

COMMUNICATIONS

CACM.ACM.ORG

OF THE

03/2014 VOL.57 NO.03

ACM

Steganography Masking Data Within Data

Big-Data Apps in the
Government Sector

The API Performance Contract

How to Build a
Bad Research Center

TaintDroid

Reading Brains

Association for
Computing Machinery





2014

ACM INTERNATIONAL CONFERENCE
ON INTERACTIVE EXPERIENCES FOR
TELEVISION AND ONLINE VIDEO

25-27 JUNE, 2014
NEWCASTLE UPON TYNE
UK

Paper Submissions by
3 February 2014

Workshop, Demo, WIP
DC, Grand Challenge
& Industrial
Submissions by
31 March 2014

Welcoming Submissions on
Content Production
Systems & Infrastructures
Devices & Interaction Techniques
Experience Design & Evaluation
Media Studies
Data Science & Recommendations
Business Models & Marketing
Innovative Concepts & Media Art

TVX2014.COM



We're more than computational theorists, database managers, UX mavens, coders and developers. We're on a mission to solve tomorrow. ACM gives us the resources, the access and the tools to invent the future. Join ACM today and receive 25% off your first year of membership.

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.

Departments

- 5 **Editor's Letter**
**Boolean Satisfiability:
Theory and Engineering**
By Moshe Y. Vardi
-
- 7 **From the President**
What If It's Us?
By Vinton G. Cerf
-
- 9 **Letters to the Editor**
**Develop Research Culture
in the Arab Middle East**
-
- 10 **BLOG@CACM**
**Capturing and Structuring
Data Mined from the Web**
Kate Matsudaira considers not only how to mine data from the Web, but what to do with it once you have it.
-
- 27 **Calendar**
-
- 107 **Careers**

Last Byte

- 109 **Puzzled**
Solutions and Sources
By Peter Winkler
-
- 112 **Q&A**
RISC and Reward
Having helped develop Reduced Instruction Set Computing and Redundant Arrays of Inexpensive Disks, David Patterson has set his sights on interdisciplinary research.
By Leah Hoffmann

News



- 12 **Reading Brains**
The first steps have been taken toward enabling a computer to perceive one's thoughts.
By Erica Klarreich
-
- 15 **World Without Wires**
Capturing electricity from ambient RF transmissions can keep low-power applications off the grid.
By Keith Kirkpatrick
-
- 18 **Playing at Health**
Developers try to tap the beneficial effects of video games.
By Neil Savage

Viewpoints

- 20 **Legally Speaking**
Mass Digitization as Fair Use
Considering the implications of the late-2013 ruling in favor of Google in the Authors Guild case.
By Pamela Samuelson
-
- 23 **Computing Ethics**
**Why Software Engineering Courses
Should Include Ethics Coverage**
Encouraging students to become comfortable exercising ethical discernment in a professional context with their peers.
*By Arvind Narayanan
and Shannon Vallor*
-
- 26 **The Profession of IT**
'Surfing Toward the Future'
A new report from Chile about improving economic competitiveness advances a novel interpretation of innovation. Timing is everything.
By Peter J. Denning
-
- 30 **Broadening Participation**
**The Impact of the United Nations
Convention on the Rights
of Persons with Disabilities**
Codifying human rights and inclusiveness in a technical context for people with disabilities.
By Richard Ladner
-
- 33 **Viewpoint**
How to Build a Bad Research Center
Sharing lessons learned from experiences creating successful multidisciplinary research centers.
By David Patterson

Practice



38

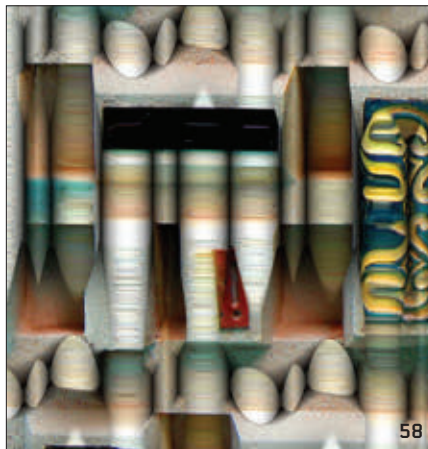
- 38 **Eventually Consistent: Not What You Were Expecting?**
Methods of quantifying consistency (or lack thereof) in eventually consistent storage systems.
By Wojciech Golab, Muntasir R. Rahman, Alvin AuYoung, Kimberly Keeton, and Xiaozhou (Steve) Li

- 45 **The API Performance Contract**
How can the expected interactions between caller and implementation be guaranteed?
By Robert F. Sproull and Jim Waldo

- 52 **Scaling Existing Lock-based Applications with Lock Elision**
Enabling existing lock-based programs to achieve performance benefits of nonblocking synchronization.
By Andi Kleen

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

Contributed Articles



58

- 58 **Making Parallel Programs Reliable with Stable Multithreading**
Stable multithreading dramatically simplifies the interleaving behaviors of parallel programs, offering new hope for making parallel programming easier.
By Junfeng Yang, Heming Cui, Jingyue Wu, Yang Tang, and Gang Hu
- 70 **Using Targeted Conferences to Recruit Women into Computer Science**
To inspire women to major in CS, take them to the Grace Hopper Celebration of Women in Computing.
By Christine Alvarado and Eugene Judson
- 78 **Big-Data Applications in the Government Sector**
In the same way businesses use big data to pursue profits, governments use it to promote the public good.
By Gang-Hoon Kim, Silvana Trimi, and Ji-Hyong Chung

Review Articles

- 86 **Trends in Steganography**
Methods for embedding secret data are more sophisticated than their ancient predecessors, but the basic principles remain unchanged.
By Elżbieta Zielińska, Wojciech Mazurczyk, and Krzysztof Szczypiorski

Research Highlights

- 98 **Technical Perspective**
Smartphone Security 'Taint' What It Used to Be
By Dan Wallach
- 99 **TaintDroid: An Information Flow Tracking System for Real-Time Privacy Monitoring on Smartphones**
By William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth



About the Cover: This month's cover story explores the art of information hiding through the ages. Steganography can be traced back to ancient Greece, where methods for embedding secret messages first took form. Today, data can be hidden in digital media and computer networks, as well as such carriers as (hint) thermochromic ink. Interactive cover design by Andriy Borys Associates.



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 SIG Services
 Donna Cappel
Director, Office of Publications
 Bernard Rous
Director, Office of Group Publishing
 Scott E. Delman

ACM COUNCIL

President
 Vinton G. Cerf
Vice-President
 Alexander L. Wolf
Secretary/Treasurer
 Vicki L. Hanson
Past President
 Alain Chesnais
Chair, SGB Board
 Erik Altman
Co-Chairs, Publications Board
 Jack Davidson and Joseph Konstan
Members-at-Large
 Eric Allman; Ricardo Baeza-Yates;
 Radia Perlman; Mary Lou Soffa;
 Eugene Spafford
SGB Council Representatives
 Brent Hailpern; Andrew Sears;
 David Wood

BOARD CHAIRS

Education Board
 Andrew McGettrick
Practitioners Board
 Stephen Bourne

REGIONAL COUNCIL CHAIRS

ACM Europe Council
 Fabrizio Gagliardi
ACM India Council
 Anand S. Deshpande, PJ Narayanan
ACM China Council
 Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs
 Jack Davidson; Joseph Konstan
Board Members
 Ronald F. Boisvert; Marie-Paule Cani;
 Nikil Dutt; Roch Guerrin; Carol Hutchins;
 Patrick Madden; Catherine McGeoch;
 M. Tamer Ozsu; Mary Lou Soffa

ACM U.S. Public Policy Office

Cameron Wilson, Director
 1828 L Street, N.W., Suite 800
 Washington, DC 20036 USA
 T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association

Chris Stephenson,
 Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

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

STAFF

DIRECTOR OF GROUP PUBLISHING

Scott E. Delman
 publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Larry Fisher

Web Editor

David Roman

Editorial Assistant

Zarina Strakhan

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Directors

Mia Angelica Balaquiot

Brian Greenberg

Production Manager

Lynn D'Addesio

Director of Media Sales

Jennifer Ruzicka

Public Relations Coordinator

Virginia Gold

Publications Assistant

Emily Williams

Columnists

David Anderson; Phillip G. Armour;
 Michael Cusumano; Peter J. Denning;
 Mark Guzdial; Thomas Haigh;
 Leah Hoffmann; Mari Sako;
 Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission

permissions@cacm.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmh@cacm.acm.org

Letters to the Editor

letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/guidelines

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY
 10121-0701
 T (212) 626-0686
 F (212) 869-0481

Director of Media Sales

Jennifer Ruzicka
 jen.ruzicka@hq.acm.org

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)

2 Penn Plaza, Suite 701
 New York, NY 10121-0701 USA
 T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF

Moshe Y. Vardi
 eic@cacm.acm.org

NEWS

Co-Chairs

Marc Najork and William Pulleyblank

Board Members

Hsiao-Wuen Hon; Mei Kobayashi;
 Michael Mitzenmacher; Rajeev Rastogi

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;
 John Leslie King;

Board Members

William Aspray; Stefan Bechtold;
 Michael L. Best; Judith Bishop;
 Stuart I. Feldman; Peter Freeman;
 Seymour Goodman; Mark Guzdial;
 Rachelle Hollander; Richard Ladner;
 Carl Landwehr; Carlos Jose Pereira de Lucena;
 Beng Chin Ooi; Loren Terveen;
 Marshall Van Alstyne; Jeannette Wing

PRACTICE

Co-Chairs

Stephen Bourne and George Neville-Neil

Board Members

Eric Allman; Charles Beeler; Bryan Cantrill;
 Terry Coatta; Stuart Feldman; Benjamin Fried;
 Pat Hanrahan; Tom Limoncelli;
 Marshall Kirk McKusick; Erik Meijer;
 Theo Schlossnagle; Jim Waldo

The Practice section of the CACM

Editorial Board also serves as the Editorial Board of *Queue*.

CONTRIBUTED ARTICLES

Co-Chairs

Al Aho and Georg Gottlob

Board Members

William Aiello; Robert Austin; Elisa Bertino;
 Gilles Brassard; Kim Bruce; Alan Bundy;
 Peter Buneman; Erran Carmel;
 Andrew Chien; Peter Druschel;
 Carlo Ghezzi; Carl Gutwin; James Larus;
 Igor Markov; Gail C. Murphy; Shree Nayar;
 Bernhard Nebel; Lionel M. Ni;
 Sriram Rajamani; Marie-Christine Rousset;
 Avi Rubin; Krishan Sabnani;
 Fred B. Schneider; Abigail Sellen;
 Ron Shamir; Yoav Shoham; Marc Snir;
 Larry Snyder; Michael Vitale;
 Wolfgang Wahlster; Hannes Werthner;
 Andy Chi-Chih Yao

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Gregory Morrisett

Board Members

Martin Abadi; Sanjeev Arora; Dan Boneh;
 Andrei Broder; Stuart K. Card; Jon Crowcroft;
 Alon Halevy; Maurice Herlihy; Norm Jouppi;
 Andrew B. Kahng; Xavier Leroy;
 Mendel Rosenblum; David Salesin;
 Guy Steele, Jr.; David Wagner;
 Margaret H. Wright

WEB

Chair

James Landay

Board Members

Gene Golovchinsky; Marti Hearst;
 Jason I. Hong; Jeff Johnson;
 Wendy E. Mackay

ACM Copyright Notice

Copyright © 2014 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@acm.org or fax (212) 869-0481.

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

Subscriptions

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

ACM Media Advertising Policy

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

Single Copies

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

COMMUNICATIONS OF THE ACM

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

POSTMASTER

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

Printed in the U.S.A.



Association for Computing Machinery





Moshe Y. Vardi

DOI:10.1145/2578043

Boolean Satisfiability: Theory and Engineering

The Boolean Satisfiability Problem (SAT, for short) asks whether a given Boolean formula, with Boolean gates such as AND and NOT, has some assignment of 0s and 1s to its input

variables such that the formula yields the value 1. SAT has been a problem of central importance in computer science since Stephen Cook proved its *NP*-completeness in 1971. To resolve the *P* vs. *NP* question, one of the most outstanding open questions in computer science and mathematics, one would have to show whether SAT is or is not in *P*, that is, whether it can be or cannot be solved in polynomial time.

At the same time, SAT is a paradigmatic constraint-satisfaction problem with numerous applications, including hardware and software design, operations research, bioinformatics, and more. Most modern-day SAT-solving algorithms are based on the work of Martin Davis and Hilary Putnam in a 1958 technical report written for the National Security Agency. While we still seem to be quite far from resolving the questions about the computational complexity of SAT, progress on the engineering side has been nothing short of spectacular. Modern SAT solvers routinely solve industrial SAT instances with millions of variables, and are being used by hardware and software designers on a daily basis.

The gap between the impressive progress on the engineering side and very slow progress on the theory side led to an unusual workshop, titled “Theoretical Foundations of Applied SAT Solving,” held last January at the Canadian Banff International Research Station for Mathematical Innovation and Discovery (see <http://www.birs.ca/events/2014/5->

day-workshops/14w5101). The workshop brought together leading SAT-complexity theorists and SAT-solving engineers, two groups that rarely meet together and barely speak the same technical language. The hope was that exposing theoreticians and engineers to state-of-the-art developments in SAT theory and engineering would narrow the chasm between these communities.

Of course, one cannot expect robust bridges between two distinct technical communities to be erected in one week, but that should not diminish the tremendous value of such bridge-building workshops. While it may take years to see if this workshop will bear concrete fruits, some fundamental research challenges already emerged.

The most fundamental challenge to emerge goes to the very heart of computational complexity theory. This theory is based typically on worst-case complexity analysis, which focuses on instances that are the most difficult to solve. Worst-case complexity analysis has proven to be quite tractable mathematically, much more, than say average-case complexity analysis. It also seems intuitive from a practical point of view; for example, a worst-case upper bound for an algorithm offers an absolute upper bound on its running time in practice. Thus, worst-case analysis is the standard approach in complexity theory. What has become clear, however, and also became painfully evident at the SAT workshop, is that worst-case analysis actually sheds very little light on the behavior of

algorithms on real-life instances. For example, theorists have demonstrated that current SAT-solving algorithms must take exponential time to solve certain families of SAT instances. Practitioners simply shrug at such bounds, while they continue to apply their solvers to very large but practically solvable SAT instances. One role of theory is to provide guidance to engineering, but worst-case (and average-case) complexity seems to offer little guidance for problems that are difficult in theory but feasible in practice. What is needed is a new computational complexity model, which will capture better the concept of “complexity in practice.”

A challenge of a different nature applied to the manner in which SAT engineers conduct their research. Current SAT engineering research is intensely heuristic. Researchers devise new heuristics and test their effectiveness on various benchmark suites. An annual SAT competition serves as a powerful motivator, as winning a track of the competition is a path to quick professional recognition. It is impossible to deny this style of research has served the community well, leading to such dramatic progress in SAT solving that researchers refer, jokingly, to “Moore’s Law for SAT.” At the same time, this heuristic approach lacks science, not only theoretical science but also empirical science. The reality is we have no understanding of why the specific sets of heuristics employed by modern SAT solvers are so effective in practice. Furthermore, we do know there are problem areas where SAT solvers are less effective, but we have no idea why. For the SAT revolution to continue unabated, we must focus also on understanding, not only on benchmarking.

So the bottom line of the workshop is that we need better theory and better engineering. Our work is cut out for us!

Moshe Y. Vardi, EDITOR-IN-CHIEF

Copyright held by Author.



CHI PLAY 2014

The ACM SIGCHI Annual Symposium on
Computer-Human Interaction in Play

Important Dates

8 May 2014, 5:00pm PT

Full papers, demos, workshops, doctoral consortium

26 June 2014, 5:00pm PT

Student competition, courses, panels and works-in-progress

Organizing Committee

Conference Chair

Lennart Nacke, University of Ontario
Institute of Technology, Canada

Technical Program Chair

Nicholas Graham, Queen's University,
Canada

Papers and Proceedings Chairs

Florian "Floyd" Mueller, RMIT University,
Australia
Regan Mandryk, University of Saskatche-
wan, Canada

Works-In-Progress, Videos and Demos Chairs

Peta Wyeth, Queensland University of
Technology, Australia
Paul Cairns, York University, UK

Student Game Design Competition Chairs

Vero vanden Abeele, University of Leuven,
Belgium
Bieke Zaman, University of Leuven,
Belgium

Industry Case Studies and Panels Chairs

Anders Drachen, Game Analytics, Denmark
Ben Medler, Electronic Arts, USA

Courses and Tutorials Chairs

Regina Bernhaupt, IRIT, University Paul
Sabatier, Toulouse III, France
Bill Kapralos, University of Ontario Institute
of Technology, Canada

Workshops Chairs

Zach Touns, New Mexico State University,
USA
Georgios Christou, European University
Cyprus, Cyprus

Doctoral Consortium Chairs

Drew Davidson, Carnegie Mellon Universi-
ty, USA
Eelke Folmer, University of Nevada, Reno,
USA

Local Arrangements Chairs

Pejman Mirza-Babaei, University of Ontario
Institute of Technology, Canada
Andrew Hogue, University of Ontario
Institute of Technology, Canada

Key Topics Include

- Game Interaction
- Novel Game Control
- Games User Research
- Gamification
- Persuasive Games
- Games for Health
- Games for Learning
- Player Experience
- Game Evaluation Methods
- Social Game Experiences
- Serious Games
- Tools for Game Creation
- Developer Experiences
- Industry Case Studies

www.chiplay.org



Vinton G. Cerf

DOI:10.1145/2577383

What If It's Us?

Near the end of January 2014 I was privileged to participate in a conference organized by the Internet Society's chapter on Interplanetary Communication

(www.ipnsig.org). The primary topic of discussion dealt with the challenges of deep space communication where the speed of light becomes an issue. Even within the solar system, we are confronted by one-way transmission delays measuring minutes to hours. Pluto is about eight hours away. In the course of the day, we learned about protocols that deal with variable delay and disruption for both deep space and even terrestrial communication. In the latter case, the issues are more about loss of connectivity, periodic or random disruption, and uncertainty brought about by store-and-forward operation in which each hop may take an arbitrarily long time.

We learned about application experiments to provide the Sami reindeer herders with communication in the far north using *data mules* (all-terrain vehicles) that carry information, drop it off in town, pick up any new information, and carry it to other towns in the area. Sounds a bit like USENET and UUCP, doesn't it?

One of the participants was an invited guest speaker and noted science fiction writer, David Brin, whose many works have been a source of inspiration and unexpected challenge to me and many readers around the world. In his talk, Brin reminded us about the famous Drake Equation, which is not about ducks but about estimates of intelligent civilizations in our galaxy (see http://en.wikipedia.org/wiki/Drake_equation).

There are a lot of parameters, most of which we do not really know how to set. The output of the equation might range from one to a very large number

depending on how the other parameter values are set. Of course, one could argue that the lower bound should be 0 (zero) based on uncertainty whether Earth's civilization(s) shows signs of intelligence. I have always thought the famous Search for Extraterrestrial Intelligence (SETI; <http://www.seti.org/>) was started because we had not found any intelligence here on Earth!

There is a famous Fermi Paradox derived from the Drake Equation, namely, if there are intelligent civilizations in our galaxy, why have we not yet detected any evidence of them? Brin had a particularly scary answer to Fermi's question. What if *we* are the ones who are supposed to *light the galaxy*? What if *our* species is destined to spread outward from Earth to populate the galaxy? I have to say, that question ran some chills up and down my spine. My first thought was "What if we don't last long enough to develop the capacity to achieve that goal?" What if we just blow the whole mission, mess up the planet, and destine our species to oblivion? Holy Moley, what if he is right about that?

I cannot speak for anyone else, but I think I would think somewhat differ-

ently about a lot of things. I would be thinking more long-term and be worried about the sustainability of our planet and the species that inhabit it. I would wonder what we should be developing to fulfill this mission. What technologies do we need to expand beyond our planet and our solar system? How should we prepare ourselves for such an ultimate goal?

There is another side to this, of course. What if, in the process of preparing ourselves to send some of our species to other systems, we encounter a similarly inclined species? How would we communicate with them? What common experience would inform our attempts to communicate? It even occurred to me this question might well be asked in contemplating communicating with our distant descendants. What Rosetta stone should we prepare? What would we want them to know about us, assuming they might struggle with this same titanic question? We might even begin by exploring our ability to communicate with other apparently sentient species here on Earth: dolphins, hominids, elephants, and others.

Can our growing skill in the use of computers help us here? Is artificial intelligence an important milestone on the path toward populating the rest of the galaxy? Are our descendants destined to be silicon analogues of human beings?

That is what I like about David Brin and science fiction in general: a lot of it makes you *really think*.

Vinton G. Cerf, ACM PRESIDENT

Copyright held by Author.

What if we are the ones who are supposed to light the galaxy?



Association for
Computing Machinery

Advancing Computing as a Science & Profession

membership application & digital library order form

Priority Code: AD13

You can join ACM in several easy ways:

Online
<http://www.acm.org/join>

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

Fax
+1-212-944-1318

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

Special rates for residents of developing countries:
<http://www.acm.org/membership/L2-3/>

Special rates for members of sister societies:
<http://www.acm.org/membership/dues.html>

Please print clearly

Name _____

Address _____

City _____ State/Province _____ Postal code/Zip _____

Country _____ E-mail address _____

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

Purposes of ACM

ACM is dedicated to:

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

I agree with the Purposes of ACM:

Signature _____

ACM Code of Ethics:

<http://www.acm.org/about/code-of-ethics>

choose one membership option:

PROFESSIONAL MEMBERSHIP:

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

STUDENT MEMBERSHIP:

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

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

payment:

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

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

Card # _____ Expiration date _____

Signature _____

RETURN COMPLETED APPLICATION TO:

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

Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

All new professional members will receive an ACM membership card.

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

Satisfaction Guaranteed!

DOI:10.1145/2578280

Develop Research Culture in the Arab Middle East

IT WAS ONLY 2002 when I first knew ACM even existed. I was preparing my master's thesis in computer engineering in Iraq. I was so amazed I thought of ACM members as movie stars. Due to many circumstances, I did not have the honor of joining as a member until 2010. Having received my bachelor's, master's, and Ph.D. in the Middle East, I have always seen a gap, a large one, between the way research is conducted there on one side and in Europe and the U.S. on the other. This feeling has been enforced whenever I get a rejection letter for a paper sent to a big computer science conference or journal. Sometimes, unfortunately, reviewers have mocked and even ridiculed instead of provided a constructive review. This happened in the early stages of my own research, and, as I learned later, to many of my colleagues as well. In most cases, when I look at Arab scientists who have published in reputable conferences and journals, I see author names of those who have either studied abroad or are working abroad, many very successful with outstanding research records. What does this say?

The Arab Middle East needs a cultural revolution in terms of research, especially in computer science. The research mentality here is quite different from other areas of the world. This is not to say it is not scientifically valid, just that research here is conducted in a different way that needs to be formalized to conform to international standards. Many researchers in the Arab world have amazing potential. Unfortunately, that potential is not being unleashed until they go abroad.

I sincerely hope ACM takes the initiative in helping spread a valid and concrete academic research culture in the Arab world. We all aim for the same thing—improving the quality of life for ourselves and for the coming generations. The lack of tools and research culture should not prevent Arab com-

puter scientists from contributing to the development of all humanity.

Mohammed M. Alani, Muscat, Oman

Still Looking for a Development Blueprint

I agree with much of Phillip G. Armour's Viewpoint "Estimation Is Not Evil" (Jan. 2014), but there were several important aspects of real-world software development environments he did not mention and which I have encountered in my own 30 years in software development. For example, I agree with his complimentary characterization of the Waterfall method as not being overly rigid or unable to handle changing requirements and encounters with unknowns. I also suspect a good part of the criticism of the Waterfall method comes from people who never actually worked in software development in the method's heyday. I also agree with Armour's praise for the Agile method's focus on generating the most value in a constrained period of time. The Agile method is great for keeping customers involved in a project, though many (in my experience) neither want nor have the time to be too closely involved; they just want the software finished and to provide the functionality they had in mind.

I also agree with Armour's description of the calculated risk-covering reserve as being part of the overall commitment. In managing software development (mostly in small companies producing custom applications for outside customers), I have found the final price and schedule are often agreed to in the sales process prior to any real-world estimation by the developers. Alternatively, even when developers do have input, I have heard upper management say, "We acknowledge your concerns but will not budget for them," even though it is a rare software project that has nothing go wrong or does not take longer than expected. I have also heard, "We acknowledge your concerns but cannot sell the project at

a price including budgeting for risks."

In the end, the software team takes the risk and is targeted if the non-risk-reserve-bearing commitment is not made. A software developer might think a workable solution would be to get the customer to buy into the risk-reserve idea, assuming, if the risk does not materialize, the customer gets the product at the lower price. In practice, however, this does not seem to occur very often. Customers know once they agree to a higher potential price, they will almost certainly pay that price. This scenario could occur for any number of reasons, but one I did not always appreciate was that once a customer agrees to the "potentially" higher amount, vendor management inevitably solidifies its own budgets based on achieving the full amount, sometimes exerting internal pressure on the development team to do so.

I understand Armour was primarily addressing the Agile method's relationship with estimation, but I was hoping he would have offered some advice that would give software developers a workable blueprint for successful software development. Even with Agile and estimations we do not seem to be there yet.

Richard H. Veith, Port Murray, NJ

Author's Response:

Veith raises an interesting point about negotiating the cost of risk with customers. In a few instances, and only where the customer-vendor relationship is on a truly honest basis, I have seen the customer engage in a useful dialogue about who should bear the risk and how much risk is appropriate. Uncertainty and risk are intrinsic to software development, and a mature process would acknowledge and optimize it. We are getting better but are not there yet.

Phillip G. Armour, Deer Park, IL

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

© 2014 ACM 0001-0782/14/03 \$15.00

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



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

DOI:10.1145/2567664

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

Capturing and Structuring Data Mined from the Web

Kate Matsudaira considers not only how to mine data from the Web, but what to do with it once you have it.



Kate Matsudaira
**Data Mining the Web
Via Crawling**

<http://bit.ly/1eJYVUH>
July 26, 2012

Central to any data-mining project is having sufficient amounts of data that can be processed to provide meaningful and statistically relevant information. But acquiring the data is only the first phase. Often collected in an unstructured form, this data must be transformed into a structured format suitable for processing.

Within the past few years there has been an increase of free Web crawler datasets (<http://bit.ly/1eDHFOO>), but for many applications it is still necessary to crawl the Web to collect information. And if the data mining pieces were not difficult enough, there are many counterintuitive challenges associated with crawling the Web to discover and collect content. The challenges become increasingly difficult when doing this on a larger scale.

In practice, capturing the content of a single Web page is quite easy. Here is a simple crawler that uses the command-line tool *wget*:

```
wget -r --html-extension --convert-links  
--mirror --progress=bar --no-verbose --no-  
parent --tries=5 -level=5 $your_url
```

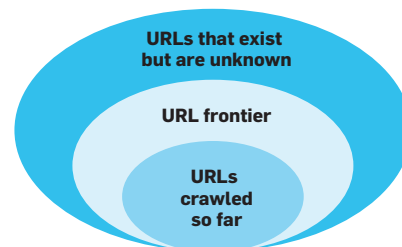
If you want to capture the content of multiple pages, there exists a wealth of open source Web crawlers, but you will find many of these solutions are overkill, often designed to crawl the Web in a very large-scale, distributed manner. Here, I focus on the challenges of collecting specific types of Web data for data mining on a smaller scale by exploring three of these challenges (creating the URL frontier, scheduling the URLs, and extracting the information) and potential solutions.

Creating the URL Frontier

A URL frontier is the collection of URLs the crawler intends to fetch and process in the future. URL frontiers generally work in one of two ways: a batch crawl or a continuous crawl. A batch crawl's frontier contains only new URLs, whereas a continuous crawl's can contain a seed set of URLs, but new ones may be added (or existing ones removed) during the crawl.

Regardless of the type of crawl, the data store used to manage the frontier

needs to be fast, because it will slow down crawling otherwise. Below is a diagram that helps illustrate the URLs in a crawl frontier.



Crawling is quite simple at its core:

1. Select a URL to crawl
2. Fetch and parse page
3. Save the important content
4. Extract URLs from page
5. Add URLs to queue
6. Repeat

Of course, the details around URL discovery are quite complicated, depending on your application (you may not want every URL on a page, for example). Adding logic around which URLs to select to crawl (selection), and which URLs to filter (often an easy restriction is to set your filters to only apply to specific domains and ignore all others) can help ensure your crawl is tightly targeted to your target content/data.

One way to create your URL frontier is to define a seed set of URLs or domains and crawl them to extract and discover new links. Post-processing the links to normalize them can help reduce duplication and is a good best practice to follow. Another example of this post-processing is filtering the links before adding them to the frontier, such as restricting links to

specific domains or using an algorithm to order the URLs in a priority that makes sense for the application and goals of the crawl. (For more, check out the Stanford paper at <http://stanford.io/1dsSy7V>.)

Crawling and Scheduling

When it comes to crawling, there are many details that allow one to do this quickly and at scale. If your crawl is less than 100 million pages and you do not need to distribute the crawling across servers, then crawling and scheduling is more manageable. The coordination of distributed crawling across several servers can get difficult quickly (and since it is much more complicated, I will have to leave those details for a future post).

When it comes to fetching and parsing a page, there are several steps involved for each URL:

1. DNS lookup
2. Fetch robots.txt
3. Fetch URL
4. Parse content
5. Store results

For the DNS lookup and robots.txt rules, one can likely cache these such that any subsequent URLs from the same IP will be much faster and will not require a trip across the network.

Tip: For DNS lookups, many of the built-in toolkits are synchronized by default, so it helps to configure this to achieve parallelism and speed.

Of course, it is important any crawler obeys the rules in robots.txt file, which may exclude all or parts of the domain. (More details on this exclusion protocol can be found at <http://bit.ly/1eRsqrn>.)

Another important aspect of crawling is choosing which URLs to crawl and in what order.

If you are doing a continuous crawl, it is important to think about when you may re-crawl a page. Certain parts of the Internet are changing all the time (such as news sites like CNN.com) and should be crawled regularly as they have new content, are high authority sites, or are important sources of data.

