front, Kelner and Spielman²⁴ exploited ideas from the smoothed analysis of the simplex method to design a (weakly) polynomial-time simplex method that functions by systematically perturbing its input program. On a more practical level, we suggest that it might be possible to solve some problems more efficiently by perturbing their inputs. For example, some algorithms in computational geometry implement variable-precision arithmetic to correctly handle exceptions that arise from geometric degeneracy.²⁹ However, degeneracies and near-degeneracies occur with exceedingly small probability under perturbations of inputs. To prevent perturbations from changing answers, one could employ quad-precision arithmetic, placing the perturbations into the least-significant half of the digits.

Our smoothed analysis of Gaussian elimination suggests a more stable solver for linear systems: When given a linear system $\mathbf{Ax} = \mathbf{b}$, we first use the standard Gaussian elimination with partial pivoting algorithm to solve $\mathbf{Ax} = \mathbf{b}$. Suppose \mathbf{x}^* is the solution computed. If $\|\mathbf{b} - \mathbf{Ax}^*\|$ is small enough, then we simply return \mathbf{x}^* . Otherwise, we can determine a parameter ε and generate a new linear system $(\mathbf{A} + \varepsilon\mathbf{G})\mathbf{y} = \mathbf{b}$, where \mathbf{G} is a Gaussian matrix with mean $\mathbf{0}$ and variance 1. Instead of solving $\mathbf{Ax} = \mathbf{b}$, we solve a perturbed linear system $(\mathbf{A} + \varepsilon\mathbf{G})\mathbf{y} = \mathbf{b}$. It follows from standard analysis that if ε is sufficiently smaller than $\kappa(\mathbf{A})$, then the solution to the perturbed linear system is a good approximation to the original one. One could use practical experience or binary search to set ε .

The new algorithm has the property that its success depends only on the machine precision and the condition

number of A , while the original algorithm may fail due to large growth factors. For example, the following is a fragment of Matlab code that first solves a linear system whose matrix is the 70×70 matrix Wilkinson designed to trip up partial pivoting, using the Matlab linear solver. We then perturb the system, and apply the Matlab solver again.


```
>> % Using the Matlab Solver
>> n = 70; A = 2*eye(n) -
    tril(ones(n)); A(:,n)=1;
>> b = randn(70,1); x = A\b;
>> norm(A*x-b)
    2.762797463910437e+004
>> % FAILED because of large
    growth factor
>> %Using the new solver
>> Ap = A + randn(n)/10^9;
    y = Ap\b;
>> norm(Ap*y-b)
    6.343500222435404e-015
>> norm(A*y-b)
    4.434147778553908e-008
```

Note that while the Matlab linear solver fails to find a good solution to the linear system, our new perturbation-based algorithm finds a good solution. While there are standard algorithms for solving linear equations that do not have the poor worst-case performance of partial pivoting, they are rarely used as they are less efficient.

For more examples of algorithm design inspired by smoothed analysis and perturbation theory, see Teng.³⁷

Acknowledgments

We would like to thank Alan Edelman for suggesting the name “Smoothed Analysis” and thank Heiko Röglin and Don Knuth for helpful comments on this writing.

Due to *Communications* space constraints, we have had to restrict our bibliography to 40 references. We apologize to those whose work we have been forced not to reference. 

References

1. Adler, I. The expected number of pivots needed to solve parametric linear programs and the efficiency of the self-dual simplex method. Technical report, University of California at Berkeley (May 1983).
2. Andoni, A. and Krauthgamer, R. The smoothed complexity of edit distance. In *Proceedings of ICALP*,

volume 5125 of *Lecture Notes in Computer Science* (Springer, 2008) 357–369.

3. Arthur, D. and Vassilvitskii, S. How slow is the k-means method? In *SOCG '06, the 22nd Annual ACM Symposium on Computational Geometry* (2006) 144–153.
4. Arthur, D. and Vassilvitskii, S. Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method. In *FOCS '06, the 47th Annual IEEE Symposium on Foundations of Computer Science* (2006) 153–164.
5. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Schfer, G., and Vredeveld, T. Average-case and smoothed competitive analysis of the multilevel feedback algorithm. *Math. Oper. Res.* 31, 1 (2006), 85–108.
6. Beier, R. and Vöcking, B. Typical properties of winners and losers in discrete optimization. In *STOC '04: the 36th Annual ACM Symposium on Theory of Computing* (2004), 343–352.
7. Blum, A. and Dunagan, J. Smoothed analysis of the perceptron algorithm for linear programming. In *SODA '02* (2002), 905–914.
8. Blum, A. and Spencer, J. Coloring random and semi-random k-colorable graphs. *J. Algorithms* 19, 2 (1995), 204–234.
9. Bohman, T., Frieze, A. and Martin, R. How many random edges make a dense graph hamiltonian? *Random Struct. Algorithms* 22, 1 (2003), 33–42.
10. Bürgisser, P., Cucker, F., and Lotz, M. Smoothed analysis of complex conic condition numbers. *J. de Mathématiques Pures Appliqués* 86 4 (2006), 293–309.
11. Damerow, V., Meyer auf der Heide, F., Räcke, H., Scheideler, C., and Sohler, C. Smoothed motion complexity. In *Proceedings of the 11th Annual European Symposium on Algorithms* (2003), 161–171.
12. Dantzig, G.B. Maximization of linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*. T.C. Koopmans, Ed. 1951, 339–347.
13. Dunagan, J., Spielman, D.A., and Teng, S.-H. Smoothed analysis of condition numbers and complexity implications for linear programming. *Mathematical Programming, Series A*, 2009. To appear. Preliminary version available at <http://arxiv.org/abs/cs/0302011v2>.
14. Edelman, A. Eigenvalue roulette and random test matrices. *Linear Algebra for Large Scale and Real-Time Applications*, Marc S. Moonen, Gene H. Golub, and Bart L. R. De Moor, eds. NATO ASI Series, 1992, 365–368.
15. Englert, M., Röglin, H., and Vöcking, B. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP: extended abstract. In *SODA '07: The 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), 1295–1304.
16. Feige, U. Refuting smoothed 3CNF formulas. In *The 48th Annual IEEE Symposium on Foundations of Computer Science* (2007), 407–417.
17. Flaxman, A. and Frieze, A.M. The diameter of randomly perturbed digraphs and some applications. In *APPROX-RANDOM* (2004), 345–356.
18. Gass, S. and Saaty, T. The computational algorithm for the parametric objective function. *Naval Res. Logist. Quart.* 2, (1955), 39–45.
19. Haimovich, M. The simplex algorithm is very good!: On the expected number of pivot steps and related properties of random linear programs. Technical report, Columbia University (April 1983).
20. Huang, L.-S. and Teng, S.-H. On the approximation and smoothed complexity of Leontief market equilibria. In *Frontiers of Algorithms Workshop* (2007), 96–107.
21. Kalai, A. and Teng, S.-H. Decision trees are PAC-learnable from most product distributions: a smoothed analysis. *ArXiv e-prints* (December 2008).
22. Karger, D. and Onak, K. Polynomial approximation schemes for smoothed and random instances of multidimensional packing problems. In *SODA '07: the 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), 1207–1216.
23. Kelner, J.A. and Nikolova, E. On the hardness and smoothed complexity of quasi-concave minimization. In *The 48th Annual IEEE Symposium on Foundations of Computer Science* (2007), 472–482.
24. Kelner, J.A. and Spielman, D.A. A randomized polynomial-time simplex algorithm for linear programming. In *The 38th Annual ACM Symposium on Theory of Computing* (2006), 51–60.
25. Klee, V. and Minty, G.J. How good is the simplex

algorithm? *Inequalities – III*. O. Shisha, ed. Academic Press, 1972, 159–175.

26. Krivelevich, M., Sudakov, B. and Tetali, P. On smoothed analysis in dense graphs and formulas. *Random Struct. Algorithms* 29 (2005), 180–193.
27. Ma, B. Why greed works for shortest common superstring problem. In *CPM '08: Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching* (2008), 244–254.
28. Manthey, B. and Röglin, H. Improved smoothed analysis of the k-means method. In *SODA '09* (2009).
29. Mehlhorn, K. and Näher, S. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, New York, 1999.
30. Mitzenmacher, M. and Vadhan, S. Why simple hash functions work: exploiting the entropy in a data stream. In *SODA '08: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2008), 746–755.
31. Röglin, H. and Vöcking, B. Smoothed analysis of integer programming. *Proceedings of the 11th International Conference on Integer Programming and Combinatorial Optimization*. M. Junger and V. Kaibel, eds. Volume 3509 of Lecture Notes in Computer Science, Springer, 2005, 276–290.
32. Rudelson, M. and Vershynin, R. The littlewood-offord problem and invertibility of random matrices. *Adv. Math.* 218, (June 2008), 600–633.
33. Sankar, A. Smoothed analysis of Gaussian elimination. Ph.D. Thesis, MIT, 2004.
34. Sankar, A., Spielman, D.A., and Teng, S.-H. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM J. Matrix Anal. Appl.* 28, 2 (2006), 446–476.
35. Spielman, D.A. and Teng, S.-H. Smoothed analysis of algorithms. In *Proceedings of the International Congress of Mathematicians* (2002), 597–606.
36. Spielman, D.A. and Teng, S.-H. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM* 51, 3 (2004), 385–463.
37. Teng, S.-H. Algorithm design and analysis with perturbations. In *Fourth International Congress of Chinese Mathematicians* (2007).
38. Vershynin, R. Beyond Hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (2006), 133–142.
39. Vu, V.H. and Tao, T. The condition number of a randomly perturbed matrix. In *STOC '07: the 39th Annual ACM Symposium on Theory of Computing* (2007), 248–255.
40. Wschebor, M. Smoothed analysis of $x(\mathbf{a})$. *J. Complexity* 20, 1 (February 2004), 97–107.

Daniel A. Spielman (spielman@cs.yale.edu) is a professor of Applied Mathematics and Computer Science at Yale University, New Haven, CT.

Shang-Hua Teng (shanghai.teng@gmail.com) is a professor of Department of Computer Science, at Boston University, and senior research scientist at Akamai Technologies, Inc.

research highlights

P. 86

**Technical
Perspective
Relational Query
Optimization—
Data Management
Meets Statistical
Estimation**

By Surajit Chaudhuri

P. 87

**Distinct-Value Synopses
for Multiset Operations**

By Kevin Beyer, Rainer Gemulla, Peter J. Haas,
Berthold Reinwald, and Yannis Sismanis

P. 96

**Technical
Perspective
Data Stream
Processing—
When You Only
Get One Look**

By Johannes Gehrke

P. 97

**Finding the Frequent
Items in Streams of Data**

By Graham Cormode and Marios Hadjieleftheriou

Technical Perspective

Relational Query Optimization—Data Management Meets Statistical Estimation

By Surajit Chaudhuri

RELATIONAL SYSTEMS HAVE made it possible to query large collections of data in a *declarative* style through languages such as SQL. The queries are translated into expressions consisting of relational operations but do not refer to any implementation details. There is a key component that is needed to support this declarative style of programming and that is the *query optimizer*. The optimizer takes the query expression as input and determines how best to execute that query. This amounts to a combinatorial optimization on a complex search space: finding a low-cost execution plan among all plans that are equivalent to the given query expression (considering possible ordering of operators, alternative implementations of logical operators, and different use of physical structures such as indexes). The success that relational databases enjoy today in supporting complex decision-support queries would not have been a reality without innovation in query optimization technology.

In trying to identify a good execution plan, the query optimizer must be aware of statistical properties of data over which the query is defined because these statistical properties strongly influence the cost of executing the query. Examples of such statistical properties are total number of rows in the relation, distribution of values of attributes of a relation, and the number of distinct values of an attribute. Because the optimizer needs to search among many alternative execution plans for the given query and tries to pick one with low cost, it needs such statistical estimation not only for the input relations, but also for many sub-expressions that it considers part of its combinatorial search. Indeed, statistical properties of the sub-expressions guide the exploration of alternatives considered by the query optimizer. Since access to a large data set can be costly, it is not feasible to determine statistical properties of these sub-expressions by executing each of


them. It is also important to be able to maintain these statistics efficiently in the face of data updates. These requirements demand a judicious trade-off between quality of estimation and the overheads of doing this estimation.

The early commercial relational systems used estimation techniques using summary structures such as simple histograms on attributes of the relation. Each bucket of the histogram represented a range of values. The histogram captured the total number of rows and number of distinct values for each bucket of the histogram. For larger query expressions, histograms were derived from histograms on its sub-expressions in an adhoc manner. Since the mid-1990s, the unique challenges of statistical estimation in the context of query optimization attracted many researchers with backgrounds and interests in algorithms and statistics. We saw development of principled approaches to these statistical estimation problems that leveraged randomized algorithms such as probabilistic counting. Keeping with the long-standing tradition of close relation between database research and the database industry, some of these solutions have been adopted in commercial database products.

The following paper by Beyer et al. showcases recent progress in statistical estimations in the context of query op-

The authors showcase recent progress in statistical estimations in the context of query optimization.

timization. It revisits the difficult problem of efficient estimation of the number of distinct values in an attribute and makes a number of contributions by building upon past work that leverages randomized algorithms. It suggests an unbiased estimator for distinct values that has a lower mean squared error than previously proposed estimators based on a single scan of data. The authors propose a summary structure (*synopsis*) for a relation such that the number of distinct values in a query using multiset union, multiset intersection, and multiset difference operations over a set of relations can be estimated from the synopses of base relations. Furthermore, if only one of the many partitions of a relation is updated, the synopsis for just that partition must be rebuilt to derive the distinct value estimations for the entire relation.

It has been 30 years since the framework of query optimization was defined by System-R and relational query optimization has been a great success story commercially. Yet, statistical estimation problems for query expressions remain one area where significant advances are needed to take the next big leap in the state of the art for query optimization. Recently, researchers are trying to understand how additional knowledge on statistical properties of data and queries can best be gleaned from past executions to enhance the core statistical estimation abilities. Although I have highlighted query optimization, such statistical estimation techniques also have potential applications in other areas such as data profiling and approximate query processing. I invite you to read the following paper to sample a subfield that lies at the intersection of database management systems, statistics, and algorithms. 

Surajit Chaudhuri (surajitc@microsoft.com) is a principal researcher and research area manager at Microsoft. He is an ACM Fellow and the recipient of the 2004 ACM SIGMOD Contributions Award.

© 2009 ACM 0001-0782/09/1000 \$10.00

Distinct-Value Synopses for Multiset Operations

By Kevin Beyer, Rainer Gemulla, Peter J. Haas, Berthold Reinwald, and Yannis Sismanis

Abstract

The task of estimating the number of distinct values (DVs) in a large dataset arises in a wide variety of settings in computer science and elsewhere. We provide DV estimation techniques for the case in which the dataset of interest is split into partitions. We create for each partition a synopsis that can be used to estimate the number of DVs in the partition. By combining and extending a number of results in the literature, we obtain both suitable synopses and DV estimators. The synopses can be created in parallel, and can be easily combined to yield synopses and DV estimates for “compound” partitions that are created from the base partitions via arbitrary multiset union, intersection, or difference operations. Our synopses can also handle deletions of individual partition elements. We prove that our DV estimators are unbiased, provide error bounds, and show how to select synopsis sizes in order to achieve a desired estimation accuracy. Experiments and theory indicate that our synopses and estimators lead to lower computational costs and more accurate DV estimates than previous approaches.

1. INTRODUCTION

The task of determining the number of distinct values (DVs) in a large dataset arises in a wide variety of settings. One classical application is population biology, where the goal is to determine the number of distinct species, based on observations of many individual animals. In computer science, applications include network monitoring, document search, predicate-selectivity estimation for database query optimization, storage-size estimation for physical database design, and discovery of metadata features such as keys and duplicates.

The number of DVs can be computed exactly by sorting the dataset and then executing a straightforward scan-and-count pass over the data; alternatively, a hash table can be constructed and used to compute the number of DVs. Neither of these approaches scales well to the massive datasets often encountered in practice, because of heavy time and memory requirements. A great deal of research over the past 25 years has therefore focused on scalable approximate methods. These methods work either by drawing a random sample of the data items and statistically extrapolating the number of DVs, or by taking a single pass through the data and using hashing techniques to compute an estimate using a small, bounded amount of memory.

Almost all of this work has focused on producing a given synopsis of the dataset, such as a random sample or bit vector, and then using the synopsis to obtain a DV estimate.

Issues related to combining and exploiting synopses in the presence of union, intersection, and difference operations on multiple datasets have been largely unexplored, as has the problem of handling deletions of items from the dataset. Such issues are the focus of this paper, which is about DV estimation methods when the dataset of interest is split into disjoint partitions, i.e., disjoint multisets.^a The idea is to create a synopsis for each partition so that (i) the synopsis can be used to estimate the number of DVs in the partition and (ii) the synopses can be combined to create synopses for “compound” partitions that are created from the base partitions using multiset union, intersection, or difference operations.

This approach permits parallel processing, and hence scalability of the DV-estimation task to massive datasets, as well as flexible processing of DV-estimation queries and graceful handling of fluctuating data-arrival rates. The partitioning approach can also be used to support automated data integration by discovering relationships between partitions. For example, suppose that the data is partitioned by its source: Amazon customers versus YouTube downloaders. Then DV estimates can be used to help discover subset-inclusion and functional-dependency relationships, as well as to approximate the Jaccard distance or other similarity metrics between the domains of two partitions; see Brown and Haas and Dasu et al.^{4,6}

Our goal is therefore to provide “partition-aware” synopses for DV estimation, as well as corresponding DV estimators that exploit these synopses. We also strive to maintain the best possible accuracy in our DV estimates, especially when the size of the synopsis is small: as discussed in the sequel, the size of the synopsis for a compound partition is limited by the size of the smallest input synopsis.

We bring together a variety of ideas from the literature to obtain a solution to our problem, resulting in best-of-breed DV estimation methods that can also handle multiset operations and deletions. We first consider,

^a Recall that a *multiset*, also called a *bag*, is an unordered collection of values, where a given value may appear multiple times, for example, $\{3,3,2,3,7,3,2\}$. Multiset union, intersection, and difference are defined in a natural way: if $n_A(v)$ and $n_B(v)$ denote the multiplicities of value v in multisets A and B , respectively, then the multiplicities of v in $A \cup B$, $A \cap B$, and $A \setminus B$ are given respectively by $n_A(v) + n_B(v)$, $\min(n_A(v), n_B(v))$, and $\max(n_A(v) - n_B(v), 0)$.

A previous version of this research paper was published in *Proceedings of the 2007 ACM SIGMOD Conference*.

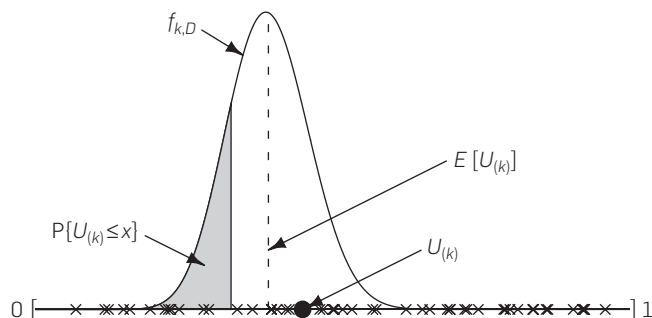
in Section 2, a simple “KMV” (K Minimum hash Values) synopsis² for a single partition—obtained by hashing the DVs in the partition and recording the K smallest hash values—along with a “basic” DV estimator based on the synopsis; see Equation 1. In Section 3, we briefly review prior work and show that virtually all prior DV estimators can be viewed as versions of, or approximations to, the basic estimator. In Section 4, we propose a new DV estimator—see Equation 2—that improves upon the basic estimator. The new estimator also uses the KMV synopsis and is a deceptively simple modification of the basic estimator. Under a probabilistic model of hashing, we show that the new estimator is unbiased and has lower mean-squared error than the basic estimator. Moreover, when there are many DVs and the synopsis size is large, we show that the new unbiased estimator has essentially the minimal possible variance of any DV estimator. To help users assess the precision of specific DV estimates produced by the unbiased estimator, we provide probabilistic error bounds. We also show how to determine appropriate synopsis sizes for achieving a desired error level.

In Section 5, we augment the KMV synopsis with counters—in the spirit of Ganguly et al. and Shukla et al.^{10, 18}—to obtain an “AKMV synopsis.” We then provide methods for combining AKMV synopses such that the collection of these synopses is “closed” under multiset operations on the parent partitions. The AKMV synopsis can also handle deletions of individual partition elements. We also show how to extend our simple unbiased estimator to exploit the AKMV synopsis and provide unbiased estimates in the presence of multiset operations, obtaining an unbiased estimate of Jaccard distance in the process; see Equations 7 and 8. Section 6 concerns some recent complements to, and extensions of, our original results in Beyer et al.³

2. A BASIC ESTIMATOR AND SYNOPSIS

The idea behind virtually all DV estimators can be viewed as follows. Each of the D DVs in a dataset is mapped to a random location on the unit interval, and we look at the position $U_{(k)}$ of the k th point from the left, for some fixed value of k ; see Figure 1. The larger the value of D , i.e., the greater the number of points on the unit interval, the smaller the value of $U_{(k)}$. Thus D can plausibly be estimated by a decreasing function of $U_{(k)}$.

Figure 1. 50 random points on the unit interval ($D = 50, k = 20$).



Specifically, if $D \gg 1$ points are placed randomly and uniformly on the unit interval, then, by symmetry, the expected distance between any two neighboring points is $1/(D + 1) \approx 1/D$, so that the expected value of $U_{(k)}$, the k th smallest point, is $E[U_{(k)}] \approx \sum_{j=1}^k (1/D) = k/D$. Thus $D \approx k/E[U_{(k)}]$. The simplest estimator of $E[U_{(k)}]$ is simply $U_{(k)}$ itself—the so-called “method-of-moments” estimator—and yields the *basic estimator*

$$\hat{D}_k^{\text{BE}} = k/U_{(k)} \quad (1)$$

The above 1-to-1 mapping from the D DVs to a set of D uniform random numbers can be constructed perfectly using $O(D \log D)$ memory, but this memory requirement is clearly infeasible for very large datasets. Fortunately, a hash function—which typically only requires an amount of memory logarithmic in D —often “looks like” a uniform random number generator. In particular, let $\mathcal{D}(A) = \{v_1, v_2, \dots, v_D\}$ be the *domain* of multiset A , i.e., the set of DVs in A , and let h be a hash function from $\mathcal{D}(A)$ to $\{0, 1, \dots, M\}$, where M is a large positive integer. For many hash functions, the sequence $h(v_1), h(v_2), \dots, h(v_D)$ looks like the realization of a sequence of independent and identically distributed (i.i.d.) samples from the discrete uniform distribution on $\{0, 1, \dots, M\}$. Provided that M is sufficiently greater than D , the sequence $U_1 = h(v_1)/M, U_2 = h(v_2)/M, \dots, U_D = h(v_D)/M$ will approximate the realization of a sequence of i.i.d. samples from the continuous uniform distribution on $[0, 1]$. This assertion requires that M be much larger than D to avoid collisions, i.e., to ensure that, with high probability, $h(v_i) \neq h(v_j)$ for all $i \neq j$. A “birthday problem” argument^{16, p. 45} shows that collisions will be avoided when $M = \Omega(D^2)$. We assume henceforth that, for all practical purposes, any hash function that arises in our discussion avoids collisions. We use the term “looks like” in an empirical sense, which suffices for applications. Thus, in practice, the estimator \hat{D}_k^{BE} can be applied with $U_{(k)}$ taken as the k th smallest hash value (normalized by a factor of $1/M$). In general, $E[1/X] > 1/E[X]$ for a non-negative random variable X ,^{17, p. 351} and hence

$$E[\hat{D}_k^{\text{BE}}] = E[k/U_{(k)}] > k/E[U_{(k)}] \approx D$$

Algorithm 1 (KMV Computation).

```

1:  $h$ : hash function from domain of dataset to  $\{0, 1, \dots, M\}$ 
2:  $L$ : list of  $k$  smallest hash values seen so far
3:  $\text{maxVal}(L)$ : returns the largest value in  $L$ 
4:
5: for each item  $x$  in the dataset do
6:    $v = h(x)$ 
7:   if  $v \notin L$  then
8:     if  $|L| < k$  then
9:       insert  $v$  into  $L$ 
10:    else if  $v < \text{maxVal}(L)$  then
11:      insert  $v$  into  $L$ 
12:      remove largest element of  $L$ 
13:    end if
14:  end if
15: end for

```

i.e., the estimator \hat{D}_k^{BE} is biased upwards for each possible value of D , so that it overestimates D on average. Indeed, it follows from the results in Section 4.1 that $E[\hat{D}_k^{\text{BE}}] = \infty$ for $k = 1$. In Section 4, we provide an unbiased estimator that also has lower mean-squared error than \hat{D}_k^{BE} .

Note that, in a certain sense, the foregoing view of hash functions—as algorithms that effectively place points on the unit interval according to a uniform distribution—represents a worst-case scenario with respect to the basic estimator. To the extent that a hash function spreads points *evenly* on $[0, 1]$, i.e., without the clumping that is a byproduct of randomness, the estimator \hat{D}_k^{BE} will yield more accurate estimates. We have observed this phenomenon experimentally.³

The foregoing discussion of the basic estimator immediately implies a choice of synopsis for a partition A . Using a hash function as above, hash all of the DVs in A and then record the k smallest hash values. We call this synopsis a *KMV synopsis* (for k minimum values). The KMV synopsis can be viewed as originating in Bar-Yossef et al.,² but there is no discussion in Bar-Yossef et al.² about implementing, constructing, or combining such synopses.

As discussed previously, we need to have $M = \Omega(D^2)$ to avoid collisions. Thus each of the k hash values requires $O(\log M) = O(\log D)$ bits of storage, and the required size of the KMV synopsis is $O(k \log D)$.

A KMV synopsis can be computed from a single scan of the data partition, using Algorithm 1. The algorithm uses a sorted list of k hash values, which can be implemented using, e.g., a priority queue. The membership check in line 7 avoids unnecessary processing of duplicate values in the input data partition, and can be implemented using a temporary hash table that is discarded after the synopsis has been built.

Assuming that the scan order of the items in a partition is independent of the items' hash values, we obtain the following result.

THEOREM 1. *The expected cost to construct a KMV synopsis of size k from a partition A comprising N data items having D distinct values is $O(N + k \log k \log D)$.*

PROOF. The hashing step and membership check in lines 6 and 7 incur a cost of $O(1)$ for each of the N items in A , for a total cost of $O(N)$. To compute the expected cost of executing the remaining steps of the algorithm, observe that the first k DVs encountered are inserted into the priority queue (line 9), and each such insertion has a cost of at most $O(\log k)$, for an overall cost of $O(k \log k)$. Each subsequent new DV encountered will incur an $O(\log k)$ cost if it is inserted (line 11), or an $O(1)$ cost otherwise. (Note that a given DV will be inserted at most once, at the time it is first encountered, regardless of the number of times that it appears in A .) The i th new DV encountered is inserted only if its normalized hash value U_i is less than M_i , the largest normalized hash value currently in the synopsis. Because points are placed uniformly, the conditional probability of this event, given the value of M_i , is $P\{U_i < M_i | M_i\} = M_i$. By the law of total expectation, $P\{U_i < M_i\} = E[P\{U_i < M_i | M_i\}] = E[M_i] = k/i$. Thus the expected cost for handling the remaining $D - k$ DVs is

$$\begin{aligned} E[\text{Cost}] &= \sum_{i=k+1}^D [(k/i)O(\log k) + (1 - (k/i))O(1)] \\ &< \sum_{i=1}^D (k/i)O(\log k) + O(D) = O(D) + O(k \log k) \sum_{i=1}^D (1/i) \\ &= O(D + k \log k \log D) \end{aligned}$$

since $\sum_{i=1}^D (1/i) = O(\log D)$. The overall expected cost is thus $O(N + D + k \log k + k \log k \log D) = O(N + k \log k \log D)$. \square

We show in Section 5 that adding counters to the KMV synopsis has a negligible effect on the construction cost, and results in a desirable “closure” property that permits efficient DV estimation under multiset operations.

3. PRIOR WORK

We now give a unified view of prior synopses and DV estimators, and discuss prior methods for handling compound partitions.

3.1. Synopses for DV estimation

In general, the literature on DV estimation does not discuss synopses explicitly, and hence does not discuss issues related to combining synopses in the presence of set operations on the corresponding partitions. We can, however, infer potential candidate synopses from the various algorithm descriptions. The literature on DV estimation is enormous, so we content ourselves with giving highlights and pointers to further references; for some helpful literature reviews, see Beyer et al.,³ Gemulla,¹¹ Gibbons¹³ and Haas and Stokes.¹⁴

Random Samples: Historically, the problem of DV estimation was originally considered for the case in which the synopsis comprises a random sample of the data items. Applications included estimation of species diversity (as discussed in the introduction), determining the number of distinct Roman coins based on the sample of coins that have survived, estimating the size of Shakespeare's vocabulary based on his extant writings, estimating the number of distinct individuals in a set of merged census lists, and so forth; see Haas and Stokes¹⁴ and references therein. The key drawback is that DV estimates computed from such a synopsis can be very inaccurate, especially when the dataset is dominated by a few highly frequent values, or when there are many DVs, each having a low frequency (but not all unique). With high probability, a sample of size k will have only one or two DVs in the former case—leading to severe underestimates—and k DVs in the latter case—leading to severe overestimates. For this reason, especially in the computer science literature, the emphasis has been on algorithms that take a complete pass through the dataset, but use limited memory. When the dataset is massive, our results are of particular interest, since we can parallelize the DV-estimate computation.

Note that if we modify the KMV synopsis to record not the hash of a value, but the value itself, then we are in effect maintaining a uniform sample of the DVs in the dataset, thereby avoiding the problems mentioned above. See Gemulla¹¹ for a thorough discussion of “distinct-value sampling” and its relationship to the DV-estimation problem.

Bit-Vector Synopses: The oldest class of synopses based on single-pass, limited-memory processing comprises various types of bit vectors. The “linear counting” technique^{1,21}

hashes each DV to a position in a bit vector V of length $M = O(D)$, and uses the number of 1-bits to estimate the DV count. Its $O(D)$ storage requirement is typically unacceptable for modern datasets.

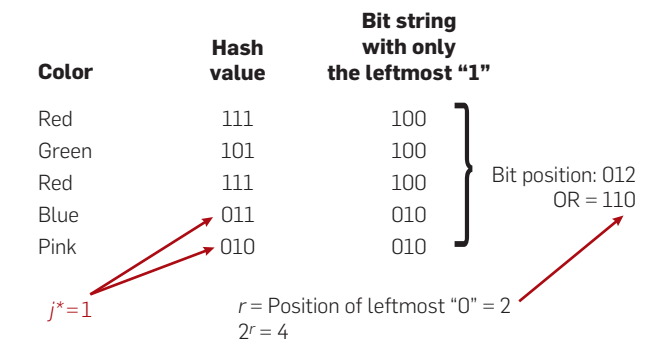
The “logarithmic counting” method of Flajolet and Martin^{1,9} uses a bit vector of length $L = O(\log D)$. The idea is to hash each of the DVs in A to the set $\{0, 1\}^L$ of binary strings of length L , and look for patterns of the form $0^j 1$ in the leftmost bits of the hash values. For a given value of j , the probability of such a pattern is $1/2^{j+1}$, so the expected observed number of such patterns after D DVs have been hashed is $D/2^{j+1}$. Assuming that the longest observed pattern, say with j^* leading 0’s, is expected to occur once, we set $D/2^{j^*+1} = 1$, so that $D = 2^{j^*+1}$; see Figure 2, which has been adapted from Astrahan et al.¹ The value of j^* is determined approximately, by taking each hash value, transforming the value by zeroing out all but the leftmost 1, and computing the logarithmic-counting synopsis as the bitwise-OR of the transformed values. Let r denote the position (counting from the left, starting at 0) of the leftmost 0 bit in the synopsis. Then r is an upper bound for j^* , and typically a lower bound for $j^* + 1$, leading to a crude (under)estimate of 2^r . For example, if $r = 2$, so that the leftmost bits of the synopsis are 110 (as in Figure 2), we know that the pattern 001 did not occur in any of the hash values, so that $j^* < 2$.

The actual DV estimate is obtained by multiplying 2^r by a factor that corrects for the downward bias, as well as for hash collisions. In the complete algorithm, several independent values of r are, in effect, averaged together (using a technique called “stochastic averaging”) and then exponentiated. Subsequent work by Durand and Flajolet⁸ improves on the storage requirement of the logarithmic counting algorithm by tracking and maintaining j^* directly. The number of bits needed to encode j^* is $O(\log \log D)$, and hence the technique is called LogLog counting.

The main drawback of the above bit-vector data structures, when used as synopses in our partitioned-data setting, is that union is the only supported set operation. One must, e.g., resort to the inclusion/exclusion formula to handle set intersections. As the number of set operations increases, this approach becomes extremely cumbersome, expensive, and inaccurate.

Several authors^{10,18} have proposed replacing each bit in the logarithmic-counting bit vector by an exact or approximate

Figure 2. Logarithmic counting.



counter, in order to permit DV estimation in the presence of both insertions and deletions to the dataset. This modification does not ameliorate the inclusion/exclusion problem, however.

Sample-Counting Synopsis: Another type of synopsis arises from the “sample counting” DV-estimation method—also called “adaptive sampling”—credited to Wegman.¹ Here the synopsis for partition A comprises a subset of $\{h(v): v \in \mathcal{S}(A)\}$, where $h: \mathcal{S}(A) \mapsto \{0, 1, \dots, M\}$ is a hash function as before. In more detail, the synopsis comprises a fixed-size buffer that holds binary strings of length $L = \log(M)$, together with a “reference” binary string s , also of length L . The idea is to hash the DVs in the partition, as in logarithmic counting, and insert the hash values into a buffer that can hold up to $k > 0$ hash values; the buffer tracks only the distinct hash values inserted into it. When the buffer fills up, it is purged by removing all hash values whose leftmost bit is not equal to the leftmost bit of s ; this operation removes roughly half of the hash values in the buffer. From this point on, a hash value is inserted into the buffer if and only if the first bit matches the first bit of s . The next time the buffer fills up, a purge step (with subsequent filtering) is performed by requiring that the two leftmost bits of each hash value in the buffer match the two leftmost bits of the reference string. This process continues until all the values in the partition have been hashed. The final DV estimate is roughly equal to $K2^r$, where r is the total number of purges that have occurred and K is the final number of values in the buffer. For sample-counting algorithms with reference string equal to $00 \dots 0$, the synopsis holds the K smallest hash values encountered, where K lies roughly between $k/2$ and k .

The Bellman Synopsis: In the context of the Bellman system, the authors in Dasu et al.⁶ propose a synopsis related to DV estimation. This synopsis comprises k entries and uses independent hash functions h_1, h_2, \dots, h_k ; the i th synopsis entry is given by the i th *minHash* value $m_i = \min_{v \in \mathcal{S}(A)} h_i(v)$. The synopsis for a partition is not actually used to directly compute the number of DVs in the partition, but rather to compute the Jaccard distance between partition domains; see Section 3.3. (The *Jaccard distance* between ordinary sets A and B is defined as $J(A, B) = |A \cap B|/|A \cup B|$. If $J(A, B) = 1$, then $A = B$; if $J(A, B) = 0$, then A and B are disjoint.) Indeed, this synopsis cannot be used directly for DV estimation because the associated DV estimator is basically \hat{D}_1^{BE} , which has infinite expectation; see Section 2. When constructing the synopsis, each scanned data item must be hashed k times for comparison to the k current minHash values; for the KMV synopsis, each scanned item need only be hashed once.

3.2. DV estimators

The basic estimator \hat{D}_k^{BE} was proposed in Bar-Yossef et al.,² along with conservative error bounds based on Chebyshev’s inequality. Interestingly, both the logarithmic and sample-counting estimators can be viewed as approximations to the basic estimator. For logarithmic counting—specifically the Flajolet–Martin algorithm—consider the binary decimal representation of the normalized hash values $h(v)/M$, where

$M = 2^l$, e.g., a hash value $h(v) = 00100110$, after normalization, will have the binary decimal representation 0.00100110. It can be seen that the smallest normalized hash value is approximately equal to 2^{-r} , so that, modulo the correction factor, the Flajolet–Martin estimator (without stochastic averaging) is $1/2^{-r}$, which roughly corresponds to \hat{D}_1^{BE} . The final F-M estimator uses stochastic averaging to average independent values of r and hence compute an estimator \hat{E} of $E[\log_2 \hat{D}_1^{\text{BE}}]$, leading to a final estimate of $\hat{D} = c2^{\hat{E}}$, where the constant c approximately unbiased the estimator. (Our new estimators are exactly unbiased.) For sample counting, suppose, without loss of generality, that the reference string is $00 \cdots 0$ and, as before, consider the normalized binary decimal representation of the hash values. Thus the first purge leaves behind normalized values of the form $0.0 \cdots$, the second purge leaves behind values of the form $0.00 \cdots$, and so forth, the last (r)th purge leaving behind only normalized hash values with r leading 0's. Thus the number 2^{-r} (which has $r - 1$ leading 0's) is roughly equal to the largest of the K normalized hash values in the size- k buffer, so that the estimate $K/2^{-r}$ is roughly equal to \hat{D}_k^{BE} .

3.3. Estimates for compound partitions

To our knowledge, the only prior discussion of how to construct DV-related estimates for compound partitions is found in Dasu et al.⁶ DV estimation for the intersection of partitions A and B is not computed directly. Instead, the Jaccard distance $\rho = J(\mathcal{S}(A), \mathcal{S}(B))$ is estimated first by an estimator $\hat{\rho}$, and then the number of values in the intersection of $\mathcal{S}(A)$ and $\mathcal{S}(B)$ is estimated as

$$\hat{D} = \frac{\hat{\rho}}{\hat{\rho} + 1} (|\mathcal{S}(A)| + |\mathcal{S}(B)|)$$

The quantities $|\mathcal{S}(A)|$ and $|\mathcal{S}(B)|$ are computed exactly, by means of GROUP BY relational queries; our proposed estimators avoid the need to compute or estimate these quantities. There is no discussion in Dasu et al.⁶ of how to handle any set operations other than the intersection of two partitions. If one uses the principle of inclusion/exclusion to handle other set operations, the resulting estimation procedure will not scale well as the number of operations increases.

4. AN IMPROVED DV ESTIMATOR

As discussed previously, the basic estimator \hat{D}_k^{BE} is biased upwards for the true number of DVs D , and so somehow needs to be adjusted downward. We therefore consider the estimator

$$\hat{D}_k^{\text{UB}} = (k - 1)/U_{(k)} \quad (2)$$

and show that, both compared to the basic estimator and in a certain absolute sense, \hat{D}_k^{UB} has superior statistical properties, including unbiasedness. The \hat{D}_k^{UB} estimator forms the basis for the extended DV estimators, discussed in Section 5, used to estimate the number of DVs in a compound partition. Henceforth, we assume without further comment that $D > k$; if $D \leq k$, then we can easily detect this situation and compute the exact value of D from the synopsis.