Remember the importance of politeness. Your crawler should not make too many simultaneous requests, as they can overwhelm underpowered servers. Best practice is to wait two seconds between requests for the same IP.

Tip: Do not sort input URLs, or you

will be waiting between calls. Randomize the order of URLs across domains.

Scheduling is not just about selecting which URLs to crawl, but also tracking the URLs crawled so far. If you have a small crawl (<100 million URLs), it probably makes sense to store these in memory; otherwise, your application performance can suffer reading from disk or across the network. It is possible to store URLs in plain text, although using a bloom filter (<http://bit.ly/1j7JGZ0>) to hash URLs makes reading and writing much faster.

Finally, as new URLs are discovered, it is important to normalize them to remove duplicates.

For example, many pages are expressed relative to the main domain, such that a link might be /about.html, but would need to be transformed into: <http://katemats.com/about.html>. Since many URLs in the wild have query parameters (such as those used for analytics or session information), it is easy to come across many different versions of a URL that reference the same Web page. These extraneous parameters must be removed or normalized so the only URL crawled is the canonical version of the page.

Using the `rel="canonical"` tag (to read more, check out Google's documentation at <http://bit.ly/K39Nlj>) can serve as a signal if the page is a duplicate, but it is not used everywhere, so it is worth implementing a fingerprinting algorithm. Using an algorithm like shingling (<http://stanford.io/Kjpnpsy>) can help detect the same or near-duplicate pages.

There are a handful of open source crawlers to consider, such as:

- ▶ Apache Nutch (<http://nutch.apache.org/>)
- ▶ Bixo (<http://bixo.101tec.com/>)
- ▶ Heritrix (<http://bit.ly/1dYRV2n>).
- ▶ Crawler4j (<http://code.google.com/p/crawler4j/>)
- ▶ Scrapy (scrapy.org)

Extracting the Information

Once the crawler is configured, the last piece of the puzzle is extracting and structuring the data from different Web pages.

The most important part of parsing Web pages is that your parser is able to deal with messy markup and will be resilient to errors. Chances are you will want some smart logic that removes the page chrome (navigation, headers/footers, search boxes, advertisements,

and so forth) so only data is returned. It is also important to consider more than ASCII text, as it is common to run across Unicode URLs and content.

If you are trying to crawl the entire Web, there is more to consider, as the Internet is full of junk such as enormous or never-ending pages, with spider traps that look like dynamic content but are actually an infinite generation of links.

Once you have the html content, it is necessary to navigate the page and DOM to find the specific parts of the page relevant or important to your cause. And then, of course, storing that information in a structured form, which may mean converting it to JSON, or transforming it to fit a schema and inserting it into the database. Thankfully, there are a lot of extractors for parsing Web pages, including:

- ▶ Scrapy (scraping and crawling framework)
- ▶ Mechanize
- ▶ BeautifulSoup (<http://bit.ly/1iVZlbl>)
- ▶ Boilerplate (<http://bit.ly/1b3ElTF>, used to remove the chrome around Web pages)

A little bit different, but good for pulling out article text from pages is Readability (<http://bit.ly/19uQ3Cp>).

There you have it! ■

Further Reading

Pant, G., Srinivasan, P., Menczer, F. *Crawling the Web*, Department of Management Sciences, School of Library and Information Science, The University of Iowa; School of Informatics, Indiana University, <http://bit.ly/1d7cCII>

Web Crawling and Indexes, Cambridge University Press, <http://stanford.io/1j7HDnX>

Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiawicz, J.D.

Tapestry: A Resilient Global-Scale Overlay for Service Development, IEEE Journal on Selected Areas in Communications, <http://bit.ly/1iWOHTx>

DeCandia, G., Hastorun, D., Madan, J., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W. *Dynamo: Amazon's Highly Available Key-value Store*, Amazon.com, <http://bit.ly/KjqGtYl>

Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*, MIT Laboratory for Computer Science, <http://bit.ly/1m6fgq1>

Kate Matsudaira is vice president of Decide.com, which was acquired by eBay in 2013.

© 2014 ACM 0001-0782/14/03 \$15.00

Reading Brains

The first steps have been taken toward enabling a computer to perceive one's thoughts.

MIND READING HAS traditionally been the domain of mystics and science fiction writers. Increasingly, however, it is becoming the province of serious science.

A new study from the laboratory of Marcel van Gerven of Radboud University Nijmegen in the Netherlands demonstrates it is possible to figure out what people are looking at by scanning their brains. When volunteers looked at handwritten letters, a computer model was able to produce fuzzy images of the letters they were seeing, based only on the volunteers' brain activity.

The new work—which builds on an earlier mathematical model by Bertrand Thirion of the Institute for Research in Computer Science and Control in Gif-sur-Yvette, France—establishes a simple, elegant brain-decoding algorithm, says Jack Gallant, a neuroscientist at the University of California, Berkeley. Such decoding algorithms eventually could be used to create more sophisticated brain-machine interfaces, he says, to allow neurologically impaired people to manipulate computers and machinery with their thoughts.

As technology improves, Gallant predicts, it eventually will be possible to use this type of algorithm to decode thoughts and visualizations, and per-



A patient wears a cap studded with electrodes during a demonstration of a noninvasive brain-machine interface by the Swiss Federal Institute of Technology of Lausanne in January 2013.

haps even dreams. “I believe that eventually, there will be something like a radar gun that you can just point at someone’s head to decode their mental state,” he says. “We have the mathematical framework we need, for the most part, so the only major limitation is how well we can measure brain activity.”

A Simple Model

In the new study, slated to appear in an upcoming issue of the journal *Neuroimage*, volunteers looked at handwritten copies of the letters B, R, A, I, N, and S, while a functional magnetic resonance imaging (fMRI) machine measured the responses of their primary visual cor-

tex (V1), the brain region that does the initial, low-level processing of visual information. The research team then used this fMRI data to train a Bayesian computer model to read the volunteers' minds when they were presented with new instances of the six letters.

"It's a very elegant study," says Thomas Naselaris, a neuroscientist at The Medical University of South Carolina in Charleston.

According to Bayes' Law, to reconstruct the handwritten image most likely to have produced a particular pattern of brain activity, it is necessary to know two things about each candidate image: the "forward model," the probability that the candidate image would produce that particular brain pattern; and the "prior," the probability of that particular image cropping up in a collection of handwritten letters. Whichever candidate image maximizes the product of these two probabilities is the most likely image for the person to have seen.

To create the forward model, the research team showed volunteers hundreds of different handwritten images of the six letters while measuring their brain activity, then used machine-learning techniques to model the most likely brain patterns that any new image would produce. To construct the prior, the team again set machine learning algorithms to work on 700 additional copies of each letter, to produce a model of the most likely arrangements of pixels when people write a letter by hand. Both models used simple linear Gaussian probability distributions, making brain decoding into a straightforward calculation, van Gerven says.

"We've shown that simple mathematical models can get good reconstructions," he says.

The research team also experimented with limiting the model's prior knowledge of the world of handwritten letters. If the model's prior information consisted only of images of the letters R, A, I, N, and S, for example, it could still produce decent reconstructions of the letter B, though not as good as when the prior included images of all six letters. The results, van Gerven says, demonstrate the decoding algorithm's ability to generalize—to reconstruct types of letters it has never "seen" before.

The human brain is, of course, the master of this kind of generalization, and this ability goes much farther than simple reconstruction of unfamiliar images. "The visual system can do something no robot can do," Naselaris says. "It can walk into a room filled with things it has never seen before and identify each thing and understand the meaning of it all."

While van Gerven's paper deals only with reconstructing the image a person has seen, other researchers have taken first steps toward deciphering the meanings a brain attaches to visual stimuli. For example, Gallant's group (including Naselaris, formerly a postdoc at Berkeley) has combined data from V1 and higher-order visual processing regions to reconstruct both the image a person has seen and the brain's interpretation of the objects in the image. More recently, in partially unpublished work, the team has done the same thing for movies, instead of still images.

"We are starting to build a repertoire of models that can predict what is going on in higher levels of the vision hierarchy, where object recognition is taking place," Naselaris says.

Other researchers are working on reading a brain's thoughts as it responds to verbal stimuli. For example, in 2010, the laboratory of Tom Mitchell at Carnegie Mellon University in Pittsburgh developed a model that could reconstruct which noun a person was reading. Van Gerven's lab is currently working on decoding the concepts volunteers consider as they listen to a children's story while inside an fMRI scanner.

Probing Thoughts

Most mind-reading research to date has focused on reconstructing the external stimuli creating a particular pattern of brain activity. A natural question is whether brain-decoding algorithms can make the leap to reconstructing a person's private thoughts and visualizations, in the absence of any specific stimulus.

The answer depends on the extent to which, for example, the brain processes mental images and real images in the same way. "The hypothesis is that perception and imagery activate the same brain regions in similar

ACM Member News

'TECHNOLOGY PRAGMATIST' ADDRESSING BRAZIL'S DIGITAL DIVIDE



"I'm a technology pragmatist," says ACM Distinguished Speaker Fernando Koch,

director of Research at the Samsung Research Institute Brazil, Advanced Technologies Group in suburban Sao Paulo. "I want to create Brazilian-bred technology tailored to solving Brazilian problems in health, education, and security."

With more than 20 years of IT industry experience in product and business development and R&D, Koch has published over 50 papers and has patents deposited or pending on such diverse topics as artificial intelligence, health informatics, cognitive computing, and intelligent mobile services.

A former Research Scientist at IBM Research Brazil, Koch says his priority has always been the practical application of next-generation mobile computing solutions, with special emphasis on health, education, connected intelligent mobile services, and secure mobile systems.

"We are just scratching the surface of mobility options. We need an 'Internet of things'; complete communication and intelligent understanding and intuition among corporate and consumer devices."

His current initiatives include closing the digital divide in Brazilian education by advancing intelligent mobile solutions like Smart School, a set of technologies to automate the digital classroom. Koch wants to cultivate new ways to optimize students' performance beyond exam-taking. "We are developing technology to analyze how long students stay on a digital page and how they interact with course material to determine what's working for them," he says.

Koch and his team are also developing low-cost sensors so Brazilian healthcare workers can use mobile devices to send high-quality mammograms, eye tests, and electrocardiograms to hospitals for review and analysis.

—Laura Didio

ways,” van Gerven says. “There have been hints that this is largely the case, but we are not there yet.”

If, Naselaris says, “highly visual processes get evoked when you are just reasoning through something—planning your day, say—then it should be possible to develop sensitive probes of internal thoughts and do something very much like mind-reading just from knowing how V1 works,” he says. “But that is a big ‘if.’”

Even if mind-reading turns out not to be as simple as decoding V1, Naselaris predicts that as neuroscientists develop forward models of the brain’s higher-level processing regions, the decoding models will almost certainly provide a portal into people’s thoughts. “I don’t think there is anything that futuristic about the idea that in five to 20 years, we will be able to make pictures of what people are thinking about, or transcribe words that people are saying to themselves,” he says.

What may prove more difficult, Gallant says, is digging up a person’s distant memories or unconscious associations. “If I ask you the name of your first-grade teacher, you can probably remember it, but we do not understand how that is being stored or represented,” he says. “For the immediate future, we will only be able to decode the active stuff you’re thinking about right now.”

Dream decoding is likely to prove another major challenge, Naselaris says. “There is so much we don’t understand about sleep,” he says. “Decoding dreams is way out in the future; that’s my guess.”

Part of the problem is that with dreams, “you never have ground truth,” Gallant says. When it comes to building a model of waking thoughts or visions, it is always possible to ask the person what he or she is thinking, or to directly control the stimuli the person’s brain is receiving, but dreams have no reality check.

The main option available to researchers, therefore, is to build models for reconstructing movies, and then treat a dream as if it were a movie playing in the person’s mind. “That is not a valid model, but we use it anyway,” Gallant says. “It is not going to be very accurate, but since we have no accuracy now, having lousy accuracy is better than nothing.”

In May 2013, a team led by Yukiyasu Kamitani of ATR Computational Neuroscience Laboratories in Kyoto, Japan, published a study in which they used fMRI data to reconstruct the categories of visual objects people experienced during hypnagogic dreams, the ones that occur as a person drifts into sleep. “They are not real dreams, but it is a proof of concept that it should be possible to decode dreams,” Gallant says.

Protecting Privacy

The dystopian future Gallant pictures, in which we could read each other’s private thoughts using something like a radar gun, is not going to happen any time soon. For now, the best tool researchers have at their disposal, fMRI, is at best a blunt instrument; instead of measuring neuronal responses directly, it can only detect blood flow in the brain—which Gallant calls “the echoes of neural activity.” The resulting reconstructions are vague shadows of the original stimuli.

What is more, fMRI-based mind reading is expensive, low-resolution, and the opposite of portable. It is also easily thwarted. “If I did not want my mind read, I could prevent it,” Naselaris says. “It is easy to generate noisy signals in an MRI; you can just move your head, blink, think about other things, or go to sleep.”

These limitations also make fMRI an ineffective tool for most kinds of brain-machine interfaces. It is conceivable fMRI could eventually be used to allow doctors to read the thoughts of patients who are not able to speak, Gallant says, but most applications of brain-machine interfaces require a much more

“The hypothesis is that perception and imagery activate the same brain regions in similar ways,” van Gerven says.

portable technology than fMRI.

However, given the extraordinary pace at which technology moves, some more effective tool will replace fMRI before too long, Gallant predicts. When that happens, the brain decoding algorithms developed by Thirion, van Gerven, and others should plug right into the new technology, Gallant says. “The math is pretty much the same framework, no matter how we measure brain activity,” he says.

Despite the potential benefit to patients who need brain-machine interfaces, Gallant is concerned by the thought of a portable mind-reading technology. “It is pretty scary, but it is going to happen,” he says. “We need to come up with privacy guidelines now, before it comes online.”

Further Reading

Horikawa, T., Tamaki, M., Miyawaki, Y., Kamitani, Y. **Neural Decoding of Visual Imagery During Sleep**, *Science* Vol. 340 No., 6132, 639-642, 3 May 2013.

Kay, K. N., Naselaris, T., Prenger, R. J., Gallant, J. L. **Identifying Natural Images from Human Brain Activity**, *Nature* 452, 352-355, March 20, 2008. <http://www.ncbi.nlm.nih.gov/pubmed/18322462>

Mitchell, T., Shinkareva, S., Carlson, A., Chang, K.-M., Malave, V., Mason, R., Just, M. **Predicting Human Brain Activity Associated with the Meanings of Nouns**, *Science* Vol. 320 No. 5880, 1191-1195, 30 May 2008. <http://www.sciencemag.org/content/320/5880/1191>

Nishimoto, S., Vu, A. T., Naselaris, T., Benjamini, Y., Yu, B., Gallant, J. L. **Reconstructing Visual Experiences from Brain Activity Evoked by Natural Movies**, *Current Biology* Vol. 21 Issue 19, 1641-1646, 11 October 2011. <http://www.sciencedirect.com/science/article/pii/S0960982211009377>

Schoenmakers, S., Barth, M., Heskens, T., van Gerven, M. **Linear Reconstruction of Perceived Images from Human Brain Activity**, *Neuroimage* 83, 951-61, December 2013. <http://www.ncbi.nlm.nih.gov/pubmed/23886984>

Thirion, B., Duchesnay, E., Hubbard, E., Dubois, J., Poline, J. B., LeBihan, D., Dehaene, S. **Inverse Retinotopy: Inferring the Visual Content of Images from Brain Activation Patterns**, *Neuroimage* 33, 1104-1116, December 2006. <http://www.ncbi.nlm.nih.gov/pubmed/17029988>

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

World Without Wires

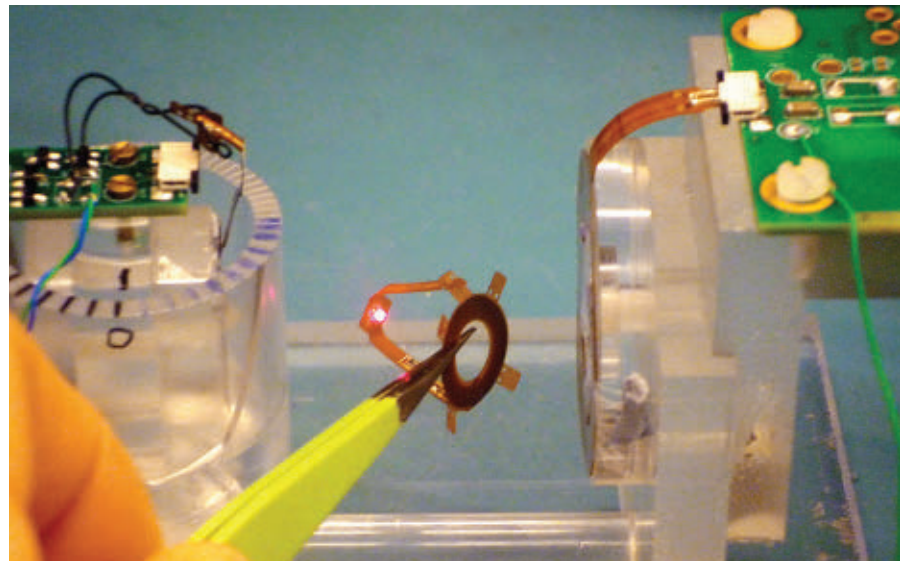
Capturing electricity from ambient RF transmissions can keep low-power applications off the grid.

MOST WIRELESS COMMUNICATION technology in use today remains firmly tethered to a very 20th-century constraint—the need for a robust, reliable and cost-efficient source of power. Indeed, common mobile devices such as cellphones, tablets, and even electric vehicles are still somewhat limited in terms of the amount of time they can operate without needing to be recharged.

As a result, efforts are under way to develop wireless technology that can either generate electricity seemingly from thin air, or utilize inductive transmission to provide a “last mile” of electricity transmission without the use of wires or advanced battery technology.

Battery technology is advancing, providing longer runtimes. However, in some cases it is impractical to use batteries to power devices, due to the vast number that may need to be replaced. For example, in large commercial or industrial buildings, networked sensors, controllers, and switches are being installed to better monitor and control lighting, HVAC, and other critical building systems. While traditional systems simply ran two sets of wires to each networked device (one to carry power and the other to carry data), fully battery-less and wireless systems can be installed with far lower labor costs, as well as reduced maintenance costs.

A primary example of this type of technology was created by EnOcean GmbH, an Oberhaching, Germany-based firm which created the technology bearing its name. The EnOcean technology allows a device to be self-powered by harvesting the energy generated as a result of the device’s operation. For example, an EnOcean light switch can harvest the kinetic energy generated when the physical switch moves from on to off or back again, creating enough energy to send small bits of data via a wireless radio link.



Researchers at the Victorian Research Laboratory of NICTA, Australia's Information Communications Technology Research Centre of Excellence, test the provision of power wirelessly to a planar spiral coil.

Similarly, an EnOcean daylight occupancy sensor can harvest ambient energy (heat) and convert it to a tiny electrical charge that will power the sensor, allowing it to send small bits of data across the network.

The greatest benefit to these devices is the ability to have a self-sustaining network of devices that do not need a hard-wired power source (which costs significantly more to install, due to the high cost of labor to string wires around a room or building), and eliminates the need for hundreds or even thousands of batteries for a network of devices, which would need to be changed every few years (a significant recurring labor cost).

Despite the added convenience and operational cost savings provided by these devices, the initial per-device unit costs are anywhere from 10 to 15 times higher than traditional wired devices, and EnOcean technology has caught on primarily in Europe, rather than in North America. However, cost is not the reason for the low rate of adoption in the U.S., says Jonathan Collins, an analyst with ABI Research, based in Oyster Bay, NY.

“Price is not the primary hurdle; the issue is around familiarity and reliability,” Collins says, noting that EnOcean and other wireless energy-harvesting technologies have not yet demonstrated the long track record of successful implementation that is usually required by building architects, engineers, and operators.

Still, the use of naturally occurring energy is not limited to building sensors. Advanced research into the harvesting and use of ambient radio frequency (RF) waves to send data and provide power to a device is under way at the University of Washington (UW).

Researchers in UW’s Department of Computer Science and Engineering’s Networks and Wireless Lab have devised a way for devices to communicate by repurposing television broadcast signals, using a technology called ambient backscatter. The technology works by equipping a device with a hardware receiver that consists of a pair of antennae that look very similar to an old “rabbit ears” television antenna, and a transistor designed to connect the two antennae together.

The antenna assembly is then connected to a power harvester tuned to capture or scatter UHF TV signals in a 50MHz-wide frequency band centered at 539MHz.

The receiver does double duty; it allows ambient RF energy to be harvested, and uses that energy to provide the power required to transmit small bits of binary data via RF signals, which can be absorbed by other receivers. Data can either be sent (when each antenna is set to work independently, thereby reflecting or “backscattering” ambient RF signals) or received (when the antennae are set to work in tandem, which permits the efficient absorption of ambient RF signals.)

Data is encoded and sent based on the binary pattern created by the switching between the absorbing and scattering of the ambient waves. The greatest benefit, however, is that the technology does not need a separate power source; the electronics and software use the energy created whenever the antenna is set to absorb radio waves, rather than requiring batteries or a hard-wired power source.

“All this communication is powered by harvesting the energy from the signals themselves,” says Shyamnath Gollakota, a research team member and assistant professor at UW. The technology is “transforming these existing ambient signals into both a source of power and also a communication medium that you

“There has to be a single standard, and as long as there are other groups trying to develop their own standards ... it is confusing.”

can piggyback onto these signals and transfer information.”

So far, prototypes have been able to send a very limited amount of data—roughly one kilobit per second—at a distance of up to about three feet, according to Gollakota. He expects the next iteration of the technology will expand that range to about 12 to 20 feet. In addition, Gollakota says his group is expanding the technology to utilize ambient backscatter with cellular signals, which generally operate at frequencies above 1GHz, compared to the 500MHz signals found in television broadcast signals.

Gollakota sees a future use for the technology in applications where low-cost and low-power data transfer is more important than high data-rate transfers, such as a replacement

for RFID tags. Devices equipped with this technology could communicate with a head-end reader or other tagged devices by leveraging existing ambient Wi-Fi or cellular signals, instead of having to set up a separate powered network of RFID readers. Furthermore, the limited amount of power that is able to be harvested from RF sources is unlikely to be useful for items that require large or constant sources of power, such as recharging batteries.

Scientists do not expect to see this technology in commercial use in the near future. Gollakota says the technology is only in prototype phase—with limited range and data throughput—and security issues have yet to be addressed.

Still, battery technology—and the creation of adjacent technologies designed to further the usability of battery-based devices—remains a key area of focus, particularly for devices that require continuous power, such as handheld personal devices and electric vehicles.

Wireless induction technology is currently being utilized to provide a more convenient way to recharge personal devices, with the end goal being a worldwide or regional standardized method for easily charging any device without requiring a physical plug-to-device link.

Wireless induction charging utilizes the underlying technology developed by Nikola Tesla and others more

Milestones

Computer Science Awards

SIMONS FOUNDATION HONORS NEWEST INVESTIGATORS

The Simons Foundation, a New York City-based private foundation devoted to advancing the frontiers of research in mathematics and the basic sciences, recently named 13 mathematicians, theoretical physicists, and theoretical computer scientists as recipients of its Simons Investigator awards.

Simons Investigator awards provide “extraordinary researchers” with long-term support “to investigate the most fundamental issues in their fields,” according to

the foundation.

The computer scientists who received this award are:

► Rajeev Alur, Zisman Family Professor in the Department of Computer and Information Science of the University of Pennsylvania, and a leading researcher in formal modeling and algorithmic analysis of computer systems.

Alur is a Fellow of the ACM, a Fellow of the IEEE, and has served as chair of ACM SIGBED (Special Interest Group on Embedded Systems).

► Piotr Indyk, professor in the Theory of Computation Group of

the Computer Science and Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, for his work on efficient approximate algorithms for high-dimensional geometric problems.

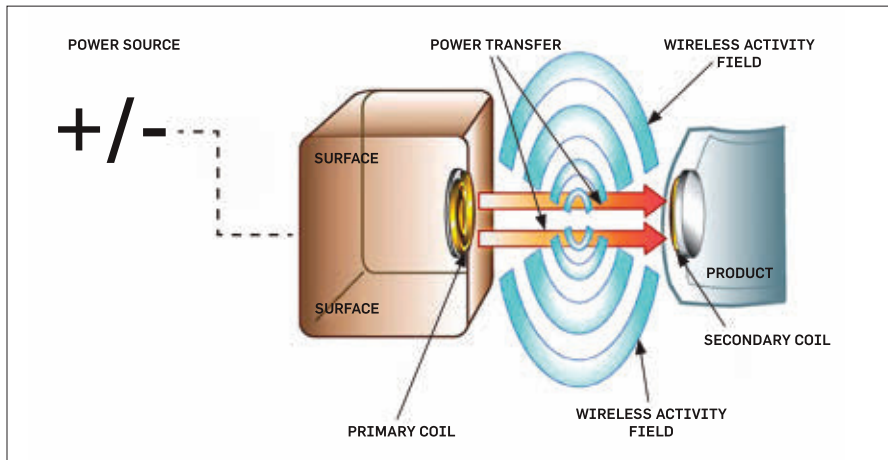
Indyk received ACM’s Paris Kanellakis Theory and Practice Award in 2012.

► Salil P. Vadhan, Vicky Joseph Professor of Computer Science and Applied Mathematics in the School of Engineering & Applied Sciences of Harvard University. Vadhan’s research includes computational complexity, cryptography, randomness in compu-

tation, and data privacy.

Vadhan served as program chair for the 43rd ACM Symposium on Theory of Computing (STOC ‘11), and is currently chair of the ACM SIGACT Committee for the Advancement of Theoretical Computer Science.

Simons Investigators are appointed for five years, with possible renewal for another five years. Each Investigator receives research support of \$100,000 per year, with another \$10,000 per year provided to the Investigator’s department.



Using inductive coupling, power is transferred from a primary coil (center) to a secondary coil in a compatible device, eliminating the need for power cords to charge the device.

than 100 years ago. Power is transmitted from a charging pad to a portable electronic device via electromagnetic waves, typically using an induction coil to create an alternating electromagnetic field from within a charging base station. A second coil, located inside the device, converts the electromagnetic field back into electrical current to charge the device's battery.

Competing technical approaches are in use, and in development. In 2012, the Power Matters Alliance (PMA) announced it was developing an open standard for inductive coupling technology, known as IEEE P2100.1. Meanwhile, the Wireless Power Consortium (WPC) has developed its own competing international standard, known as Qi (pronounced "Chi"), an open standard designed to promote interoperability between wireless devices and charging stations.

"There has to be a single standard, and as long as there are other groups trying to develop their own standards—in many cases, very similar standards—it is confusing," says John Perzow, vice president of market development for the Wireless Power Consortium. The lack of an international standard "will delay the adoption of this technology."

Induction is also being used on a much larger scale, to provide ongoing wireless charging of city buses in Gumi, South Korea. In this test setting, a wireless induction system from the Korea Advanced Institute of Science and Technology (KAIST) utilizes 180kW power strips embedded along a road that can create a mag-

netic field above them, which can be used to wirelessly transfer energy to a receiver underneath a bus. In this system, the strips in the road create the magnetic field, and a receiver located underneath the bus converts the magnetic field back into an electric current, which flows to the battery and motor, powering the bus. The power strip is switched on only when a bus is approaching or departing based on a proximity sensor, eliminating the possibility that a power strip will create a magnetic field when regular vehicles pass above it.

The buses receive up to 85% of the power generated by passing the receiver through the magnetic field, enabling the buses to use onboard batteries about one-third the size of a typical car battery, reducing the system's weight and cost. Charging takes place while the buses are being driven along their routes, eliminating downtime that would be required using traditional electric vehicle charging technology, while also reducing the vehicle emissions that are common with traditional internal combustion engines.

The challenge in bringing this technology to scale is based on the cost of the in-road infrastructure. Although KAIST claims the technology will only require 5% to 15% of an existing stretch of road to be rebuilt, Dong-Ho Cho, one of the main researchers involved with the project, admits that reducing the number of power strips in the road could impact performance and require larger batteries, eliminating the advantages of this system.

Cho notes that for the Gumi 7.5-mile test road, six power strips were installed at a cost of \$1.2 million, and the receiver included on each bus cost \$500,000, which indicates the commercial implementation of this technology may be a hard sell, particularly for cash-strapped municipalities that generally own and operate the majority of roads around the world.

Nevertheless, the concept of wireless induction charging has been proven in the Italian cities of Turin and Genoa. For the last 10 years, electric buses that use stationary induction charging (at the depot and via charging coils located under bus stops) have been used successfully, resulting in a system that provided economic payback of the initial investment in just four years. The major difference between these systems and the one proposed by KAIST is that the South Korean system allows buses to be charged while moving, instead of being charged while stopped. □

For Further Reading

Liu, V., Parks, A., Talla, V., Gollakota, S., Wetherall, D., Smith, J.

"Ambient Backscatter: Wireless Communication Out of Thin Air," University of Washington, <http://abc.cs.washington.edu/files/comm153-liu.pdf>

Charging by Induction, The Physics Classroom, <http://www.physicsclassroom.com/class/estatics/u8l2b.cfm>

"KAIST's wireless Online Electric Vehicle, OLEV, runs inner city roads," Korea Advanced Institute of Science and Technology (KAIST), http://www.eurekalert.org/pub_releases/2013-08/tkai-kwo080513.php

EnOcean Technology – Energy Harvesting Wireless, EnOcean GmbH, at http://www.enocean.com/fileadmin/redaktion/pdf/white_paper/WP_EnOcean_Technology_en_Jul11.pdf

Video

University of Washington- Ambient Backscatter: <http://www.youtube.com/watch?v=gX9cbxLS0kE>

KAIST OLEV Vehicle Demonstrations: <http://www.youtube.com/watch?v=BbzKzOmwgdY>

Keith Kirkpatrick is principal of 4K Research & Consulting, LLC, based in Lynbrook, NY.

© 2014 ACM 0001-0782/14/03 \$15.00

Playing at Health

Developers try to tap the beneficial effects of video games.

PATRICIA KAHLBAUGH'S DAD WAS an avid tennis player in his day, but after he lost his leg to diabetes, he was sidelined permanently. Once his family bought him a Wii, however, he would spend hours playing the virtual version of the game, and seemed engaged in the device's imagined universe.

That sparked Kahlbaugh, an associate professor of psychology at Southern Connecticut State University, to wonder whether playing such "exergames" could benefit the elderly beyond such obvious gains as providing exercise. She recruited 35 people with an average age of 82 from independent living elderly housing in the New Haven area and divided them into three groups. One group was assigned to play a Wii game of their choice for one hour a week with a graduate student; the second spent an hour a week watching television with a student; the third watched TV by themselves.

Over the 10 weeks of the study, participants who played Wii—they all chose bowling as their game—reported being in a better mood and feeling less lonely. Those who did not play did not show those gains.

Because the study was small and limited in scope, Kahlbaugh says, it is difficult to tell if it was the game itself, or the exercise and social interaction that went with it, that benefited the participants. "Is it the competition? Is it just being stimulated by a computer?" she asks. What did seem clear, though, was that playing the game made people feel better.

Many researchers are coming to believe that video games can, in fact, be good for you. The games have to be properly designed and appropriately used, and researchers are still gathering evidence of what exactly the effects might be, but they say the right games might teach both physical and social skills, affect mood, and generally improve well-being.



Metia Interactive managing director Maru Nihoniho demonstrating the SPARX video game, which teaches teenagers how to deal with feeling depressed or stressed.

Research from the University of California, San Diego (UCSD), seems to back up Kahlbaugh. Scientists there recruited adults between the ages of 63 and 94 and had them play Wii Sports three times a week for 12 weeks. They assessed players for symptoms of mild depression, and found that at the end of the 12-week period, those symptoms had decreased. Those benefits were still there when researchers followed up two months and three months later.

Ipsit Vahia, a geriatric psychiatrist who helped perform the UCSD study, says the game certainly seemed to help with what is known as subsyndromal depression—fleeting or intermittent episodes of sadness that don't rise to the level of full depression, but nevertheless hurt people's quality of life. Vahia says such depression is about three times as common as major depression in older adults, and not easily treated with antidepressant medication, so having games as an alternative form of therapy is appealing.

There has not yet been enough study for the Food and Drug Administration to approve games for therapeutic use. Researchers are working to build that evidence, and in the meantime Vahia says there is no harm in older people using exergames to improve their moods. "I have recommended it for clinical use with reasonably good results," he says.

He and other scientists worry, though, that some companies that market games as tools for training memory are exaggerating the benefits of those games. A 2010 study at Kings College London and the University of Manchester, for instance, found that brain training games improved performance on a task they trained users for, but the skills were not transferred to other tasks. "There is contradictory evidence about those games that are designed for memory and cognition and brain speed," Vahia says.

The area where there has been the most study is in the effects of violent

video games on the behavior of children, which show that “playground-level aggression” does in fact increase, says Douglas Gentile, a child psychologist at Iowa State University, although that is only one of many risk factors for childhood aggression. “The overall meta-analysis made it clear there is an effect; not a huge one,” Gentile says.

It is not surprising, when children learn that violence helps them succeed in a game, that they would transfer that to other areas of life, he says. “Whatever you’re practicing, you get better at,” Gentile observes. In the same way, though, games that are more “pro-social” can teach empathy and cooperation. Gentile recently completed a study that followed more than 2,000 children in Singapore who played pro-social games for two years, which showed their scores on standard methods of measuring childhood empathy improved over that period.

Gentile says studying both the positive and negative effects of video games can help game designers come up with new strategies for building beneficial games. “The dichotomous way of thinking, that games are good or bad, is probably the wrong way to think about it,” Gentile says. “Once we know what the effects are, we can be more thoughtful, so we are maximizing the benefits and minimizing the harms.”

Adam Gazzaley, a neuroscientist at the University of California, San Francisco, is trying to develop games that can bolster cognitive control—a variety of skills that include sustained attention, selective attention, task switching, and working memory (the ability to temporarily remember bits of information necessary to a task). Cognitive control is important to a variety of everyday tasks, such as driving a car, that draw on multiple mental skills at the same time. Gazzaley and a team of researchers developed NeuroRacer, a video game in which players use a joystick to navigate a car along a windy road while watching out for road signs. If a green circle pops up, which it does at random intervals, players were to shoot it, while ignoring green, blue, or red pentagons and squares.

After four weeks of playing the game, the players, adults between 60 and 85 years old, had improved their perfor-

Many researchers are coming to believe that video games can, in fact, be good for you.

mance, scoring better than 20-year-olds who had not been trained on the game. Follow-up tests six months later showed the players had maintained the skills they had acquired, without practice. Even cognitive skills not specifically targeted by the game improved for all the players.

“We know cognitive control abilities are a critical tool that people use to navigate the world around them,” Gazzaley says. “The leap is not so far that if these skills are better, they might translate into improvements in their lives.”

The study also used electroencephalograms to look at what was happening in the brains of the game players. Participants who improved the most also had the greatest increase in brain activity in the pre-frontal cortex, which Gazzaley says makes researchers more confident they are right about how such training affects the brain.

Gazzaley is also interested in how games can improve cognitive abilities in children. His lab, which includes computer scientists and game developers, is working on five more video games with the aim of testing them for both educational and therapeutic uses. He is also the founder of a company, Akili Interactive Labs, in Boston, MA, which is developing therapeutic games for mobile platforms. He hopes the research will reach the point where therapeutic games could win FDA approval.

Other companies are also working on developing games that can be certified as beneficial. Posit Science, of San Francisco, CA, is conducting clinical trials in the hopes of winning FDA approval for a game that can help treat schizophrenia. Atentiv, of Waltham, MA, is working on a game to treat Attention Deficit Hyperactivity Disorder.

Cogmed, a European company bought in 2010 by the London-based educational publishing company Pearson, makes games it says are designed to improve working memory.

Alvaro Fernandez, CEO of SharpBrains, a San Francisco market research firm that tracks what it calls the cognitive and brain fitness market, says such companies sprang up to fill a niche that was being ignored by established video game manufacturers. Those manufacturers are not interested in marketing games as beneficial, because they feel it makes them sound less fun, Fernandez says. “The mainstream publishers could not care less about this,” he says, but companies in this market have seen a substantial increase in worldwide revenues, from \$200 million in 2005 to \$1 billion in 2012.

Gazzaley says properly designed and scientifically validated video games might be made to aid in all sorts of areas, from education to therapy to social skills. “Games are just so broad in what they can do that if you really take your time and think through the principles, you can pretty much work anything into a game platform,” he says. ■

Further Reading

Revving up Brain Skills <https://www.youtube.com/watch?v=wL92mkXna7k>

Anguera, JA, Boccanfuso, J, Rintoul, JL, Al-Hashimi, O, Faraji, F, Janowich, J, Kong, E, Larraburo, Y, Rolfe, C, Johnston, E, Gazzaley, A **Video game training enhances cognitive control in older adults**, *Nature* 501, Sept. 5, 2013.

Owen, AM, Hampshire, A, Grahm, JA, Stenton, R, Dajani, S, Burns, AS, Howard, RJ, Ballard, CG **Putting brain training to the test**, *Nature* 465, June 10, 2010.

Rosenberg, D, Depp, CA, Vahia, IV, Reichstadt, J, Palmer, BW, Kerr, J, Norman, G, Jeste, DV **Exergames for Subsyndromal Depression in Older Adults: A Pilot Study of a Novel Intervention**, *Am J Geriatr Psychiatry* 18(3), March 2010.

Kahlbaugh, PE, Sperandio, AJ, Carlson, AL, Hauselt, J **Effects of Playing Wii on Well-Being in the Elderly: Physical Activity, Loneliness, and Mood**, *Activities, Adaptation & Aging* 35, Dec. 16, 2011.

Neil Savage is a science and technology writer based in Lowell, MA.

© 2014 ACM 0001-0782/14/03 \$15.00



DOI:10.1145/2566965

Pamela Samuelson

Legally Speaking Mass Digitization as Fair Use

Considering the implications of the late-2013 ruling in favor of Google in the Authors Guild case.

GOOGLE HAS SCANNED 20 million books for its Google Book Search (GBS) project. In most countries of the world, this mass digitization would be deemed copyright infringement as to books not in the public domain. In the U.S., however, a doctrine known as fair use makes it possible to argue that scanning books to make an index and to provide snippets of their contents in response to search queries does not infringe copyrights.

The Authors Guild, a nonprofit organization representing approximately 8,500 professional authors, believes that Google's scanning of in-copyright books from the collections of research library partners is copyright infringement. In 2005, it brought a lawsuit to challenge this practice.

In 2008, Google announced a settlement of this lawsuit as well as a similar lawsuit brought by five trade publishers. That settlement would have given Google the right, among other things, to commercialize all out-of-print books in the GBS corpus through

three different business models and to display up to 20% of book contents in response to search queries unless rights holders affirmatively stepped forward to opt out of these business models and displays. Judge Denny Chin disapproved the proposed settlement in March 2011.

After further negotiations failed to produce a new settlement, Judge Chin established a schedule for proceeding with the litigation. Accordingly, the Authors Guild and Google filed motions for summary judgment. (This allows judges to resolve disputes if no key facts are in dispute and the litigants just disagree about how the law applies to those facts).

Less than two months after oral argument on these motions, Judge Chin ruled in November 2013 that Google's scanning, indexing, and snippet-providing were fair and non-infringing uses of the in-copyright books in the GBS corpus. The Authors Guild has announced it will appeal. This column will explain why I predict the appellate court review will result in an affirmation of Judge Chin's decision.

What Is Fair Use?

Making fair uses of copyrighted works is lawful under U.S. law. Courts typically consider four factors in deciding whether a challenged use is fair or infringing: the purpose of the defendant's use; the nature of the copyrighted work; the amount and substantiality of the taking; and the harm to the market for the work.

Quoting a few sentences for a book review, parodying a popular song, reproducing parts of a politician's speech for news coverage, photocopying a news article relevant to a research project, and making time-shift copies of television programs are among the conventional uses that U.S. law would generally consider fair.

Most countries in the world do not have a flexible doctrine such as fair use in their copyright laws. Instead, legislatures typically identify specific types of uses that should be exempted. Quotation, news reporting, parody, and private study privileges are common in national copyright laws.

What is different about fair use is that it can be—and often is—applied



to situations not contemplated by the legislature. When Congress enacted a new copyright law in 1976, it did not think about whether making time-shift copies of television programs should be treated as infringement, let alone whether makers of videotape recorders (VTRs) should be held indirectly liable for any infringing uses of the devices. Courts were able to look to fair use as a way to balance the interests of copyright owners and users of VTRs and reach a judgment on these issues.

The Supreme Court's 1984 *Sony v. Universal City Studios* decision ruled that time-shift copying was fair use. The use was private and noncommercial. The programs copied had been shown on broadcast television for free. Whole programs were typically copied but consumers usually taped over the copy after seeing the program at a later time so the copies were not being retained. Universal had conceded no harm had yet occurred from time-shifting. Evidence offered about possible future harm was, in the Court's view, too speculative to matter.

Because Sony's VTR had substantial non-infringing uses, the Court concluded that consumers should be able to buy VTRs for these uses. Hence, Sony was not liable for infringing uses by some VTR users.

Subsequent fair use cases have addressed similar challenges posed by technological advances, including MP3 players, add-on software, and search engines.

The Authors Guild's Position

Although Google is not directly monetizing the books whose contents it displays when serving up snippets from in-copyright books, the Guild has asserted Google had unquestionably scanned the books for commercial purposes (for example, to improve its search engine technologies). Commercial purposes usually cut against fair use.

Unlike the reviewer who quotes from a book or a musician who parodies a popular song, Google is not creating a new work of authorship that builds upon pre-existing works and contributes to knowledge or culture.

Because of this, the Guild argues that Google's use of the books is "non-transformative." This also tends to cut against fair use.

Google has, moreover, made copies of millions of highly creative books without seeking rights holder permissions. Indeed, it has made multiple copies of each book in the corpus. And it has archived these copies on its servers. All of these factors, in the Guild's view, disfavor fair use.

The Authors Guild has raised three main arguments about harm. First, it invoked Supreme Court decisions saying that harm should be presumed when defendants make non-transformative uses of protected works for commercial purposes.

Second, the Guild has asserted that Google is interfering with a market that would have developed to license the kinds of uses that Google has made of these books. The Guild has contended that this potential licensing market has been harmed.

Third, the Guild has pointed to the risk that clever hackers might break into Google servers and "liberate" mil-

lions of in-copyright books, thereby harming the market for purchases of books. Or clever researchers could display much of a particular book's contents by running a series of search queries that would allow them to consume a substantial amount of the book without purchasing it.

In the Guild's view, no one should be able to make systematic copies of entire books without seeking copyright owner permission.

Judge Chin Agreed With Google

While accepting that there was some commerciality in the GBS project, Judge Chin agreed with Google that it had made "transformative" uses of the books. The Supreme Court has interpreted this term as including uses for a different purpose than the original. Google "transforms [the] expressive text [of books] into a comprehensive word index that helps readers, scholars, researchers, and others [to] find books," so its use of the books was not just transformative, but highly so.

Judge Chin relied upon some prior decisions saying that search engine "thumbnails" of images found on the open Web were transformative because they improved access to information. This fulfills the U.S. constitutional purposes of copyright law ("to promote the progress of science and useful arts"). Snippets of text, like thumbnails, enable users to find works in which they are interested. Thus, the purpose-of-the-use factor weighed strongly in favor of fair use.

The nature-of-the-work factor was not given much weight, although Judge Chin noted the vast majority of GBS books were non-fiction books from research library collections that were out-of-print. This somewhat favored fair use.

The Guild's harm assertions did not persuade Judge Chin. It was more likely, in his view, that GBS would be beneficial to authors and publishers. After all, GBS enables users to discover books in which they are interested and provides links to websites of booksellers so they can purchase relevant books.

The one factor that disfavored fair use, albeit only slightly, was the amount-and-substantiality-of-the-taking factor. Whole books were unquestionably copied. Yet

The Guild has asserted Google had unquestionably scanned the books for commercial purposes.

Judge Chin recognized this copying was necessary to make the index. It also mattered that Google displayed only a few snippets from each book in response to search queries.

Secondary Liability?

The Authors Guild claimed that Google was secondarily liable for copyright infringement because it provided its research library partners with digital copies of books that Google had scanned from their collections. The libraries have pooled these copies into a digital repository known as the HathiTrust.

The Guild brought a separate copyright infringement lawsuit against HathiTrust and Google's library partners for uses of books in the HathiTrust corpus. HathiTrust, like Google, has principally relied on fair use to justify its actions.

HathiTrust has asserted three purposes as favoring its fair use defense. Digitization preserves books in partner library collections. It enables text-mining so that researchers can discover books relevant to their interests. It enhances access to books for print-disabled persons. Once books are in digital form, it is relatively simple to convert the texts into braille or aural forms.

The *Authors Guild v. HathiTrust* case, like the *Authors Guild v. Google* case, was decided by a trial court on a summary judgment motion. Judge Baer, who presided over the *HathiTrust* case, decided in October 2012 that HathiTrust's uses of the digital books were fair.

In considering the secondary liability claim in the Google case, Judge Chin agreed with Judge Baer that the HathiTrust uses were fair. Google cannot be held secondarily liable for copyright infringement if the libraries' uses are fair, as Judges Baer and Chin have decided they are.

What Will Happen on Appeal?

The Authors Guild has appealed its loss in the *HathiTrust* case and plans to appeal in the *Google* case. A panel of three appellate court judges heard oral argument in the *HathiTrust* case in October 2013. Two of the judges seemed quite interested in and supportive of HathiTrust's fair use defense.

These two judges will also be on the panel that hears the Guild's appeal in the *Google* case. The third judge that will be on the *Google* panel is a famous fair use expert. He was sympathetic toward Google's fair use defense when that panel overturned a class certification ruling from which Google had appealed. This panel of judges ordered the fair use appeal to come back to them. One never knows, of course, what will actually happen on any appeal, but my prediction is the appellate judges will affirm both fair use rulings.

This is not only good news for Google, but also for U.S.-based libraries, archives, historical societies, and similar institutions that want to undertake mass-digitization projects to increase public access to their collections.

Even though these institutions might be confident that mass-digitization projects would, if tested in litigation, be deemed fair uses, they typically lack financial resources to defend fair uses. They may also be intimidated by the risk of large statutory damage awards if they lost the cases. Google has both the resources and will to take on this monumental struggle and win. Of course, Google did not fight the Authors Guild's lawsuit in order to benefit the public, but if this is a by-product of the ruling in the case, so much the better. □

Pamela Samuelson (pam@law.berkeley.edu) is the Richard M. Sherman Distinguished Professor of Law and Information at the University of California, Berkeley.

Copyright held by Author/Owner(s).

Computing Ethics

Why Software Engineering Courses Should Include Ethics Coverage

Encouraging students to become comfortable exercising ethical discernment in a professional context with their peers.

S SOFTWARE DEVELOPERS CREATE the architectures that govern our online and increasingly our offline lives—from software-controlled cars and medical systems to digital content consumption and behavioral advertising. In fact, software helps shape, not just reflect, our societal values.^a Are the creators of code aware of this power and the responsibilities that go with it? How, and to what extent, are they trained in the ethics of their discipline?

Like medical, legal, and business ethics, engineering ethics is a well-developed area of professional ethics. In 2000, the organization that accredits university programs and degrees in engineering (ABET) began to formally require the study of engineering ethics in all accredited programs.^b

Yet most engineering ethics textbooks focus primarily on ethical issues faced by civil, mechanical, or electrical engineers. The case studies they typically include—the Challenger explosion, the Ford Pinto fires, the Union Carbide/Bhopal disaster—depict harms caused by ethical lapses in those fields. Of course, the cars and rockets and bridges built today depend upon critical soft-



ware for their safe operation, and failure of these software systems can result in death or grievous injury. However, the distinctive ethical dilemmas that arise in the software engineering context are not yet being sufficiently addressed.

In the Internet era, the software development and deployment process has some peculiarities that exacerbate the ethical issues for software engineers. First, the shortened life cycle has

weakened and in some cases obliterated software review by management and legal teams. So software engineers may deploy code directly to end users—in stark contrast to, say, a civil engineering project with a years- or decades-long life cycle and multiple layers of oversight. For Web applications such as Facebook, individual engineers or small groups of engineers code and deploy features directly, and indeed the

a See <http://www.nyu.edu/projects/nissenbaum/vid/about.html>

b See <http://www.abet.org/engineering-change/>

culture takes pride in this. Even where more traditional development practices prevail, some deployments like bug fixes are shipped with only technical (and not ethical) oversight. In such circumstances, the individuals deploying the code may have to rely on their own familiarity with ethics when faced with the old question: it may be legal to do this, but *should we*?

Second is the issue of scale—perhaps the defining feature of the software revolution. For software, the entire world is typically part of the addressable market. This scale creates the potential for individual software engineers to produce great good, but with it naturally comes the ability to cause great harm, especially when combined with the ability to deploy code directly to end users. Here is a benign but illustrative example. On June 9, 2011, Google released a “doodle” honoring Les Paul, which users found addictive to play with. This is a type of project that is typically done by an individual engineer on his or her “20% time,” in a day or two. A third party, RescueTime, later estimated that 5.3 million hours were ultimately spent by people playing this game.^c

Consider that 5.3 million hours equates to about *eight lifetimes*. Did the doodle make a positive contribution to the world? Do engineers at Google have an obligation to consider this question before releasing the feature? What principle(s) should they use to determine their answer about the benefits and/or harms of their work? Often, individual software engineers must grapple directly with such issues, instead of relying on management or anyone else.

Finally, software engenders ethical concerns other than concrete harms. Software embeds moral and cultural values and inevitably nudges society toward these values. Today’s Web and information services are designed around the centralized collection and control of personal data. One effect is that social interactions happen more often in public view; another is the changing balance of power between users, companies, and governments. Further, the lack of geographic constraints means software engineers are

generally unfamiliar with the culture and values of many of the end users of their products. Cost cutting often leaves little room for user studies or consultations with experts that might allow software development firms to acquire this familiarity.

Nevertheless, software engineers share with everyone a desire to flourish and do well in life and work. Thus, ethical obligations have a professional *and* a personal dimension. Without a sense of personal ethics, a software engineer would be indifferent to her actions’ effects on the lives of others in circumstances not explicitly addressed by a professional code of ethics. But for professionals whose work impacts public welfare, personal ethics is not enough. Without a sense of professional ethics, individuals might justify to themselves conduct that would be much more difficult to justify in front of others. Additionally, professional ethics help us understand how ethical standards and values apply to a particular type of work. For example, what does integrity look like in a software engineering context? What sort of specific coding practices demonstrate integrity, or a lack of it? This is something that professional codes of ethics—and discipline-specific ethics training—can help students learn to see.

Being a professional means being a part of a moral community of others who share the same profound responsibilities. Embedding coverage of ethics in software engineering courses would help students draw strength and wisdom from dialogue with other future members of their profession—colleagues who will face the same types of moral dilemmas, struggle with the same sorts of tough decisions, and ulti-

Software embeds moral and cultural values and inevitably nudges society toward these values.

mately seek to earn, in similar ways, the respect of their peers and the broader public. Software engineering professors are in the best position to spark that dialogue. So why is applied ethics currently not taught much in computer science and software engineering courses? We cannot know for sure, but there are some plausible reasons.

First, the algorithmic and engineering techniques students learn are fantastically general (all that computers do, at an underlying level, is manipulate ones and zeroes). The flip side is that computer science is generally taught in a manner divorced from practical context. This abstraction makes computer science education powerful; it would be silly to train students to work specifically in the music or enterprise software industry. However, the exclusive focus on general techniques leaves students with the impression the implications and consequences of their work product are not their proper concern. When such students graduate to solving problems in the real world, they are likely to adopt the type of thinking that prevails in many parts of the industry—that anything technically feasible is fair game, and that ethical issues are best handled by compliance teams and Terms of Service documents.

Second, these technologies are inherently morally ambiguous. For example, the same digital signatures that make the lock icon appear in a browser (indicating you are not connected to a spoofed site) are used by malware authors to ensure zombie machines obey only commands from their true bot-masters. Or consider machine learning and collaborative filtering systems used to recommend new bands or books you might like, or to detect credit card fraud: inevitably, these systems introduce systematic biases into our patterns of consumption and behavior. The so-called “filter bubble” arises when algorithmic systems, such as Google search or the Facebook news feed, decide what information to show a user based on his or her past pattern of searches and clicks.^d The worry is that users will be fed reinforcing viewpoints and find themselves isolated in a personalized “echo chamber.”

c See <http://blog.rescuetime.com/2011/06/09/google-doodle-strikes-again/>

d See <http://www.amazon.com/Filter-Bubble-What-Internet-Hiding-ebook/dp/B004IYJE6A/>

At the level of demographics, the seemingly fair principle of treating “similar” users similarly can lead to deepening disparities. Online ads have been shown to display racial bias,³ and prices online have been shown to vary based on users’ personal attributes.⁴ In the political sphere, public-interest groups have been investigating the implications of campaign messages tailored to the individual.² Perhaps most worryingly, when systems with direct power over our lives, such as the no-fly list, use opaque machine-learning based techniques to make decisions, we lose the safeguards of due process.¹

Thus, when confronted with the bewildering variety of ethical questions that may arise from a single technology, engineering professors might well prefer to leave the whole topic to some “ethics professionals.”

A third factor is that it is often unclear how coding practices might mitigate these harms and risks. In all likelihood, for example, the racial bias of online ads was not the result of explicit intent by engineers but rather an emergent property of a system aiming to maximize the click-through rate. Even defining the notion of fairness of an algorithmic system in a mathematical way that computers can interpret remains elusive—let alone a general procedure for designing systems that satisfy this goal.⁶

But these are all reasons for teaching ethics in software engineering courses, not ignoring it. Ethical judgment, what philosophers often call *practical wisdom*, is needed most when moral dilemmas arise for which there are no easy solutions. And the software engineers are often the only ones who fully understand both the benefits and the dangers engendered by new technologies.

How to teach software engineering ethics? One choice is between a separate ethics course versus integrating ethics discussion into every course. Both approaches are valuable, but the latter is perhaps more immediately useful. We propose that computer science educators include a discussion of ethics with every significant technology they teach. Hypotheticals and case studies are two powerful and complementary tools for this purpose. Hypotheticals

e See C. Dwork et al., *Fairness Through Awareness*; <http://arxiv.org/abs/1104.3913>

Students should be in the habit of considering how the code they write serves the public good.

allow students to quickly isolate important ethical principles in an artificially simplified context; for example, one might ask students what ethical principles or values come into play if a manager suggests promising a customer ‘fictionware’—a desirable feature that is actually impossible to develop or deliver. Discussions of case studies, on the other hand, allow students to confront the tricky interplay between the sometimes competing ethical values and principles relevant in real-world settings. For example, the Google Street View case might be used to tease out the ethical conflicts between individual and cultural privacy expectations, the principle of informed consent, Street View’s public value as a service, its potential impact on human perceptions and behaviors, and its commercial value to Google and its shareholders. Then, to bring students back to the practical space of ethical action, the professor might pose a realistic hypothetical, asking students to explain and defend how, as a Google project manager, they would evaluate a proposal to bring Street View technology to a remote African village. What questions should be asked? Who should be consulted? What benefits, risks and safeguards considered? What trade-offs weighed?

What matters in these exercises is not that students can arrive at the ‘right’ answers; nor even that the instructor have them in hand. In many real-life cases there is no single right answer, only a range of more or less ethically informed and wise responses. What matters is that students get comfortable exercising ethical discernment in a professional context alongside their peers.

Educators should also seek to instill professionalism in students; cur-

rently many software engineers seem indifferent to or even actively reject this aspect of their work. Collaborative activities could help reinforce the sense of belonging. For example, students could be tasked with doing a collective survey of ethical lapses in the software industry, along with a survey of ethical attitudes among employees of various companies. Since these companies are prospective employers, the results will be of immediate value to students, increasing their motivation.

Habits are powerful: Students should be in the habit of considering how the code they write serves the public good, how it might fail or be misused, who will control it; and their teachers should be in the habit of calling these issues to their attention. Students may resist a bit—after all, one of the things that draws some people to programming is the opportunity to retreat into a happy zone of bit twiddling detached from the world, and we are proposing to habituate them out of this. Other students, however, may be more drawn to such an approach.^f Between confronting and evading ethics, confronting ethics is the only defensible choice. ■

f See <http://www.smartplanet.com/blog/bulletin/computer-science-is-for-women-too/>

References

1. Citron, D.K. Technological due process. *Washington University Law Review* 85 (2007), 1249–1313; http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1012360.
2. Larson, J. Message machine starts providing answers. *ProPublica* (Oct. 18, 2012); <http://www.propublica.org/article/message-machine-starts-providing-answers>.
3. Sweeney, L. Discrimination in online ad delivery. *Commun. ACM* 56, 5 (May 2013), 44–54; doi>10.1145/2447976.2447990.
4. Valentinio-Devries, J., Singer-Vine, J., and Soltani, A. Websites vary prices, deals based on users’ information. *The Wall Street Journal* (Dec. 24, 2013); <http://on.wsj.com/19pz4S5>.

Arvind Narayanan (arvindn@cs.princeton.edu) is an assistant professor of computer science at Princeton University, NJ.

Shannon Vallor (svallor@scu.edu) is an associate professor of philosophy at Santa Clara University, CA.

The authors thank Irina Raicu for her feedback and edits. The authors are also grateful for the comments received from numerous people including the anonymous reviewer and David Robinson.

A portion of this article appears in the introductory section of “An Introduction to Software Engineering Ethics”—a concise curriculum module available at no cost, to all professors, from the Markkula Center for Applied Ethics at Santa Clara University.

Copyright held by Author/Owner(s).



Peter J. Denning

DOI:10.1145/2566967

The Profession of IT 'Surfing Toward the Future'

A new report from Chile about improving economic competitiveness advances a novel interpretation of innovation. Timing is everything.

EVERYBODY, IT SEEMS, is interested in innovation. Many professionals actively seek innovations to deal with immediate concerns, such as product designs, and with long-term concerns such as education, pensions, and healthcare. Despite huge efforts, success rates are low. A new report, *Surfing Towards the Future*, by the Chilean National Council on Innovation for Competitiveness led by Fernando Flores,³ gives an unprecedented account of how innovations emerge. It proposes a skill set, “surfing history,” based on reading waves of possibilities and riding them to success. Crucial elements are the climate of exploration and adventure, the timing, and balance when buffeted by the unpredictable. I organized my reflections on the report as an interview.

You say that innovations are historical emergences. What does that mean?

When you take a long view, you see that innovations seem to appear at moments in history when the conditions are most conducive. The conditions are a need in the social community, and the existence of a suitable technology base. The innovation begins when someone proposes a new combination of existing technologies and components to meet the need. Timing is delicate and critical. Once the conditions are “ripe,” several people may propose the same innovation around the same time. Many innovation proposals fail because of poor timing: there is too little interest in the social community or



the technologies needed to make them work do not exist.

I hear frequently that innovations fail because of management problems or lack of creativity, not because of timing. Would not more management discipline or cleverness lead to better results?

Strategic planning is one of management's favorite tools. Unfortunately, it is a broken concept. It assumes implicitly that there is a fixed future and we can position ourselves for maximum return when that future

arrives. But there is no such thing as a future-that-will-happen, only a set of possibilities for what might happen. Unpredicted events can suddenly change the picture, blocking our path or opening up new and unanticipated paths. These unpredictable events can be almost anything such as a natural disaster, an emergency, a new declaration by a leader, a new technology, or a chance meeting of a new person. There is no fixed path—such as a plan would have you follow—to navigate through such uncertainty.

PHOTOGRAPH FROM EPIC STOCK MEDIA

The design thinking movement advocates creativity and imagination. From a business perspective, we often have an abundance of creative and imaginative proposals for solving our problems. How do we choose which ones to pursue? How do we turn the chosen ones into reality? It is all too common that people who plunge into creative design workshops are initially exhilarated by the possibilities they create, and then later frustrated by the indifference others display toward their designs and by the difficulties of getting people to agree to adopt them. Creativity and imagination are important, but not enough.

Wait a minute. We are peppered with stories of great inventors and geniuses who changed the world. Do you mean that their creativity and imagination was not the reason for their success?

We are very skeptical that innovation is caused by geniuses. We believe innovations emerge at moments in time when all the conditions are right. The person we credit in retrospect was an agent of history. We need to understand the conditions that make the time for an invention “ripe” and the dispositions that enable an inventor to sense the convergence and act on it.