4.1. Moments and error bounds

Let U_1, U_2, \dots, U_D be the normalized hash values of the D distinct items in the dataset; for our analysis, we model these values as a sequence of i.i.d. random variables from the uniform $[0, 1]$ distribution—see the discussion in Section 2. As before, denote by $U_{(k)}$ the k th smallest of U_1, U_2, \dots, U_D , that is, $U_{(k)}$ is the k th uniform order statistic. We can now apply results from the classical theory of order statistics to establish properties of the estimator $\hat{D}_k^{\text{UB}} = (k - 1)/U_{(k)}$. We focus on moments and error bounds; additional analysis of \hat{D}_k^{UB} can be found in Beyer et al.³

Our analysis rests on the fact that the probability density function (pdf) of $U_{(k)}$ is given by

$$f_{k,D}(t) = t^{k-1}(1-t)^{D-k}/B(k, D-k+1) \quad (3)$$

where $B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt$ denotes the standard beta function; see Figure 1. To verify (3), fix $x \in [0, 1]$ and observe that, for each $1 \leq i \leq D$, we have $P\{U_i \leq x\} = x$ by definition of the uniform distribution. The probability that exactly k of the D uniform random variables are less than or equal to x is a binomial probability: $\binom{D}{k} x^k (1-x)^{D-k}$. Thus the probability that $U_{(k)} \leq x$ is equal to the probability that at least k random variables are less than or equal to x , which is a sum of binomial probabilities:

$$P\{X_{(k)} \leq x\} = \sum_{j=k}^D \binom{D}{j} x^j (1-x)^{D-j} = \int_0^x f_{k,D}(t) dt$$

The second equality can be established by integrating the rightmost term by parts repeatedly and using the identity

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} = \frac{(a-1)!(b-1)!}{(a+b-1)!} \quad (4)$$

where $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the standard gamma function and the rightmost equality is valid for integer a and b . The result in (3) now follows by differentiation.

With (3) in hand, we can now determine the moments of \hat{D}_k^{UB} , in particular, the mean and variance. Denote by a^b the falling power $a(a-1) \cdots (a-b+1)$.

THEOREM 2. *Suppose that $r \geq 0$ is an integer with $r < k$. Then*

$$E[(\hat{D}_k^{\text{UB}})^r] = (k-1)^r D^r / (k-1)^r \quad (5)$$

In particular, $E[\hat{D}_k^{\text{UB}}] = D$ provided that $k > 1$, and $\text{Var}[\hat{D}_k^{\text{UB}}] = D(D-k+1)/(k-2)$ provided that $k > 2$.

PROOF. For $k > r \geq 0$, it follows from (3) that

$$E[U_{(k)}^{-r}] = \int_0^1 \frac{f_{k,D}(t)}{t^r} dt = \frac{B(k-r, D-k+1)}{B(k, D-k+1)}$$

and the first assertion of the theorem follows directly from (4). Setting $r = 1$ in (5) yields the next assertion, and the final assertion follows from (5) and the relation $\text{Var}[\hat{D}_k^{\text{UB}}] = E[(\hat{D}_k^{\text{UB}})^2] - E^2[\hat{D}_k^{\text{UB}}]$. \square

Recall that the mean squared error (MSE) of a statistical estimator X of an unknown quantity μ is defined as $\text{MSE}[X] = E[(X - \mu)^2] = \text{Var}[X] + \text{Bias}^2[X]$. To compare the MSE of the

basic and unbiased estimators, note that, by (5), $E[\hat{D}_k^{BE}] = kD/(k-1)$ and

$$\text{MSE}[\hat{D}_k^{BE}] = \left(\frac{k}{k-1}\right)^2 \text{MSE}[\hat{D}_k^{UB}] + \left(\frac{D}{k-1}\right)^2.$$

Thus \hat{D}_k^{BE} is biased high for D , as discussed earlier, and has higher MSE than \hat{D}_k^{UB} .

We can also use the result in (3) to obtain probabilistic (relative) error bounds for the estimator \hat{D}_k^{UB} . Specifically, set $I_x(a, b) = \int_0^x t^{a-1} (1-t)^{b-1} dt / B(a, b)$, so that $P\{U_{(k)} \leq x\} = I_x(k, D-k+1)$. Then, for $0 < \varepsilon < 1$ and $k \geq 1$, we have

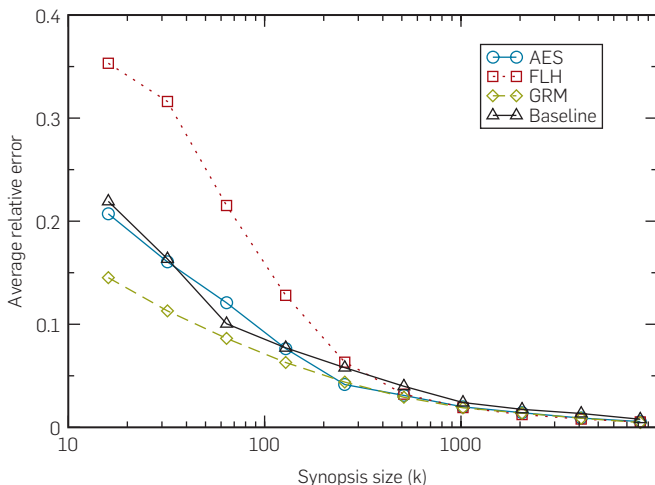
$$\begin{aligned} P\left\{\left|\frac{\hat{D}_k^{UB} - D}{D}\right| \leq \varepsilon\right\} &= P\left\{\frac{k-1}{(1+\varepsilon)D} \leq U_{(k)} \leq \frac{k-1}{(1-\varepsilon)D}\right\} \\ &= I_{\frac{k-1}{(1-\varepsilon)D}}(k, D-k+1) - I_{\frac{k-1}{(1+\varepsilon)D}}(k, D-k+1). \end{aligned} \quad (6)$$

For example, if $D = 10^6$ and the KMV synopsis size is $k = 1024$, then, with probability $\delta = 0.95$, the estimator \hat{D}_k^{UB} will be within $\pm 4\%$ of the true value; this result is obtained by equating the right side of (6) to δ and solving for ε numerically. In practice, of course, D will be unknown, but we can assess the precision of \hat{D}_k^{UB} approximately by replacing D with \hat{D}_k^{UB} in the right side of (6) prior to solving for ε .

Error bounds can also be derived for \hat{D}_k^{BE} . As discussed in Beyer et al.,³ \hat{D}_k^{UB} is noticeably superior to \hat{D}_k^{BE} when k is small; for example, when $k = 16$ and $\delta = 0.95$, use of the unbiased estimator yields close to a 20% reduction in ε . As k increases, $k-1 \approx k$ and both estimators perform similarly.

The foregoing development assumes that the hash function behaves as a 1-to-1 mapping from the D DVs in the dataset to a set of D uniform random numbers. In practice, we must use real-world hash functions that only approximate this ideal; see Section 2. Figure 3 shows the effect of using real hash functions on real data. The RDW database was obtained from the data warehouse of a large financial company, and consists of

Figure 3. Hashing Effect on the RDW Dataset.



24 relational tables, with a total of 504 attributes and roughly 2.6 million tuples. For several different hash functions, we computed the average value of the relative error $\text{RE}(\hat{D}_k^{UB}) = |\hat{D}_k^{UB} - D|/D$ over multiple datasets in the database. The hash functions are described in detail in Beyer et al.³; for example, the Advanced Encryption Standard (AES) hash function is a well established cipher function that has been studied extensively. The “baseline” curve corresponds to an idealized hash function as used in our analysis. As can be seen, the real-world accuracies are consistent with the idealized results, and reasonable accuracy can be obtained even for synopsis sizes of $k < 100$. In Beyer et al.,³ we found that the relative performance of different hash functions was sensitive to the degree of regularity in the data; the AES hash function is relatively robust to such data properties, and is our recommended hash function.

4.2. Analysis with many DVs

When the number of DVs is known to be large, we can establish a minimum-variance property of the \hat{D}_k^{UB} estimator, and also develop useful approximate error bounds that can be used to select a synopsis size prior to data processing.

Minimum Variance Property: The classical statistical approach to estimating unknown parameters based on a data sample is the method of maximum likelihood.^{7, Sec. 4.2} A basic result for maximum-likelihood estimators^{17, Sec. 4.2.2} asserts that an MLE of an unknown parameter has the minimum variance over all possible parameter estimates as the sample size becomes large. We show that \hat{D}_k^{UB} is asymptotically equivalent to the maximum-likelihood estimator (MLE) as D and k become large. Thus, for $D \gg k \gg 1$, the estimator \hat{D}_k^{UB} has, to a good approximation, the minimal possible variance for any estimator of D .

To find the MLE, we cast our DV-estimation problem as a parameter estimation problem. Specifically, recall that $U_{(k)}$ has the pdf $f_{k,D}$ given in (3). The MLE estimate of D is defined as the value \hat{D} that maximizes the likelihood $L(D; U_{(k)})$ of the observation $U_{(k)}$, defined as $L(D; U_{(k)}) = f_{k,D}(U_{(k)})$. That is, roughly speaking, \hat{D} maximizes the probability of seeing the value of $U_{(k)}$ that was actually observed. We find this maximizing value by solving the equation $L'(D; U_{(k)}) = 0$, where the prime denotes differentiation with respect to D . We have $L'(D; U_{(k)}) = \ln(1 - U_{(k)}) - \Psi(D - k + 1) + \Psi(D + 1)$, where Ψ denotes the digamma function. If x is sufficiently large, then $\Psi(x) \approx \ln(x - 1) + \gamma$, where $\gamma \approx 0.5772$ denotes Euler’s constant. Applying this approximation, we obtain $\hat{D}_k^{MLE} \approx k/U_{(k)}$, so that the MLE estimator roughly resembles the basic estimator \hat{D}_k^{BE} provided that $D \gg k \gg 1$. In fact, our experiments indicated that \hat{D}_k^{MLE} and \hat{D}_k^{BE} are indistinguishable from a practical point of view. It follows that $\hat{D}_k^{MLE} \approx (k/k - 1)\hat{D}_k^{UB}$ is asymptotically equivalent to \hat{D}_k^{UB} as $k \rightarrow \infty$.

Approximate Error Bounds: To obtain approximate probabilistic error bounds when the number of DVs is large, we simply let $D \rightarrow \infty$ in (6), yielding

$$\begin{aligned} P\{\text{RE}(\hat{D}_k^{UB}) \leq \varepsilon\} &\approx e^{-\frac{k-1}{1+\varepsilon}} \left(1 + \sum_{i=1}^{k-1} \frac{(k-1)^i}{(1+\varepsilon)^i i!}\right) \\ &\quad - e^{-\frac{k-1}{1-\varepsilon}} \left(1 + \sum_{i=1}^{k-1} \frac{(k-1)^i}{(1-\varepsilon)^i i!}\right). \end{aligned}$$

An alternative derivation is given in Beyer et al.³ using a powerful proof technique that exploits well known properties of the exponential distribution. The approximate error bounds have the advantageous property that, unlike the exact bounds, they do not involve the unknown quantity D . Thus, given desired values of ϵ and δ , they can be used to help determine target synopsis sizes prior to processing the data. When D is not large, the resulting recommended synopsis sizes are slightly larger than necessary, but not too wasteful. It is also shown in Beyer et al.³ that $E[\text{RE}(\hat{D}_k^{\text{UB}})] \approx (\pi(k-2)/2)^{-1/2}$ for large D , further clarifying the relationship between synopsis size and accuracy.

5. FLEXIBLE AND SCALABLE ESTIMATION

The discussion up until now has focused on improving the process of creating and using a synopsis to estimate the number of DVs in a single base partition, i.e., in a single dataset. As discussed in the introduction, however, the true power of our techniques lies in the ability to split up a massive dataset into partitions, create synopses for the partitions in parallel, and then compose the synopses to obtain DV estimates for arbitrary combinations of partitions, e.g., if the combination of interest is in fact the union of all the partitions, then, as in the classical setting, we can estimate the number of DVs in the entire dataset, while parallelizing the traditional sequential scan of the data. On the other hand, estimates of the number of DVs in the intersection of partitions can be used to estimate similarity measures such as Jaccard distance.

We therefore shift our focus to DV estimation for a compound partition that is created from a set of base partitions using the multiset operations of intersection, union, and difference. To handle compound partitions, we augment our KMV synopses with counters; we show that the resulting AKMV synopses are “closed” under multiset operations on the parent partitions. The closure property implies that if A and B are compound partitions and E is obtained from A and B via a multiset operation, then we can compute an AKMV synopsis for E from the corresponding AKMV synopses for A and B , and unbiasedly estimate the number of DVs in E from this resulting synopsis. This procedure avoids the need to access the synopsis for each of the base partitions that were used to create A and B . The AKMV synopsis can also handle deletions of individual items from the dataset. As discussed below, the actual DV estimators that we use for compound partitions are, in general, extensions of the simple \hat{D}_k^{UB} estimator developed in Section 4.

5.1. AKMV synopses

We assume throughout that all synopses are created using the same hash function $h: \mathcal{S} \mapsto \{0, 1, \dots, M\}$, where \mathcal{S} denotes the domain of the data values that appear in the partitions and $M = \Omega(|\mathcal{S}|^2)$ as discussed in Section 2. We start by focusing on insertion-only environments, and discuss deletions in Section 5.3.

We first define the AKMV synopsis of a base partition A , where A has a KMV synopsis $L = (h(v_1), h(v_2), \dots, h(v_k))$ of size k , with $h(v_1) < h(v_2) < \dots < h(v_k)$. The AKMV synopsis of A is defined as $L^+ = (L, c)$, where $c = (c(v_1), c(v_2), \dots, c(v_k))$

is a set of k non-negative counters. The quantity $c(v)$ is the multiplicity in A of the value v . The first two lines in Figure 4 show the normalized hash values and corresponding counter values in the AKMV synopses L_A^+ and L_B^+ of two base partitions A and B , respectively. (Circles and squares represent normalized hash values; inscribed numbers represent counter values.)

The size of the AKMV synopsis is $O(k \log D + k \log N)$, where N is the number of data items in A . It is easy to modify Algorithm 1 to create and maintain counters via $O(1)$ operations. The modified synopsis retains the original construction complexity of $O(N + k \log k \log D)$.

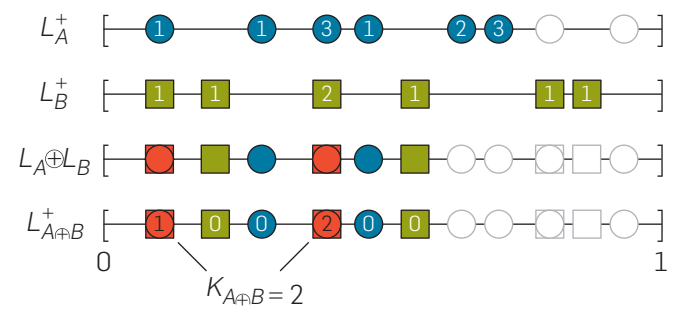
To define an AKMV synopsis for compound partitions, we first define an operator \oplus for combining KMV synopses. Consider two partitions A and B , along with their KMV synopses L_A and L_B of sizes k_A and k_B , respectively. Define $L_A \oplus L_B$ to be the ordered list comprising the k smallest values in $L_A \cup L_B$, where $k = \min(k_A, k_B)$ and we temporarily treat L_A and L_B as sets rather than ordered lists. (See line 3 of Figure 4; the yellow points correspond to values that occur in both L_A and L_B .) Observe that the \oplus operator is symmetric and associative.

THEOREM 3. *The ordered list $L = L_A \oplus L_B$ is the size- k KMV synopsis of $A \uplus B$, where $k = \min(k_A, k_B)$.*

PROOF. We again temporarily treat L_A and L_B as sets. For a multiset S with $\mathcal{S}(S) \subseteq \mathcal{S}$, write $h(S) = \{h(v): v \in \mathcal{S}(S)\}$, and denote by G the set of k smallest values in $h(A \uplus B)$. Observe that G contains the k' smallest values in $h(A)$ for some $k' \leq k$, and these k' values therefore are also contained in L_A , i.e., $G \cap h(A) \subseteq L_A$. Similarly, $G \cap h(B) \subseteq L_B$, so that $G \subseteq L_A \cup L_B$. For any $h \in (L_A \cup L_B) \setminus G$, we have that $h > \max_{h' \in G} h'$ by definition of G , because $h \in h(A \uplus B)$. Thus G in fact comprises the k smallest values in $L_A \cup L_B$, so that $L = G$. Now observe that, by definition, G is precisely the size- k KMV synopsis of $A \uplus B$. \square

We next define the AKMV synopsis for a compound partition E created from $n \geq 2$ base partitions A_1, A_2, \dots, A_n using the multiset union, intersection, and set-difference operators. Some examples are $E = A_1 \uplus A_2$ and $E = ((A_1 \uplus A_2) \cap (A_3 \uplus A_4)) \setminus A_5$. The AKMV synopsis for E is defined as $L_E^+ = (L_E, c_E)$, where $L_E = L_{A_1} \oplus L_{A_2} \oplus \dots \oplus L_{A_n}$ is of size $k = \min(k_{A_1}, k_{A_2}, \dots, k_{A_n})$, and, for $v \in \cup_{i=1}^n \mathcal{S}(A_i)$, the counter $c_E(v)$ is the multiplicity

Figure 4. Combining two AKMV synopses of size $k = 6$ (synopsis elements are colored).



of v in E ; observe that $c_E(v) = 0$ if v is not present in E , so that there may be one or more zero-valued counters; see line 4 of Figure 4 for the case $E = A \sqcap B$.

With these definitions, the collection of AKMV synopses over compound partitions is closed under multiset operations, and an AKMV synopsis can be built up incrementally from the synopses for the base partitions. Specifically, suppose that we combine (base or compound) partitions A and B —having respective AKMV synopses $L_A^+ = (L_A, c_A)$ and $L_B^+ = (L_B, c_B)$ —to create $E = A \diamond B$, where $\diamond \in \{\cup, \sqcap, \setminus, \vdash\}$. Then the AKMV synopsis for E is $(L_A \oplus L_B, c_E)$, where

$$c_E(v) = \begin{cases} c_A(v) + c_B(v) & \text{if } \diamond = \cup; \\ \min(c_A(v), c_B(v)) & \text{if } \diamond = \sqcap; \\ \max(c_A(v) - c_B(v), 0) & \text{if } \diamond = \setminus. \end{cases}$$

As discussed in Beyer et al. and Gemulla,^{3, 11} the AKMV synopsis can be sometimes be simplified. If, for example, all partitions are ordinary sets (not multisets) and ordinary set operations are used to create compound partitions, then the AKMV counters can be replaced by a compact bit vector, yielding an $O(k \log D)$ synopsis size.

5.2. DV estimator for AKMV synopses

We now show how to estimate the number of DVs for a compound partition using the partition's AKMV synopsis; to this end, we need to generalize the unbiased DV estimator of Section 4. Consider a compound partition E created from $n \geq 2$ base partitions A_1, A_2, \dots, A_n using multiset operations, along with the AKMV synopsis $L_E^+ = (L_E, c_E)$, where $L_E = (h(v_1), h(v_2), \dots, h(v_k))$. Denote by $V_L = \{v_1, v_2, \dots, v_k\}$ the set of data values corresponding to the elements of L_E . (Recall our assumption that there are no hash collisions.) Set $K_E = |\{v \in V_L : c_E(v) > 0\}|$; for the example $E = A \sqcap B$ in Figure 4, $K_E = 2$. It follows from Theorem 3 that L_E is a size- k KMV synopsis of the multiset $A_{\cup} = A_1 \cup A_2 \cup \dots \cup A_n$. The key observation is that, under our random hashing model, V_L can be viewed as a random sample of size k drawn uniformly and without replacement from $\mathcal{S}(A_{\cup})$; denote by $D_{\cup} = |\mathcal{S}(A_{\cup})|$ the number of DVs in A_{\cup} . The quantity K_E is a random variable that represents the number of elements in V_L that also belong to the set $\mathcal{S}(E)$. It follows that K_E has a hypergeometric distribution: setting $D_E = |\mathcal{S}(E)|$, we have $P\{K_E = j\} = \binom{D_E}{j} \binom{D_{\cup} - D_E}{k-j} / \binom{D_{\cup}}{k}$. We now estimate D_E using K_E , as defined above, and $U_{(k)}$, the largest hash value in L_E . From Section 4.1 and Theorem 3, we know that $\hat{D}_{\cup} = (k-1)/U_{(k)}$ is an unbiased estimator of D_{\cup} ; we would like to “correct” this estimator via multiplication by the ratio $\rho = D_E/D_{\cup}$. We do not know ρ , but a reasonable estimate is

$$\hat{\rho} = K_E/k \tag{7}$$

the fraction of elements in the sample $V_L \subseteq \mathcal{S}(A_{\cup})$ that belong to $\mathcal{S}(E)$. This leads to our proposed estimator

$$\hat{D}_E = \frac{K_E}{k} \left(\frac{k-1}{U_{(k)}} \right) \tag{8}$$

THEOREM 4. *If $k > 1$ then $E[\hat{D}_E] = D_E$. If $k > 2$, then*

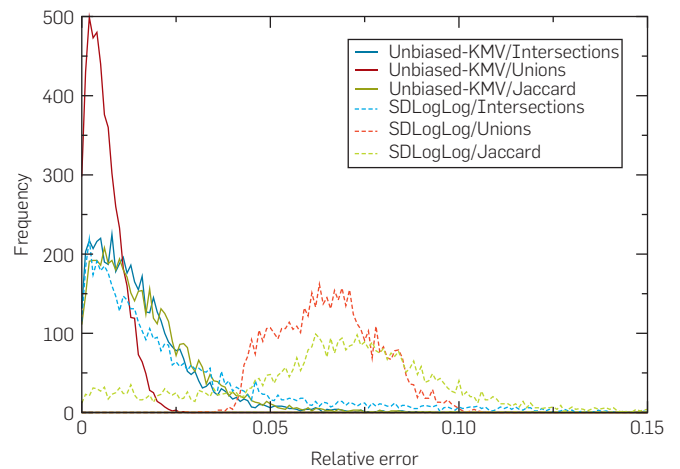
$$\text{Var}[\hat{D}_E] = \frac{D_E(kD_{\cup} - k^2 - D_{\cup} + k + D_E)}{k(k-2)}$$

PROOF. The distribution of K_E does not depend on the hash values $\{h(v) : v \in \mathcal{S}(A_{\cup})\}$. It follows that the random variables K_E and $U_{(k)}$, and hence the variables $\hat{\rho}$ and $U_{(k)}$, are statistically independent. By standard properties of the hypergeometric distribution, $E[K_E] = kD_E/D_{\cup}$, so that $E[\hat{D}_E] = E[\hat{\rho}\hat{D}_{\cup}] = E[\hat{\rho}]E[\hat{D}_{\cup}] = \rho D_{\cup} = D_E$. The second assertion follows in a similar manner. \square

Thus \hat{D}_E is unbiased for D_E . It also follows from the proof that the estimator $\hat{\rho}$ is unbiased for D_E/D_{\cup} . In the special case where $E = A \cap B$ for two ordinary sets A and B , the ratio ρ corresponds to the Jaccard distance $J(A, B)$, and we obtain an unbiased estimator of this quantity. We can also obtain probability bounds that generalize the bounds in (6); see Beyer et al.³ for details.

Figure 5 displays the accuracy of the AKMV estimator when estimating the number of DVs in a compound partition. For this experiment, we computed a KMV synopsis of size $k = 8192$ for each dataset in the RDW database. Then, for every possible pair of synopses, we used \hat{D}_E to estimate the DV count for the union and intersection, and also estimated the Jaccard distance between set domains using our new unbiased estimator $\hat{\rho}$ defined in (7). We also estimated these quantities using a best-of-breed estimator: the SDLogLog estimator, which is a highly tuned implementation of the LogLog estimator given in Durand and Flajolet.⁸ For this latter estimator, we estimated the number of DVs in the union directly, and then used the inclusion/exclusion rule to estimate the DV count for the intersection and then for the Jaccard distance. The figure displays, for each estimator, a relative-error histogram for each of these three multiset operations. (The histogram shows, for each possible RE value, the number of dataset pairs for which the DV estimate yielded that value.) For the

Figure 5. Accuracy Comparison for Union, Intersection, and Jaccard Distance on the RDW Dataset.



majority of the datasets, the unbiased estimator based on the KMV synopsis provides estimates that are almost ten times more accurate than the SDLogLog estimates, even though both methods used exactly the same amount of available memory.

5.3. Deletions

We now show how AKMV synopses can easily support deletions of individual items. Consider a partition A that receives a stream of transactions of the form $+v$ or $-v$, corresponding to the insertion or deletion, respectively, of value v . A naive approach maintains two AKMV synopses: a synopsis L_i^+ for the multiset A_i of inserted items and a synopsis L_d^+ for the multiset A_d of deleted items. Computing the AKMV synopsis of the multiset difference $A_i \setminus A_d$ yields the AKMV synopsis L_A^+ of the true multiset A . We do not actually need two synopses: simply maintain the counters in a single AKMV synopsis L by incrementing the counter at each insertion and decrementing at each deletion. If we retain synopsis entries having counter values equal to 0, we produce precisely the synopsis L_A^+ described above. If too many counters become equal to 0, the quality of synopsis-based DV estimates will deteriorate. Whenever the number of deletions causes the error bounds to become unacceptable, the data can be scanned to compute a fresh synopsis.

6. RECENT WORK

A number of developments have occurred both in parallel with, and subsequent to, the work described in Beyer et al.³ Duffield et al.⁷ devised a sampling scheme for weighted items, called “priority sampling,” for the purpose of estimating “subset sums,” i.e., sums of weights over subsets of items. Priority sampling can be viewed as assigning a priority to an item i with weight w_i by generating random number U_i uniformly on $[0, 1/w_i]$, and storing the k smallest-priority items. In the special case of unit weights, if we use a hash function instead of generating random numbers and if we eliminate duplicate priorities, the synopsis reduces to the KMV synopsis and, when the subset in question corresponds to the entire population, the subset-sum estimator coincides with \hat{D}_k^{UB} . If duplicate priorities are counted rather than eliminated, the AKMV synopsis is obtained (but we then use different estimators). An “almost minimum” variance property of the priority-sampling-based estimate has been established in Szegegy²⁰; this result carries over to the KMV synopsis, and strengthens our asymptotic minimum-variance result. In Cohen et al.,⁵ the priority-sampling approach has recently been generalized to a variety of priority-assignment schemes.

Since the publication of Beyer et al.,³ there have been several applications of KMV-type synopses and corresponding estimators to estimate the number of items in a time-based sliding window,¹² the selectivity of set-similarity queries,¹⁵ the intersection sizes of posting lists for purposes of OLAP-style querying of content-management systems,¹⁹ and document-key statistics that are useful for identifying promising publishers in a publish-subscribe system.²²

7. SUMMARY AND CONCLUSION

We have revisited the classical problem of DV estimation, but from a synopsis-oriented point of view. By combining and extending previous results on DV estimation, we have obtained the AKMV synopsis, along with corresponding unbiased DV estimators. Our synopses are relatively inexpensive to construct, yield superior accuracy, and can be combined to easily handle compound partitions, permitting flexible and scalable DV estimation. □

References

1. Astrahan, M., Schkolnick, M., Whang, K. Approximating the number of unique values of an attribute without sorting. *Inf. Sys. 12* (1987), 11–15.
2. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L. Counting distinct elements in a data stream. In *Proc. RANDOM* (2002), 1–10.
3. Beyer, K.S., Haas, P.J., Reinwald, B., Sismanis, Y., Gemulla, R. On synopses for distinct-value estimation under multiset operations. In *Proc. ACM SIGMOD* (2007), 199–210.
4. Brown, P.G., Haas, P.J. Techniques for warehouse of sample data. In *Proc. ICDE* (2006).
5. Cohen, E., Kaplan, H. Tighter estimation using bottom k sketches. *Proc. VLDB Endow. 1, 1* (2008), 213–224.
6. Dasu, T., Johnson, T., Muthukrishnan, S., Shkapenyuk, V. Mining database structure; or, how to build a data quality browser. In *Proc. ACM SIGMOD* (2002), 240–251.
7. Duffield, N., Lund, C., Thorup, M. Priority sampling for estimation of arbitrary subset sums. *J. ACM 54, 6* (2007), 32.
8. Durand, M., Flajolet, P. Loglog counting of large cardinalities. In *Proc. ESA* (2003), 605–617.
9. Flajolet, P., Martin, G.N. Probabilistic counting algorithms for data base applications. *J. Comp. Sys. Sci. 31* (1985), 182–209.
10. Ganguly, S., Garofalakis, M., Rastogi, R. Tracking set-expression cardinalities over continuous update streams. *VLDB J. 13* (2004), 354–369.
11. Gemulla, R. *Sampling Algorithms for Evolving Datasets*. Ph.D. thesis, TU Dresden, Dept. of CS, 2008. <http://nbn-resolving.de/urn:nbn:de:bsz:14-ds-1224861856184-11644>.
12. Gemulla, R., Lehner, W. Sampling time-based sliding windows in bounded space. In *Proc. SIGMOD* (2008), 379–392.
13. Gibbons, P. Distinct-values estimation over data streams. In M. Garofalakis, J. Gehrke, and R. Rastogi, editors, *Data Stream Management: Processing High-Speed Data Streams*. Springer, 2009. To appear.
14. Haas, P.J., Stokes, L. Estimating the number of classes in a finite population. *J. Amer. Statist. Assoc. 93* (1998), 1475–1487.
15. Hadjieleftheriou, M., Yu, X., Koudas, N., Srivastava, D. Hashed samples: selectivity estimators for set similarity selection queries. *Proc. VLDB Endow. 1, 1* (2008), 201–212.
16. Motwani, R., Raghavan, P. *Randomized Algorithms*. Cambridge University Press (1995).
17. Serfling, R.J. *Approximation Theorems of Mathematical Statistics*. Wiley, New York (1980).
18. Shukla, A., Deshpande, P., Naughton, J.F., Ramasamy, K. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *Proc. VLDB* (1996), 522–531.
19. Simitsis, A., Baid, A., Sismanis, Y., Reinwald, B. Multidimensional content exploration. *Proc. VLDB Endow. 1, 1* (2008), 660–671.
20. Szegegy, M. The DLT priority sampling is essentially optimal. In *Proc. STOC* (2006), 150–158.
21. Whang, K., Vander-Zanden, B.T., Taylor, H.M. A linear-time probabilistic counting algorithm for database applications. *ACM Trans. Database Sys. 15* (1990), 208–229.
22. Zimmer, C., Tryfonopoulos, C., Weikum, G. Exploiting correlated keywords to improve approximate information filtering. In *Proc. SIGIR* (2008), 323–330.

Kevin Beyer, Rainer Gemulla, Peter J. Haas, Berthold Reinwald, and Yannis Sismanis (kbeyer,rgemull,phaas,reinwald,syannis@us.ibm.com), IBM Almaden Research Center, San Jose, CA.

Technical Perspective

Data Stream Processing— When You Only Get One Look

By Johannes Gehrke

THE DATABASE AND systems communities have made great progress in developing database systems that allow us to store and query huge amounts of data. My first computer cost about \$1,000; it was a Commodore 64 with a 170KB floppy disk drive. Today (September 2009), I can configure a 1.7TB file server for the same price. Companies are responding to this explosion in storage availability by building bigger and bigger data warehouses, where every digital trail we leave is stored for later analysis. Data arrives 24/7, and real-time “on-the-fly” analysis—where answers are always available—is becoming mandatory. Here is where data stream processing comes to the rescue.

In data stream processing scenarios, data arrives at high speeds and must be analyzed in the order it is received using a limited amount of memory. The area has two main directions: a systems side and an algorithmic side. On the systems side, researchers have developed data stream processing systems that work like database systems turned upside down: Long-running queries are registered with the system and data is streamed through the system. Startups now sell systems that analyze streaming data for solutions in areas such as fraud detection, algorithmic trading, and network monitoring. They often offer at least an order of magnitude performance improvement over traditional database systems.

On the algorithmic side, there has been much research on novel one-pass algorithms. These algorithms have no need for secondary index

structures, and they do not require expensive sorting operations. They are online—an answer of the query over the current prefix of the stream is available at any time. These so-called data stream algorithms achieve these properties by trading exact query answers against approximate answers, but these approximations come with provable quality guarantees.

The following paper by Graham Cormode and Marios Hadjieleftheriou gives an overview of recent progress for the important primitive of finding frequent items in a data stream. Informally, an item is frequent in a prefix of the stream if its relative frequency exceeds a user-defined threshold. Another formulation of the problem just looks for the most frequently occurring items. The authors present an algorithmic framework that encompasses previous work and shows the results of a thorough experimental comparison of the different approaches.

This paper is especially timely since some of these algorithms are already in use (and those that are in use are not necessarily the best, according to the authors). For example, inside Google’s analysis infrastructure, in the map-reduce framework, there exist several prepackaged “aggregators” that in one pass collect statistics over a huge dataset. The “quantile” aggregate, which collects a value at each quantile of the data, uses a previously developed algorithm that is covered in the paper, and the “top” aggregate estimates the most popular values in a dataset, again using an algorithm captured by the framework in the paper.

Within AT&T (the home institution of the authors) a variety of streaming algorithms are deployed today for network monitoring, based on real-time analysis of packet header data. For example, quantile aggregates are used to track the distribution of round-trip delays between different points in the network over time. Similarly, heavy-hitter aggregates are used to find sources that send the most traffic to a given destination over time.

Although the paper surveys about 30 years of research, there is still much progress in the area. Moreover, finding frequent items is just one statistic. In practice, much more sophisticated queries such as frequent combinations of items, mining clusters, or other statistical models require data stream algorithms with quality guarantees. For example, recent work from Yahoo! for content optimization shows how to use time series models that are built online to predict the click-through rate of an article based on the stream of user clicks.

I think we have only scratched the surface both for applications and in novel algorithms, and I am looking forward to another 30 years of innovation. I recommend this paper to learn about the types of techniques that have been developed over the years and see how ideas from algorithms, statistics, and databases have come together in this problem. **□**

Johannes Gehrke (johannes@cs.cornell.edu) is an associate professor at Cornell University, Ithaca, NY.

© 2009 ACM 0001-0782/09/1000 \$10.00

Finding the Frequent Items in Streams of Data

By Graham Cormode and Marios Hadjieleftheriou

Abstract

The frequent items problem is to process a stream of items and find all those which occur more than a given fraction of the time. It is one of the most heavily studied problems in mining data streams, dating back to the 1980s. Many other applications rely directly or indirectly on finding the frequent items, and implementations are in use in large-scale industrial systems. In this paper, we describe the most important algorithms for this problem in a common framework. We place the different solutions in their historical context, and describe the connections between them, with the aim of clarifying some of the confusion that has surrounded their properties.

To further illustrate the different properties of the algorithms, we provide baseline implementations. This allows us to give empirical evidence that there is considerable variation in the performance of frequent items algorithms. The best methods can be implemented to find frequent items with high accuracy using only tens of kilobytes of memory, at rates of millions of items per second on cheap modern hardware.

1. INTRODUCTION

Many data generation processes can be modeled as *data streams*. They produce huge numbers of pieces of data, each of which is simple in isolation, but which taken together lead to a complex whole. For example, the sequence of queries posed to an Internet search engine can be thought of as a stream, as can the collection of transactions across all branches of a supermarket chain. In aggregate, this data can arrive at enormous rates, easily in the realm of hundreds of gigabytes per day or higher. While this data may be archived and indexed within a data warehouse, it is also important to process the data “as it happens,” to provide up to the minute analysis and statistics on current trends. Methods to achieve this must be quick to respond to each new piece of information, and use resources which are very small when compared to the total quantity of data.

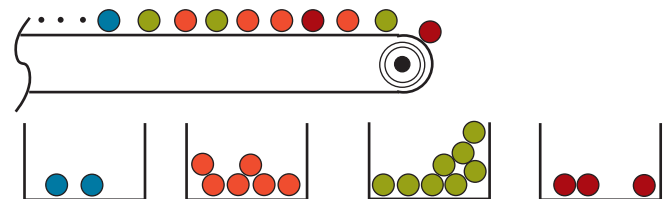
These applications and others like them have led to the formulation of the so-called “streaming model.” In this abstraction, algorithms take only a single pass over their input, and must accurately compute various functions while using resources (space and time per item) that are strictly sublinear in the size of the input—ideally, polynomial in the logarithm of the input size. The output must be produced at the end of the stream, or when queried on the prefix of the stream that has been observed so far. (Other variations ask for the output to be maintained continuously in the presence of updates, or on a “sliding

window” of only the most recent updates.) Some problems are simple in this model: for example, given a stream of transactions, finding the mean and standard deviation of the bill totals can be accomplished by retaining a few “sufficient statistics” (sum of all values, sum of squared values, etc.). Others can be shown to require a large amount of information to be stored, such as determining whether a particular search query has already appeared anywhere within a large stream of queries. Determining which problems can be solved effectively within this model remains an active research area.

The *frequent items problem* (also known as the *heavy hitters problem*) is one of the most heavily studied questions in data streams. The problem is popular due to its simplicity to state, and its intuitive interest and value. It is important both in itself, and as a subroutine within more advanced data stream computations. Informally, given a sequence of items, the problem is simply to find those items which occur most frequently. Typically, this is formalized as finding all items whose frequency exceeds a specified fraction of the total number of items. This is shown in Figure 1. Variations arise when the items are given weights, and further when these weights can also be negative.

This abstract problem captures a wide variety of settings. The items can represent packets on the Internet, and the weights are the size of the packets. Then the frequent items represent the most popular destinations, or the heaviest bandwidth users (depending on how the items are extracted from the flow identifiers). This knowledge can help in optimizing routing decisions, for in-network caching, and for planning where to add new capacity. Or, the items can represent queries

Figure 1. A stream of items defines a frequency distribution over items. In this example, with a threshold of $\phi = 20\%$ over the 19 items grouped in bins, the problem is to find all items with frequency at least 3.8—in this case, the green and red items (middle two bins).



A previous version of this paper was published in *Proceedings of the International Conference on Very Large Data Bases* (Aug. 2008).

made to an Internet search engine, and the frequent items are now the (currently) popular terms. These are not simply hypothetical examples, but genuine cases where algorithms for this problem have been applied by large corporations: AT&T¹ and Google,^{2,3} respectively. Given the size of the data (which is being generated at high speed), it is important to find algorithms which are capable of processing each new update very quickly, without blocking. It also helps if the working space of the algorithm is very small, so that the analysis can happen over many different groups in parallel, and because small structures are likely to have better cache behavior and hence further help increase the throughput.