Consider Louis Pasteur, who gave us a rabies vaccine in the 1890s. He had a history of other major innovations and was widely regarded as a genius. Pasteur was a master chemist. Two hundred years earlier, there was no field of chemistry. Pasteur could not have existed at that time. Similarly, if he were born 50 years later, others in the 1890s time period, who were also working on vaccines, would in all likelihood have discovered a rabies vaccine; it would be a done deal in his new time. The contribution we remember him for was possible only at that moment of time in France.

Come on. Pasteur clearly had something to do with it. He was not just a lucky bystander.

We agree. We think Pasteur had a skill we call “surfing” that enabled him to ride the waves of possibilities swirling around him and find a path that led to great value. An example of Pasteur’s surfing skill arose in the

early 1880s, when he was searching for an anthrax vaccine. After he discovered the vaccine and verified in his lab that it worked, he did not go public, to the great frustration of his colleagues and advisors. The right moment showed up when a famous veterinarian challenged him to a public test of the vaccine. He accepted the challenge. In the test, no vaccinated animal got anthrax and every unvaccinated animal perished. The theatre of the test and its subsequent publicity propelled Pasteur, his anthrax vaccine, and his germ theory of disease into public prominence and earned him encomiums like “benefactor of humanity.” His timing and sense of drama were impeccable. We argue that Pasteur was a genius at surfing waves of possibilities. We would also speculate that if Pasteur lived in a different time or place, his surfing skill would have helped him make other achievements, but not anthrax or rabies vaccines.

What about Thomas Edison? He is celebrated for his creativity and imagination. Was he a surfer?

Edison was recognized as a genius and, yes, he was a virtuoso surfer as well. Contrary to the popular story, he did not invent the light bulb. A long search for electric lighting had begun in the early 1800s. He joined the search in the 1870s because he saw how to build an electrical distribution system that would power lamps in homes and shops. He performed hundreds of experiments until he found a lamp design that would last long enough to be useful. He announced his incandescent lamp in 1879. A year later, he patented an electrical distribution system and got one operating in 1882 on Hudson Street in New York City. He said lamps were useless without electricity and he pledged to make electricity so cheap that only the rich would burn candles. He undertook the lamp experiments only when he believed that by the time he found a robust lamp he would have a solution for cheap electrical distribution. He timed his entry onto the waves of those possibilities and rode them to a convergence. He set off an avalanche of people moving to use electric rather than gas lighting.

Calendar of Events

May 6–9

ACM The First Annual International Conference on Nanoscale Computing and Communication
Atlanta, GA,
Contact: Ian F. Akyildiz,
Email: ian@ece.gatech.edu

May 7–9

Gender and IT Appropriation, Science and Praxis in Dialogue — Forum for Interdisciplinary Exchange
Siegen, Germany,
Contact: Wulf Volker,
Email: volker.wulf@uni-siegen.de

May 18–21

SIGSIM Principles of Advanced Discrete Simulation
Denver, CO,
Sponsored: SIGSIM,
Contact: John A. Hamilton Jr.,
Email: hamilton@auburn.edu

May 20–22

Computing Frontiers Conference
Cagliari, Italy,
Sponsored: SIGMICRO,
Contact: Pedro Trancoso,
Email: pedro@cs.ucy.ac.cy

May 21–23

Great Lakes Symposium on VLSI 2014
Houston, TX,
Sponsored: SIGDA,
Contact: Joseph R. Cavallar,
Email: cavallar@rice.edu

May 29–31

2014 Computers and People Research Conference
Singapore,
Sponsored: SIGMIS,
Contact: Damien Joseph,
Email: adjoseph@ntu.edu.sg

May 31–June 3

Symposium on Theory of Computing
New York, NY,
Sponsored: SIGACT,
Contact: Howard J. Karloff,
Email: howard@cc.gatech.edu

June 1–7

36th International Conference on Software Engineering
Hyderabad, India,
Sponsored: SIGSOFT,
Contact: Pankaj Jalote,
Email: jalote@iiitd.ac.in



Association for
Computing Machinery

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

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

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

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



What do you mean by avalanche?

Avalanche is an important metaphor for innovation.¹ It is a cascading series of events where each one triggers new events. Entrepreneurs try to anticipate the right moment—the tipping point⁴—when an innovation proposal will trigger an avalanche of people adopting the proposal. If the timing is too early, the proposal will not be a tipping point. If it is too late, someone else will have started the avalanche. The metaphor is borrowed from complexity theory, where snow avalanches or sandpile avalanches can only be described by the frequency and duration of a cascade, but it is impossible to predict whether any particular event will trigger a cascade or how long it will last. Edison banked on cheap electricity causing an avalanche of people moving from gas to electric lighting. Intel Chairman Andy Grove once said any technology that could do something 10 times better than any current technology creates a high risk of an avalanche in favor of that technology. In the Internet, the introduction of the Mosaic browser in 1994 triggered an avalanche into the World Wide Web and into commercial use of the Internet; prior to that time, commercial uses were discouraged. Today, a number of education leaders see MOOCs as the beginning of an avalanche that will transform education.

What about research in science and technology? If we would pay for more research and development, we would have more components to solve our problems.

We agree that science and technology R&D are important, but R&D is not a reliable precursor of innovation. Technologies are deeply woven into social systems and their practices. Even when we have an exploitable scientific recurrence, we cannot predict how the corresponding technology will be used because that depends on how the social system moves with it.

What about putting those three elements together? Suppose we pay attention to creativity, science, and technology. Can we then get the innovations we need?

Still no. That is the Silicon Valley Illusion. Silicon Valley tries to reduce

innovation to an equation: science + technology + creativity = innovation. There are many social processes and practices in the background that enable Silicon Valley to be a technology innovator. No one has been able to replicate Silicon Valley in other places because they do not understand all the social processes, and even if they did, the processes probably would not fit with the culture of another region. It is far better to establish close ties with people in innovation regions than to try to replicate their regions in yours.

Now I am at a loss. If none of the usual approaches—planning, genius, science, technology, and creativity—is sufficient for innovation, how does someone go about it?

This was on our minds when we started our work on the report. If Chile winds up adopting policies based on assumptions that do not work, it will not succeed in its goal of improving its competitiveness. This is why we stepped back to ask how innovations actually happen. We produced the new interpretation you see in the report, and that allowed us to make policy recommendations that are likely to help Chile. And any other country as well.

Our new interpretation is that innovations emerge in history. Every innovation bears the imprint of its history and its era. The more common notions of new ideas and adopted practices hide the historical emergence. Unless you can see how historical emergence works, you cannot create policies that foster it.

Our examples of Pasteur and Edison point out that innovation is not simply a moment of insight, it is the product of the era (the times). The ingredients of an innovation are a concern or problem for which a resolution would be valuable, a set of existing components (including technologies) that enable or constrain possible resolutions, and a proposal for a particular combination of components (a design). Pasteur's concern was to protect people against rabies; existing components included methods for producing animal vaccines, expertise in chemistry, expertise in microscopy, and a validated theory of germs; the proposal was a rabies vaccine. These three elements are all highly depen-

dent on the historical moment. Many of the needed components did not exist years before. The rabies vaccine proposal would seem routine years later. Pasteur's genius included a critical skill we call "surfing history." He was a master at chemistry and had unusual insights about connections between disease, germs, and life science. But his ability to bring all that into the service of mankind flowed not from his scientific genius, but from his surfing mastery.

OK, now you have my attention. What is "surfing history"?

This is first and foremost a metaphor. Think of the pictures you have seen of surfers riding the waves. They seem to have a special sense of when is the exact right moment to jump on their board and start the ride; too soon and they can be washed under, and too late they flub. They find balance points between giant waves and ride them through narrow openings between the turbulence of the peaks and valleys. It takes a lot of practice and research on waves to develop the sense of how to do it. And surfers are always thinking about the waves and looking for new ways to ride them. They learn how the shape of the shoreline and the rocks hidden in the deep affect the movements of the waves and adjust their approach to match. It is a beauty to watch.

In innovation, the waves represent movements of possibilities. Some possibilities develop a momentum and a form as they gather believers and followers. There are many waves of possibilities moving toward the horizon we call the future. Sometimes they clash and cancel, sometimes they reinforce. Like the surfer, the innovator develops a feel for the movements of these possibility waves, a sense of how to ride their balance points, and a sense of when it is too early or too late to join a wave.

Reality is more complicated. Ocean surfers ride waves but do not alter them. The history surfer worries not only how to ride existing possibilities but to generate new ones to ride after that.

Waves of water are real, but possibilities are abstract. The feel of a balance point on ocean waves is not the same

In innovation, the waves represent movements of possibilities.

as chasing an opportunity or dodging a threat in business.

Ah, you are mistaken; possibilities are quite real. Think of all the things you wanted but could not get. The possibilities did not materialize, and the failures could be just as life changing as being wiped out by a crashing wave.

So where do we find these waves of possibilities?

They appear in the conversations, moods, and emotions of a community. Every conversation opens new possibilities and is affected by the moods and emotions of its participants. The innovator looks for signs that a possibility has many followers (or does not) and which emotions are manifested in the conversations and in people's behaviors. Conversations and moods that continue over time within a large group are big waves worth watching. Some of the new social media, such as Twitter, are cultivating many new observers of shifting waves of moods in social communities. Since the conversations and the commitments they fulfill cause a little bit of history to be made, we can say that the waves of conversations are making history. That is why we can say that innovators are surfers of history.

What kinds of "possibility waves" must I surf?

We think there are six, depending on the time horizons during which the waves move. Just like waves on the ocean, possibilities have different wavelengths and directions. Starting with the widest, the horizons are:

1. The "fulgor." That Spanish word, also found in English, means something dazzling and bright. We use it as a metaphor for an insight that opens

up a new world of possibilities for dealing with a concern.

2. Next is basic science, which is a search for recurrences that explain what is going on and suggest new technologies. These searches can be triggered by fulgors. Research centers, laboratories, and universities, and universities are common locations.

3. Next is a search for applications of basic science principles.

4. Next is a search for new products that embody concrete applications.

5. Next is opening new markets for products supporting the applications.

6. Finally, is the everyday use of things.


The experienced history surfer is aware of movements at all these levels at the same time.

What conversations make me into a surfer?

There are two kinds. Pragmatic conversations are the ones in which we request, promise, offer, declare, assess, and assert. These conversations make commitments, and commitments produce actions. If we are good at fulfilling our commitments, we build trust.

The other kind is the world-opening conversation. Sadly, this kind is much less common.² In these conversations, we tune in on concerns, sense moods, articulate possibilities, notice anomalies (disharmonies), harbor contradictions, and suspend judgments.

What do you recommend for those wanting to learn to be surfers of history?

This is a big topic that will get us into the challenges now faced by our education institutions. Let us come back to it in a future interview. 

References

1. Barber, M., Donnelly, K., and Rizvi, S. *An Avalanche is Coming: Higher Education and the Revolution Ahead*; <http://www.pearson.com/avalanche>
2. Denning, P. The other side of language. *Commun. ACM* 56, 9 (Sept. 2012), 35–37.
3. Flores, F. Report of Consejo Nacional de Innovación para la Competitividad, *Surfing Towards the Future: Chile on the 2025 Horizon*. (2013); <http://chile-california.org/surfing-towards-the-future-intocounselor-fernando-flores-vision-for-innovation>
4. Gladwell, M. *The Tipping Point*. Back Bay Books, 2002.

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, is Editor of *ACM Ubiquity*, and is a past president of ACM. The author's views expressed here are not necessarily those of his employer or the U.S. federal government.

Copyright held by Owner/Author(s).

Broadening Participation

The Impact of the United Nations Convention on the Rights of Persons with Disabilities

Codifying human rights and inclusiveness in a technical context for people with disabilities.

THE UNITED NATIONS Convention on the Rights of Persons with Disabilities (CRPD) is an international treaty that outlines the rights of persons with disabilities and how nations who sign and ratify the convention should ensure those rights. The CRPD with its Optional Protocol were adopted in on December 13, 2006. To date, 158 countries signed and 138 have ratified the convention. According to the World Health Organization, there are approximately one billion people in the world who have a disability of one kind or another. A disability may be congenital or happen during life through wars, accidents, diseases, and even life-saving medical procedures. Everyone who becomes old tends to become disabled. There was a time, and it is still the case in some parts of the world, when those with disabilities were hidden away and considered to be a burden on society. Over time there has been a realization that disability is just another dimension in the diversity of human life and that people with disabilities deserve the same human rights and dignity that those without a disability have. The CRPD is an attempt to codify those rights and delineate how they might be achieved.

In this column, I focus on two ideas in the CRPD that speak to us as computer professionals:



- ▶ What the CRPD says about technology for people with disabilities.

- ▶ What the CRPD says about including people with disabilities in the technology workforce.

These two themes are intertwined because technology can help enable people who have a disability to become students and eventual creators of the next generation of technology.

Further, with more people with disabilities in the technology workforce, it is more likely the technology developed will be accessible from the beginning, not as an afterthought. The challenge is: how can we as computer professionals, and collectively in our organizations, help realize the vision of the CRPD. I begin with an overview of the CRPD.

What Is the CRPD?

An excellent source of information about the CRPD is the United Nations Enable website.^a In 2001, the General Assembly of the United Nations established an Ad Hoc Committee “to consider proposals for a comprehensive and integral international convention to promote and protect the rights and dignity of persons with disabilities, based on the holistic approach in the work done in the fields of social development, human rights and non-discrimination and taking into account the recommendations of the Commission on Human Rights and the Commission for Social Development.” After five years of work by representatives of many countries and groups such as Disabled Peoples’ International the Convention was finalized and opened for signature and ratification by the nations of the world.

The CRPD has 50 articles with the purpose stated in part in Article 1: *The purpose of the present Convention is to promote, protect and ensure the full and equal enjoyment of all human rights and fundamental freedoms by all persons with disabilities, and to promote respect for their inherent dignity.*

The CRPD governed by the principles stated in Article 3:

- a. Respect for inherent dignity, individual autonomy including the freedom to make one’s own choices, and independence of persons;
- b. Non-discrimination;
- c. Full and effective participation and inclusion in society;
- d. Respect for difference and acceptance of persons with disabilities as part of human diversity and humanity;
- e. Equality of opportunity;
- f. Accessibility;
- g. Equality between men and women;
- h. Respect for the evolving capacities of children with disabilities and respect for the right of children with disabilities to preserve their identities.

It is important to note the CRDP represents the social model of disability, not the medical model that focuses on functional impairments. By contrast, the social model reflects the reality that most long-term functional impairments will not have a cure in one’s

Although the CRPD is directed at national governments and public policy, parts of it are a blueprint for computing organizations.

lifetime, so it is important to think of the person with a disability holistically, as just another member of humankind in its great variety.

What the CRPD Says About Technology

Technology plays a significant role in providing access to the activities of life, employment, and social well-being for people with disabilities. This is recognized in eight articles of which I point out two that are relevant to the computing field. In Article 4, General Obligations, there is one section that addresses technology: *g. To undertake or promote research and development of, and to promote the availability and use of new technologies, including information and communications technologies, mobility aids, devices and assistive technologies, suitable for persons with disabilities, giving priority to technologies at an affordable cost;*

In Article 9, Accessibility, there are two more sections that address technology: *g. Promote access for persons with disabilities to new information and communications technologies and systems, including the Internet; h. Promote the design, development, production and distribution of accessible information and communications technologies and systems at an early stage, so that these technologies and systems become accessible at minimum cost.*

Article 4.g. recognizes that research and development on new technologies will continue and that most people in the world, especially those with disabilities, are poor. Thus, a priority should be given to technologies that have a lower cost. Article 4.g. basically states

that the Internet and other information and communication technologies should be accessible. It is generally true that all popular browsers support accessibility, but individual websites may not be accessible because they were not designed following accessibility standards like WCAG 2.0.² Article 4.h. says accessibility should be built into technology from the beginning, not as an afterthought. This last point is consistent with the arguments that Vint Cerf made in his *Communications* President’s Letter column entitled “Why Is Accessibility So Hard?”¹ Fortunately, some companies such as Apple have decided that accessibility is important enough to build it into their mainstream products. One form of this is Apple’s VoiceOver screen reader that allows a blind iPhone or iPad user to navigate and read the touch screen easily by employing intuitive gestures and text-to-speech technology.

What the CRPD Says About Workforce Inclusion

Article 27, Work and employment, directly addresses workforce inclusion in this statement: *States Parties recognize the right of persons with disabilities to work, on an equal basis with others; this includes the right to the opportunity to gain a living by work freely chosen or accepted in a labor market and work environment that is open, inclusive and accessible to persons with disabilities.*

It is stressed further in the article that persons with disabilities should have the opportunity for all kinds of employment in the public and private sectors, and as owners of their own businesses. Considering that technology is so important to the well-being of people with disabilities, it is natural they would be attracted to and would be very valuable in computing fields. This is not to say all computer professionals with disabilities should all be working on accessible technology. The way to think of it is that if there were more computer professionals with disabilities, then because of their expertise and perspectives, it is more likely that new technologies will meet the needs of those one billion people who have disabilities. Realize too that all of us as we age are increasingly likely to join this group of one billion. These are a lot of customers.

a See United Nations Enable; <http://un.org/disabilities>

Security and privacy for augmented reality systems

Who does what in a MOOC?

Formally verified mathematics

Fusing functional and object-oriented programming with Scala

Bounded biharmonic weights for real-time deformation

Small data, where $n = m$

Rate-limiting state

Is multicore hardware for general-purpose parallel processing broken?

Plus the latest news about simultaneous translation technology, how computers are changing medical diagnostics, and ways computers can replace animal testing.

Implications of the CRPD to ACM and Other Computing Organizations

Although the CRPD is directed at national governments and public policy, parts of it are a blueprint for computing organizations. ACM is the largest organization of computing professionals in the world, which means it should be leading the computing field in the right direction with regard to inclusion of people with disabilities in the computing workforce. ACM sponsors the annual ACM Richard Tapia Celebration of Diversity in Computing, which in the past few years has recognized disability in its scope. One of the featured speakers at 2014 conference is the famous blind computer scientist Chieko Asakawa. Three special interest groups SIGCHI, SIGCSE, and SIGACCESS are making systemic changes to become more welcoming and accessible to members with disabilities at their conferences. A notable new feature of the recent ACM SIGACCESS International Symposium on Computers and Accessibility (ASSETS) was a user experience panel of three panelists: one blind, one deaf-blind, and one with severe motor and speech disabilities. All three described their personal experiences with technology as well as that of others with their disability. This direct interaction of disabled users of technology with accessibility researchers was a powerful encounter.

In the U.S., the relatively new organization the National Center for Minorities and People with Disabilities in Information Technology (CMD-IT), under the leadership of Valerie Taylor, has taken the lead in including disability in its mission and title. Most of the major activities of CMD-IT are in workforce development and are welcoming and accessible to people with disabilities.

Role of AccessComputing

AccessComputing (Alliance for Access to Computing Careers) is a U.S. National Science Foundation-funded alliance that started in 2006.^b Its goal is to increase the number and success of students pursuing computing careers. AccessComputing has three main approaches toward achieving its goal.

► **Direct interventions** with students with disabilities through workshops, internships, and other activities.

► **Institutional change** through ca-

Many students with disabilities have a need for technology so they should be involved in its creation.

capacity building institutes and working with departments and organizations to help them become more welcoming and accessible to those students.

► **Information dissemination** through articles, webpages, online courses, and searchable knowledgebase.

AccessComputing has 35 institutional and organizational partners and many more collaborators who share the same goal and work on activities toward that goal. Because of the close relationship between technology and disability it makes sense for an organization like AccessComputing to exist. Many students with disabilities have a need for technology so they should be involved in its creation. This AccessComputing model is applicable in all the nations who have signed or ratified the CRPD. As the Principal Investigator for AccessComputing, I invite readers from around the world to contact me to help you start your own AccessComputing Alliance or let me know of some other similar program in your own country. ■

^b See AccessComputing; <http://www.washington.edu/accesscomputing/>

References

1. Cerf, V.G. Why is accessibility so hard? *Commun. ACM* 55, 11 (Oct. 2012), 7.
2. Web Content Accessibility Guidelines (WCAG 2.0). <http://www.w3.org/TR/WCAG20/>.

Richard E. Ladner (ladner@cs.washington.edu) is Professor in Computer Science and Engineering at the University of Washington. He is Principal Investigator for AccessComputing, an NSF-funded alliance with the goal of increasing the success and number of students with disabilities in computing fields.

The writing of this column was supported by the National Science Foundation Grant No. CNS-1042260. Any questions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the U.S. federal government.

Copyright held by Author/Owner(s).

Viewpoint

How to Build a Bad Research Center

Sharing lessons learned from experiences creating successful multidisciplinary research centers.

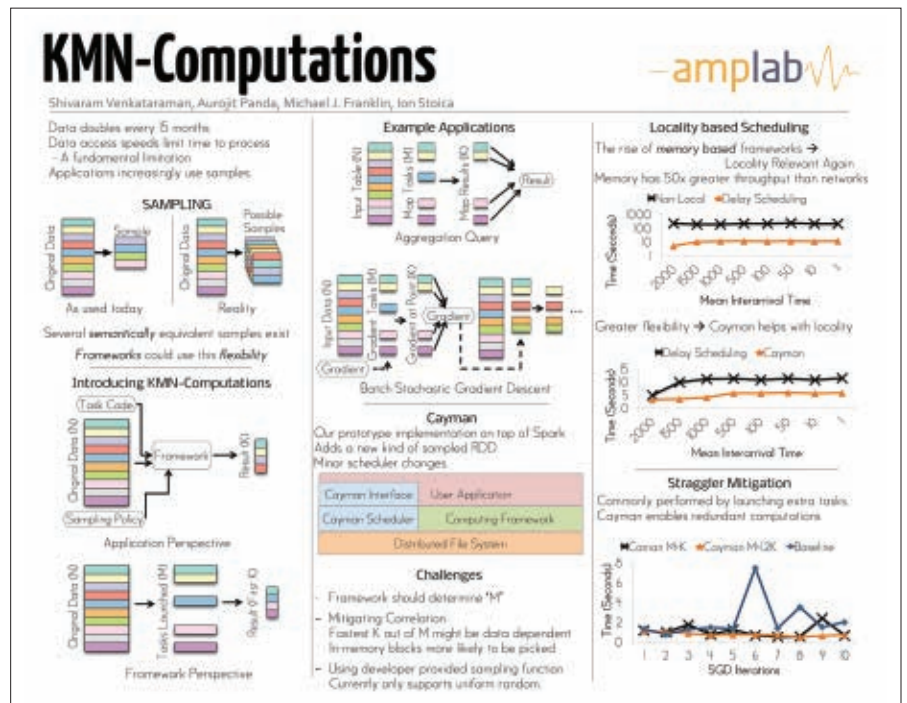
A MAJOR CENTER JUST COMPLETED,¹⁰ so I finally have time to collect my thoughts on centers. I have been part of a dozen centers in computer systems, often as director (see the accompanying table). By center, I mean a research project with at least three faculty, a dozen students, and a common vision. This Viewpoint is from the perspective of an academic in computer systems, but I hope it has wider applicability, even beyond academia. I do not advocate centers for all research; a lone researcher is best for many topics. Why care about the Berkeley experience? Alas, establishing credentials is a lot like bragging so let me apologize in advance, which I will also need to do later. The *U.S. News and World Report* ranked universities for the computer systems field four times since 2002. In every survey, our peers rank UC Berkeley first. In addition, the National Research Council published a study of information technology research that led to multibillion-dollar industries.⁶ UC Berkeley was associated with seven such industries, more than any other university, primarily for its system projects.

The Eight Commandments for a Bad Center

Following the template of my earlier pieces^{a,b} I offer Eight Command-

a D. Patterson, <http://www.cs.berkeley.edu/~pattsrn/talks/BadTalk.pdf>, 1983.

b D. Patterson, <http://www.cs.berkeley.edu/~pattsrn/talks/BadCareer.pdf>, 1997.



A sample poster from the 2013 Berkeley EECS Annual Research Symposium open house at UC Berkeley's AMPLab.

ments on “How to Build a Bad Research Center.” I later suggest how to avoid bad centers.

Bad Commandment 1. Thou shalt not mix disciplines in a center. It is difficult for people from different disciplines to talk to each other, as they do not share a common culture or vocabulary. Thus, multiple disciplines waste time, and therefore precious research funding. Instead, remain pure.

Bad Commandment 2. Thou shalt expand centers. Expanse is measured

geographically, not intellectually. For example, in the U.S. the ideal is having investigators from 50 institutions in all 50 states, as this would make a funding agency look good to the U.S. Senate.

Bad Commandment 3. Thou shalt not limit the duration of a center. To demonstrate your faith in the mission of the center, you should be willing to promise to work on it for decades. (Or at least until the funding runs out.)

Bad Commandment 4. Thou shalt not build a graven prototype. Integrat-

ing results in a centerwide prototype takes time away from researchers' own, more important, private research.

Bad Commandment 5. Thou shalt not disturb thy neighbors. Good walls make good researchers; isolation reduces the chances of being distracted from your work.

Bad Commandment 6. Thou shalt not talk to strangers. Do not waste time convening meetings to present research to outsiders; following the eighth commandment, reviews of your many papers supply sufficient feedback.

Bad Commandment 7. Thou shalt make decisions as a consensus of equals. The U.S. Congress is a sterling example of making progress via consensus.

Bad Commandment 8. Thou shalt honor thy paper publishers. Researchers of research measure productivity by the number of papers and the citations to them. Thus, to ensure center success, you must write, write, write and cite, cite, cite. If the conference acceptance rate is 1/X, then obviously you should submit at least X papers, for otherwise chances are that your center will not have a paper at every conference, which is a catastrophe.

Alternatives to a Bad Research Center

Creating alternatives to bad centers requires breaking all eight command-

ments. While I use my Berkeley experience for concrete examples, I polled the alumni from the accompanying table and found that projects at CMU, Google, Harvard, Wisconsin, and UC San Diego are breaking these commandments as well.

Good Commandment 1. Thou shalt mix disciplines in a center.

Good Commandment 2. Thou shalt limit the expanse of a center. The rapid change in underlying technologies, and hence our fields, leads to new opportunities. While industry has more resources, it is often difficult for companies to innovate across conventional interfaces. The psychological support of others also increases the collective courage of a group. Multidisciplinary teams, which increasingly involve disciplines outside computer science, have greater opportunity if they are willing to take chances that individuals and companies will not. I believe in our fast-moving fields there are now more chances for impact between disciplines than within them.

Proposers often promise nearly anything to increase the chances of getting funding, with little regard to running a center should funding be procured. For example, many believe that including numerous faculty and institutions increases chances of funding. One excuse is simply the

difficulty of evaluating research; as it takes time to judge center impact, there is little downside to proposing unwieldy centers. One study suggests following both of these good commandments. After examining 62 NSF-funded centers in computer science, the researchers found that multiple disciplines increase chances of research success, while research done in multiple institutions—especially when covering a large expanse—decreases them: “The multi-university projects we studied were less successful, on average, than projects located at a single university. ... Projects with many disciplines involved excelled when they were carried out within one university.”²

A downside to multidisciplinary research is that it does take time to understand the differences in culture and vocabulary. If you believe multidisciplinary centers offer the best chance for having impact, however, then the benefits outweigh the costs.

Good Commandment 3. Thou shalt limit the duration of a center.

My career has been based on five-year research centers, as the table attests. This sunset clause arose from three observations:

► *To hit home runs, it is wise to have many at bats.* Fortunately, people remember research home runs and not the near misses. My experience has been that the chance for home runs is more a function of the number of research projects than of the years spent, so shorter projects give you more chances for success.

► *It is difficult to predict information technology trends much longer than five years.* We start a center based on our best guess of what new opportunities will present themselves in seven to 10 years, which is the right target for a five-year research center. It is much, much more difficult to guess what the opportunities will be in 15 to 20 years, which you would need for longer projects.

► *U.S. graduate student lifetimes are about five years.* It is much easier to run a center if there is no turnover of the people. As we are in academia, at the start of each new project we recruit a new crop of students, as they will not graduate for five years.

You need a decade after a center finishes to judge if it was a home run.

David Patterson's research centers (the research center director is listed first in the third column). The growth in project size over time probably has as much to do with our increasing success at fund-raising as it does with trying to tackle bigger research problems.

Years	Title	Profs: Director, Co-PIs	Students
1977–1981	X-Tree: Tree Multiprocessor	Despain, Patterson, Sequin	12
1980–1984	RISC: Reduced Instructions	Patterson, Ousterhout, Sequin	17
1983–1986	SOAR: Smalltalk On A RISC	Patterson, Ousterhout	22
1985–1989	SPUR: Symbolic Processing Using RISCs	Patterson, Fateman, Hilfinger, Hodges, Katz, Ousterhout	21
1988–1992	RAID: Redundant Array of Inexpensive Disks	Katz, Ousterhout, Patterson, Stonebraker	16
1993–1998	NOW: Network of Workstations	Culler, Anderson, Brewer, Patterson	25
1997–2002	IRAM: Intelligent RAM	Patterson, Kubiawicz, Wawrzynek, Yelick	12
2001–2005	ROC: Recovery Oriented Computing Systems	Patterson, Fox	11
2005–2011	RAD Lab: Reliable Adaptive Distributed Computing Lab	Patterson, Fox, Jordan, Joseph, Katz, Shenker, Stoica	30
2007–2013	Par Lab: Parallel Computing Lab	Patterson, Asanovic, Demmel, Fox, Keutzer, Kubiawicz, Sen, Yelick	40
2011–2017	AMP Lab: Algorithms, Machines, and People	Franklin, Jordan, Joseph, Katz, Patterson, Recht, Shenker, Stoica	40
2013–2018	ASPIRE Lab	Asanovic, Alon, Bachrach, Demmel, Fox, Keutzer, Nikolic, Patterson, Sen, Wawrzynek	40

Just eight of the 12 centers listed in the table are old enough, and only three of them—RISC, RAID, and the Network of Workstations^c center—could be considered home runs. If slugging .375 is good, then I am glad that I had many five-year centers rather than fewer long ones.

A downside to sunset clauses is that it is easier to recruit students and sustain funding as a center grows in reputation than to recruit to or to fund a new center. However, if the goal is to maximize home runs versus recruiting or funding, in our fast-moving fields it is better to declare victory after five years and look afresh for new opportunities, and then to recruit the right team to pursue them.