Obtaining efficient and scalable solutions to the frequent items problem is also important since many streaming applications need to find frequent items as a subroutine of another, more complex computation. Most directly, mining frequent *itemsets* inherently builds on finding frequent *items* as a basic building block. Finding the entropy of a stream requires learning the most frequent items in order to directly compute their contribution to the entropy, and remove their contribution before approximating the entropy of the residual stream.⁸ The HSS (Hierarchical Sampling from Sketches) technique uses hashing to derive multiple substreams, the frequent elements of which are extracted to estimate the frequency moments of the stream.⁴ The frequent items problem is also related to the recently popular area of Compressed Sensing.

Other work solves generalized versions of frequent items problems by building on algorithms for the “vanilla” version of the problem. Several techniques for finding the frequent items in a “sliding window” of recent updates (instead of all updates) operate by keeping track of the frequent items in many sub-windows.^{2,13} In the “heavy hitters distinct” problem, with applications to detecting network scanning attacks, the count of an item is the number of *distinct* pairs containing that item paired with a secondary item. It is typically solved extending a frequent items algorithm with distinct counting algorithms.²⁵ Frequent items have also been applied to models of probabilistic streaming data,¹⁷ and within faster “skipping” techniques.³

Thus the problem is an important one to understand and study in order to produce efficient streaming implementations. It remains an active area, with many new research contributions produced every year on the core problem and its variations. Due to the amount of work on this problem, it is easy to miss out some important references or fail to appreciate the properties of certain algorithms. There are several cases where algorithms first published in the 1980s have been “rediscovered” two decades later; existing work is sometimes claimed to be incapable of a certain guarantee, which in truth it can provide with only minor modifications; and experimental evaluations do not always compare against the most suitable methods.

In this paper, we present the main ideas in this area, by describing some of the most significant algorithms for the core problem of finding frequent items using common notation and terminology. In doing so, we also present the historical development of these algorithms. Studying these algorithms is instructive, as they are relatively simple, but can be shown

to provide formal guarantees on the quality of their output as a function of an accuracy parameter ϵ . We also provide baseline implementations of many of these algorithms against which future algorithms can be compared, and on top of which algorithms for different problems can be built. We perform experimental evaluation of the algorithms over a variety of data sets to indicate their performance in practice. From this, we are able to identify clear distinctions among the algorithms that are not apparent from their theoretical analysis alone.

2. DEFINITIONS

We first provide formal definition of the stream and the frequencies f_i of the items within the stream as the number of times item i is seen in the stream.

Definition 1. Given a stream S of n items $t_1 \dots t_n$, the frequency of an item i is $f_i = |\{j | t_j = i\}|$. The *exact ϕ -frequent items* comprise the set $\{i | f_i > \phi n\}$.

Example. The stream $S = (a, b, a, c, c, a, b, d)$ has $f_a = 3, f_b = 2, f_c = 2, f_d = 1$. For $\phi = 0.2$, the frequent items are a, b , and c .

A streaming algorithm which finds the exact ϕ -frequent items must use a lot of space, even for large values of ϕ , based on the following information-theoretic argument. Given an algorithm that claims to solve this problem for $\phi = 50\%$, we could insert a set S of N items, where every item has frequency 1. Then, we could also insert $N - 1$ copies of item i . If i is now reported as a frequent item (occurring more than 50% of the time) then $i \in S$, else $i \notin S$. Consequently, since correctly storing a set of size N requires $\Omega(N)$ space, $\Omega(N)$ space is also required to solve the frequent items problem. That is, any algorithm which promises to solve the exact problem on a stream of length n must (in the worst case) store an amount of information that is proportional to the length of the stream, which is impractical for the large stream sizes we consider.

Because of this fundamental difficulty in solving the exact problem, an approximate version is defined based on a tolerance for error, which is parametrized by ϵ .

Definition 2. Given a stream S of n items, the ϵ -approximate frequent items problem is to return a set of items F so that for all items $i \in F, f_i > (\phi - \epsilon)n$, and there is no $i \notin F$ such that $f_i > \phi n$.

Since the exact ($\epsilon = 0$) frequent items problem is hard in general, we use “frequent items” or “the frequent items problem” to refer to the ϵ -approximate frequent items problem. A closely related problem is to estimate the frequency of items on demand.

Definition 3. Given a stream S of n items defining frequencies f_i as above, the frequency estimation problem is to process a stream so that, given any i , an \hat{f}_i is returned satisfying $f_i \leq \hat{f}_i \leq f_i + \epsilon n$.

A solution to the frequency estimation problem allows the frequent items problem to be solved (slowly): one can estimate the frequency of every possible item i , and report those i 's whose frequency is estimated above $(\phi - \epsilon)n$. Exhaustively enumerating all items can be very time consuming (or infeasible,

e.g., when the items can be arbitrary strings). However, all the algorithms we study here solve both the approximate frequent items problem and the frequency estimation problem at the same time. Most solutions are deterministic, but we also discuss randomized solutions, which allow a small user-specified probability of making a mistake.

3. FREQUENT ITEMS ALGORITHMS

We discuss two main classes of algorithms for finding the frequent items. Counter-based algorithms track a subset of items from the input, and monitor counts associated with these items. We also discuss sketch algorithms, which are (randomized) linear projections of the input viewed as a vector, and solve the frequency estimation problem. They therefore do not explicitly store items from the input. Furthermore, sketch algorithms can support deletion of items (corresponding to updates with a negative weight, discussed in more detail below), in contrast with counter-based schemes, at the cost of increased space usage and update time.

These are by no means the only solutions possible for this problem. Other solutions are based on various notions of randomly sampling items from the input, and of summarizing the distribution of items in order to find *quantiles*, from which the frequent items can be discovered. These solution types have attracted less interest for the frequent items problem, and are less effective based on our full experimental evaluations.¹⁰

3.1. Counter-based algorithms

Counter-based algorithms decide for each new arrival whether to store this item or not, and if so, what counts to associate with it. A common feature of these algorithms is that when given a new item, they test whether it is one of k being stored by the algorithm, and if so, increment its count. The cost of supporting this “dictionary” operation depends on the model of computation assumed. A simple solution is to use a hash table storing the current set of items, but this means that an otherwise deterministic solution becomes randomized in its time cost, since it takes *expected* $O(1)$ operations to perform this step. Other models assume that there is hardware support for these operations (such as Content Addressable Memory), or else that deterministic “dynamic dictionary algorithms” are used. We sidestep this issue in this presentation by just counting the number of “dictionary” operations in the algorithms.

Majority Algorithm: The problem of frequent items dates back at least to a problem proposed by Moore in 1980. It was published as a “problem” in the *Journal of Algorithms* in the June 1981 issue, as

[J.Alg 2, P208–209] Suppose we have a list of n numbers, representing the “votes” of n processors on the result of some computation. We wish to decide if there is a majority vote and what the vote is.

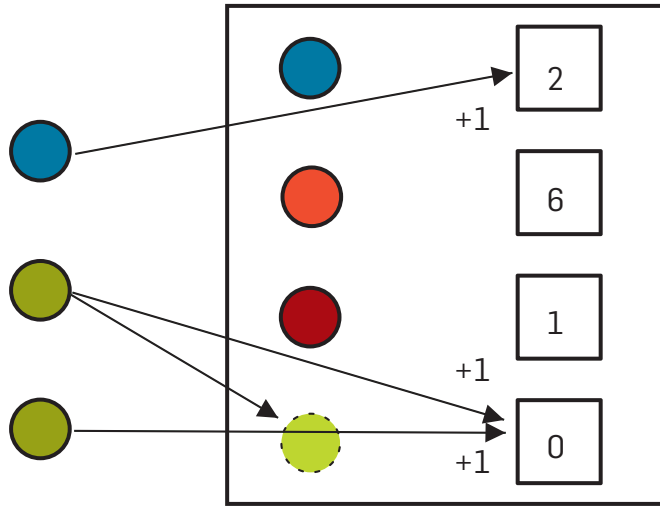
Moore, with Boyer, also invented the MAJORITY algorithm in 1980, described in a technical report from early 1981.⁶ To them, this was mostly of interest from the perspective of automatically proving the correctness of the solution (the details of this were published in 1991, along with a partial

history⁷). In the December 1982, *Journal of Algorithms*, a solution provided by Fischer and Salzburg was published.¹⁵ Their proposed algorithm, although presented differently, was essentially identical to MAJORITY, and was accompanied by an analysis of the number of comparisons needed to solve the problem. MAJORITY can be stated as follows: store the first item and a counter, initialized to 1. For each subsequent item, if it is the same as the currently stored item, increment the counter. If it differs, and the counter is zero, then store the new item and set the counter to 1; else, decrement the counter. After processing all items, the algorithm guarantees that if there is a majority vote, then it must be the item stored by the algorithm. The correctness of this algorithm is based on a pairing argument: if every nonmajority item is paired with a majority item, then there should still remain an excess of majority items. Although not posed as a streaming problem, the algorithm has a streaming flavor: it takes only one pass through the input (which can be ordered arbitrarily) to find a majority item. To verify that the stored item really is a majority, a second pass is needed to simply count the true number of occurrences of the stored item. Without this second pass, the algorithm has a partial guarantee: if there is an exact majority item, it is found at the end of the first pass, but the algorithm is unable to determine whether this is the case. Note that as the hardness results for Definition 1 show, no algorithm can correctly distinguish the cases when an item is just above or just below the threshold in a single pass without using a large amount of space.

The “Frequent” Algorithm: Twenty years later, the problem of streaming algorithms was an active research area, and a generalization of the Majority algorithm was shown to solve the problem of finding all items in a sequence whose frequency exceeds a $1/k$ fraction of the total count.^{14,18} Instead of keeping a single counter and item from the input, the FREQUENT algorithm stores $k - 1$ (item, counter) pairs. The natural generalization of the Majority algorithm is to compare each new item against the stored items T , and increment the corresponding counter if it is among them. Else, if there is some counter with a zero count, it is allocated to the new item, and the counter set to 1. If all $k - 1$ counters are allocated to distinct items, then all are decremented by 1. A grouping argument is used to argue that any item which occurs more than n/k times must be stored by the algorithm when it terminates. Figure 2 illustrates some of the operations on this data structure. Pseudocode to illustrate this algorithm is given in Algorithm 1, making use of set notation to represent the dictionary operations on the set of stored items T : items are added and removed from this set using set union and set subtraction, respectively, and we allow ranging over the members of this set (any implementation will have to choose how to support these operations). We also assume that each item j stored in T has an associated counter c_j . For items not stored in T , then c_j is implicitly defined as 0 and does not need to be explicitly stored.

In fact, this generalization was first proposed by Misra and Gries as “Algorithm 3”²² in 1982: the papers published in 2002 (which cite Fischer¹⁵ but not Misra²²) were actually rediscoveries of their algorithm. In deference to its initial discovery, this algorithm has been referred to as the “Misra–Gries” algorithm

Figure 2. Counter-based data structure: the blue (top) item is already stored, so its count is incremented when it is seen. The green (middle) item takes up an unused counter, then a second occurrence increments it.



in more recent work on streaming algorithms. In the same paper, an “Algorithm 2” correctly solves the problem but has only speculated worst-case space bounds. Some works have asserted that the FREQUENT algorithm does not solve the frequency estimation problem accurately, but this is erroneous. As observed by Bose et al.,⁵ executing this algorithm with $k = 1/\epsilon$ ensures that the count associated with each item on termination is at most ϵn below the true value.

The time cost of the algorithm is dominated by the $O(1)$ dictionary operations per update, and the cost of decrementing counts. Misra and Gries use a balanced search tree, and argue that the decrement cost is amortized $O(1)$; Karp et al. propose a hash table to implement the dictionary¹⁸; and Demaine et al. show how the cost of decrementing can be made worst-case $O(1)$ by representing the counts using offsets and maintaining multiple linked lists.¹⁴

LossyCounting: The LOSSYCOUNTING algorithm was proposed by Manku and Motwani in 2002,¹⁹ in addition to a randomized sampling-based algorithm and techniques for extending from frequent items to frequent itemsets. The

algorithm stores tuples which comprise an item, a lower bound on its count, and a “delta” (Δ) value which records the difference between the upper bound and the lower bound. When processing the i th item in the stream, if information is currently stored about the item then its lower bound is increased by one; else, a new tuple for the item is created with the lower bound set to one, and Δ set to $\lfloor i/k \rfloor$. Periodically, all tuples whose upper bound is less than $\lfloor i/k \rfloor$ are deleted. These are correct upper and lower bounds on the count of each item, so at the end of the stream, all items whose count exceeds n/k must be stored. As with FREQUENT, setting $k = 1/\epsilon$ ensures that the error in any approximate count is at most ϵn . A careful argument demonstrates that the worst-case space used by this algorithm is $O(\frac{1}{\epsilon} \log \epsilon n)$, and for certain time-invariant input distributions it is $O(\frac{1}{\epsilon})$.

Storing the delta values ensures that highly frequent items which first appear early on in the stream have very accurate approximated counts. But this adds to the storage cost. A variant of this algorithm is presented by Manku in a presentation of the paper,²⁰ which dispenses with explicitly storing the delta values, and instead has all items sharing an implicit value of $\Delta(i) = \lfloor i/k \rfloor$. The modified algorithm stores (item, count) pairs. For each item in the stream, if it is stored, then the count is incremented; otherwise, it is initialized with a count of 1. Every time $\Delta(i)$ increases, all counts are decremented by 1, and all items with zero count are removed from the data structure. The same proof suffices to show that the space bound is $O(\frac{1}{\epsilon} \log \epsilon n)$. This version of the algorithm is quite similar to Algorithm 2 presented in Misra²²; but in Manku,²⁰ a space bound is proven. The time cost is $O(1)$ dictionary operations, plus the periodic compress operations which require a linear scan of the stored items. This can be performed once every $O(\frac{1}{\epsilon} \log \epsilon n)$ updates, in which time the number of items stored has at most doubled, meaning that the amortized cost of compressing is $O(1)$. We give pseudocode for this version of the algorithm in Algorithm 2, where again T represents the set of currently monitored items, updated by set operations, and c_i are corresponding counts.

SpaceSaving: All the deterministic algorithms presented so far have a similar flavor: a set of items and counters are kept, and various simple rules are applied when a new item arrives (as illustrated in Figure 2). The SPACESAVING algorithm introduced in 2005 by Metwally et al. also fits this template.²¹ Here, k (item, count) pairs are stored, initialized by the first k distinct

Algorithm 1: FREQUENT(k)

```

n ← 0;
T ← ∅;
foreach i do
  n ← n + 1;
  if i ∈ T then
    ci ← ci + 1;
  else if |T| < k - 1 then
    T ← T ∪ {i};
    ci ← 1;
  else forall j ∈ T do
    cj ← cj - 1;
    if cj = 0 then T ← T \ {j};

```

Algorithm 2: LOSSYCOUNTING(k)

```

n ← 0; Δ ← 0; T ← ∅;
foreach i do
  n ← n + 1;
  if i ∈ T then ci ← ci + 1;
  else
    T ← T ∪ {i};
    ci ← 1 + Δ;
  if ⌊n/k⌋ ≠ Δ then
    Δ ← ⌊n/k⌋;
    forall j ∈ T do
      if cj < Δ then T ← T \ {j};

```

Algorithm 3: SPACESAVING(k)

```

n ← 0;
T ← ∅;
foreach i do
  n ← n + 1;
  if i ∈ T then ci ← ci + 1;
  else if |T| < k then
    T ← T ∪ {i};
    ci ← 1;
  else
    j ← arg minj ∈ T cj;
    cj ← cj - 1;
    T ← T ∪ {i} \ {j};

```

items and their exact counts. As usual, when the next item in the sequence corresponds to a monitored item, its count is incremented; but when the next item does not match a monitored item, the (item, count) pair with the smallest count has its item value replaced with the new item, and the count incremented. So the space required is $O(k)$ (respectively $O(\frac{1}{\epsilon})$), and a short proof demonstrates that the counts of all stored items solve the frequency estimation problem with error n/k (respectively ϵn). It also shares a useful property with LOSSYCOUNTING, that items which are stored by the algorithm early in the stream and are not removed have very accurate estimated counts. The algorithm appears in Algorithm 3. The time cost is bounded by the dictionary operation of finding if an item is stored, and of finding and maintaining the item with minimum count. Simple heap implementations can track the smallest count item in $O(\log 1/\epsilon)$ time per update. When all updates are unitary (+1), a faster approach is to borrow ideas from the Demaine et al. implementation of FREQUENT, and keep the items in groups with equal counts. By tracking a pointer to the group with smallest count, the find minimum operation takes constant time, while incrementing counts take $O(1)$ pointer operations (the “Stream-Summary” data structure described by Metwally et al.²¹).

3.2. Sketch algorithms

Here, we use the term “sketch” to denote a data structure which can be thought of as a linear projection of the input. That is, if we imagine the stream as implicitly defining a vector whose i th entry is f_i , the sketch is the product of this vector with a matrix. For the algorithm to use small space, this matrix will be implicitly defined by a small number of bits. The sketch algorithms described here use hash functions to define a (very sparse) linear projection. Both views (hashing or linear projection) can be helpful in explaining the methods, and it is usually possible to alternate between the two without confusion. Because of their linearity, it follows immediately that updates with negative values can easily be accommodated by such sketching methods. This allows us to model the removal of items (to denote the conclusion of a packet flow; or the return of a previously bought item, say) as an update with negative weight.

The two sketch algorithms outlined below solve the frequency estimation problem. They need additional data information to solve the frequent items problem, so we also describe algorithms which augment the stored sketch to find frequent items quickly. The algorithms are randomized, which means that in addition to the accuracy parameter ϵ , they also take a failure probability δ so that (over the random choices made in choosing the hash functions) the probability of failure is at most δ . Typically, δ can be chosen to be very small (e.g., 10^{-6}) while keeping the space used by the sketch low.

CountSketch: The first sketch in the sense that we use the term was the AMS or Tug-of-war sketch due to Alon et al.¹ This was used to estimate the second frequency moment, $F_2 = \sum_i f_i^2$. It was subsequently observed that the same data structure could be used to estimate the inner product of two frequency distributions, i.e., $\sum_i f_i f'_i$ for two distributions given (in a stream) by f_i and f'_i . But this means that if f_i is defined by a stream, at query time we can find the product with $f'_i = 1$ and

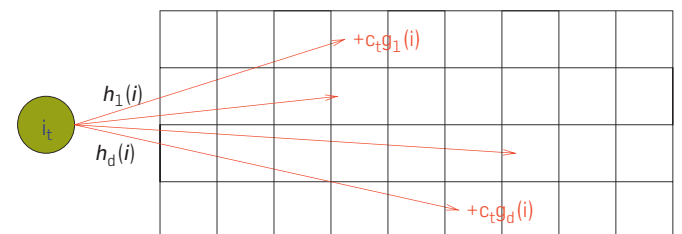
$f'_i = 0$ for all $j \neq i$. Then, the true answer to the inner product should be exactly f_i . The error guaranteed by the sketch turns out to be $\epsilon F_2^{1/2} \leq \epsilon n$ with probability of at least $1 - \delta$ for a sketch of size $O(\frac{1}{\epsilon^2} \log 1/\delta)$. The ostensibly dissimilar technique of “Random Subset Sums”¹⁶ (on close inspection) turns out to be isomorphic to this instance of the algorithm.

Maintaining the AMS data structure is slow, since it requires updating the whole sketch for every new item in the stream. The COUNTSKETCH algorithm of Charikar et al.⁹ dramatically improves the speed by showing that the same underlying technique works if each update only affects a small subset of the sketch, instead of the entire summary. The sketch consists of a two-dimensional array C with d rows of w counters each. There are two hash functions for each row, h_j which maps input items onto $[w]$, and g_j which maps input items onto $\{-1, +1\}$. Each input item i causes $g_j(i)$ to be added on to entry $C[j, h_j(i)]$ in row j , for $1 \leq j \leq d$. For any row j , the value $g_j(i)$ is an unbiased estimator for f_i . The estimate \hat{f}_i is the median of these estimates over the d rows. Setting $d = \log \frac{4}{\delta}$ and $w = O(\frac{1}{\epsilon^2})$ ensures that \hat{f}_i has error at most $\epsilon F_2^{1/2} \leq \epsilon n$ with probability of at least $1 - \delta$. This guarantee requires that the hash functions are chosen randomly from a family of “four-wise independent” hash functions.²⁴ The total space used is $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$, and the time per update is $O(\log \frac{1}{\delta})$ worst case. Figure 3 shows a schematic of the data structure under the update procedure: the new item i gets mapped to a different location in each row, where $g_j(i)$ is added on to the current counter value in that location. Pseudocode for the core of the update algorithm is shown in Algorithm 4.

CountMin Sketch: The COUNTMIN sketch algorithm of Cormode and Muthukrishnan¹² can be described in similar terms to COUNTSKETCH. An array of $d \times w$ counters is maintained, along with d hash functions h_j . Each update is mapped onto d entries in the array, each of which is incremented. Now $\hat{f}_i = \min_{1 \leq j \leq d} C[j, h_j(i)]$. The Markov inequality is used to show that the estimate for each j overestimates by less than n/w , and repeating d times reduces the probability of error exponentially. So setting $d = \log \frac{1}{\delta}$ and $w = O(\frac{1}{\epsilon})$ ensures that \hat{f}_i has error at most ϵn with probability of at least $1 - \delta$. Consequently, the space is $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ and the time per update is $O(\log \frac{1}{\delta})$. The data structure and update procedure is consequently much like that illustrated for the COUNTSKETCH in Figure 3, with $g_j(i)$ always equal to 1. The update algorithm is shown in Algorithm 5.

Finding Frequent Items Using a Hierarchy: Since sketches solve the case when item frequencies can decrease, more

Figure 3. Sketch data structure: each new item is mapped to a set of counters, which are incremented.



Algorithm 4: COUNTSKETCH(w, d)

```

 $C[1, 1] \dots C[d, w] \leftarrow 0;$ 
for  $j \leftarrow 1$  to  $d$  do
  | Initialize  $g_j, h_j;$ 
foreach  $i$  do
  |  $n \leftarrow n + 1;$ 
  | for  $j \leftarrow 1$  to  $d$  do
  | |  $C[j, h_j(i)] \leftarrow C[j, h_j(i), j] + g_j(i);$ 

```

Algorithm 5: COUNTMIN(w, d)

```

 $C[1, 1] \dots C[d, w] \leftarrow 0;$ 
for  $j \leftarrow 1$  to  $d$  do
  | Initialize  $g_j;$ 
foreach  $i$  do
  |  $n \leftarrow n + 1;$ 
  | for  $j \leftarrow 1$  to  $d$  do
  | |  $C[j, h_j(i)] \leftarrow C[j, h_j(i)] + 1;$ 

```

complex algorithms are needed to find the frequent items. Here, we assume a “strict” case, where negative updates are possible but no negative frequencies are allowed. In this strict case, an approach based on divide-and-conquer will work: additional sketches are used to determine which ranges of items are frequent.¹² If a range is frequent, then it can be split into b nonoverlapping subranges and the frequency of each subrange estimated from an appropriate sketch, until a single item is returned. The choice of b trades off update time against query time: if all items $i \in \{1 \dots U\}$, then $\lceil \log_b U \rceil$ sketches suffice, but each potential range is split into $b > 1$ subranges when answering queries. Thus, updates take $O(\log_b U \log \frac{1}{\delta})$ hashing operations, and $O(1)$ counter updates for each hash. Typically, moderate constant values of b are used (between 2 and 256, say); choosing b to be a power of two allows fast bit-shifts to be used in query and update operations instead of slower divide and mod operations. This results in COUNTMIN sketch Hierarchical and COUNTSKETCH Hierarchical algorithms.

Finding Frequent Items Using Group Testing: An alternate approach is based on “combinatorial group testing” (CGT), which randomly divides the input into buckets so that we expect at most one frequent item in each group. Within each bucket, the items are divided into subgroups so that the “weight” of each group indicates the identity of the frequent item. For example, separating the counts of the items with odd identifiers and even identifiers will indicate whether the heavy item is odd or even; repeating this for all bit positions reveals the full identity of the item. This can be seen as an extension of the COUNTMIN sketch, since the structure resembles the buckets of the sketch, with additional information on subgroups of each bucket (based on the binary representation of items falling in the bucket); further, the analysis and properties are quite close to those of a Hierarchical COUNTMIN sketch. This increases the space to $O(\frac{1}{\epsilon} \log U \log \delta)$ when the binary representation takes $\log U$ bits. Each update requires $O(\log \frac{1}{\delta})$ hashes as before, and updating $O(\log U)$ counters per hash.

4. EXPERIMENTAL COMPARISON**4.1. Setup**

We compared these algorithms under a common implementation framework to test as accurately as possible their relative performance. All algorithms were implemented using C++, and used common subroutines for similar tasks (e.g., hash tables) to increase comparability. We ran experiments on a 4 Dual Core Intel(R) Xeon(R) 2.66 GHz with 16GB of RAM running Windows 2003 Server. The code was compiled using Microsoft’s Visual C++ 2005 compiler and g++ 3.4.4 on cygwin. We did not observe significant differences between the two compilers. We report here results obtained using Visual C++ 2005. The code is available from <http://www.research.att.com/~marioh/frequent-items/>.

For every algorithm we tested a number of implementations, using different data structures to implement the basic set operations. For some algorithms the most robust implementation choice was clear; for others we present results of competing solutions. For counter-based algorithms we examine: FREQUENT using the Demaine et al. implementation technique of linked lists (F), LOSSYCOUNTING keeping separate delta values for each item (LCD), LOSSYCOUNTING without deltas (LC), SPACESAVING using a heap (SSH), and SPACESAVING using linked lists (SSL). We also examine sketch-based methods: hierarchical COUNTSKETCH (CS), hierarchical COUNTMIN sketch (CMH), and the CGT variant of COUNTMIN.

We ran experiments using 10 million packets of HTTP traffic, representing 24 hours of traffic from a backbone router in a major network. Experiments on other real and synthetic datasets are shown in an extended version of this article.¹⁰ We varied the frequency threshold ϕ , from 0.0001 to 0.01. In our experiments, we set the error guarantee $\epsilon = \phi$, since our results showed that this was sufficient to give high accuracy in practice.

We compare the efficiency of the algorithms with respect to

- Update throughput, measured in number of updates per millisecond.
- Space consumed, measured in bytes.
- Recall, measured in the total number of true heavy hitters reported over the number of true heavy hitters given by an exact algorithm.
- Precision, measured in total number of true heavy hitters reported over the total number of answers reported. Precision quantifies the number of false positives reported.
- Average relative error of the reported frequencies: We measure separately the average relative error of the frequencies of the true heavy hitters, and the average relative error of the frequencies of the false positive answers. Let the true frequency of an item be f and the measured frequency \tilde{f} . The absolute relative error is defined by $\delta f = \frac{|f - \tilde{f}|}{f}$. We average the absolute relative errors over all measured frequencies.

For all of the above, we perform 20 runs per experiment (by dividing the input data into 20 chunks and querying the algorithms once at the end of each run). We report averages on all graphs, along with the 5th and 95th percentiles as error bars.

4.2. Counter-based algorithms

Space and Time Costs: Figure 4(a) shows the update throughput of the algorithms as a function of increasing frequency threshold (ϕ). The `SPACE SAVING` and `FREQUENT` algorithms are fastest, while the two variations of `LOSSY COUNTING` are appreciably slower. On this data set, `SSL` and `SSH` are equally very fast, but on some other data sets `SSL` was significantly faster than `SSH`, showing how data structure choices can affect performance. The range of frequency thresholds (ϕ) considered did not affect update throughput (notice the log scale on the horizontal axis). The space used by these algorithms at the finest accuracy level was less than 1MB. `SSL` used 700KB for $\phi = 0.0001$, while the other algorithms all required approximately 400KB. Since the space cost varies with $1/\phi$, for $\phi = 0.01$, the cost was 100 times less, i.e., a matter of kilobytes. This range of sizes is small enough to fit within a modern second level cache, so there is no obvious effect due to crossing memory boundaries on the architectures tested on. A naive solution that maintains one counter per input item would consume many megabytes (and this grows linearly with the input size). This is at least 12 times larger than `SSH` for $\phi = 0.0001$ (which is the most robust algorithm in terms of space), and over a thousand times larger than all algorithms for $\phi = 0.01$. Clearly, the space benefit of these algorithms, even for small frequency thresholds, is substantial in practice.

Precision, Recall, and Error: All algorithms tested guarantee perfect recall (they will recover every item that is frequent). Figure 4(b) plots the *precision*. We also show the 5th and 95th percentiles in the graphs as error bars. Precision is the total number of true answers returned over the total number of answers. Precision is an indication of the number of false positives returned. Higher precision means smaller number of false positive answers. There is a clear distinction between different algorithms in this case. When using $\epsilon = \phi$, `F` results in a very large number of false positive answers, while `LC` and `LCD` result in approximately 50% false positives for small ϕ parameters, but their precision improves as skewness increases. Decreasing ϵ relative to ϕ would improve this at the cost of increasing the space used. However, `SSL` and `SSH` yield 100% accuracy in all cases (i.e., no false positives), with about the same or better space usage. Note that these implement the same algorithm and so have the same output, only differing in the underlying implementation of certain

data structures. Finally, notice that by keeping additional per-item information, `LCD` can sometimes distinguish between truly frequent and potentially frequent items better than `LC`.

Figure 4(c) plots the average relative error in the frequency estimation of the truly frequent items. The graph also plots the 5th and 95th percentiles as error bars. The relative error of `F` decreases with ϕ , while the error of `Lossy Counting` increases with ϕ . Note that `F` always returns an underestimate of the true count of any item; `LC` and `LCD` always return overestimates based on a Δ value, and so yield inflated estimates of the frequencies of infrequent items.

Conclusions: Overall, the `SPACE SAVING` algorithm appears conclusively better than other counter-based algorithms, across a wide range of data types and parameters. Of the two implementations compared, `SSH` exhibits very good performance in practice. It yields very good estimates, typically achieving 100% recall and precision, consumes very small space, and is fairly fast to update (faster than `LC` and `LCD`). Alternatively, `SSL` is the fastest algorithm with all the good characteristics of `SSH`, but consumes twice as much space on average. If space is not a critical issue, `SSL` is the implementation of choice.

4.3. Sketch algorithms

The advantage of sketches is that they support deletions, and hence are the only alternative in fully dynamic environments. This comes at the cost of increased space consumption and slower update performance. We used a hierarchy with branching factor $b = 16$ for all algorithms, after running experiments with several values and choosing the best trade-off between speed, size, and precision. The sketch depth is set to $d = 4$ throughout, and the width to $w = 2/\phi$, based on the analysis of the `COUNTMIN` sketch. This keeps the space usage of `CS` and `CMH` relatively similar, and `CGT` is larger by constant factors.

Space and Time Cost: Figure 5(a) shows the update throughput of the algorithms. Update throughput is mostly unaffected by variations in ϕ , though `CMH` does seem to become slower for larger values of ϕ . `CS` has the slowest update rate among all algorithms, due to the larger number of hashing operations needed. Still, the fastest sketch algorithm is from 5 up to 10 times slower than the fastest counter-based algorithm. Figure 5(b) plots the space consumed. The size of the sketches is fairly large compared to counter-based algorithms: of the order of several megabytes

Figure 4. Performance of counter-based algorithms on real network data (a) speed, (b) precision, and (c) average relative error.

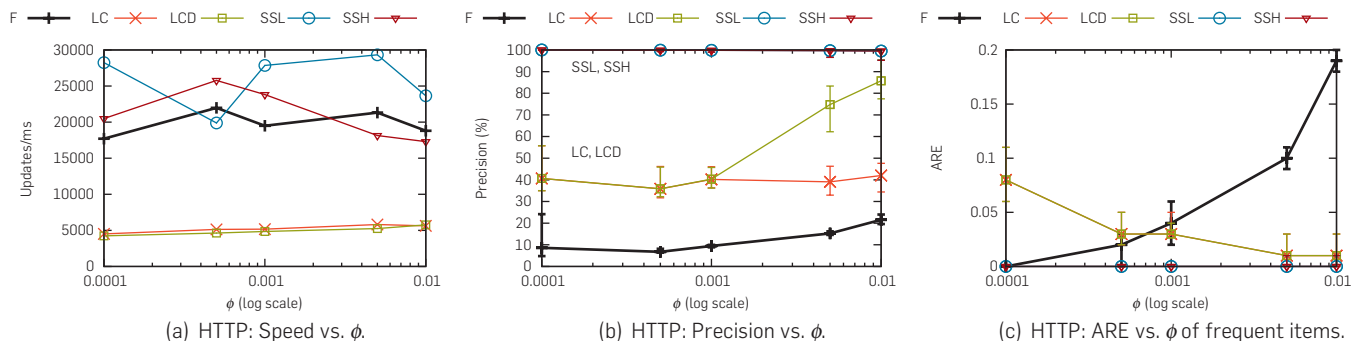
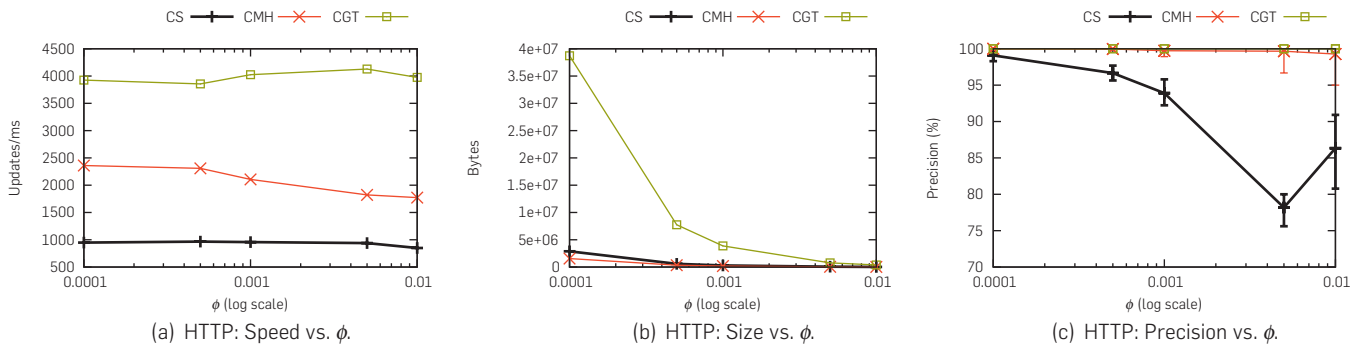


Figure 5. Performance of sketch algorithms on real network data (a) speed, (b) size, and (c) precision.

for small values of ϕ . CMH is the most space efficient sketch and still consumes space three times as large as the least space efficient counter-based algorithm.

Precision, Recall, and Error: The sketch algorithms all have near perfect recall, as is the case with the counter-based algorithms. Figure 5(c) shows that they also have good precision, with CS reporting the largest number of false positives. Nevertheless, on some other datasets we tested (not shown here), the results were reversed. We also tested the average relative error in the frequency estimation of the truly frequent items. For sufficiently skewed distributions all algorithms can estimate item frequencies very accurately, and the results from all sketches were similar since all hierarchical sketch algorithms essentially correspond to a single instance of a COUNTSKETCH or COUNTMIN sketch of equal size.

Conclusions: There is no clear winner among the sketch algorithms. CMH has small size and high update throughput, but is only accurate for highly skewed distributions. CGT consumes a lot of space but it is the fastest sketch and is very accurate in all cases, with high precision and good frequency estimation accuracy. CS has low space consumption and is very accurate in most cases, but has slow update rate and exhibits some random behavior.

5. CONCLUDING REMARKS

We have attempted to present algorithms for finding frequent items in streams, and give an experimental comparison of their behavior to serve as a baseline for comparison. For insert-only streams, the clear conclusion of our experiments is that the SPACESAVING algorithm, a relative newcomer, has surprisingly clear benefits over others. We observed that implementation choices, such as whether to use a heap or lists of items grouped by frequencies, trade-off speed, and space. For sketches to find frequent items over streams including negative updates, there is not such a clear answer, with different algorithms excelling at different aspects of the problem. We do not consider this the end of the story, and continue to experiment with other implementation choices. Our source code and experimental test scripts are available from <http://www.research.att.com/~mariah/frequent-items/> so that others can use these as baseline implementations.

We conclude by outlining some of the many variations of the problem

- In the *weighted input* case, each update comes with an associated weight (such as a number of bytes, or number of units sold). Here, sketching algorithms directly handle weighted updates because of their linearity. The SPACESAVING algorithm also extends to the weighted case, but this is not known to be the case for the other counter-based algorithms discussed.
- In the *distributed data* case, different parts of the input are seen by different parties (different routers in a network, or different stores making sales). The problem is then to find items which are frequent over the union of all the inputs. Again due to their linearity properties, sketches can easily solve such problems. It is less clear whether one can merge together multiple counter-based summaries to obtain a summary with the same accuracy and worst-case space bounds.
- Often, the item frequencies are known to follow some statistical distribution, such as the Zipfian distribution. With this assumption, it is sometimes possible to prove a smaller space requirement on the algorithm, as a function of the amount of “skewness” in the distribution.^{9,21}
- In some applications, it is important to find how many *distinct* observations there have been, leading to a *distinct heavy hitters* problem. Now the input stream S is of the form (i, j) , and f_i is defined as $|\{j | (i, j) \in S\}|$. Multiple occurrences of (i, j) only count once towards f_i . Techniques for “distinct frequent items” rely on combining frequent items algorithms with “count distinct” algorithms.²⁵
- While processing a long stream, it may be desirable to weight more recent items more heavily than older ones. Various models of *time decay* have been proposed to achieve this. In a sliding window, only the most recent items are considered to define the frequent items.² More generally time decay can be formalized via a function which assigns a weight to each item in the stream as a function of its (current) age, and the frequency of an item is the sum of its decayed weights.