Good Commandment 4. Thou shalt build a centerwide prototype. A common feature of the centers in the table is the collective creation of a prototype that demonstrates the vision of the center, which helps ensure the pieces are compatible. While systems students like building their part of the center vision, they do not necessarily want to work with others to make everything fit together. However, the educational power of such multidisciplinary centers comes from students of different backgrounds working together, as the experience improves their understanding and taste in research. This process also enhances students' system-building skills by requiring them to do a real prototype, not just toy demos. Such prototypes can even lead to open source projects, which simultaneously aid technology transfer and expand the workforce building the prototype. In fact, open source success generally means more developers outside versus inside the center.

One downside of such centers is that faculty must convince students of the benefits of spending some of their time working for the common good versus working just on one's own piece. However, once they start seeing how the research of others benefits their results, they no longer require encouragement to do so.

^c The NOW project showed clusters of workstations helped everything from encryption to sorting. The Inktomi search engine was built on NOW. The startup Inktomi Inc. in turn proved the value of the value of clusters of many low-cost computers versus fewer high-end servers, which Google and others in the Internet industry later followed.

I believe in our fast-moving fields there are now more chances for impact between disciplines than within them.

Good Commandment 5. Thou shalt disturb thy neighbors. Researchers have found that innovation is enhanced if all participants work in a single open space less than 50 meters across, as it encourages spontaneous discussions across disciplines.¹ The goal is for the space to support concentration *and* communication. For example, for the last four centers listed in the table, the faculty gave up their private offices to be embedded with students and postdocs in open space, where only the meeting rooms have walls.⁸ Faculty access draws students from their home offices to the lab, which increases chances of interactions.

A downside of shared space is the cost of remodeling to create an attractive open space. This is a one-time capital expense, since following projects will use it, but even so, the cost was equal to just two students over the life of a center. Shared open space is certainly more beneficial to a center than a few more students.

Good Commandment 6. Thou shalt talk to strangers. A key to the success of our centers has been feedback from outsiders. Twice a year we hold three-day retreats with everyone in the center plus dozens of guests from other institutions. You can think of it as having a "program committee" that meets with everyone for six days a year for five years, or a month of feedback per center. Their value is indicated by 100% of the respondents in my alumni poll using them.

Having long discussions with outsiders often reshapes the views of the students and the faculty, as our guests bring fresh perspectives. More impor-

tantly, at the end of each retreat we get frank, informed, and thorough feedback from our valued guests. Researchers desperately need insightful and constructive criticism, but rarely get it. During retreats, we are not allowed to argue with feedback when it is given; instead, we go over it carefully on our own afterward. In fact, we warn that we are taking their advice seriously, so be careful what you ask for.

Retreats fulfill other important roles:

- ▶ They provide real milestones, which are scarce in academia. It is one thing to promise your advisor you will get something done by January, but quite another when you are scheduled to give a talk on it to respected visitors.

- ▶ They build esprit de corps, in that everyone in the center spends three days together. There is free time to play as well as work, so students in these centers often become lifelong friends. In fact, I was delighted to learn from the poll that perhaps due to their shared experience, there is even a sense of connection between the generations of systems alumni.

- ▶ Over their five-year center lifetimes, all students get 10 chances to present posters or give talks, which gives them as much exposure as they desire and improves their communication skills.

A downside to retreats is again cost, which is about as much as one or two students. As before, exchanging retreats for a few more students would be ill advised. You also need to be careful which outsiders you invite to the retreat to be sure that they are really engaged and that they will offer useful and constructive criticism.

Good Commandment 7. Thou shalt find a leader. To make progress, you need to find someone willing to spend the time to pull people together, to create a shared vision, to build team spirit, and in who investigators believe will make decisions in the best interest of the center. When I am the director, I am playing the game to be on the winning team rather than to be the most celebrated coach. Hence, I recruit faculty who are team players, leaving prima donnas for others to manage. I try to lead by example, working hard for the center's success while hoping my teammates are inspired to follow. My guiding motto is "There are no losers on a winning team, and no winners on a losing team."

Good Commandment 8. Thou shalt honor impact. While papers are surely associated with high-impact research, they are not a direct measure of it. High-impact research can even be difficult to publish. The first paper on the LLVM compiler, winner of the 2012 ACM Software Systems Award, was initially rejected by the main compiler conference (PLDI), and the first paper on the World Wide Web by Tim Berners-Lee was rejected by Hypertext '91.

Nor are best paper awards necessarily predictors of high impact. One study counts citations of best paper awards, documenting the modest influence of most of them.^d For example, the conference program committee for the first MapReduce paper, which led to the 2012 ACM Infosys Award, named two other papers as best! Test of time awards, given 10+ years after the conference, are a much better indicator of success but they are trailing indicators.

For my successful projects, the early indicators were not papers or awards, but non-researchers trying to use our ideas. For RISC and NOW, the first adopters were startups or at least young companies. For RAID, it was hungry companies trying to gain market share. The market leaders were happy with the status quo; they developed related products only *after* others had commercial success.

Educational Impact of Centers

Some may be surprised to learn that centers can improve undergraduate education, which is a principle of research universities. Less than a decade after seeing how to pipeline microprocessors in the RISC project, universities taught undergraduates to design their own. The RAD Lab led to a reinvention of our software engineering course, which has long been problematic in CS departments.³ It grew quickly from 35 to 240 Berkeley students, and inspired massive open online courses where 10,000 earned certificates in 2012. These classroom innovations led to three textbooks, which further expand educational impact.^{4,5,9}

Good research centers are really

federations of subprojects that share a common vision, with each subproject having one or two professors and several students. Each professor is an expert in a different field, which facilitates multidisciplinary research. These subprojects give students the same faculty attention as single-investigator project, but in addition students:

- ▶ Learn by working and negotiating with dozens of smart, hard working, and stubborn collaborators, some of who go on to notoriety.

- ▶ Acquire good taste in research that comes from working on a common prototype, which leads to clear-eyed dissertations.

- ▶ Get a user base and feedback for new tools and approaches.

- ▶ Have senior students as role models in the open space, which can lead to explicit mentoring.

- ▶ Have fewer feelings of loneliness that is all too common in the Ph.D. process, due to the esprit de corps that comes from the open space and retreats.

- ▶ Can be reenergized by praise from retreat visitors.

I cannot be modest while espousing the successes of this model, so as I warned earlier, let me yet again apologize in advance. When DARPA cut the research funding to universities in the last decade,⁷ our pitch to companies was that we needed multidisciplinary centers to keep producing sterling systems students. Happily, our claim proved to be true. For example, one student from the first cohort in the open space of the RAD Lab came up with the idea for the Spark cluster-computing framework¹¹ after overhearing machine-learning students in the lab gripe about MapReduce. Not only did he get job offers from all companies he visited, he got them from all the top universities too.

In case you were curious, these projects do not just produce single superstars. For example, the Par Lab averaged three students per paper and each student averaged three first-papers. More importantly, Burton Smith opined: "These are the best graduate students I have ever seen." This is high praise from a Microsoft Fellow and National Academy of Engineering (NAE) member. What Smith said in 2013 was obviously gratifying, but I was struck

that this is exactly what Mark Weiser, the father of ubiquitous computing, said about SPUR students in 1989.

Reflections on Five-Year, Multidisciplinary Research Centers

If such centers are good for faculty as well as for their students, then by now we should be able to tell. One measure is election to NAE and the National Academy of Sciences: only five to 10 computer scientists are selected each year for NAE (with half from industry), and just two or three for NAS. Thus far, nine of the 24 Berkeley CS faculty who worked in the centers listed in the table are in NAE (~40%), and three are also in NAS. In fact, the majority of the Berkeley CS faculty in NAE or in NAS worked on these projects. In case you were wondering, they were elected after joining these centers.

Whereas early computing problems were more likely to be solved by a single investigator within a single discipline, I believe the fraction of computing problems requiring multidisciplinary teams will increase. If so, then learning how to build good centers could become even more important for next 40 years than it has been for the past 40. □

References

1. Allen, T. and Henn, G. *The Organization and Architecture of Innovation: Managing the Flow of Technology*. Butterworth-Heinemann, 2006.
2. Cummings, J. and Kiesler, S. Collaborative research across disciplinary and organizational boundaries. *Social Studies of Science* 35, 5 (2005), 703–722.
3. Fox, A. and Patterson, D. Crossing the software education chasm. *Commun. ACM* 55, 5 (May 2012), 44–49.
4. Fox, A. and Patterson, D. *Engineering Software as a Service: An Agile Approach Using Cloud Computing*. First Edition. Strawberry Canyon, 2014.
5. Hennessy, J. and Patterson, D. *Computer Architecture: A Quantitative Approach*. Fifth Edition. Morgan Kaufmann, 2011.
6. *Innovation in Information Technology*. National Research Council Press, 2003.
7. Lazowska, E. and Patterson, D. An endless frontier postponed. *Science* 308, 5723 (2005), 757.
8. Patterson, D. Your students are your legacy. *Commun. ACM* 52, 3 (Mar. 2009), 30–33.
9. Patterson, D. and Hennessy, J. *Computer Organization and Design: The Hardware/Software Interface*. Fifth Edition. Morgan Kaufmann, 2013.
10. *The Berkeley Par Lab: Progress in the Parallel Computing Landscape*. D. Patterson, D. Gannon, and M. Wrinn, Eds., 2013.
11. Zaharia, M. et al. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing* (2010).

David Patterson (pattsrn@cs.berkeley.edu) is the E.H. and M.E. Pardee Chair of Computer Science at UC Berkeley and is a past president of ACM.

Copyright held by Author/Owner(s).

An interview with David Patterson appears on page 112.

^d See Best Papers vs. Top Cited Papers in Computer Science; <http://arnetminer.org/conferencebestpapers>.



ACM Books



MORGAN & CLAYPOOL
PUBLISHERS

Publish your next book in the ACM Digital Library

ACM Books is a new series of advanced level books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers.

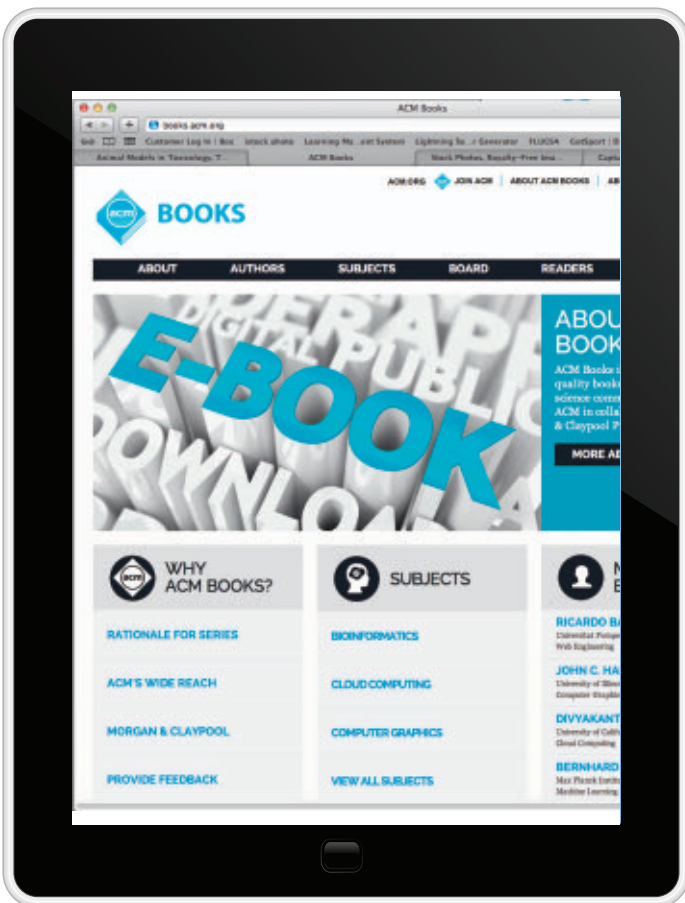
I'm pleased that ACM Books is directed by a volunteer organization headed by a dynamic, informed, energetic, visionary Editor-in-Chief (Tamer Özsu), working closely with a forward-looking publisher (Morgan and Claypool).

—Richard Snodgrass, University of Arizona

books.acm.org

ACM Books

- ◆ will include books from across the entire spectrum of computer science subject matter and will appeal to computing practitioners, researchers, educators, and students.
- ◆ will publish graduate level texts; research monographs/overviews of established and emerging fields; practitioner-level professional books; and books devoted to the history and social impact of computing.
- ◆ will be quickly and attractively published as ebooks and print volumes at affordable prices, and widely distributed in both print and digital formats through booksellers and to libraries and individual ACM members via the ACM Digital Library platform.
- ◆ is led by EIC M. Tamer Özsu, University of Waterloo, and a distinguished editorial board representing most areas of CS.



Proposals and inquiries welcome!

Contact: **M. Tamer Özsu**, Editor in Chief
booksubmissions@acm.org



Association for
Computing Machinery

Advancing Computing as a Science & Profession

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

Methods of quantifying consistency (or lack thereof) in eventually consistent storage systems.

BY WOJCIECH GOLAB, MUNTASIR R. RAHMAN, ALVIN AUYOUNG, KIMBERLY KEETON, AND XIAOZHOU (STEVE) LI

Eventually Consistent: Not What You Were Expecting?

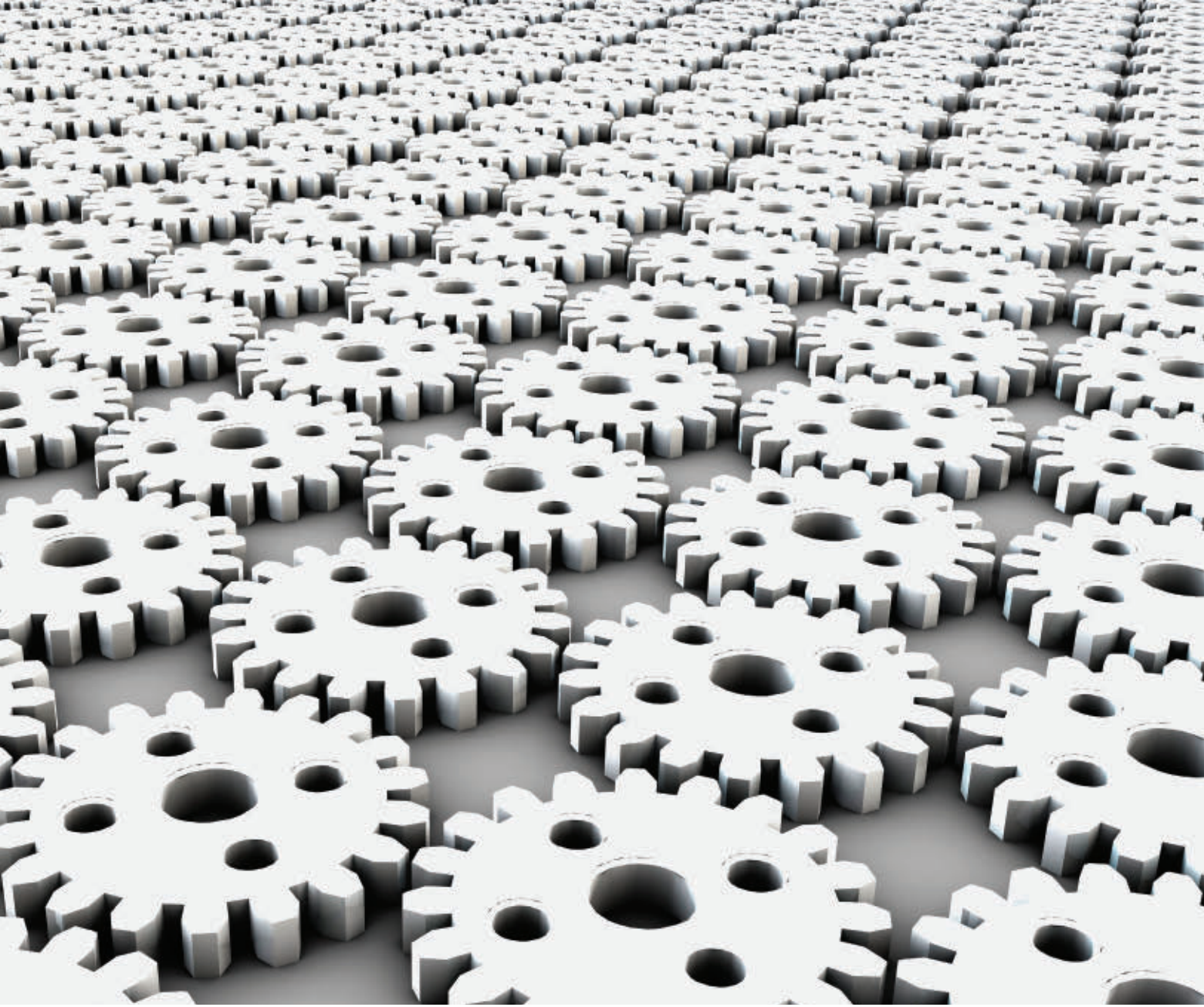
STORAGE SYSTEMS CONTINUE to lay the foundation for modern Internet services such as Web search, e-commerce, and social networking. Pressures caused by rapidly growing user bases and datasets have driven system designs away from conventional centralized databases and toward more scalable distributed solutions, including simple NoSQL key-value storage systems, as well as more elaborate NewSQL databases that support transactions at scale.

Distributed key-value storage systems are among the simplest and most scalable specimens in the modern storage ecosystem. Such systems forgo many of the luxuries of conventional databases, including



ACID (atomicity, consistency, isolation, durability) transactions, joins, and referential integrity constraints, but retain fundamental abstractions such as tables and indexes. As a result, application developers are sheltered from technicalities such as normalizing the relational schema, selecting the optimal transaction isolation level, and dealing with deadlocks.

Despite their simple interface and data model, distributed key-value storage systems are complex internally as they must replicate data on two or more servers to achieve higher read performance and greater availability in the face of node or network failures. Keeping these replicas synchronized requires a distributed replication protocol that can add substantial latency to storage operations, especially in geo-



replicated systems. Specifically, write operations must update a subset of the replicas before the system acknowledges the completion of the write to the client, and reads may fetch data from a subset of the replicas, adopting the latest value observed (for example, one having the highest timestamp) as the response.

As it turns out, choosing which subset of replicas to contact for a storage operation profoundly impacts the behavior of distributed storage systems. For strong consistency a client may read and write a majority of replicas. Since majorities overlap, this ensures each read “sees” the latest write. In contrast, eventually consistent systems may read and write non-overlapping subsets.^{26,28} Once a write is acknowledged, the new value is propa-

gated to the remaining replicas; thus, all replicas are eventually updated unless a failure occurs. In the meantime, readers may observe stale values if they fetch data from replicas that have not yet received the update.

Although many applications benefit from strong consistency, latency-sensitive applications such as shopping carts in e-commerce websites choose eventual consistency to gain lower latency.¹ This can lead to consistency anomalies such as items lost from a shopping cart or oversold. However, since a user can detect and correct the problem during checkout, such anomalies are tolerable provided they are short-lived and occur infrequently. The critical task for application developers and system administrators, therefore, is to understand

how consistency and latency are impacted by various storage and application configuration parameters, as well as the workload that the application places on the storage system. Only with proper insight into this subtle issue can administrators and developers make sensible decisions regarding the configuration of the storage system or the choice of consistency model for a given application.

This article looks at methods of quantifying consistency (or lack thereof) in eventually consistent storage systems. These methods are necessary for meaningful comparisons among different system configurations and workloads. First, the article defines eventual consistency more precisely and relates it to other notions of weak consistency. It then drills down into metrics, fo-


cusing on the dimension of staleness, and surveys different techniques for predicting and measuring staleness. Finally, the relative merits of these techniques are evaluated, and any remaining open questions are identified.

Defining Eventual Consistency


Eventual consistency can be defined as either a property of the underlying storage system, or a behavior observed by a client application. For example, Doug Terry et al. give the following definition of eventual consistency in the context of the Bayou storage system: “All replicas eventually receive all writes (assuming sufficient network connectivity and reasonable reconciliation schedules), and any two replicas that have received the same set of writes have identical databases.”²⁶ This informal definition addresses both the propagation of writes to replicas (that is, the *eventual*) and convergence onto a well-defined state—for example, through timestamp-based reconciliation—that is, the *consistency*).

This definition, while simple and elegant, does not say precisely what clients observe. Instead, it captures a range of behaviors encompassing both strongly consistent relational databases, and weakly consistent systems that may return stale or out-of-order data without bound. In contrast, Werner Vogels describes the consistency observed by clients in a concrete way: “The storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value.”²⁸ Vogels’s definition captures in an intuitive way the convergence of replicas to the last updated value, but only in the special case where updates are suspended while clients read an object—a scenario very different from the online environment in which many eventually consistent systems are deployed.

Since weak consistency is difficult to reason about, application developers often seek stronger properties such as a consistent prefix, monotonic reads, “read my writes,” or causal consistency.²⁵ Indeed, some systems support such stronger-than-eventual properties (for example, COPS [Clusters of Order-Preserving Servers²²] and Pileus²⁷), but the commercial success of systems such as Amazon’s



Eventual consistency can be defined as either a property of the underlying storage system, or a behavior observed by a client application.



Dynamo¹² proves that eventual consistency can be good enough in practice. To understand why, researchers have sought to describe the range of behaviors of eventually consistent systems in more precise terms than abstract definitions in the style of Terry et al. and Vogels. Existing approaches in this endeavor fall into three categories: relaxed consistency metrics, system modeling and prediction, and empirical measurement tools. The next three sections survey representative works in each category.

Relaxed Consistency Properties

Abstract definitions of eventual consistency leave open a number of questions regarding system behavior in the presence of concurrent accesses, as well as in a failure-prone environment. For example: How quickly do updates propagate to replicas in practice, and how often do replicas agree on the latest value of an object? When replicas do not agree, how does that affect the clients’ view of the data? What exactly do clients observe during an extended failure that partitions the network, or in the moments shortly after the network is healed?

An emerging approach for describing these behaviors is to relate them to a strict form of consistency called *linearizability*.¹⁸ Informally speaking, this property states that the storage system behaves as though it executes operations one at a time, in some serial order, despite actually executing some operations in parallel. As will be explained later, this serial order is further constrained in a manner that forbids stale reads. As a result, linearizability is a suitable gold standard against which eventually consistent systems can be judged. In other words, the observed behavior can be described in terms of deviations from this standard, which in this context can be regarded as consistency violations. For example, one can think of Amazon’s Dynamo as violating linearizability when reads return stale values, even though the system promises only eventual consistency.

Whereas linearizability is ubiquitous in centralized systems—for example, in multithreaded data structures—Brewer’s CAP principle⁹ states that such a strong consistency property (C) is unattainable in always-available (A)

and partition-tolerant (P) distributed systems. (Database experts should read *consistency* in this context as *transaction isolation*.) As a result, one cannot expect any storage system to accept updates and yet remain consistent during a network partition. Although this does not preclude strong consistency during failure-free operation, even in that case the system may be configured to sacrifice consistency for better latency.⁴ For example, in the Cassandra key-value store, clients can effect this trade-off by requesting various “consistency levels,” which control the number of replicas that respond to a read or write.²⁰

The side effect of weakening consistency is increased staleness, which so far has been discussed informally. More precisely, a value is considered *fresh* from the moment it is written into a storage system until it is overwritten, and *stale* thereafter. Thus, staleness describes the age of a value returned by a read relative to the last updated value, and hence quantifies how badly a system’s behavior deviates from the gold standard. Two interpretations of this concept have been discussed in the literature,^{2,15} arising from different ways to define “age”:

- *Version-based staleness* defines age by counting versions of an object (for example, a read returns the k th-latest version).

- *Time-based staleness* defines age in terms of wall-clock time (for example, a read returns a value t time units older than the last updated value).

The concept of staleness, although intuitive, is fraught with technical subtleties. In the simple case where operations are executed one at a time, there is a natural order in which clients expect these operations to take effect, and so stale reads can be identified easily. When operations are executed in parallel, however, their order can be very difficult to determine from the client’s perspective.

To make sense of eventual consistency properties— k -atomicity and Δ -atomicity—which give precise meaning to the notions of version-based and time-based staleness, respectively. Both properties are relaxed forms of linearizability,¹⁸ but for historical reasons they refer in name to Lamport’s *atomicity* property,²¹ which is similar in spirit.

Linearizability (The “Gold Standard”). Consider the trace of operations in Figure 1a, showing three operations applied to an object denoted by X . First, a write operation $W(X,1)$ assigns 1 to object X , and then this value is updated to 2 by a second write operation, $W(X,2)$. The third operation $R(X)$ is a read of X and begins after both writes have ended. In this case, linearizability dictates that $W(X,1)$ should appear to take effect before $W(X,2)$, because $W(X,1)$ ends before $W(X,2)$ begins. Thus, 2 is the last updated value of X when $R(X)$ is applied, but $R(X)$ returns 1 instead, so the trace is not linearizable. The more complex case when operations overlap in time is addressed by the formal definition of linearizability,¹⁸ a discussion that is beyond the scope of this article.

K-Atomicity (Version-Based Staleness). The k -atomicity property was introduced by Amitanand Aiyer et al.² Like linearizability, it requires that operations appear to take effect in an order that is constrained by their start and finish times; within this order, however, a read may return *any* of the k last updated values. For example, the trace shown in Figure 1a is k -atomic for $k = 2$ but not $k = 1$.

Δ -Atomicity (Time-based Staleness). The Δ -atomicity property was proposed by Wojciech Golab et al.¹⁵ Similar to k -atomicity, it relaxes linearizability by allowing reads to return stale values. Staleness, however, is defined in terms of time: a read may return a value that is up to Δ time units stale. For example, the trace in Figure 1a is Δ -atomic if Δ is defined as the width of the gap

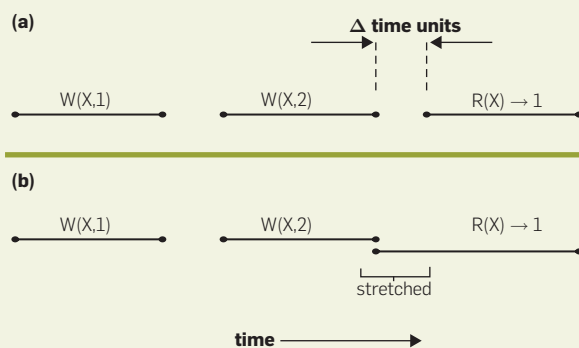
between $W(X,2)$ and $R(X)$. Linearizability permits $R(X)$ to take effect before $W(X,2)$ if the two operations overlap in time, as shown in Figure 1b, whereas it requires $R(X)$ to take effect after $W(X,2)$ in Figure 1a. If $R(X)$ is hypothetically “stretched” to the left by Δ time units (that is, if $R(X)$ had started Δ time units earlier), as in Figure 1b, then the trace becomes linearizable. Thus, the response of $R(X)$ is considered only Δ time units stale in Figure 1a.

Prediction

Another method used to characterize eventual consistency is based on a combination of system modeling and prediction. Peter Bailis et al.⁶ present the PBS (Probabilistically Bounded Staleness) framework, in which a white-box system model is proposed to predict data staleness observed by client applications. The PBS model estimates the probability of $\langle k, t \rangle$ staleness—the condition that the read, which begins t time units after the end of the write, returns the value assigned by one of the last k writes. This condition is similar to k -atomicity but considers only a single read at a fixed distance t from the write. When $k = 1$, it captures the probability that the read returns the latest value (that is, is not stale).

The PBS model makes two simplifying assumptions. First, like Vogels’s definition of eventual consistency,²⁸ PBS does not consider workloads where writes overlap in time with reads, in which the width t of the gap between writes and reads is not well defined. Second, PBS does not model failures explicitly. Although storage node failures can be simulated using longer

Figure 1. Example trace of operations illustrating linearizability, k -atomicity, and Δ -atomicity.



latencies, PBS does not account for network partitions. Follow-up work¹¹ extends the PBS model by considering node failures.

Empirical Measurement

Thus far, this article has addressed techniques for quantifying staleness in eventually consistent systems. This section discusses approaches that measure consistency empirically from the perspective of both the system and the client.

Measuring eventual consistency “in the wild” is as difficult as defining it precisely. Concurrent operations make it difficult to identify the order in which operations take effect, and hence to classify reads as stale or not. To make matters worse, in the event of a network partition, clients on opposite sides of the divide may observe the last updated value differently, even if no new updates are made to an object after some point in time. Such

an anomaly is possible because in an always-available system each partition will continue to accept writes.

Despite these challenges, a number of techniques have been devised for measuring eventual consistency, particularly data staleness. This article looks at two fundamentally different methodologies: *active measurement*, in which the storage system is exercised in an artificial way to determine the time lag from when a new value is written to the storage system until this value becomes visible to clients; and *passive analysis*, in which a trace of operations is recorded for an arbitrary workload and analyzed mathematically to obtain a measurement of staleness.

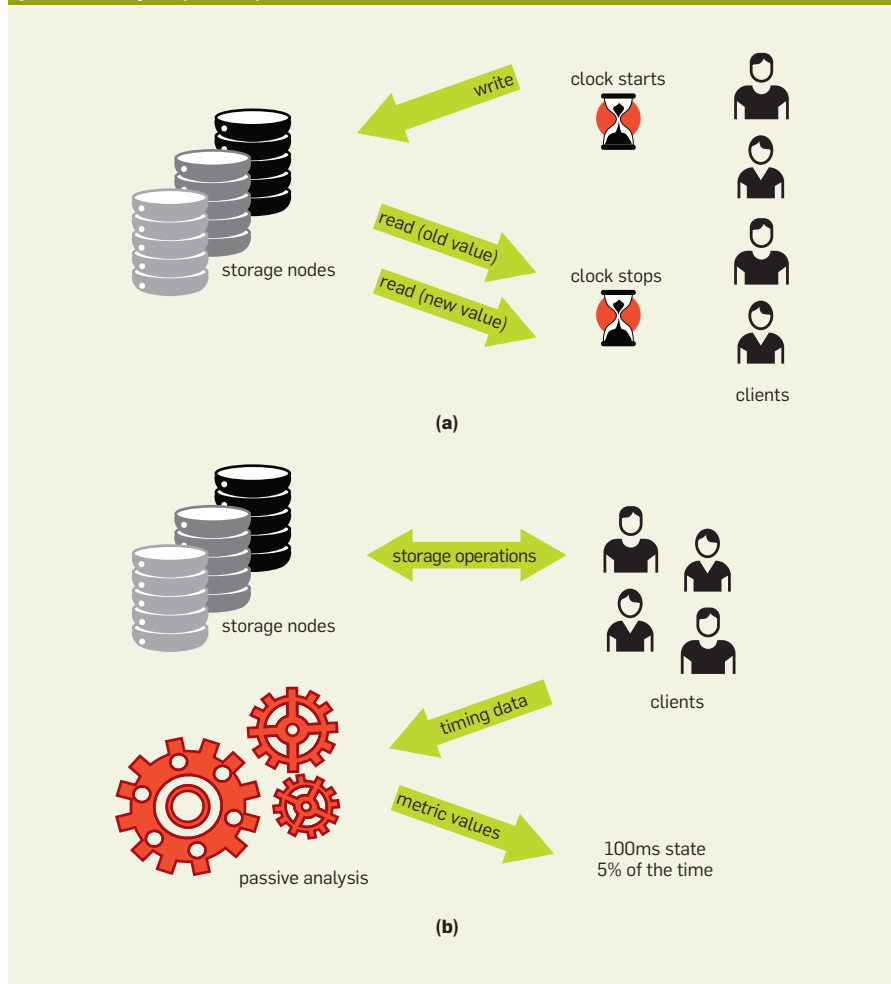
Active measurement underlies early studies of consistency in cloud storage systems. In this category of techniques, one client writes a new value to a key, and a different client then reads the same key repeatedly until the new value is returned. As in Vogels’s definition

of eventual consistency,²⁸ writes do not overlap in time with reads in this scenario. The time from the write to the last read that returns the old value—or alternatively, the first read of the new value—answers the question, “How eventual?” and can be regarded as an estimate of the *convergence time* of the replication protocol—the time needed to propagate a new value to all the replicas of an object.

At a technical level, the main challenge in active measurement is to determine the difference in time between operations executed at different client nodes—namely, a write and a read. Figure 2 illustrates how this difference can be computed using a collection of clients. To capture precisely the moment when the last replica receives the updated value, Hiroshi Wada et al. apply reads 50 times per second using one or more clients.²⁹ Bernbach et al. go one step further and use a collection of geographically distributed readers to ensure the reads hit all possible replicas.⁸ Staleness measurement in YCSB++ (Yahoo! Cloud-serving Benchmark)²³ follows a similar approach but uses a ZooKeeper producer-consumer queue¹⁹ to synchronize writers and readers. This approach circumvents issues related to clock skew but introduces additional latencies caused by queue operations, which may limit precision.

Active measurement can be used to discover the range of update propagation times in an eventually consistent system. For example, in experiments involving Amazon’s SimpleDB,⁴ Wada et al. report that convergence occurred in at most one second in more than 90% of the runs, but took more than four seconds in a few (less than 1%) cases. On the other hand, active measurement does not indicate what proportion of reads in a real workload will return stale values, as this quantity depends on how the data objects are accessed. For example, the proportion of stale reads would be expected to vary with the rate at which operations are applied on a given object—close to zero when operations are applied infrequently and the gaps between them exceed the convergence time, and larger when reads follow writes more closely. Active measurement does not separate these two cases, as it is based on a

Figure 2. Infrastructure for consistency measurement: active measurement (top), and passive analysis (bottom).



controlled workload designed to measure convergence time.

Passive analysis. In the earlier section on relaxed consistency properties, the discussion focused on ways of defining staleness in a precise and meaningful way and provided a hint of how a consistency metric can be derived from such definitions. It begins with a “black-or-white” consistency property, such as linearizability, which is either satisfied or not satisfied by a system or an execution trace; next, this property is relaxed by introducing a parameter that bounds the staleness of reads (for example, Δ -atomicity); the last step is to determine the parameter value that most accurately describes the system’s behavior (for example, Δ -atomic for $\Delta \geq 10$ ms). Passive analysis refers to this final step and entails examining the operations recorded in an execution trace to determine the order in which they appear to take effect.

The most immediate technical challenge in passive analysis is the collection of the trace, which records for each operation its type, start, and finish times, as well as its arguments and response (for example, a read of object X , starting at time t_1 and ending at t_2 , returning the value 5). This trace can be obtained at clients, as shown in Figure 2.

Because the trace is obtained by merging data from multiple clients or storage nodes over a finite period of time, it is prone to two types of anomalies: *dangling reads*, which return values that lack a corresponding write (that is, a write that assigns the value returned by the read); and *reversed operations*, whereby a read appears before its corresponding write in the trace. These anomalies must be removed if they occur; otherwise, the k -atomicity and Δ -atomicity properties are undefined and cannot be used for computing staleness.

Dangling reads indicate missing information, such as when the first access to an object in a trace is a read and the corresponding write occurs before the start of the trace. In this case the dangling read is identified easily and can be dropped from the trace with no ill effects. Reversed operations are equally easy to detect, but more difficult to remedy. Clock synchronization techniques such as atomic clocks and GPS¹⁰ can eliminate reversed operations altogether. Even ordinary Network Time

Protocol (NTP) is sufficient if the synchronization of clocks is tight enough. Alternatively, one can also estimate clock skew directly from reversed operations and adjust the trace accordingly.

Given a trace free from dangling reads and reversed operations, the next challenge is to compute staleness metrics. Efficient algorithms are critical in this context because the trace may be very long, which means that both the space and computation required is high. In fact, the problem of testing whether a trace is linearizable was shown by Gibbons and Korach to be intractable.¹³ Testing whether a trace is k -atomic or Δ -atomic for fixed k and Δ is at least as hard since these properties are equivalent to linearizability when $k = 1$ and $\Delta = 0$. Computing k and Δ from a trace is harder still, and hence also intractable.

Fortunately, the intractability result breaks in the special case when every write on a given object assigns a unique value. This condition is straightforward to enforce (for example, by embedding a unique token in each value written, such as a node ID and timestamp), which opens the door to efficient algorithms for computing staleness metrics from traces in practice. For Δ -atomicity, such an algorithm is known,¹⁵ and has been used to analyze time-based staleness in traces obtained using Cassandra.²⁴ For k -atomicity, an efficient algorithm is known only for deciding whether a trace is 2-atomic.¹⁴

Passive analysis in general is not limited to staleness metrics, as illustrated by the technique of *cycle analysis*,^{5,30} which detects linearizability violations by analyzing a *conflict graph* representing the execution trace. The absence of cycles in such a graph indicates the trace is linearizable, and so the number and length of such cycles can be defined as metrics for inconsistency.^{5,30} The relationship of these metrics to staleness is not known precisely.

Comparison and Discussion

Prediction, active measurement, and passive analysis are all useful ways to study eventual consistency. Although each method has specific strengths and weaknesses, it is difficult to compare them directly, as they meet different goals. Passive analysis is *client-*

centric in that it reflects the manner in which the client application interacts with the system. As a result, passive analysis can be used to compare staleness observed in two workloads applied to the same storage system. Active measurement, on the other hand, is *system-centric* in that it measures the convergence time of a system’s replication protocol for a controlled workload. Thus, active measurement is best suited for comparing different systems in terms of staleness, or the same system under different software or hardware configurations. The PBS framework⁶ is also designed around a controlled workload, and for that reason it is classified as system-centric.

The techniques discussed in this article also vary in terms of conceptual models of eventual consistency. Active measurement and PBS are based upon a *simplified model*, similar to Vogels’s definition,²⁸ in which writes occur before reads. In contrast, passive analysis considers a *general model* in which writes may overlap in time with reads and with other writes. As explained in another article by Golab et al.,¹⁶ storage systems may behave differently in these two models when clients follow the “ $R + W > N$ ” rule,²⁸ which ensures read and write operations access overlapping subsets of replicas. The latter condition has long been considered sufficient for “strong consistency,”^{6,28} and in fact guarantees linearizability in the simplified model but permits linearizability violations in the general model.

Despite the somewhat high cost of analyzing a detailed trace of operations, passive analysis plays an important role in understanding eventual consistency. Whereas any perspective on consistency is ultimately affected by the state of data in replicas in a storage system, which can be thought of as the “ground truth,” passive analysis is most closely tied to the actual consistency observed by client applications. Specifically, it reflects data-level anomalies only when they manifest themselves to clients—for example, as stale reads.

Future Work

For modern online service providers, small decreases in latency are known to have a measurable effect on user experi-

ence and, as a consequence, revenue.¹⁷ As weak consistency continues to gain traction as the means for reducing latency, the ability to reason clearly about this trade-off will be an important competitive advantage for service providers.

Consistency-aware pricing schemes and guarantees are already being adopted by industry. Amazon's DynamoDB supports differentiated pricing for consistent reads and eventually consistent reads, with the former guarantee costing the application developer twice as much.³ More recently, Terry et al. describe a system that allows client applications to select different grades of weak consistency, such as "at most 200-ms latency and at most 5-minute staleness," through novel SLAs (service-level agreements).²⁷ Other read guarantees, such as monotonic reads or causal consistency, can be requested, and a "wish list" of different combinations can be specified with different utilities. This opens the door to even more fine-grained pricing schemes, making it more important than ever for users to understand the utility of different points in consistency-related trade-offs.

Whereas the understanding of eventual consistency is expected to improve with additional empirical studies involving measurement, fundamental technical problems also remain to be solved. In particular, the problem of consistency *verification* remains an open and important challenge. Storage system designers need verification techniques to test whether their implementation correctly fulfills application-specified SLAs, and, likewise, application developers could use such tools to verify the service quality actually received.

One of the open problems related to verification algorithms is to determine the computational complexity of deciding k -atomicity. Existing algorithms handle only the special case $k < 3$,¹⁴ which limits their use in practice. Similar technical ideas can be applied when $k \geq 3$ but only for a restricted class of traces, and it is not known whether an efficient algorithm exists for deciding k -atomicity in the general case.

Conclusion

Eventual consistency is increasingly viewed as a spectrum of behaviors that can be quantified along various

dimensions, rather than a binary property that a storage system either satisfies or fails to satisfy. Advances in characterizing and verifying these behaviors will enable service providers to offer an increasingly rich set of service levels of differentiated performance, ultimately improving the end user's experience.

Acknowledgments

We are grateful to Jay J. Wylie, Indranil Gupta, and Doug Terry for their helpful feedback. C

Related articles on queue.acm.org

Eventually Consistent

Werner Vogels

<http://queue.acm.org/detail.cfm?id=1466448>

Eventual Consistency Today: Limitations, Extensions, and Beyond

Peter Bailis and Ali Ghodsi

<http://queue.acm.org/detail.cfm?id=2462076>

BASE: An Acid Alternative

Dan Pritchett

<http://queue.acm.org/detail.cfm?id=1394128>

References

- Abadi, D. Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. *IEEE Computer* 45, 2 (2012), 37–42.
- Aiyer, A., Alvisi, L. and Bazzi, R.A. On the availability of non-strict quorum systems. In *Proceedings of the 19th International Symposium on Distributed Computing*, (2005), 48–62.
- Amazon Web Services. DynamoDB; <http://aws.amazon.com/dynamodb/>.
- Amazon Web Services. SimpleDB; <http://aws.amazon.com/simpledb/>.
- Anderson, E., Li, X., Shah, M.A., Tucek, J. and Wylie, J.J. What consistency does your key-value store actually provide? In *Proceedings of the 6th Usenix Workshop on Hot Topics in System Dependability*, 2010.
- Bailis, P., Venkataraman, S., Franklin, M.J., Hellerstein, J.M. and Stoica, I. Probabilistically bounded staleness for practical partial quorums. In *Proceedings of the VLDB Endowment* 5, 8 (2012), 776–787.
- Bermbach, D., Sakr, S. and Zhao, L. Towards comprehensive measurement of consistency guarantees for cloud-hosted data storage services. In *Proceedings of the 5th TPC (Transaction Processing Performance Council) Technology Conference on Performance Evaluation and Benchmarking*, 2013.
- Bermbach, D. and Tai, S. Eventual consistency: how soon is eventual? An evaluation of Amazon S3's consistency behavior. In *Proceedings of the 6th Workshop on Middleware for Service-oriented Computing*, 2011.
- Brewer, E.A. Towards robust distributed systems (invited talk). In *Proceedings of the 19th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2000.
- Corbet, J.C., et al. Spanner: Google's globally distributed database. In *Proceedings of the 10th Usenix Conference on Operating Systems Design and Implementation*, (2012), 251–264.
- Davidson, A., Rubinstein, A., Todi, A., Bailis, P. and Venkataraman, S. Adaptive hybrid quorums in practical settings, 2012; http://www.cs.berkeley.edu/~kubitron/courses/cs262a-F12/projects/reports/project12_report_ver2.pdf.
- Decandia, G., Hastorun, D., Jampani, M. et al. Dynamo: Amazon's highly available key-value store. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles*, 2007.
- Gibbons, P.B. and Korach, E. Testing shared memories.

Society for Industrial and Applied Mathematics Journal on Computing 26, 4 (1997), 1208–1244.

- Golab, W., Hurwitz, J. and Li, X. On the k -atomicity-verification problem. In *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems*, 2013.
- Golab, W., Li, X. and Shah and M.A. Analyzing consistency properties for fun and profit. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, (2011), 197–206.
- Golab, W., Rahman, M.R., AuYoung, A., Keeton, K. and Gupta, I. Client-centric benchmarking of eventual consistency for cloud storage systems. Manuscript under submission, 2013.
- Hamilton, J. The cost of latency; <http://perspectives.mvdiirona.com/2009/10/31/TheCostOfLatency.aspx>.
- Herlihy, M. and Wing, J.M. Linearizability: a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems* 12, 3 (1990), 463–492.
- Hunt, P., Konar, M., Junqueira, F.P. and Reed, B. ZooKeeper: Wait-free coordination for Internet-scale systems. In *Proceedings of the 2010 Usenix Annual Technical Conference*, (2010), 11–25.
- Lakshman, A. and Malik, P. Cassandra: A decentralized structured storage system. *ACM SIGOPS Operating Systems Review* 44, 2 (2010), 35–40.
- Lamport, L. On interprocess communication. Part I: basic formalism; and Part II: algorithms. *Distributed Computing* 1, 2 (1986), 77–101.
- Lloyd, W., Freedman, M.J., Kaminsky, M. and Andersen, D.G. Don't settle for eventual: Scalable causal consistency for wide-area storage with COPS. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, (2011), 401–416.
- Patil, S., Polte, M., Ren, K., et al. YCSB++: benchmarking and performance debugging advanced features in scalable table stores. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, (2011), 9:1–9:14.
- Rahman, M.R., Golab, W., AuYoung, A., Keeton, K. and Wylie, J.J. Toward a principled framework for benchmarking consistency. In *Proceedings of the 8th Usenix Workshop on Hot Topics in System Dependability*, 2012.
- Terry, D. 2013. Replicated data consistency explained through baseball. *Communications of the ACM* 56, 12 (Dec. 2013), 82–89.
- Terry, D., Petersen, K., Spreitzer, M. and Theimer, M. The case for non-transparent replication: examples from Bayou. *IEEE Data Eng. Bulletin* 21 (1998), 12–20.
- Terry, D.B., Prabhakaran, V., Kotla, R., Balakrishnan, M., Aguilera, M.K. and Abu-Libdeh, H. Consistency-based service-level agreements for cloud storage. In *Proceedings of the 24th ACM Symposium on Operating Systems Principles*, 2013.
- Vogels, W. Eventually consistent. *ACM Queue* 6, 6 (2008), 14–19.
- Wada, H., Fekete, A., Zhao, L., Lee, K. and Liu, A. Data consistency properties and the trade-offs in commercial cloud storage: the consumers' perspective. In *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research*, 2011, 134–143.
- Zellag, K. and Kemme, B. How consistent is your cloud application? In *Proceedings of the 3rd ACM Symposium on Cloud Computing*, (2012), 1–14.

Wojciech Golab is an assistant professor in electrical and computer engineering at the University of Waterloo. Previously, he worked as a researcher in storage systems at Hewlett-Packard Labs in Palo Alto.

Muntasir Raihan Rahman is a computer science Ph.D. student in the Distributed Protocols Research Group at the University of Illinois at Urbana-Champaign.

Alvin AuYoung is a senior researcher at Hewlett-Packard Labs. He has also worked in areas such as resource scheduling and distributed graph processing.

Kimberly Keeton is a principal researcher at Hewlett-Packard Laboratories. She has also worked in areas such as storage management and intelligent storage.

Xiaozhou (Steve) Li is a software engineer at Google. Previously, he worked at Hewlett-Packard Labs as a researcher and at Microsoft as a software design engineer.

How can the expected interactions between caller and implementation be guaranteed?

BY ROBERT F. SPROULL AND JIM WALDO

The API Performance Contract

WHEN YOU CALL functions in an API, you expect them to work correctly; sometimes this expectation is called a *contract* between the caller and the implementation. Callers also have performance expectations about these functions, and often the success of a software system depends on the API meeting these expectations.

So there is a *performance contract* as well as a *correctness contract*. The performance contract is usually implicit, often vague, and sometimes breached (by caller or implementation). How can this aspect of API design and documentation be improved?

Any significant software system today depends on the work of others: sure, you write some code, but you call functions in the operating system and in a variety of software packages through APIs that cut down on the amount of code you must write. In some cases you even outsource work to distant servers hooked to you by cranky networks. You depend on these functions and services for correct operation, but you also depend on them for performing well enough so the overall system performs well. There is bound to be performance variation in complex systems that involve paging, net-

work delays, sharing resources such as disks, and so on. Even in simple setups such as a stand-alone computer with all program and data in memory, however, you can be surprised when an API or operating system does not meet performance expectations.

People are accustomed to speaking of the contract between an application and the implementation of an API to describe correct behavior when an API function is invoked. The caller must meet certain initial requirements, and the function is then required to perform as specified. (These dual responsibilities are like the pre- and post-conditions that appear in Floyd-Hoare logic for proving program correctness.) While today's API specifications do not make correctness criteria explicit in a way that leads to correctness proofs, the type declarations and textual documentation for an API function strive

to be unambiguous about its logical behavior. If a function is described as “add the element e to the end of list l ” where e and l are described by types in the calling sequence, then the caller knows what behavior to expect.

There is more to an API function than correctness, however. What resources does it consume, and how fast is it? People often make assumptions based on their own judgments of what the implementation of the function being performed should be. Adding an element to the end of a list should be “cheap.” Reversing a list might be “a linear function of list length.” For many simple functions, these intuitions suffice to avoid trouble—though not always, as described later in this article. For functions of any complexity, however, such as “draw string s in font f at (x,y) in window w ,” or “find the average of the values stored on a remote file,” there is room for surprise if not frustration at the performance. Further, the API documentation provides no hints about which functions are cheap and which are costly.

To complicate matters, after you have tuned your application to the performance characteristics of an API, a new version of the API implementation arrives or a new remote server stores the data file, and rather than overall performance increasing (the eternal hope for something new), it tanks. In short, the performance contracts that accompany the composition of software components in systems deserve more attention.

A Performance Taxonomy

Programmers begin building intuitive API performance models early on (see Sidebar 1). To be useful, the model need not be very accurate. Here is a simple taxonomy:

Always cheap (examples: `toupper`, `isdigit`, `java.util.HashMap.get`). The first two functions are always cheap, usually inlined table lookups. Lookup in a properly sized hash table is expected to be fast, but a hash collision may slow down an occasional access.

Usually cheap (examples: `fgetc`, `java.util.HashMap.put`). Many functions are designed to be fast most of the time but require occasionally invoking more complex code; `fgetc`

Some libraries offer more than one way of performing a function, usually because the alternatives have quite different performance.

must occasionally read a new character buffer. Storing a new item in a hash table may make the table so full the implementation enlarges it and rehashes all its entries.

The documentation for `java.util.HashMap` makes a commendable start at exposing a performance contract: “This implementation provides constant-time performance for the basic operations (`get` and `put`), assuming the hash function disperses the elements properly among the buckets. Iteration over collection views requires time proportional to the “capacity” of the `HashMap`...”³

The performance of `fgetc` depends on properties of the underlying stream. If it is a disk file, then the function will normally read from a user-memory buffer without requiring an operating-system call, but it must occasionally invoke the operating system to read a new buffer. If it is reading input from a keyboard, then the implementation probably invokes the operating system for each character read.

Predictable (examples: `qsort`, `regexec`). Performance of these functions varies with properties of their arguments, (for example, the size of an array to sort or the length of a string to search). These functions are often data-structure or common-algorithm utilities that use well-known algorithms and do not require system calls. You can usually judge performance based on expectations for the underlying algorithms (for example, that sort will take $n \log n$ time). When using complex data structures (for example, B-trees) or generic collections (where it may be difficult to identify the underlying concrete implementation), it may be more difficult to estimate performance. It is important to understand the predictability may be only probable; `regexec` is generally predictable based on its input, but there are pathological expressions that will cause exponential time explosions.

Unknown (examples: `fopen`, `fseek`, `pthread_create`, many “initialization” functions, and any call that traverses a network). These functions are not cheap and often have high variance in their performance. They allocate resources from pools (threads, memory, disk, operating system objects), often requiring exclusive access to shared

operating system or I/O resources. There is usually substantial initialization required. Calling across a network is always expensive (relative to local access), but the variation in the expense can be larger still, making the formation of a reasonable performance model much more difficult.

Threads libraries are easy marks for performance issues. The Posix standard took many years to settle down, and implementations are still plagued with problems.⁶ Portability of threaded applications remains dicey. Some of the reasons for difficulties with threads are: the need for tight integration with operating systems, almost all of which (including Unix and Linux) were not designed originally with threads in mind; interaction with other libraries, especially making functions thread-safe and dealing with performance problems thereby induced; and several different design points for threads implementations, loosely characterized as lightweight and heavyweight.

Segmenting APIs By Performance

Some libraries offer more than one way of performing a function, usually because the alternatives have quite different performance.

Returning to sidebar 1, note that most programmers are told that using a library function to fetch each character is not the fastest way (even if the code for the function is inlined to avoid the function-call overhead). The more performance-conscious will read a very large array of characters and extract each one using array or pointer operations in the programming language. In extreme cases the application can map pages of the file into memory pages to avoid copying data into an array. In return for performance, these functions put a larger burden on the caller (for example, to get buffer arithmetic correct and to make the implementation consistent with other library calls such as a call to `fseek` that would require adjusting buffer pointers and perhaps contents).

Programmers are always counseled to avoid premature optimization in their programs, thus delaying extreme revisions until simpler ones have proven inadequate. The only way to be sure about performance is to measure

SIDEBAR 1.

Forming Your Own Model of Performance

In learning to program, you very early on acquire rules of thumb about performance. The pseudocode below is a pattern for reasonably efficient processing of a modest-size file of characters:

```
fs = fopen("~/dan/weather-data.txt", "r"); //(1)
for ( i=0; i<10000; i++) {
    ch = fgetc(fs); // (2)
    // process character ch
}
. . .
```

The function call (1) is expected to take a while, but the call to get a character (2) is expected to be *cheap*. This makes intuitive sense: to process a file, a stream need be opened only once, but the “get the next character” function will be called often, perhaps thousands or millions of times.

These two stream functions are implemented by a library. The documentation for the library² clearly states what these functions do—an informal presentation of the correctness contract between the implementation and the application. There is no mention of performance, nor any hint to the programmer that the two functions differ substantially in performance. Therefore, programmers build models of performance based on experience, not specifications.⁷

Not all functions induce obvious performance properties. For example:

```
fseek(fs, ptr, SEEK_SET); //(3)
```

This function might be cheap when the target file data is already in a buffer. In the general case, it will involve an operating-system call and perhaps I/O. In a wild case, it might require reeling off several thousand feet of magnetic tape. It is also possible that the library function is not cheap even in the simple case: the implementation may simply store the pointer and set a flag that will cause the hard work to be done on the next stream call that reads or writes data, thus pushing the performance uncertainty onto an otherwise-cheap function.

SIDEBAR 2.

Function Creep

```
SetFontSize(f, 10);
SetDrawPosition(w, 200, 20);
DrawText(w, f, "This is a passage.");
```

This example is from a fictitious window system: it sets a font size and a drawing position, then draws some text. You might expect all of these functions to be quite cheap, because rendering text windows on a screen should be fast. Indeed, in early window systems, such as QuickDraw on the Macintosh, you would be right.

The functionality of today’s window systems has, rightly, crept upward. Character rasters—even those rendered on a display—are computed from geometric outlines, so preparing a font of a given size may take a while. Modern window systems speed up rendering text by generous use of caching—even saving prepared rasters on disk so that they are available to multiple applications and are preserved across system restarts. Application programmers, however, have no idea whether the fonts they are requesting have been prepared and cached, whether `SetFontSize` will do the (considerable) computation for all of a font’s characters, or whether characters will be converted one at a time as needed by strings passed to `DrawText`.

Augmented functionality may also allow storing font data on network-attached file servers, so font access may take a while, or even fail after a long timeout if the server does not respond.

In many cases the details will not matter, but if the delays are substantial or highly variable, the application might choose to prepare during its initialization all the fonts it plans to use. Most window-system APIs do not have such a function.

it. Programmers usually write the entire program before confronting mismatches between performance expectations (or estimates) and the reality delivered by an implementation.

Performance variation. The performance of “predictable” functions can be estimated from properties of their arguments. The “unknown” func-

tions may also vary greatly depending on what they are asked to do. The time required to open a stream on a storage device will certainly depend on access times, and perhaps data-transfer rates, of the underlying device. Storage accessed via network protocols may be especially costly; certainly it will be variable.

SIDEBAR 3.

When the Performance Contract Breaks

```
SetColor(g, COLOR_RED);
DrawLine(g, 100, 200, 200, 600);
```

Graphics library functions are usually fast. It is reasonable to assume that you draw many lines of varied colors by setting the color for each line; if you are worried that the `SetColor` function is not cheap, you might call it only when the color changes. At one point, the graphics hardware offering of a workstation company required clearing the (hardware) geometry pipeline to change a line’s color, so color change was a costly operation. To be fast, the application was, for example, forced to sort lines by color and draw all red lines together. This was not the intuitive way to write the program, and it forced a major change to the structure and programming of the application.

The expensive `SetColor` operation came to be understood as a mistake and was fixed in subsequent hardware. The code that had been written to overcome the mistake, however, may remain, becoming an inexplicable complexity for anyone maintaining the code who does not know the performance history of the hardware.

SIDEBAR 4.

The Long Story of `malloc` and Dynamic Memory Allocation

It would be nice if dynamic memory allocation with the `malloc()` function could be characterized as “usually cheap,” but that would be wrong since memory allocation—and `malloc` in particular—is one of the first suspects when programmers start hunting for performance problems. As part of their education in performance intuition, programmers learn that if they are calling `malloc` tens of thousands of times, especially to allocate small fixed-size blocks, they are better off allocating a single larger chunk of memory with `malloc`, chopping it up into the fixed-size blocks, and managing their own list of free blocks.

Implementers of `malloc` have struggled over the years to make it usually cheap in the presence of wide variations in usage patterns and in properties of the hardware/software system on which it runs.⁴ Systems that offer virtual memory, threading, and very large memories all present challenges to “cheap,” and `malloc`—along with its complement, `free()`—must trade off efficiency and evils of certain usage patterns such as memory fragmentation.⁸

Some software systems, such as Lisp and Java, use automatic memory allocation along with garbage collection to manage free storage. While this is a great convenience, a programmer concerned with performance must be aware of the costs. A Java programmer, for example, should be taught early about the difference between the `String` object, which can be modified only by making a new copy in new memory, and a `StringBuffer` object, which contains space to accommodate lengthening the string. As garbage-collection systems improve, they make the unpredictable pauses for garbage collection less common; this can lure the programmer into complacency, believing the automated reclamation of memory will never be a performance problem, when in fact it is only less often a performance problem.

Many cheap functions are cheap only *most* of the time, or they have an expected cheap cost. A “get a character” routine must occasionally refill a buffer using an operating system call that always takes much longer than fetching a character from a full buffer—and might occasionally take a very long time indeed (for example, reading a file on a heavily loaded file server or from a disk that is dying and will succeed only after multiple read retries).

The functions with “unknown” performance are likely to exhibit wide performance variations for a variety of reasons. One reason is function creep (see sidebar 2), where a generic function becomes more powerful over time. I/O streams are a good example: a call to open a stream invokes very different code in libraries and operating systems depending on the type of stream being opened (local disk file, network-served file, pipe, network stream, string in memory, and so on). As the range of I/O devices and types of files expands, the variance of the performance can only increase. The common life cycle of most APIs—to add functionality incrementally over time—inevitably increases performance variations.

A large source of variation is differences between ports of libraries to different platforms. Of course, the underlying speed of the platform—both hardware and operating system—will vary, but library ports can lead to changes in the *relative* performance of functions within an API or performance across APIs. It’s not uncommon for a quick-and-dirty initial port to have many performance problems, which are gradually fixed. Some libraries, such as those for handling threads, have notoriously wide porting-performance variation. Thread anomalies can surface as extreme behaviors—an impossibly slow application or even deadlock.

These variations are one reason why constructing precise performance contracts is difficult. It is usually not necessary to know performance with great precision, but extreme variations from expected behavior can cause problems.

Failure performance. The specifications for an API include details of behavior when a call fails. Returning an

error code and throwing an exception are common ways of telling the caller the function did not succeed. As with specifications for normal behavior, however, the *performance* of the failure is not specified. Here are three important cases:

► *Fail fast.* A call fails quickly—as fast or faster than its normal behavior. Calling `sqrt(-1)` fails fast. Even when a `malloc` call fails because no more memory is available, the call should return about as fast as any `malloc` call that must request more memory from the operating system. A call to open a stream for reading a nonexistent disk file is likely to return about as fast as a successful call.

► *Fail slow.* Sometimes a call fails very slowly—so slowly the application program might have wanted to proceed in other ways. For example, a request to open a network connection to another computer may fail only after several long time-outs expire.

► *Fail forever.* Sometimes a call simply stalls and does not allow the application program to proceed at all. For example, a call whose implementation waits on a synchronization lock that is never released may never return.