Each of these problems has also led to considerable effort from the research community to propose and analyze algorithms. This research is ongoing, cementing the position of the frequent items problem as one of the most enduring and intriguing in the realm of algorithms for data streams. **□**

References

1. Alon, N., Matias, Y., Szegedy, M. The space complexity of approximating the frequency moments. In *ACM Symposium on Theory of Computing*, (1996), 20–29. Journal version in *J. Comp. Syst. Sci.* 58 (1999), 137–147.
2. Arasu, A., Manku, G.S. Approximate counts and quantiles over sliding windows. In *ACM Principles of Database Systems* (2004).
3. Bhattacharya, S., Madeira, A., Muthukrishnan, S., Ye, T. How to scalably skip past streams. In *Scalable Stream Processing Systems (SSPS) Workshop with ICDE 2007* (2007).
4. Bhuvanagiri, L., Ganguly, S., Kesh, D., Saha, C. Simpler algorithm for estimating frequency moments of data streams. In *ACM-SIAM Symposium on Discrete Algorithms* (2006).
5. Bose, P., Kranakis, E., Morin, P., Tang, Y. Bounds for frequency estimation of packet streams. In *SIROCCO* (2003).
6. Boyer, R.S., Moore, J.S. A fast majority vote algorithm. Technical Report ICSCA-CMP-32, Institute for Computer Science, University of Texas (Feb. 1981).
7. Boyer, R.S., Moore, J.S. MJRTY—a fast majority vote algorithm. In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, Automated Reasoning Series. Kluwer Academic Publishers, 1991, 105–117.
8. Chakrabarti, A., Cormode, G., McGregor, A. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms* (2007).
9. Charikar, M., Chen, K., Farach-Colton, M. Finding frequent items in data streams. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)* (2002).
10. Cormode, G., Hadjieleftheriou, M. Finding frequent items in data streams. In *International Conference on Very Large Data Bases* (2008).
11. Cormode, G., Korn, F., Muthukrishnan, S., Johnson, T., Spatscheck, O., Srivastava, D. Holistic UDAFs at streaming speeds. In *ACM SIGMOD International Conference on Management of Data* (2004), 35–46.
12. Cormode, G., Muthukrishnan, S. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75.
13. Datar, M., Gionis, A., Indyk, P., Motwani, R. Maintaining stream statistics over sliding windows. In *ACM-SIAM Symposium on Discrete Algorithms* (2002).
14. Demaine, E., López-Ortiz, A., Munro, J.I. Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms (ESA)* (2002).
15. Fischer, M., Salzburg, S. Finding a majority among n votes: Solution to problem 81–5. *J. Algorithms* 3, 4 (1982), 376–379.
16. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M. How to summarize the universe: Dynamic maintenance of quantiles. In *International Conference on Very Large Data Bases* (2002), 454–465.
17. Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E. Estimating statistical aggregates on probabilistic data streams. In *ACM Principles of Database Systems* (2007).
18. Karp, R., Papadimitriou, C., Shenker, S. A simple algorithm for finding frequent elements in sets and bags. *ACM Trans. Database Syst.* 28 (2003), 51–55.
19. Manku, G., Motwani, R. Approximate frequency counts over data streams. In *International Conference on Very Large Data Bases* (2002).
20. Manku, G.S. Frequency counts over data streams. <http://www.cse.ust.hk/vldb2002/VLDB2002-proceedings/slides/S10P03slides.pdf> (2002).
21. Metwally, A., Agrawal, D., Abbadi, A.E. Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory* (2005).
22. Misra, J., Gries, D. Finding repeated elements. *Sci. Comput. Programming* 2 (1982), 143–152.
23. Pike, D., Dorward, S., Griesemer, R., Quinlan, S. Interpreting the data: Parallel analysis with sawzall. *Dyn. Grids Worldwide Comput.* 13, 4 (2005), 277–298.
24. Thorup, M., Zhang, Y. Tabulation-based 4-universal hashing with applications to second moment estimation. In *ACM-SIAM Symposium on Discrete Algorithms* (2004).
25. Venkataraman, S., Song, D.X., Gibbons, P.B., Blum, A. New streaming algorithms for fast detection of superspreaders. In *Network and Distributed System Security Symposium NDSS* (2005).

Graham Cormode and Marios Hadjieleftheriou
([graham,marioh]@research.att.com),
AT&T Labs—Research, Florham Park, NJ.

© 2009 ACM 0001-0782/09/1000 \$10.00

Take Advantage of ACM's Lifetime Membership Plan!

- ◆ **ACM Professional Members** can enjoy the convenience of making a single payment for their entire tenure as an ACM Member, and also be protected from future price increases by taking advantage of **ACM's Lifetime Membership** option.
- ◆ **ACM Lifetime Membership** dues may be tax deductible under certain circumstances, so becoming a Lifetime Member can have additional advantages if you act before the end of 2009. (Please consult with your tax advisor.)
- ◆ Lifetime Members receive a certificate of recognition suitable for framing, and enjoy all of the benefits of **ACM Professional Membership**.

Learn more and apply at:

<http://www.acm.org/life>



Association for
Computing Machinery

Advancing Computing as a Science & Profession



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

Open Positions at INRIA For Faculty and Research Scientists

INRIA is a French public research institute in Computer Science and Applied Mathematics. It is an outstanding and highly visible scientific organization, a major player in European research and development programs.

INRIA has eight research centers in Paris, Bordeaux, Grenoble, Lille, Nancy, Nice - Sophia Antipolis, Rennes and Saclay. These centers host more than 170 groups in partnership with universities and other research organizations. INRIA focuses the activity of over 1200 researchers and faculty members, 1200 PhD students and 500 post-docs and engineers, on fundamental research at the best international level, as well as on development and transfer activities in the following areas:

- ▶ Modeling, simulation and optimization of complex dynamic systems
- ▶ Formal methods in programming secure and reliable computing systems
- ▶ Networks and ubiquitous information, computation and communication systems
- ▶ Vision and human-computer interaction modalities, virtual worlds and robotics
- ▶ Computational Engineering, Computational Sciences and Computational Medicine
- ▶ In 2010, INRIA will be opening several positions in its 8 research centers across France:
 - ▶ Junior and senior level positions,
 - ▶ Tenured and tenure-track positions,
 - ▶ Research and joint faculty positions with universities

These positions cover all the above research areas.

INRIA centers provide outstanding scientific environments and excellent working conditions. The institute welcomes applications from all nationalities. It offers competitive salaries and social benefit programs. French schooling, medical coverage and social programs are highly regarded. Visa and working permits for the applicant and the spouse will be provided.

Calendar and detailed application information at:

<http://www.inria.fr/travailler/index.en.html>

More about our research groups:

<http://www.inria.fr/recherche/equipements/index.en.html>

More about INRIA: www.inria.fr

Japan Advanced Institute of Science and Technology (JAIST) Assistant Professor

Japan Advanced Institute of Science and Technology invites applicants for an Assistant Professor position of the field of knowledge media in SCHOOL OF KNOWLEDGE SCIENCE. The appointee is expected to start her/his academic and educational activities in JAIST at an earlier time after **1st April, 2010**.

Candidates have to be highly competent in conducting research and education in the areas of human science, i.e., anthropometry, ergonomics, human factors, brain science and/or skill science.

1. Mission:

A primary mission of this position is to promote international research and education in the field of knowledge science based on the human science.

2. Qualification:

Applicants have to hold a Ph.D. degree, and be qualified for highly scientific activities by participating in domestic and international research initiatives. The search committee may ask candidates to present research activities together with a research plan in Japanese and/or English.

3. Contract: 5 years (extendable)

4. Selection:

The search committee shall evaluate the candidates' expertise, research activities, and teaching skills. The evaluation result of the each applicant will not be released after the selection. The evaluation procedure is strictly impartial, unbiased, and fair. An Equal Opportunity Employer, JAIST values diversity and strongly encourages applications from foreigners and women.

5. Applicants must submit some applications:

Detailed can be found at:

http://www.jaist.ac.jp/jimu/syomu/koubo/knowledge_media-e.htm

6. Deadline: Applications must be received no later than November 30, 2009.

For more information, please contact:

<http://www.jaist.ac.jp/index-e.html>

KAIST Professor

KAIST, the top Science & Engineering University in Korea, invites applications for tenure-track positions at all levels in Computer Science. We welcome applications from outstanding candidates in all major fields of computer science and its interdisciplinary areas.

KAIST offers a competitive start-up research fund and joint appointment with KAIST Institutes, which will expand opportunities in interdisciplinary research and funding. KAIST also provides housing for five years. KAIST is committed to increasing the number of female and non-Korean faculty members.

Required qualification is a Ph.D. or an equivalent degree in computer science or a closely related field by the time of appointment. Strong candidates who are expected to receive the Ph.D. degrees within a year can be offered our appointment. Applicants must demonstrate strong research potential and commitment to teaching. KAIST attracts nationwide top students pursuing B.S., M.S., and Ph.D. degrees. The teaching load is three hours per semester.

For more information on available positions, please visit our website:

<http://cs.kaist.ac.kr/service/employment.cs>

National Taiwan University Professor-Associate Professor-Assistant Professor

The Department of Computer Science and Information Engineering, the Graduate Institute of Networking and Multimedia, and the Graduate Institute of Biomedical Electronics and Bioinformatics, all of National Taiwan University, have faculty openings at all ranks beginning in February 2010. Highly qualified candidates in all areas of computer science/engineering and bioinformatics are invited to apply. A Ph.D. or its equivalent is required. Applicants are expected to conduct outstanding research and be committed to teaching. Candidates should send a curriculum vitae, three letters of reference, and supporting materials before October 15, 2009, to Prof Kun-Mao Chao, Department of Computer Science and Information Engineering, National Taiwan University, No 1, Sec 4, Roosevelt Rd., Taipei 106, Taiwan.

NEC Laboratories America, Inc. Research Staff Members - Data Management

NEC Laboratories America, Inc., (www.nec-labs.com) is a vibrant industrial research center renowned for technical excellence and high-impact innovations that conducts research and development in support of global businesses by building upon NEC's 100-year history of innovation. Our research programs cover a wide range of technology areas and maintain a balanced mix of fundamental and applied research as they focus on innovations which are ripe for technical breakthrough. Our progressive environment provides exposure to industry-leading technologies and nurtures close collaborations with leading research universities and institutions. Our collaborative atmosphere, commitment to developing talent, and extremely competitive benefits ensure that we attract the sharpest minds in their respective fields. NEC Labs is headquartered in Princeton, NJ and has a second location in Cupertino, CA.

We are seeking researchers to join our Data Management group in Cupertino, CA. The current research focus of the group is to create cutting edge technologies for Data Management in the Cloud. Candidates must have a Ph.D. in Computer Science (or related fields) with solid data management background and strong publication record in related areas, must be proactive with a "can-do" attitude, and able to conduct research independently. Experience in Cloud Computing, SaaS, Service Oriented Computing areas is a major plus.

The requirements for one position are:

- ▶ Deep understanding of data management systems and database internals
- ▶ Strong hands-on system building and prototyping skills
- ▶ Experience in distributed data management
- ▶ Good knowledge of emerging data models and data processing techniques (e.g.,

- ▶ Key/Value Stores, Column-Oriented Databases, MapReduce, etc.)
- ▶ Knowledge of middleware technologies

For consideration, please forward resume and research statement to recruit@nec-labs.com and reference "DM-R1" in the subject line.

For another position the researcher will create new models to capture, analyze, and predict the state of the data management systems deployed in cloud environment and combine the insights provided by those models with the database internals to deliver leading-edge data management technologies for unparalleled efficiency gains. The requirements are:

- ▶ Demonstrated knowledge of statistical and probabilistic models in large scale data and system analysis
- ▶ Strong experience in data mining and data analytics
- ▶ Good hands-on system building and prototyping skills
- ▶ Experience in data warehousing

For consideration, please forward resume and research statement to recruit@nec-labs.com and reference "DM-R2" in the subject line.

EOE/AA/MFDV

Texas State University-San Marcos
Department of Computer Science

Applications are invited for a tenure-track position at the rank of Assistant, Associate or Professor. Consult the department recruiting page at

<http://www.cs.txstate.edu/recruitment/> for job duties, required and preferred qualifications, application procedures, and information about the university and the department.

Texas State University-San Marcos is an equal opportunity educational institution and as such does not discriminate on grounds of race, religion, sex, national origin, age, physical or mental disabilities, or status as a disabled or Vietnam era veteran. Texas State is committed to increasing the number of women and minorities in faculty and senior administrative positions. Texas State University-San Marcos is a member of the Texas State University System.

Toyota Technological Institute at Chicago (TTI-C)

Computer Science at TTI-Chicago
Faculty Positions at All Levels

Toyota Technological Institute at Chicago (TTI-C) is a philanthropically endowed degree-granting institute for computer science located on the University of Chicago campus. The Institute is expected to soon reach a steady-state of 12 traditional faculty (tenure and tenure track), and 12 limited term faculty. Applications are being accepted in all areas, but we are particularly interested in:

- ▶ Theoretical computer science
- ▶ Speech processing
- ▶ Machine learning
- ▶ Computational linguistics
- ▶ Computer vision
- ▶ Scientific computing
- ▶ Programming languages

Positions are available at all ranks, and we have a large number of limited term positions currently available.

For all positions we require a Ph.D. Degree or Ph.D. candidacy, with the degree conferred prior to date of hire. Submit your application electronically at:

<http://ttic.uchicago.edu/facapp/>


Toyota Technological Institute at Chicago is an Equal Opportunity Employer

University of Chicago
Professor, Associate Professor, Assistant Professor, and Instructor

The Department of Computer Science at the University of Chicago invites applications from exceptionally qualified candidates in all areas of Computer Science for faculty positions at the ranks of Professor, Associate Professor, Assistant Professor, and Instructor. The University of Chicago has the highest standards for scholarship and faculty quality, and encourages collaboration across disciplines.


The Chicago metropolitan area provides a diverse and exciting environment. The local economy is vigorous, with international stature in banking, trade, commerce, manufacturing, and transportation, while the cultural scene includes diverse cultures, vibrant theater, world-renowned symphony, opera, jazz, and blues. The University is located in Hyde Park, a pleasant Chicago neighborhood on the Lake Michigan shore.

All applicants must apply through the University's Academic Jobs website, <http://academiccareers>.



SWARTHMORE COLLEGE invites applications for a tenure-track appointment as **Assistant or Associate Professor of Engineering in the area of Computer Engineering** to begin in September 2010. A doctorate in Computer or Electrical Engineering, or a related field, is required, along with strong interests in undergraduate teaching and in developing a laboratory research program involving undergraduates. Teaching responsibilities include robotics and digital design, and elective courses in the candidate's area of specialization, examples of which could include image processing/vision, embedded systems, graphics, and other areas related to computer hardware. Supervision of student research and senior design projects, as well as student advising, is required. Sabbatical leave with support is available every fourth year.

Swarthmore College is an undergraduate liberal arts institution with 1500 students on a suburban arboretum campus 12 miles southwest of Philadelphia. Eight faculty in the Department of Engineering offer a rigorous ABET-accredited program for the BS in Engineering to approximately 120 students. The department has an endowed equipment budget, and there is support for faculty/student collaborative research. For program details, see <http://engin.swarthmore.edu>. Interested candidates should submit a c.v., brief statements describing teaching philosophy and research interests, and undergraduate and graduate transcripts, along with three letters of reference to: Chair, Department of Engineering, Swarthmore College, 500 College Avenue, Swarthmore, PA 19081-1390, or to Imolter1@swarthmore.edu, with the word "candidate" in the subject line, by December 1, 2009. Swarthmore College is an equal opportunity employer; women and minority candidates are strongly encouraged to apply.



Graduate School of Computer and Information Sciences Dean

Nova Southeastern University (NSU) invites applications for Dean of its Graduate School of Computer and Information Sciences. The School offers a unique mix of innovative M.S. and Ph.D. programs in computer science, information systems, information security, and educational technology.

As the chief academic and administrative officer of the Graduate School of Computer and Information Sciences (GSCIS), the Dean will be responsible for leadership of the school's academic and administrative affairs. The Dean will provide innovative vision and leadership in order to maintain and advance the stature of the GSCIS. The Dean will foster and enhance the multidisciplinary structure of the GSCIS that includes Computer Science, Information Systems, and Information Technology in Education disciplines. The Dean will ensure that quality educational services are provided to students.

Qualifications include a doctoral degree in computer science, information systems, or related field. Candidates should have the ability to work with faculty in their continued pursuit of academic excellence and a shared vision towards preeminence in research and scholarship. Candidates should have experience related to graduate education, a demonstrated record of developing and facilitating research, and senior administrative experience. Candidates should have a sophisticated knowledge of the use of technology in the delivery of education and distance learning and/or hybrid curricula.

Located on a beautiful 330-acre campus in Fort Lauderdale, Florida, NSU has more than 28,000 students and is the sixth largest independent, not for-profit university in the United States. NSU awards associate's, bachelor's, master's, educational specialist, doctoral, and first-professional degrees in more than 100 disciplines. It has a college of arts and sciences and schools of medicine, dentistry, pharmacy, allied health and nursing, optometry, law, computer and information sciences, psychology, education, business, oceanography, and humanities and social sciences.

Applications should be submitted online at www.nsujobs.com for position #997648

Visit our websites: www.nova.edu & <http://scis.nova.edu>

Nova Southeastern University is an Equal Opportunity/Affirmative Action Employer.

uchicago.edu/applicants/Central?quickFind=50533. A cover letter, curriculum vitae including a list of publications, a statement describing past and current research accomplishments and outlining future research plans, a description of teaching experience, and a list of references must be uploaded to be considered as an applicant. Candidates may also post a representative set of publications, to this website. The reference letters can be sent by mail or e-mail to:

Chair, Department of Computer Science
The University of Chicago
1100 E. 58th Street, Ryerson Hall
Chicago, IL. 60637-1581

Or to:

recommend-50533@mailman.cs.uchicago.edu (attachments can be in pdf, postscript or Microsoft Word).

Please note that at least three reference letters need to be mailed or e-mailed to the above addresses and one of them must address the candidate's teaching ability. Applicants must have completed all requirements for the PhD except the dissertation at time of application, and must have completed all requirements for the PhD at time of appointment. The PhD should be in Computer Science or a related field such as Mathematics or Statistics. To ensure full consideration of your application all materials [and letters] must be received by November 15. Screening will continue until all available positions are filled. The University of Chicago is an Affirmative Action/Equal Opportunity Employer.

University Of Detroit Mercy Computer Science / Software Engineering

University of Detroit Mercy is seeking a faculty member for a tenure track position in Computer Science or Software Engineering at the Assistant Professor level. A Ph.D. is required. Deadline Sep 30, 2009. More information is available at <http://eng-sci.udmercy.edu/opportunities/index.htm>

University of Geneva – CS Dept. Associate/Assistant Professor

The CS Dept. University of Geneva announces the opening of one Associate/Assistant Professor Position

The candidate must have a research and teaching record in one or more of the following areas: management and modelling of trust, security and privacy when handling multimedia data, knowledge management and information search, massively distributed information retrieval.

Documents to apply: The cv and the list of the publications, at the top of which the candidate indicates 5 or 6 publications which are the most representative of his/her work (two copies of these publications must be included). A certified copy of the highest diploma obtained by the candidate, or a certificate of the institution that delivered it, must be joined to the application. The candidate must provide 3 letters of reference. The application with the required documents must be sent before the 30th. of September 2009 by email as one file only to Lisette.Marques@unige.ch to whom any enquiry regarding this position should be addressed.

University of Illinois at Chicago Assistant Professor/Associate Professor

University of Illinois at Chicago, Department of Mathematics, Statistics, and Computer Science

The Department has active research programs in a broad spectrum of centrally important areas of pure mathematics, computational and applied mathematics, combinatorics, mathematical computer science and scientific computing, probability and statistics, and mathematics education. See <http://www.math.uic.edu> for more information.

Applications are invited for tenure track Assistant Professor or tenured Associate Professor positions, effective August 16, 2010. Preference will be given to applicants in statistics and related areas, but outstanding applicants in all specialties will be considered. Final authorization of the position is subject to the availability of state funding.

Applicants must have a Ph.D. or equivalent degree in mathematics, computer science, statistics, mathematics education or related field, an outstanding research record, and evidence of strong teaching ability. The salary is negotiable.