Intuition about failure performance is rarely as good as that for normal performance. One reason is simply that writing, debugging, and tuning programs provides far less experience with failure events than with normal events. Another is that a function call can fail in many, many ways, some of them fatal, and not all described in the API's specs. Even exception mechanisms, which are intended to describe more precisely the handling of errors, do not make all possible exceptions visible. Moreover, as library functionality increases, so do the opportunities for failure. For example, APIs that wrap network services (ODBC, JDBC, UPnP, ...) intrinsically subscribe to the vast array of network-failure mechanisms.

A diligent application programmer uses massive artillery to deal with unlikely failures. A common technique is to surround rather large parts of a program with `try...catch` blocks that can retry whole sections that fail. Interactive programs can try to save a user's work using a giant `try...catch` around the entire program, the effect

of which is to mitigate failure of the main program by saving, in a disk file, key logs, or data structures that record the effects of the work done by the user before the failure.

The only way to deal with stalls or deadlocks is to set up a watchdog thread, which expects a properly running application program to check in periodically with the watchdog, saying, in effect, "I'm still running properly." If too much time elapses between check-ins, the watchdog takes action—for example, saving state, aborting the main thread, and restarting the entire application. If an interactive program responds to a user's command by calling functions that may fail slowly, it may use a watchdog to abort the entire command and return to a known state that allows the user to proceed with other commands. This gives rise to a defensive programming style that plans for the possible abortion of every command.

Why Does the Performance Contract Matter?

Why must an API adhere to a performance contract? *Because major structures of an application program may depend on adherence to such a contract.* A programmer chooses APIs, data structures, and overall program structures based in part on expectations of API performance. If the expectations or the performance are grossly wrong, then the programmer cannot recover merely by tuning API calls but must rewrite large, possibly major, portions of the program. The defensive structure of the interactive program previously mentioned is another example.

In effect, a serious violation of the performance contract leads to a composition failure: a program written to the contract cannot be mated to (composed with) an implementation that fails to uphold the contract.

Of course, there are many programs whose structure and performance are influenced very little by library performance (scientific computations and large simulations are often in this category); however, much of today's "routine IT," especially the software that pervades Web-based services, makes extensive use of libraries whose performance is vital to overall performance.

Even small variations in perfor-

mance can cause major changes in the perception of the program by its users. This is especially true in programs dealing with various kinds of media. Dropping occasional frames of a video stream might be acceptable (indeed, more acceptable than allowing the frame rate to lag other media), but humans have evolved to detect even slight dropouts in audio, so minor changes in the performance of that type of media may have major impacts on the acceptability of the overall program. Such worries have led to considerable interest in notions of quality of service, which in many ways is the attempt to ensure performance at a high level.

How can contract violations be avoided? Although performance-contract violations are rare and rarely catastrophic, paying attention to the role of performance when using a software library can help lead to more robust software. Here are some precautions and strategies to use:

1. *Choose APIs and program structures carefully.* If you have the luxury of writing a program from scratch, ponder performance-contract implications as you begin. If the program starts out as a prototype and then remains in service for a while, it will undoubtedly be rewritten at least once; a rewrite is an opportunity to rethink API and structure choices.

2. *API implementers have an obligation to present a consistent performance contract as new versions and ports are released.* Even a new experimental API will garner users who will begin to derive a performance model of the API. Thereafter, changing the performance contract will surely irritate developers and may cause them to rewrite their programs.

Once an API is mature, it is essential the performance contract not change. In fact, the most universal APIs (for example, `libc`) arguably have become so in part because their performance contracts have been stable during the evolution of the API. The same goes for ports of APIs.

One could hope that API implementers might routinely test new versions to verify they have introduced no performance quirks. Unfortunately, such testing is rarely done. This does not mean, however, that you cannot


have your own tests for the parts of an API you depend on. With a profiler, a program can often be found to rely on a small number of APIs. Writing a performance test suite that will compare new versions of a library against the recorded performance of earlier versions can give programmers an early warning the performance of their own code is going to change with the release of the new library.

Many programmers expect computers and their software to get faster with time—*uniformly*. That is, they expect each new release of a library or a computer system to scale up performance of all API functions equally, especially the “cheap” ones. This is in fact very difficult for vendors to guarantee, but it so closely describes actual practice that customers believe it. Many workstation customers expected newer versions of graphics libraries, drivers, and hardware to increase performance of *all* graphics applications, but they were equally keen on functionality improvements of many kinds, which usually reduce performance of older functions, even if only slightly.


One could also hope that API specifications would make the performance contract explicit, so that those using, modifying, or porting the code would honor the contract. Note that a function’s use of dynamic memory allocation, whether implicit or automatic, should be part of this documentation.

3. *Defensive programming can help.* A programmer can use special care when invoking API functions with unknown or highly variable performance; this is especially true for considerations of failure performance. You can move initializations outside performance-critical areas and try to warm up any cached data an API may use (for example, fonts). APIs that exhibit large performance variance or have a lot of internally cached data could help by providing functions for passing “hints” from the application to the API about how to allocate or initialize these structures. Occasional pings to servers that are known to be contacted can establish a list of those that might be unavailable, allowing some long failure pauses to be avoided.

One technique sometimes used in graphics applications is to do a dry run,



Many programmers expect computers and their software to get faster with time—uniformly.



rendering a window of graphics into an off-screen (not visible) window, merely to warm up caches of fonts and other graphics data structures.

4. *Tune parameters exposed by the API.* Some libraries offer explicit ways of influencing performance (for example, controlling the size of buffers allocated for files, the initial sizes of tables, or the size of a cache). Operating systems also provide tuning options. Adjusting these parameters can improve performance *within* the confines of a performance contract; tuning does not remedy gross problems but can mitigate otherwise fixed choices embedded in libraries that heavily influence performance.

Some libraries provide alternative implementations of functions with identical semantics, usually in the form of concrete implementations of generic APIs. Tuning by choosing the best concrete implementation is usually very easy. The Java Collections package is a good example of this structure.

Increasingly, APIs are designed to adapt to usage on the fly, freeing the programmer from choosing the best parameter settings. If a hash table gets too full, it is automatically expanded and rehashed (a virtue balanced by, alas, the performance hit of the occasional expansions). If a file is being read sequentially, then more buffers can be allocated so that it is read in larger chunks.

5. *Measure performance to verify assumptions.* Common advice to programmers is to instrument key data structures to determine whether each structure is being used correctly. For example, you might measure how full a hash table is or how often hash collisions occur. Or you might verify that a structure designed to be fast for reading at the expense of write performance is actually being read more than written.

Adding sufficient instrumentation to measure the performance of many API calls accurately is difficult, a large amount of work, and probably not worth the information it yields. Adding instrumentation on those API calls that are central to the performance of an application (assuming you can identify them and your identification is correct), however, can save a lot of time when

problems arise. Note that much of this code can be reused as part of the performance monitor for the next release of the library, as mentioned earlier.

None of this is meant to discourage dreamers from working on tools that would automate such instrumentation and measurement, or on ways to specify performance contracts so that performance measurements can establish compliance with the contract. These are not easy goals, and the return may not be huge.

It is often possible to make performance measurements without having instrumented the software in advance (for example, by using profilers or tools such as DTrace⁵). These have the advantage of not requiring any work until there is a problem to track down. They can also help diagnose problems that creep in when modifications to your code or to libraries upset performance. As *Programming Pearls* author Jon Bentley recommends, “Profile routinely; measure performance drift from a trusted base.”¹

6. *Use logs: Detect and record anomalies.* Increasingly, violations of performance contracts appear in the field when distributed services are composed to form a complex system. (Note that major services offered via network interfaces sometimes have SLAs [service-level agreements] that specify acceptable performance. In many configurations, a measurement process occasionally issues service requests to check the SLA is met.) Since these services are invoked over a network connection using methods akin to API function calls (for example, remote procedure call or its variants such as XML-RPC, SOAP, or REST), the expectation of a performance contract applies. Applications detect failure of these services and often adapt gracefully. Slow response, however, especially when there are dozens of such services that depend on each other, can destroy system performance very quickly. Professionally managed services environments, such as those that provide Web services at major Internet sites, have elaborate instrumentation and tooling to monitor Web-service performance and to tackle problems that arise. The far more modest computer collections in homes also depend on such services—many that are

part of the operating system on each laptop computer and some that are embedded in appliances on the network (for example, a printer, network file system, or file backup service)—but there are no aids to help detect and deal with performance problems.

It would be helpful if clients of these services made note of the performance they expect and produce log entries that help diagnose problems (this is what `syslog` is for). When your file backup seems unreasonably slow (one hour to back up 200MB), is it slower than yesterday? Slower than it was before the latest operating system software update? Given that several computers may be sharing the backup device, is it slower than you should expect? Or is there some logical explanation (for example, the backup system finds a damaged data structure and embarks on a long procedure to rebuild it)?

Diagnosing performance problems in compositions of opaque software (where no source code is available, and there are no details of the modules and APIs that form the composition) requires the software play a role in reporting performance and detecting problems. While you cannot address performance problems within the software itself (it is opaque), you can make adjustments or repairs to operating systems and the network. If the backup device is slow because its disk is almost full, then you can surely add more disk space. Good logs and associated tools would help; sadly, logs are an undervalued and neglected area of computer system evolution.

Making Composition Work

Today’s software systems depend on composing independently developed components in such a way that they work—meaning they perform the desired computations at acceptable speed. The dream of statically checking a composition to guarantee the composition will be correct (“correct by composition”) remains elusive. Instead, software-engineering practice has developed methods for testing components and compositions that work pretty well. Every time an application binds to a dynamic library or launches atop an operating-system interface, the correctness of the composition is required.

Despite its importance, the performance of a composition—whether the client and the provider of an interface adhere to the performance contract between them—has been given short shrift. True, this contract is not as important as the correctness contract, but the full power of composition depends on it.

Acknowledgments

Thanks to Butler Lampson, who coined the term *cheap* to describe functions whose performance is fast enough to dispel all attempts to optimize them away; it has a wonderful ring of “economical yet serviceable,” Eric Allman, for his helpful comments, and Jon Bentley, an inveterate performance debugger. □

Related articles on queue.acm.org

API Design Matters

Michi Henning

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

Revisiting Network I/O APIs: The netmap Framework

Luigi Rizzo

<http://queue.acm.org/detail.cfm?id=2103536>

Passing a Language through the Eye of a Needle

Roberto Ierusalimsky, Luiz Henrique de Figueiredo and Waldemar Celes

<http://queue.acm.org/detail.cfm?id=1983083>

References

- Bentley, J. Personal communication.
- GNU C Library; http://www.gnu.org/software/libc/manual/html_node/index.html.
- Java Platform, Standard Edition 7. API Specification; <http://docs.oracle.com/javase/7/docs/api/index.html>.
- Korn, D.G., Vo, K.-P. In search of a better malloc. In *Proceedings of the Summer '85 Usenix Conference*, 489–506.
- Oracle. Solaris Dynamic Tracing Guide; <http://docs.oracle.com/cd/E19253-01/817-6223/>.
- Pthreads(7) manual page; <http://www.kernel.org/doc/man-pages/online/pages/man7/pthreads.7.html>; <http://man7.org/linux/man-pages/man7/pthreads.7.html>.
- Saltzer, J.H., Kaashoek, M.F. Principle of least astonishment. In *Principles of Computer System Design*. Morgan Kaufmann, 2009, 85.
- Vo, K.-P. Vmalloc: a general and efficient memory allocator. *Software Practice and Experience* 26, 3 (1996), 357–374; <http://www2.research.att.com/~astopen/download/ref/vmalloc/vmalloc-spe.pdf>.

Robert F. Sproull is an adjunct faculty member of the computer science department at the University of Massachusetts (Amherst), a position he assumed after retiring as director of Sun Microsystems Laboratories.

Jim Waldo is a professor of the practice of computer science at Harvard University, where he is also the chief technology officer, a position he assumed after leaving Sun Microsystems Laboratories.

© 2014 ACM 0001-0782/14/03 \$15.00

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

Enabling existing lock-based programs to achieve performance benefits of nonblocking synchronization.

BY ANDI KLEEN

Scaling Existing Lock-based Applications with Lock Elision

MULTITHREADED APPLICATIONS TAKE advantage of increasing core counts to achieve high performance. Such programs, however, typically require programmers to reason about data shared among multiple threads. Programmers use synchronization mechanisms such as mutual-exclusion locks to ensure correct updates to shared data in the presence of accesses from multiple threads. Unfortunately,

these mechanisms serialize thread accesses to the data and limit scalability.

Often, lock-based programs do not scale because of the long block times caused by serialization, as well as the excessive communication overhead to coordinate synchronization. To reduce the impact on scalability, programmers use fine-grained locking, where instead of using a few locks to protect all shared data (coarse granularity), they use many locks to protect data at a finer granularity. This is a complex and error-prone process^{11,12} that also often impacts single-thread performance.

Extensive work exists to improve synchronization performance. Lock-free data structures support concurrent operations without mutually exclusive locks.¹³ Such algorithms are often quite complex.

A rich variety of locking algorithms of varying complexity exist.¹¹ Other approaches such as optimistic locking avoid synchronization overhead—for example, by using sequence numbers to protect reading shared data and retrying the accesses if necessary. While effective under certain conditions, extending this to be generally usable is quite difficult.

Transactional memory⁵ proposed hardware mechanisms to simplify the development of lock-free data structures. They rely on mechanisms other than locks for forward progress and exploit the underlying cache-coherence mechanisms to detect conflict among threads.

Lock elision¹⁵ was another proposal to expose concurrency in lock-based programs by executing them in a lock-less fast path. It uses the hardware capability of modern processors and the underlying cache-coherence protocol to execute critical sections optimistically, without acquiring a lock. The lock is acquired only when actually required to resolve a data conflict.

In spite of the numerous proposals, high-performance synchronization remains difficult: programmers must use information known in advance to determine when to serialize, and scal-



able locking is complex, leading to conservative lock placement.

Recently, commercial processors from Intel Corporation and IBM have introduced hardware support to improve synchronization.^{6,7,9,17} This presents a unique opportunity for software developers to improve scalability of their software.

In this article, the focus is on improving the existing lock-based programming model, and thus, looks at lock elision as the primary usage model.

Hardware Support For Lock Elision

Programmers must decide at development time how to control access to shared data—whether it is using coarse-grained locks where a few locks protect all data or fine-grained locks to protect different data. The dynamic behavior eventually determines sharing patterns. Finding errors with

incorrectly used synchronization is quite difficult.

What is needed is a mechanism that has the usability of coarse-grained locks with the scalability of fine-grained locks. This is exactly what lock elision provides. The programmer must still use locks to protect shared data but can adopt a more coarse-grained approach. The hardware determines dynamically whether threads need to serialize in a critical section and executes the lock-based programs in a lock-free manner, where possible.

The processor executes lock-protected critical sections (called transactional regions) transactionally. Such an execution only reads the lock; it does not acquire or write to it, thus exposing concurrency. Because the lock is elided and the execution optimistic, the hardware buffers any updates and checks for conflicts with other threads.

During the execution, the hardware does not make any updates visible to other threads.

On a successful transactional execution, the hardware atomically commits the updates to ensure all memory operations appear to occur instantaneously when viewed from other processors. This atomic commit allows the execution to occur optimistically without acquiring the lock; the execution, instead of relying on the lock, now relies on hardware to ensure correct updates. If synchronization is unnecessary, the execution can commit without any cross-threaded serialization.

A transactional execution may be unsuccessful because of abort conditions such as data conflicts with other threads. When this happens, the processor will do a transactional abort. This means the processor discards all updates performed in the region,

restores the architectural state to appear as if the optimistic execution never occurred, and resumes execution nontransactionally. Depending on the policy in place, the execution may retry lock elision or skip it and acquire the lock.

To ensure an atomic commit, the hardware must be able to detect violations of atomicity. It does this by maintaining a read and a write set for the transactional region. The read set consists of addresses read from within the transactional region; and the write set consists of addresses written to, from within the same region.

A conflicting data access occurs if another logical processor reads a location that is part of the transactional region's write-set or writes a location that is a part of the read- or write-set of the transactional region. This is referred to as a *data conflict*.

Transactional aborts may also occur as a result of limited transactional resources. For example, the amount of data accessed in the region may exceed the buffering capacity. Some operations that cannot be transactionally executed, such as I/O, always cause aborts.

Intel TSX

Intel Transactional Synchronization Extensions (TSX) provides two instruction-set interfaces to define transaction regions. The first is Hardware Lock Elision (HLE), which uses legacy compatible instruction prefixes to enable lock elision for existing atomic-lock instructions. The other is Restricted Transactional Memory (RTM), which provides new XBEGIN and XEND instructions to control transactional execution and supports an explicit fallback handler.

Programmers who want to run Intel TSX-enabled software on legacy hardware could use the HLE interface to implement lock elision. On the other hand, with more complex locking primitives or when more flexibility is needed, the RTM interface can be used to implement lock elision.

This article focuses on the RTM interface in TSX. IBM systems provide instructions roughly similar to RTM.^{6,9,17}

Fallback handlers. To use the RTM interface, programmers add a lock-elision wrapper to the synchronization routines. Instead of acquiring the lock, the wrapper uses the XBEGIN instruction and falls back to the lock if the transaction aborts and retries do not succeed.

The Intel TSX architecture (and others, except IBM zSeries, which has limited support for guaranteed small transactions⁹) does not guarantee that a transactional execution will ever succeed. Software must have a fallback nontransactional path to execute on a transactional abort. Transactional execution does not directly enable new algorithms but improves the performance of existing ones. The fallback code must have the capability of ensuring eventual forward progress; it must not simply keep retrying the transactional execution forever.

To implement lock elision and improve existing lock-based programming models, the programmers would test the lock within the transactional region and acquire the lock in the nontransactional fallback path, and then reexecute the critical region. This enables a classic lock-based programming model.

Alternatively, the programmer may use the transactional support to improve the performance and implementation of existing nonblocking heavyweight algorithms, using a fast and simple path to execute transactionally, but a slower and more complex nonblocking path if the transactional execution aborts. The programmer must ensure proper interaction between the two paths.

The programmer may also use the transactional support for implementing new programming models such as transactional memory. One approach uses the transactional support in hardware for a fast path, with an STM

Figure 1. Lock elision.

Elided lock:

```
/* Start transactional region. On abort we come back here. */
if (_xbegin() == _XBEGIN_STARTED) {
    /* Put lock into read-set and abort if lock is busy */
    if (lock variable is not free)
        _xabort(_XABORT_LOCK_BUSY);
} else {
    /* Fallback path */
    /* Come here when abort or lock not free */
    lock lock;
}
/* Execute critical region either transaction or with lock */
```

Elided unlock:

```
/* Critical region ends */
/* Was this lock elided? */
if (lock is free)
    _xend();
else
    unlock lock
```

Figure 2. An example synchronization of transactions through the lock variable.

```
Thread 1                Thread 2
Transaction A
Read Lock X

                        Transaction B
                        Read Lock X
                        ... abort for some reason ...
                        Fallback handler
                        Writes to lock X to acquire
* DATA CONFLICT ON LOCK *

Abort
Waits on lock to become free
```


(software transactional memory) implementation for the fallback path² if the large overhead of STM on fallback is acceptable.¹

Implementing lock elision. Lock elision uses a simple transactional wrapper around existing lock code, as shown in Figure 1.

`_xbegin()` tries to execute the critical section transactionally. If the execution does not commit, then `_xbegin` returns a status word different from `_XBEGIN_STARTED`, indicating the abort cause. After doing some retries, the program can then acquire the lock.

On unlock, the code assumes that if the lock is free, then the critical region was elided, and it commits with `_xend()`. (This assumes the program does not unlock free locks). Otherwise, the lock is unlocked normally. This is a simplified (but functional) example. A practical implementation would likely implement some more optimizations to improve transaction success. GNU Compiler Collection (GCC) provides detailed documentation for the intrinsics used here.⁴ GitHub has a reference implementation of the TSX intrinsics.¹⁰

A thread that keeps aborting and goes to the fallback handler eventually acquires the lock. All threads speculating on the same lock have the lock in their read-set and abort their transactional execution when they detect a write conflict on the lock variable. If that happens, then eventually they will all take the fallback path. This synchronization mechanism between fallback path and speculative execution is important to avoid races and preserve the locking semantics.

This also requires that the lock variable is visible to the elision wrapper, as shown in Figure 2.

Enabling lock elision in existing programs. Most existing applications use synchronization libraries to implement locks. Adding elision support to only these libraries is often sufficient to enable lock elision for these applications. This may be as simple as adding an elision wrapper in the existing library code. The application does not need changes for this step, as lock elision maintains the lock-based programming model.

We implemented lock elision for

the POSIX standard pthread mutexes using Intel TSX and integrated it into Linux glibc 2.18. An implementation to elide pthread read/write locks has not yet been integrated into glibc.

The glibc mutex interface is binary compatible. This allows existing programs, using pthread mutexes for locking, to benefit from elision. Similarly, other lock libraries can be enabled for elision.

The next step is to evaluate performance and analyze transactional execution. Such an analysis may suggest changes to applications to make them more elision friendly. Such changes also typically improve performance without elision, by avoiding unnecessary communication among cores.

Analyzing transactional execution.

The behavior of explicit speculation is different from existing programming models. An effective way of analyzing transactional execution is with a transaction-aware profiler using hardware performance-monitoring counters.

Intel TSX provides extensive support to monitor performance, including sampling, abort rates, abort-cause profiling, and cycle-level analysis for successful and unsuccessful transactional executions. Often, the abort rates can be significantly reduced through minor changes to application data structures or code (for example, by avoiding false sharing).

The performance-monitoring infrastructure provides information about hardware behavior that is not directly visible to programmers. This is often critical for performance tuning. Programmers can also instrument transactional regions to understand their behavior. This provides only a limited

picture, however, because transactional aborts discard updates, or the instrumentation itself may contribute to aborts.

Elision Results

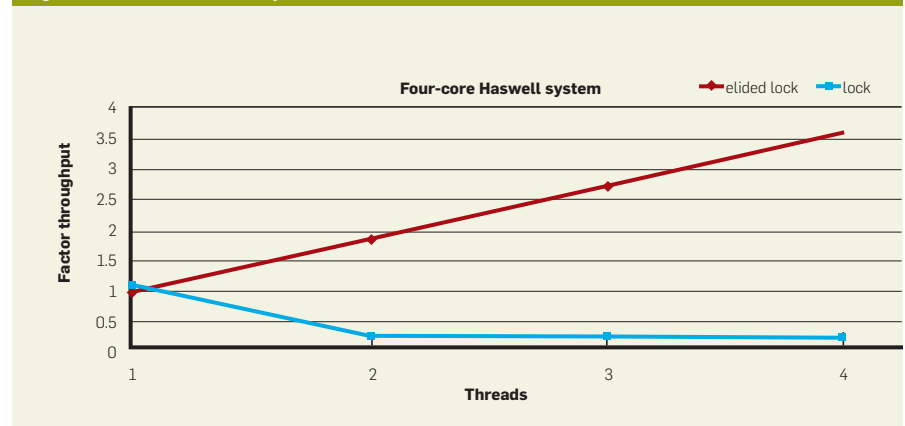
Figure 3 compares the average number of operations per second when reading a shared hash table protected with a global lock versus an elided global lock. The tests were run on a four-core Intel Core (Haswell) system without SMT. The elided lock provides near-perfect scaling from one to four cores, while the global lock degrades quickly. A recent paper¹⁸ analyzes lock elision using Intel TSX for larger applications and reports compelling benefits for a wide range of critical section types.

When does elision help? Data structures that do not encounter significant data conflicts are excellent candidates for elision. An elided lock can allow multiple readers and nonconflicting writers to execute concurrently. This is because lock elision does not result in any write operations on the lock, thus behaving as a reader-writer lock. It also eliminates any cache-line communication of the lock variable. As a result, programs with fine-grained locking may also see gains.

Not all programs benefit from lock improvements. These programs are either single threaded, do not use contended locks, or use some other algorithm for synchronization. Data structures that repeatedly conflict do not see a concurrency benefit.

If the program uses a complex non-blocking or fine-grained algorithm, then a simpler transactional fast path can speed it up. Further, some pro-

Figure 3. Hash table lookup with elision.



grams using system calls or similar unfriendly operations inside their critical section will see repeated aborts.

The cost of aborts. Not all transactional aborts make the program slower. The abort may occur where otherwise the thread would simply have been waiting for a lock. Such transactional regions that abort may also serve to prefetch data. Frequent, persistent aborts hurt performance, however, and developers must analyze aborts to reduce their probability.

Adaptive elision. Synchronization libraries can adapt their elision policies according to transactional behavior. For example, the glibc implementation uses a simple adaptive algorithm to skip elision as needed. The synchronization library wrapper detects transactional aborts and disables lock elision on unsuccessful transactional execution. The library reenables elision for the lock after a period of time, in case the situation has changed. Developing innovative adaptive heuristics is an important area of future work.^{14,16}

Adaptive elision combined with elision wrappers added to existing synchronization libraries can effectively enable lock elision seamlessly for all locks in existing programs. Developers should still use profiling to identify causes for transactional aborts, however, and address the expensive ones. That remains the most effective way to improve performance.

As an alternative to building adaptivity into the elision library, programmers may selectively identify which locks to apply elision on. This approach may work for some applications but may require developers to understand the dynamic behavior of all locks in the program, and it may result in missed opportunities. Both approaches can be combined.

The lemming effect. When a transactional abort occurs, the synchronization library may either retry elision or explicitly skip it and acquire the lock. How the library retries elision is important. For example, when a thread falls back to explicitly acquiring a lock, it results in aborting other threads eliding the same lock. A naïve implementation on the other threads would immediately retry elision. These threads would find the lock

held, abort, and retry elision, thus quickly reaching their retry threshold without any opportunity to have found the lock free. As a result, these threads quickly transition to an execution where they skip elision. It is easy to see how all threads end up in a situation where no thread reattempts lock elision for longer time periods. This phenomenon is the lemming effect³ where threads enter extended periods of nonelided execution. This prevents software from taking advantage of the underlying elision hardware.

A simple and effective fix in the synchronization library is to retry elision only if the lock is free and spin/wait nontransactionally—and limit retry counts. Some lock algorithms implement a queue to enforce an order for threads if they find the lock unavailable. Queues are incompatible with parallel speculation. For such code, programmers must use the elision wrapper, including retry support, prior to the actual queuing code. Other mitigation strategies are possible.

Once programmers understand the underlying behavior, simple fixes can go a long way in improving unexpected performance behaviors.

Lock elision is a new technique in the scaling toolbox that is available on modern systems. It enables existing lock-based programs to achieve the performance benefits of non-blocking synchronization and fine-grained locking with minor software engineering effort.

Acknowledgments

Thanks to Noel Arnold, Jim Cownie, Roman Dementiev, Ravi Rajwar, Arch Robison, Joanne Strickland, and Konrad Lai for their feedback and improvements to the article. C

Related articles on queue.acm.org

Proving the Correctness of Nonblocking Data Structures

Mathieu Desnoyers

<http://queue.acm.org/detail.cfm?id=2490873>

Erlang for Concurrent Programming

Jim Larson

<http://queue.acm.org/detail.cfm?id=1454463>

Trials and Tribulations of Debugging Concurrency

Kang Su Gatlin

<http://queue.acm.org/detail.cfm?id=1035623>

References

- Cascaval, C., Blundell, C., Michael, M., Cain, H. W., Wu, P., Chiras, S. and Chatterjee, S. Software transactional memory: Why is it only a research toy? *Commun. ACM* 51, 11 (Nov. 2008), 40–46.
- Dallessandro, L., Carouge, F., White, S., Lev, Y., Moir, M., Scott, M.L. and Spear, M.F. Hybrid NORE: A case study in the effectiveness of best effort hardware transactional memory. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2011, 39–52.
- Dice, D., Herlihy, M., Lea, D., Lev, Y., Luchangco, V., Mesard, W., Moir, M., Moore, K. and Nussbaum, D. Applications of the adaptive transactional memory test platform. In *Proceedings of the 3rd Annual ACM SIGPLAN Workshop on Transactional Computing*, 2008.
- GCC. X86 transaction memory intrinsics, 2013; <http://gcc.gnu.org/onlinedocs/gcc-4.8.2/gcc/X86-transactional-memory-intrinsics.html#X86-transactional-memory-intrinsics>.
- Herlihy, M. and Moss, J.E.B. Transactional memory: Architectural support for lock-free data structures. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, 1993, 289–300.
- IBM. Power ISA, 2013; <http://www.power.org/documentation/power-isa-version-2-07/>.
- Intel. *Intel 64 and IA-32 Architectures Software Developer Manuals*, Vol. 1, Chapt. 14, 2012; <http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>.
- Intel. *IA Optimization Manual*, Chapt. 12, TSX Optimization; <http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf>. Web resources; <http://www.intel.com/software/tsx>.
- Jacobi, C., Slegel, T. and Dreiner, D. Transactional memory architecture and implementation for IBM System Z. In *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, 25–36.
- Kleen, A. 2013. TSX-tools; <http://github.com/andikleen/tsx-tools>.
- McKenney, P.E. Is parallel programming hard, and, if so, what can you do about it? (2013); <https://www.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html>.
- McVoy, L. SMP scaling considered harmful, 1999; <http://www.bitmover.com/lnl/smp.pdf>.
- Michael, M. The balancing act of choosing nonblocking features. *Commun. ACM* 56, 9 (Sept. 2013), 46–53.
- Pohlack, M. and Diestelhorst, S. 2011. From lightweight hardware transactional memory to lightweight lock elision. In the Sixth Annual ACM SIGPLAN Workshop on Transactional Computing.
- Rajwar, R. and Goodman, J.R. Speculative lock elision. In *Proceedings of the 34th Annual ACM/IEEE International Symposium on Microarchitecture*, 2001, 294–305.
- Usui, T., Behrends, R., Evans, J. and Smaragdakis, Y. Adaptive locks: combining transactions and locks for efficient concurrency. In *Proceedings of the 4th ACM SIGPLAN Workshop on Transactional Computing*, 2009.
- Wang, A., Gaudet, M., Wu, P., Amaral, J.N., Ohmacht, M., Barton, C., Silvera, R. and Michael, M. Evaluation of Blue Gene/Q hardware support for transactional memory. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, 2012, 127–136.
- Yoo, R.M., Hughes, C.J., Rajwar, R. and Lai, K. Performance evaluation of Intel Transaction Synchronization Extensions for high-performance computing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2013.