Send vita and at least three (3) letters of recommendation, clearly indicating the position being applied for, to: Appointments Committee; Dept. of Mathematics, Statistics, and Computer Science; University of Illinois at Chicago; 851 S. Morgan (m/c 249); Box T; Chicago, IL 60607. Applications through mathjobs.org are encouraged. No e-mail applications will be accepted. To ensure full consideration, materials must be received by November 16, 2009. However, we will continue considering candidates until all positions have been filled. Minorities, persons with disabilities, and women are particularly encouraged to apply. UIC is an AA/EOE.

University of Illinois at Chicago Research Assistant Professor

University of Illinois at Chicago, Department of Mathematics, Statistics, and Computer Science

The Department has active research programs in a broad spectrum of centrally important areas of pure mathematics, computational and applied mathematics, combinatorics, mathematical computer science and scientific computing, probability and statistics, and mathematics education. See <http://www.math.uic.edu> for more information.

Applications are invited for the following position, effective August 16, 2010.

Research Assistant Professorship. This is a non-tenure track position, normally renewable annually to a maximum of three years. This position carries a teaching responsibility of three courses per year, and the expectation that the incumbent play a significant role in the research life of the Department. The salary for AY 2009-2010 for this position is \$54,500. Applicants must have a Ph.D. or equivalent degree in mathematics, computer science, statistics, mathematics education or related field, and evidence of outstanding research potential. Preference will be given to candidates in areas related to number theory or dynamical systems.

Send vita and at least three (3) letters of recommendation, clearly indicating the position being applied for, to: Appointments Committee; Dept. of Mathematics, Statistics, and Computer Science; University of Illinois at Chicago; 851 S. Morgan (m/c 249); Box R; Chicago, IL 60607. Applications through mathjobs.org are encouraged.

No e-mail applications will be accepted. To ensure full consideration, materials must be received by December 31, 2009. However, we will continue considering candidates until all positions have been filled. Minorities, persons with disabilities, and women are particularly encouraged to apply. UIC is an AA/EOE.

University of Oregon Department of Computer and Information Science Faculty Position

The Computer and Information Science (CIS) Department at the University of Oregon seeks applicants for one or more full-time, tenure-track faculty positions beginning fall, 2010, at the rank of Assistant Professor. The University of Oregon is an AAU research university located in Eugene, two hours south of Portland, and within one hour's drive of both the Pacific Ocean and the snow-capped Cascade Mountains.

The CIS Department is part of the College of Arts and Sciences and is housed within the Lorry Lokey Science Complex. The department offers B.S., M.S. and Ph.D. degrees. More information about the department, its programs and faculty can be found at <http://www.cs.uoregon.edu>, or by contacting the search committee at faculty.search@cs.uoregon.edu.

We offer a stimulating, friendly environment for collaborative research both within the department and with other departments on campus. Faculty in the department are affiliated with the Cognitive and Decision Sciences Institute, the Computational Science Institute, and the Neuro-Informatics Center.

Computer science is a rapidly evolving academic discipline. The department accordingly seeks to hire faculty in established areas as well as emerging directions in computer science. Applicants interested in interdisciplinary research are encouraged to apply. Applicants must have a Ph.D. in computer science or closely related field, a demonstrated record of excellence in research, and a strong commitment to teaching. A successful candidate will be expected to conduct a vigorous research program and to teach at both the undergraduate and graduate levels.

Applications will be accepted electronically through the department's web site (only). Application information can be found at <http://www.cs.uoregon.edu/Employment/>. Review of applications will begin January 4, 2010 and continue until the position is filled. Please address any questions to faculty.search@cs.uoregon.edu.

The University of Oregon is an equal opportunity/affirmative action institution committed to cultural diversity and is compliant with the Americans with Disabilities Act. We are committed to creating a more inclusive and diverse institution and seek candidates with demonstrated potential to contribute positively to its diverse community.

University of Pennsylvania Faculty

The University of Pennsylvania invites applicants for tenure-track appointments in computer science to start **July 1, 2010**. Tenured appointments will also be considered.

The Department of Computer and Information Science seeks individuals with exceptional promise for, or a proven record of, research achievement who will excel in teaching undergraduate and graduate courses and take a position of international leadership in defining their field of study. While exceptional candidates in all areas of core computer science may apply, of particular interest this year are candidates in who are working on the foundations of Market and Social Systems Engineering - the formalization, analysis, optimization, and realization of systems that increasingly integrate engineering, computational, and economic systems and methods. Candidates should have a vision and interest in defining the research and educational frontiers of this rapidly growing field.

The University of Pennsylvania is an Equal Opportunity/Affirmative Action Employer.

The Penn CIS Faculty is sensitive to "two-body problems" and would be pleased to assist with opportunities in the Philadelphia region.

For more detailed information regarding this position and application link please visit:

<http://www.cis.upenn.edu/departmental/facultyRecruiting.shtml>

University of Pennsylvania Lecturer

The University of Pennsylvania invites applicants for the position of Lecturer in Computer Science to start **July 1, 2010**. Applicants should hold a graduate degree (preferably a Ph.D.) in Computer Science or Computer Engineering, and have a strong interest in teaching with practical application. Lecturer duties include undergraduate and graduate level courses within the Master of Computer and Information Technology program, (www.cis.upenn.edu/grad/mcit/). Of particular interest are applicants with expertise and/or interest in teaching computer hardware and architecture. The position is for one year and is renewable annually up to three years. Successful applicants will find Penn to be a stimulating environment conducive to professional growth in both teaching and research.

The University of Pennsylvania is an Equal Opportunity/Affirmative Action Employer.

The Penn CIS Faculty is sensitive to "two-body problems" and would be pleased to assist with opportunities in the Philadelphia region.

For more detailed information regarding this position and application link please visit:

<http://www.cis.upenn.edu/departmental/facultyRecruiting.shtml>

University of San Francisco Tenure-track Position

The Department of Computer Science at the University of San Francisco invites applications for a tenure-track position beginning in August, 2010. While special interest will be given to candidates in bioinformatics, game engineering, systems, and networking, qualified applicants from all areas of Computer Science are encouraged to apply. For full consideration, applications should be submitted by December 1, 2009.

More details, including how to apply, can be found here:

<http://www.cs.usfca.edu/job>

University of Tartu Senior Research Fellow

The successful candidate will lead an industry-driven research project in the area of agile software development environments. The position is for 4 years with a monthly salary of 2500-4000 euro. For details see: <http://tinyurl.com/pnn5qn>

The University of Washington Bothell Assistant Professor – Software Engineering

The University of Washington Bothell Computing and Software Systems Program invites applications for a tenure track position in Software Engineering to begin fall 2010. Areas of research and teaching interest include: Requirements Engineering, Quality Assurance, Testing Methodologies, Software Development Processes, Software Design Methodologies, Software Project Management, and Collaborative and Team Development.

The Bothell campus of the University of Washington was founded in 1990 as an innovative, interdisciplinary campus within the University of Washington system – one of the premier institutions of higher education in the US. Faculty members have full access to the resources of a major research university, with the culture and close relationships with students of a small liberal arts college.

For additional information, including application procedures, please see our website at <http://www.uwb.edu/CSS/>. All University faculty engage in teaching, research, and service. The University of Washington, Bothell is an affirmative action, equal opportunity employer.

US Air Force Academy Visiting Professor of Computer Science

The United States Air Force Academy Department of Computer Science is accepting applications for a Visiting Professor position for the 2010-11 academic year. Interested candidates should contact the Search Committee Chair, Dr. Barry Fagin, at barry.fagin@usafa.edu or 719-333-7377. Detailed information is available at <http://www.usafa.edu/df/dfcs/visiting/index.cfm>.

Wake Forest University Faculty Position in Computational Biophysics

Wake Forest University invites applications for a tenure track faculty position at the level of Assistant Professor with a joint appointment in the Departments of Computer Science and Physics to begin in the fall semester of 2010. Applicants should have completed a PhD in an appropriate field by the time of appointment. Wake Forest University is a highly ranked, private university with about 4500 undergraduates, 750 graduate students, and 1700 students in the professional schools of medicine, law, divinity and business. The Physics Department has a major concentration in biophysics with approximately one third of the departmental faculty working in that field.

Several computer science faculty are actively engaged in scientific computing, computational systems biology, biological modeling and bioinformatics. Interdisciplinary research is highly valued and encouraged by the departments and University. The successful candidate will have a strong research record in computational biophysics. The candidate should also have demonstrated ability to teach courses relating to topics in physics, biophysics, or computer science. The successful candidate will be expected to teach in both departments at the undergraduate and graduate levels. Excellence in research, teaching, and obtaining external funding will be expected. Applicants should send a copy of their CV, statements regarding their research interests and teaching philosophy, and the names of three references to the Computational Biophysics Search Committee, Box 7507, Wake Forest University, Winston-Salem, NC 27109-7507. Application materials can be sent electronically in the form of a single PDF document to physcsrecruit@lists.wfu.edu. Review of applications will begin November 1, 2009 and will continue through January 15, 2010. More information is available at <http://www.wfu.edu/csphy/recruiting/>. Wake Forest University is an equal opportunity/affirmative action employer

Williams College Visiting Faculty Position

The Department of Computer Science at Williams College invites applications for an anticipated one-semester, half-time visiting faculty position in the spring of 2010.

We are particularly interested in candidates who can teach an undergraduate course in artificial intelligence or a related field. Candidates should either hold or be pursuing a Ph.D. in computer science or a closely related discipline. This position might be particularly attractive to candidates who are pursuing an advanced degree and seek the opportunity to incorporate additional classroom experience in their professional preparation.

Applications in the form of a vita, a teaching statement, and three letters of reference, at least one of which speaks to the candidate's promise as a teacher, may be sent to:

Professor Thomas Murtagh, Chair
Department of Computer Science
TCL, 47 Lab Campus Drive
Williams College
Williamstown, MA 01267

Electronic mail may be sent to cssearch@cs.williams.edu. Applications should be submitted by October 15, 2009 and will be considered until the position is filled.

The Department of Computer Science consists of eight faculty members supporting a thriving undergraduate computer science major in a congenial working environment with small classes, excellent students, and state-of-the-art facilities. Williams College is a highly selective, coeducational, liberal arts college of 2100 students located in the scenic Berkshires of Western Massachusetts. Beyond meeting fully its legal obligations for non-discrimination, Williams College is committed to building a diverse and inclusive community where members from all backgrounds can live, learn, and thrive.

CALL FOR NOMINATIONS EDITOR-IN-CHIEF

ACM INTERNATIONAL CONFERENCE PROCEEDINGS SERIES

ACM is looking for an Editor-in-Chief (EIC) for its International Conference Proceedings Series (ICPS). ACM has initiated ICPS to publish proceedings of high quality conferences, symposia, and workshops that are not sponsored by ACM or its SIGs. ICPS has been in existence since 2002, providing conference organizers a means of electronically publishing proceedings that ensures high visibility and wide distribution (ICPS proceedings are made available in the ACM Digital Library).

As a result of the sharp growth of the ICPS, the ACM Publications Board has found it necessary to initiate a more formal editorial management structure for the series to continue to ensure ACM-level quality. The series will be managed similar to other ACM publications and will have an EIC who, along with the Editorial Board, will control the content of the series. The EIC will be further assisted by an Advisory Board, which will include representation from ACM SIGs.

ACM Publications Board has set up a nominating committee to assist the Board in selecting the ICPS EIC. The nominating committee members (in alphabetical order) are:

Michel Beaudouin-Lafon,
Univ. Paris-Sud

Beng Chin Ooi,
National University of Singapore

Bashar Nuseibeh,
The Open University, UK

Tamer Ozsu (Committee Chair),
University of Waterloo

Dan R. Olsen,
Brigham Young University

Moshe Vardi,
Rice University

The nominating committee would like to receive nominations for the EIC position (self nominations are welcome). Please send nominations, accompanied with a short statement (up to one page) justifying why the nominee is suitable for this position, to Tamer Ozsu (tozsu@cs.uwaterloo.ca) by October 15, 2009.



Association for
Computing Machinery

Advancing Computing as a Science & Profession

[CONTINUED FROM P. 112] Milgram's experiments were. Essentially, what e'd done was create something that a computer scientist would recognize as a decentralized algorithm.

You're referring to an experiment in which Milgram asked a group of people in the Midwest to forward a letter to a friend of his near Boston.

Right. He gave them the man's name and mailing address and some basic personal details, but the rules were they had to mail it to someone they knew on a first-name basis. And no one had a bird's-eye view of the network, but collectively people were able to pass these messages to the target very effectively.

How does your own work fit into that picture?

What I did was use the 'algorithm' almost as a scientific probe of the structure of the network. I proposed a model for social networks that allows people to succeed at this kind of search, to land as close to the target as possible. There was a mathematical definition and linking trees and theorems. But the key feature is that we should create links at many different scales of resolution and in equal proportions.

Meaning?

There are levels of resolution we can use to think about the world. Geographically, there are people who are close neighbors, people who are nearby, people who are in the same region and country, and so on. It's important to have links across all these scales. If we only link locally, we can't get messages far away very quickly, but if we only create links at long ranges, then, for example, it would be very easy to get a message to the Boston area, but there would be no local structure to go the final few steps.

Have you been able to apply those findings to more technical topics?

One area where they prove to be useful is in the design of peer-to-peer networks. If you think about what a network is trying to do, it's trying to get information between pairs of computers that need to communicate, and it's trying to do this without any global knowledge of the network.

"A lot of the way we experience information online now is in a different form. It's coming to us continuously, in bite-size pieces, through our social networks—blogs we read, things we see on Twitter, things our friends email us."

You've also been working on a different kind of Web search.

In the classical model of Web search, some centralized thing like a search engine gathers up lots of Web pages, and users come and issue queries and get answers. A lot of the way we experience information online now is in a different form. It's coming to us continuously, in bite-size pieces, through our social networks—blogs we read, things we see on Twitter, things our friends email us.

How should our search tools and algorithms evolve to fit this new model?

One thing we should look at is a sort of temporal pattern, a pattern across time rather than across the network. So measures like chatter, bursts of activity, and upward and downward trends are going to be important, because they help us organize this kind of information. You see this increasingly on sites like Google News and Twitter search, which are essentially trying to give a time signature for certain stories. ■

Leah Hoffmann is Brooklyn-based technology writer.

© 2009 ACM 0001-0782/09/1000 \$10.00

Q&A

The Networker

Jon Kleinberg talks about algorithms, information flow, and the connections between Web search and social networks.

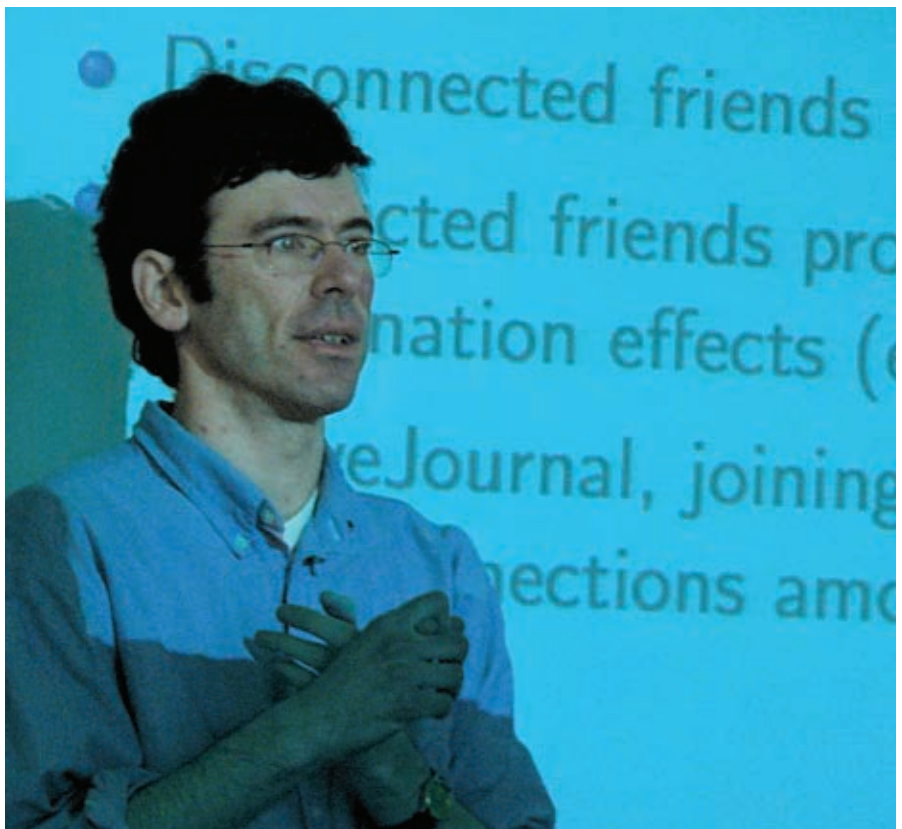
A PROFESSOR OF computer science at Cornell University, Jon Kleinberg has been studying how people maintain social connections—both on- and offline—for more than a decade. Kleinberg has received numerous honors and awards, most recently the 2008 ACM-Infosys Foundation Award in Computing Sciences.

How did you first come to think about social networks?

It started with Web search, and appreciating that in order to understand Web search—which is about people looking for information—we have to understand networks, and in particular networks that are created by people and reflect the social structure. You might look at the network within an organization to try to find the most central or important people. You can ask a similar question about the links among Web pages. And that becomes a way of finding the information that's been most endorsed by the community.

This was the motivation behind the hubs and authorities algorithm, which you developed in the mid-1990s and which uses the structure of the Internet to try to find the most authoritative Web pages.

In a sense, you can think about it as an attempt to automate something you could carry out manually. Suppose I were trying to buy a new laptop, for example. I'd find lots of people blogging, writing product reviews, and so on. And I'd see certain things be-



ing mentioned over and over, certain brands and laptops, and I might get some sense that here are the most popular ones. Those become the authorities, and the people who are best at mentioning them are the hubs. The best hubs reflect the consensus of the community, while the best authorities are that consensus, and the two reinforce each other.

Where did your work go from there?

Once I had created the algorithm, I realized there's something very basic

about using these links and endorsements to evaluate things in a network. And that led to other areas, like citation analysis and, more broadly, the field of social networks.

Some of the most famous research in that field was done by Stanley Milgram, whose small world experiments in the 1960s established that we're all linked by short paths—the proverbial “six degrees of separation.”

The thing that intrigued me was how creative [CONTINUED ON P. 111]

Emerging Software Technologies



O R L A N D O 2 0 0 9

Seeding the Clouds
The New World of Software Development
Software Engineering Tools:
The Next Generation

**Disney's Contemporary
Resort, Orlando, FL**

October 25-29 2009

Keynotes & Invited Speakers:

Barbara Liskov (Turing Award), Tom Malone (MIT),
Jeanette Wing (CMU, NSF), Gerard J. Holzman (NASA JPL),
Robert Johnson (Facebook), Jimmy Wales (Wikipedia)



PROGRAM CHAIR
Gary T. Leavens
University of Central Florida
papers@oopsla.org

CONFERENCE CHAIR
Shail Arora, Gradepoint Inc.
chair@oopsla.org

ONWARD! CHAIR
Bernd Brügge
Technische Universität München
chair@onward-conference.org



Association for
Computing Machinery

Advancing Computing as a Science & Profession

OOPSLA is sponsored by
ACM SIGPLAN in cooperation with SIGSOFT

www.oopsla.org



Think Parallel.....

It's not just what we make.
It's what we make possible.

Advancing Technology Curriculum
Driving Software Evolution
Fostering Tomorrow's Innovators

Learn more at: www.intel.com/thinkparallel