Andi Kleen is a software engineer at Intel's open source technology center. He worked on the x86-64 part of the Linux kernel and other kernel areas, including networking, performance, and error recovery. He currently focuses on performance tuning, scalability to many cores, and performance analysis.

32nd ACM Conference on Human Factors in Computing Systems

**April 26th - May 1st
Toronto, Canada**



CHI 2014
One of a CHInd

Inspiring keynotes: Margaret Atwood, Scott Jenson

World-leading research findings

Courses from HCI legends including Bill Buxton & Don Norman

Industry focused HCI in practice & case study sessions

Attend. Be changed. Change the world.

chi2014.acm.org @sig_chi

Conference Chairs: Matt Jones, Philippe Palanque
Technical Program Chairs: Tovi Grossman, Albrecht Schmidt

DOI:10.1145/2500875

Stable multithreading dramatically simplifies the interleaving behaviors of parallel programs, offering new hope for making parallel programming easier.

BY JUNFENG YANG, HEMING CUI, JINGYUE WU,
YANG TANG, AND GANG HU

Making Parallel Programs Reliable with Stable Multithreading

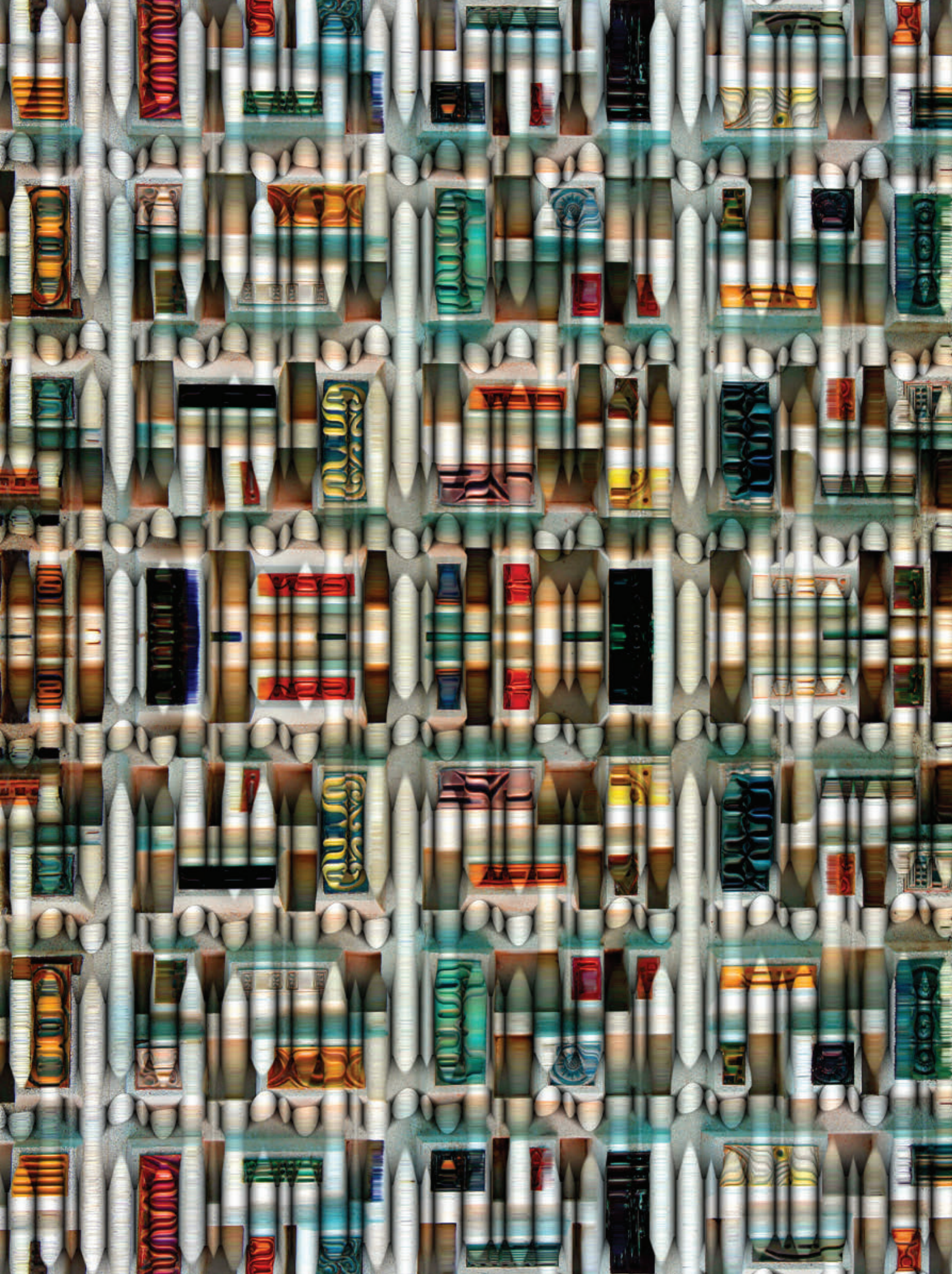
RELIABLE SOFTWARE HAS long been the dream of most researchers, practitioners, and users. In the past decade or so, several research and engineering breakthroughs have greatly improved the reliability of sequential programs (or the sequential aspect of parallel programs); successful examples include Coverity's source code analyzer,⁶ Microsoft's Static Driver Verifier,³ Valgrind memory checker,¹⁷ and certified operating systems and compilers.²⁰

However, the same level of success has not yet propagated to parallel programs, which are notoriously difficult to write, test, analyze, debug, and verify, much more so than the sequential versions. Experts consider reliable parallelism "something of a black art"⁸ and one of the grand challenges in computing.^{1,18} But widespread parallel programs are plagued with insidious concurrency bugs¹⁵ (such as data races, including concurrent accesses to the same memory location with at least one write, and deadlocks, including threads circularly waiting for resources). Some of the worst of them killed people in the Therac-25 radiation-therapy incidents by generating massive overdoses of radiation and caused the Northeast power blackout of 2003. These bugs can be exploited by attackers violating confidentiality, integrity, and availability of critical systems.²⁴

Over the past decade, two technology trends have made the challenge of reliable parallelism even more urgent: The first is the rise of multicore hardware; the speed of a single processor core is limited by fundamental physical constraints, forcing processors into multicore designs and developers resorting to parallel code for best performance on multicore processors. The second is accelerating computational demand. Scientific computing, video and image processing, financial simulation, big-data analytics, Web search, and online social networking all involve massive computations and various kinds of parallel programs to

» key insights

- **Getting multithreaded programs right is difficult because they have too many possible thread interleavings, or schedules, not because they are nondeterministic.**
- **Not all schedules are necessary; for many programs, a small set is enough to process all possible inputs.**
- **StableMT dramatically reduces the set of schedules while keeping overhead low, greatly enhancing almost all reliability techniques, including testing, debugging, and program analysis.**




maximize performance.

If reliable software is an overarching challenge of computer science, reliable parallelism is surely the key. To make parallel programs reliable, researchers have devoted decades of effort, producing numerous ideas and systems, ranging from new hardware, programming languages, and programming models to tools that detect, diagnose, avoid, or fix concurrency bugs. As usual, new hardware, languages, and models take years, if not forever, to adopt. Tools are helpful but tend to attack derived problems, not the root cause.


We look to attack fundamental, open problems in making shared-memory multithreaded programs reliable. These programs express concurrency through threads, essentially lightweight, sequential processes that share memory. We target them because they are the most widespread type of parallel programs, with mature support from hardware, operating systems, libraries, and programming languages. Moreover, they will likely remain prevalent for the foreseeable future.

Unlike sequential programs, repeated executions of the same multithreaded program on the same input could yield behaviors (such as correct vs. buggy), depending on how the threads interleave. Conventional wisdom has long blamed this nondeterminism for the challenges in reliable multithreading;¹³ threads are nondeterministic by default, and it is the (tricky) job of developers to account for this nondeterminism. Nondeterminism has direct implications on reliability; for instance, it makes testing less effective. A program may run correctly on an input in the testing lab because the interleavings tested happen to be correct, but executions on the same exact input may still fail in the field when the program hits a buggy, untested interleaving.

To eliminate nondeterminism, several groups of researchers, including us, have dedicated themselves to building deterministic multithreading (DMT) systems^{2,4,5,7,12,14,19} that force multithreaded programs to always execute the same thread interleaving, or schedule, on the same input, always resulting in the same behavior. By mapping each input to only one schedule, DMT brings determinism, a key prop-



Removing unnecessary schedules from the haystack would make the needles easier to find.



erty of sequential computing, into multithreading.

However, nondeterminism is only a small piece of the puzzle, and determinism (the cure for nondeterminism) is not as useful as commonly perceived, being neither sufficient nor necessary for reliability. It is not sufficient because a perfectly deterministic system can map each input to an arbitrary schedule, so small input perturbations lead to vastly different schedules, artificially reducing a program's robustness and stability. It is not necessary because a nondeterministic system with a small set of schedules for all inputs can be made reliable by exhaustively checking all schedules.

What makes multithreading difficult for programmers to get right is quantitative; multithreaded programs have too many schedules. The number of schedules for each input is already enormous because the parallel threads can interleave in many ways, depending on such factors as hardware timing and operating-system scheduling. Aggregated over all inputs, the number is even greater. Finding a few schedules that trigger concurrency errors out of all schedules (so developers can prevent them) is like finding needles in a haystack. Although DMT reduces schedules for each input, it could map each input to a different schedule, so the total set of schedules for all inputs remains enormous.

We have attacked this root cause by asking if all the many schedules are necessary. Our 2010 study found that many real-world programs can use a small set of schedules to efficiently process a range of inputs.¹⁰ Leveraging this insight, we envision a new approach we call stable multithreading, or StableMT, that reuses each schedule on a range of inputs, mapping all inputs to a dramatically reduced set of schedules. By vastly shrinking the haystack, the needles are much easier to find. By mapping many inputs to the same schedule, program behaviors are stabilized against small input perturbations. StableMT and DMT are not mutually exclusive; a system can be both deterministic and stable.

To realize our vision of StableMT, we built several systems: TERN¹⁰ and PEREGRINE;¹¹ two compiler and run-

time implementations of StableMT; and a program-analysis framework that leverages StableMT to achieve high coverage and precision unmatched by its counterparts.²² They address three complementary challenges, two of which are long open in related areas. TERN addresses how to compute highly reusable schedules. The more reusable the schedules, the fewer of them are needed. Unfortunately, computing reusable schedules is undecidable at compile time and costly at runtime. PEREGRINE addresses how to efficiently make executions follow schedules and not deviate, a decades-old challenge in the area of deterministic execution and replay. Our analysis framework addresses how to effectively analyze multithreaded programs, a well-known open problem in program analysis. Our implementations of these systems are mostly transparent to developers and fully compatible with existing hardware, operating systems, thread libraries, and programming languages, simplifying adoption.

Our initial results are promising. Evaluation on a diverse set of multithreaded programs, including the Apache Web server and the MySQL database, shows TERN and PEREGRINE reduce schedules dramatically; for instance, under a typical setup, they reduce the number of schedules needed by parallel compression utility PBZip2 down to two schedules for each different number of threads, regardless of

file content. Their overhead is moderate, less than 15% for most programs. Our program-analysis framework enables construction of many program analyses with precision and coverage unmatched by their counterparts; for instance, a race detector we built found previously unknown bugs in extensively checked code with almost no false bug reports.

Difficult to Get Right

Here, we start with preliminaries, describe the challenges caused by nondeterminism and by too many schedules, and explain why nondeterminism is a lesser cause than too many schedules.

Inputs, schedules, and buggy schedules. We say “input” to broadly refer to the data a program reads from its execution environment, including not only the data read from files and sockets but command-line arguments, return values of external functions (such as `gettimeofday`), and any external data that can affect program execution. We say “schedule” to broadly refer to the (partially or totally) ordered set of communication operations in a multithreaded execution, including synchronizations (such as `lock` and `unlock` operations) and shared memory accesses (such as `load` and `store` instructions to shared memory). Of all the schedules, most run fine, but some trigger concurrency errors, causing program crashes, incorrect computations, deadlocked executions, and other failures. Consider this toy program:

```
// thread      // thread 2
lock(1);      lock(1);
*p = . . . ;   p = NULL;
unlock(1);    unlock(1);
```

The schedule in which thread 2 gets the lock before thread 1 causes a dereference-of-NULL failure. Now consider another example, this one with data races on `balance`:

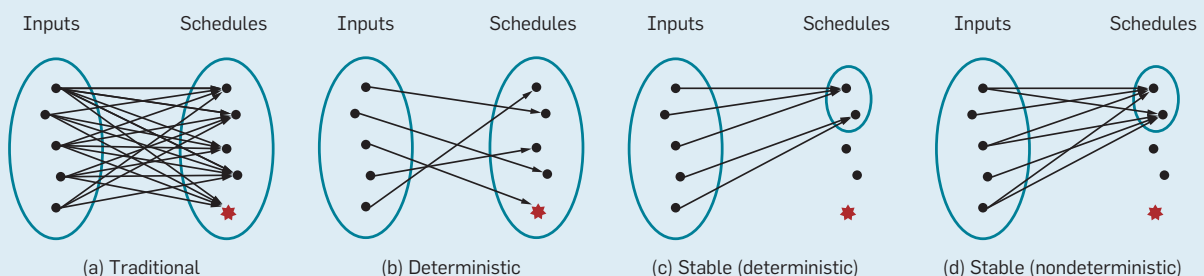
```
// thread 1    // thread 2
// deposit 100 // withdraw 100
t = balance + 100;
                    balance =
                    balance - 100;
balance = t;
```

The schedule with the statements executed in this order corrupts `balance`. We call the schedules that trigger concurrency errors “buggy schedules.” Strictly speaking, the errors are in the programs, triggered by a combination of inputs and schedules. However, typical concurrency errors (such as in Lu et al.¹⁵ and Yang et al.²⁴) depend much more on the schedules than on inputs (such as once the schedule is fixed, the bug occurs for all inputs allowed by the schedule). Recent research on testing multithreaded programs (such as Musuvathi et al.¹⁶) focuses on testing schedules to find the buggy ones.

Challenges caused by nondeterminism. A multithreaded program is nondeterministic because, even with the same program and input, different schedules can still lead to different behaviors; for

Figure 1. Multithreading approaches.

Red stars represent buggy schedules. Traditional multithreading (a) is a conceptual many-to-many mapping where one input may execute under many schedules due to nondeterminism and many inputs may execute under one schedule because the schedule fixes the order of the communication operations but allows the local computations to operate on any input data. DMT (b) can map each input to an arbitrary schedule, reducing a program’s robustness on input perturbations. StableMT (c and d) reduces the total set of schedules for all inputs (represented by the shrunk ellipses), increasing robustness and improving reliability. StableMT and DMT are orthogonal; a StableMT system can be either deterministic (c) or nondeterministic (d).



instance, the two toy programs do not always run into the bugs. Except for the schedules described, the other schedules lead to correct executions.

This nondeterminism involves many challenges, especially in testing and debugging. Suppose an input can execute under n schedules. Testing $n - 1$ schedules is not enough for complete reliability because the single untested schedule could still be buggy. An execution in the field may hit this untested schedule and fail. Debugging is challenging, too. To reproduce a field failure for diagnosis, the exact input alone is not enough; developers must also manage to reconstruct the buggy schedule out of n possibilities.

Figure 1a outlines the traditional multithreading approach, conceptually, a many-to-many mapping, where one input can execute under many schedules due to nondeterminism, and many inputs can execute under one schedule because a schedule fixes the order of the communication operations but allows the local computations to operate on any input data.

Challenges caused by too many schedules. A typical multithreaded program includes an enormous number of schedules. For a single input, the number is asymptotically exponential in the schedule length; for instance, given m threads, each competing for a lock k times, each order of lock acquisitions forms a schedule, easily yielding $\frac{(mk)!}{(k!)^m} \geq (m!)^k$ total schedules, a number exponential in both m and k . Aggregated over all inputs, the number of schedules is even greater.

Finding a few buggy schedules in these exponentially many schedules raises a series of needle-in-a-haystack challenges; for instance, to write correct multithreaded programs, developers must carefully synchronize their code to weed out the buggy schedules. As usual, humans err when they must scrutinize many possibilities to locate corner cases. Various forms of testing tools suffer, too. Stress testing is the common method for (indirectly) testing schedules, but often redundantly tests the same schedules while missing others. Recent tools (such as by Musuvathi et al.¹⁶) systematically test schedules for bugs, but developers lack resources to cover more than a tiny fraction of all the exponentially

many schedules.

Determinism not as useful as commonly perceived. To address the challenges due to nondeterminism, researchers, including us, have dedicated much effort and built several systems that force a multithreaded program to always run the same schedule on the same input, thus bringing determinism to multithreading. This determinism has value for reliability; for instance, one testing execution now validates all future executions on the same input, and reproducing a concurrency error now requires only the input.

Meanwhile, little has been done to solve the challenges caused by too many schedules. The research community has assigned to nondeterminism more than its share of guilt but overlooked the main culprit—a rather quantitative cause, that multithreaded programs simply have too many schedules. Although determinism has value, that value is less than commonly perceived and neither sufficient nor necessary for reliability.

Determinism \nRightarrow reliability. Determinism is a narrow property: same input + same program = same behavior. It has no jurisdiction if the input or program changes, however slightly. Yet developers often expect a program to be robust or stable against slight program changes or input perturbations; for instance, adding a debug `printf` should in principle not make the bug disappear. Likewise, a single bit flip of a file should usually not cause a compression utility to crash. Unfortunately, determinism does not provide this stability and if naively implemented even undermines it.

To illustrate, consider the system in Figure 1b that maps each input to an arbitrary schedule. This mapping is perfectly deterministic but destabilizes program behaviors on multiple inputs. A single bit flip could force a program to discard a correct schedule and wander into a vastly different, buggy schedule.

This instability is counterintuitive at least, raising new reliability challenges; for instance, testing one input provides little assurance on very similar inputs, despite the differences in input not invalidating the tested schedule. Debugging now requires every bit of the bug-inducing input,

including not only the data a user typed but also environment variables, shared libraries, even a different user name or if an error report lacks credit card numbers. The bug may never be reproduced, regardless of how many times developers retry, because the schedule chosen by the deterministic system for the altered input happens to be correct. Note even a correct sequential program may show different behaviors for small input changes across boundary conditions, but these conditions are typically infrequent, and the different behaviors are intended by developers. In contrast, the instability introduced by the system in Figure 1b is artificial and on all inputs.

Besides inputs, naively implemented determinism can destabilize program behaviors on minor code changes, so adding a debug `printf` causes the bug to deterministically disappear. Another problem is that the number of all possible schedules remains enormous, so the coverage of schedule testing tools remains low.

In practice, to mitigate such problems, researchers augment determinism with other techniques. To support debug `printf`, some have proposed temporarily reverting to nondeterministic execution.¹² Moreover, DMP,¹² as well as CoreDet⁴ and Kendo,¹⁹ change schedules only if the inputs change low-level instructions being executed. Although better than mapping each input to an arbitrary schedule, these systems still allow small input perturbations to destabilize schedules unnecessarily when the perturbations change the low-level instructions executed (such as one extra load executed) we have observed in our experiments.¹⁰ Our TERN and PEREGRINE systems and Liu et al.'s DTHREADS¹⁴ built after TERN combine DMT with StableMT to frequently reuse schedules on a range of inputs for stability.

Reliability \nRightarrow determinism. Determinism is a binary property; if an input maps to $n > 1$ schedules, executions on this input may be nondeterministic, however small n is. Yet a nondeterministic system with a small set of total schedules is easily made reliable. Consider an extreme case of the nondeterministic system in Figure

1d, mapping all inputs to at most two schedules. In it, developers are able to easily solve the needle-in-a-haystack challenges due to nondeterminism; for instance, to reproduce a field failure given an input, they can afford to search for one of only two schedules. As an analogy, a coin toss is nondeterministic, but humans have no problem understanding and doing it, as only two outcomes are possible.

Shrinking the Haystack

Motivated by the limitations of determinism and the challenges due to exponentially many schedules, we investigated a central research issue: whether all the exponentially many schedules are necessary. A schedule is necessary if it is the only one that can process specific inputs or yield good performance under specific scenarios. Removing unnecessary schedules from the haystack would make the needles easier to find.

We investigated on a diverse set of popular multithreaded programs, ranging from server programs (such as Apache) to desktop utilities (such as parallel compression utility PBZip2) to parallel implementations of computation-intensive algorithms (such as fast Fourier transformation). They use diverse synchronization primitives (such as locks, semaphores, condition variables, and barriers). This produced two insights: First, for many programs, a range of many inputs share the same equivalent class of schedules. One schedule out of the class is enough to process the entire input range. Intuitively, an input often contains two types of data: metadata that controls the communication of the execution (such as number of threads to spawn) and computational data the threads compute on locally. A schedule requires the input metadata to have certain values but also allows the computational data to vary. That is, it can process any input that has the same metadata; for instance, consider PBZip2, which splits an input file among multiple threads, each compressing one file block. The communication, or which thread gets which file block, is independent of the thread-local compression. Under a typical setup (such as no read failures or signals), for each different number

of threads set by a user, PBZip2 can use two schedules, one if the file can be evenly divided by the number of threads, another to otherwise compress any file, regardless of file data.

This loose coupling of inputs and schedules is not unique to PBZip2; many other programs also exhibit this property. Table 1 is a sample of our findings, including three real-world programs—Apache, PBZip2, and *aget* (a parallel file download utility)—and five implementations of computation-intensive algorithms from two widely used benchmark suites—Stanford’s SPLASH2 and Princeton’s PARSEC.

The second is that the overhead of enforcing a schedule on different inputs is low. The exponentially many schedules presumably allow the runtime system to react to various timing factors and select an efficient schedule. However, results from our StableMT systems invalidated the presumption. With carefully designed schedule representations, they incurred less than 15% overhead enforcing schedules on different inputs for most evaluated programs. This moderate overhead is worth the gains in reliability.

Leveraging the insights, we invented stable multithreading, or StableMT, a new multithreading approach that reuses each schedule on a range of inputs, mapping all inputs to a dramatically reduced set of schedules. Vastly shrinking the haystack at

once. In addition, StableMT stabilizes program behaviors on inputs that map to the same schedule and minor program changes that do not affect the schedules, providing the robustness and stability expected by developers and users alike.

StableMT and DMT are orthogonal. StableMT aims to reduce the set of schedules for all inputs, whereas DMT aims to reduce the schedules for each input (down to one). A StableMT system may be either deterministic or nondeterministic. Figure 1c and Figure 1d depict two StableMT systems; the many-to-one mapping in Figure 1c is deterministic, while the many-to-few mapping in Figure 1d is nondeterministic. A many-to-few mapping improves performance because the runtime system can choose an efficient schedule out of a few schedules for an input based on current timing factors but increases the effort and resources needed for reliability. Fortunately, only a few choices of schedules (such as a small constant, like two) are available, so the challenges caused by nondeterminism are easily solved.

Benefits. By vastly reducing the set of schedules, StableMT brings numerous reliability benefits to multithreading:

Testing. StableMT automatically increases the coverage of schedule testing tools, with coverage defined as the ratio of tested schedules over all schedules; for instance, consider PBZip2 again, which needs only two schedules

Table 1. Constraints on inputs sharing the same equivalent class of schedules. For each program, one schedule out of the class is enough to process any input satisfying the constraints in the third column under typical setups (such as no system-call failures or signals).

Program	Purpose	Constraints on inputs sharing schedules
Apache	Web server	For a group of typical HTTP GET requests, same cache status
PBZip2	Compression	Same number of threads
aget	File download	Same number of threads, similar file sizes
barnes	N-body simulation	Same number of threads, same values of two configuration variables
fft	Fast Fourier transform	Same number of threads
lu-contig	Matrix decomposition	Same number of threads, similar sizes of matrices and blocks
blackscholes	Option pricing	Same number of threads, number of options no less than number of threads
swaptions	Swaption pricing	Same number of threads, number of swaptions no less than number of threads

for each different number of threads under typical setups. Testing 32 schedules covers from one to 16 threads. Given that PBZip2 achieves peak performance when the number of threads is identical or close to the number of cores, and a typical machine has up to 16 cores, 32 tested schedules can cover most schedules executed in the field.

Debugging. Reproducing a bug does not require the exact input, as long as the original and the altered inputs map to the same schedule. It also does not require the exact program, as long as the changes to the program do not affect the schedule. Users can remove private information (such as credit-card numbers) from their bug reports, and developers can reproduce the bugs in different environments or add `printf` statements.

Analyzing and verifying programs. Static analysis can focus on the set of schedules enforced in the field to gain precision. Dynamic analysis enjoys the same benefits as testing. Model

checking can check dramatically fewer schedules, mitigating the so-called “state explosion” problem.⁹ Interactive theorem proving becomes easier, too, because verifiers must prove theorems on only the set of schedules enforced in the field.

Avoiding errors at runtime. Programs can also adaptively learn correct schedules in the field, then reuse them on future inputs to avoid unknown, potentially buggy schedules.

Caveats. StableMT is not for every multithreaded program. It works well with programs with schedules loosely coupled with inputs, but a program may decide to, say, spawn threads or invoke synchronizations based on intricate conditions involving many bits in the input. An example is the parallel `grep`-like utility `pfscan`, which searches for a keyword in a set of files using multiple threads, and for each match grabs a lock to increment a counter. A schedule computed on one set of files is unlikely to suit another. To

increase the input range each schedule covers, developers can exclude the operations on this lock from the schedule using annotations.

StableMT provides robustness and stability on small input and program perturbations when they do not affect schedules, though there is room for improvement; for instance, when developers change their programs by adding synchronizations, it may be more efficient to update previously computed schedules rather than recompute from scratch. We leave this idea for future work.

Building Stable Multithreading Systems

Although the vision of stable multithreading is appealing, its realization faces numerous challenges, including the following:

Computing schedules. How can StableMT systems compute the schedules to map inputs to? The schedules must be feasible in real-world situations so executions reusing them do not get stuck. They should also be highly reusable;

Enforcing schedules. How can StableMT systems enforce schedules deterministically and efficiently? “Deterministically” so executions that reuse a schedule cannot deviate, even when there are data races, and “efficiently” so overhead does not offset reliability benefits, a challenge decades old in the area of deterministic execution and replay; and

Handling threads. How can StableMT systems handle multithreaded server programs? They often run for a long time, reacting to each client request as it arrives, thus making their schedules very specific to a stream of requests and difficult to reuse.

We have been tackling these challenges since 2009 in building our two StableMT prototypes—TERN¹⁰ and PEREGRINE¹¹—that frequently reuse schedules with low overhead. Here, we describe our solutions, though they are by no means the only ones; for example, subsequent to TERN, other researchers built a system that stabilizes schedules for general multithreaded programs.¹⁴

Computing schedules. Crucial to implementing StableMT is how to compute the set of schedules for process-

Figure 2. Example program based on parallel compression utility PBZip2, which spawns `nthread` worker threads, splits a file among the threads, and compresses the file blocks in parallel.

```

1: main(int argc, char *argv[ ]) {
2:   int i, nthread = atoi(argv[1]);
3:   for(i=0; i<nthread; ++i)
4:     pthread_create(worker); // create worker threads
5:   for(i=0; i<nthread; ++i)
6:     worklist.add(read_block(i)); // add block to work list
7:   // Error: missing pthread_join() operations
8:   worklist.clear(); // clear work list
9:   ...
10: }
11: worker() { // worker threads for compressing file blocks
12:   block = worklist.get(); // get a file block from work list
13:   compress(block);
14: }
15: compress(block t block) {
16:   if(block.data[0] == block.data[1])
17:     ...
18: }
```

Figure 3. A synchronization schedule of the example program. Each synchronization is labeled with its line number in Figure 2.

```

// main           // worker 1           // worker 2
4: pthread_create(worker);
4: pthread_create(worker);
6: worklist.add();
           12: worklist.get();
6: worklist.add();
           12: worklist.get();
8: worklist.clear();
```

ing inputs. At bare minimum, a schedule must be feasible when enforced on an input so the execution does not get stuck or deviate from the schedule. Ideally, the set of schedules should also be small for the sake of reliability. One idea is to precompute schedules using static source-code analysis, but the halting problem makes it undecidable to statically compute schedules guaranteed to work dynamically. Another is to compute schedules on the fly as a program runs, but the computations may be complex and their overhead prohibitively high.

Our systems compute schedules by recording them from past executions; the recorded schedules can then be reused on future inputs to stabilize program behaviors. TERN works like this: At runtime, it maintains a persistent cache of schedules recorded from past executions. When an input arrives, it searches the cache for a schedule compatible with the input. If it finds one, it simply runs the program while enforcing the schedule. Otherwise, it runs the program as is while recording a new schedule from the execution, saving the schedule into the cache for future reuse.

The TERN approach to computing schedules has several benefits: First, by reusing schedules shown to work, it might avoid potential errors in unknown schedules, improving reliability. A real-world analogy is the natural human (and animal) tendency to follow familiar routes to avoid possible hazards along unknown routes. Migrating birds often follow, for example, fixed flyways. (The name TERN comes from the Arctic Tern, a bird species that migrates farthest among all animals.) Why are our multithreading systems unable to learn from them and reuse familiar schedules?

Second, TERN explicitly stores schedules, so developers and users can flexibly choose what schedules to record and when; for instance, developers can populate a cache of correct schedules during testing, then deploy the cache together with their program, improving testing effectiveness and avoiding the overhead to record schedules on user machines. Moreover, they can run their favorite checking tools on the schedules to detect a variety of errors and choose to keep only the cor-

rect schedules in the cache.

TERN is efficient because it can amortize the cost of computing schedules. Recording and checking a schedule is more expensive than reusing a schedule, but, fortunately, TERN does it only once for each schedule, then reuses the schedule on many inputs, amortizing the cost.

A key challenge for TERN is to check that an input is compatible with a schedule before executing the input under the schedule. Otherwise, if it tries to enforce a schedule of, say, two threads on an input requiring four, the execution would not follow the schedule. This challenge turns out to be the most difficult one we had to solve in building TERN. Our solution leverages several advanced program-analysis techniques, including two new ones we invented; see Cui¹⁰ and Cui¹¹ for details.

When recording a schedule, TERN tracks how the synchronizations in the schedule depend on the input, capturing these dependencies in a relaxed, quickly checkable set of constraints we call the “precondition of the schedule.” It then reuses the schedule on all inputs satisfying the precondition, avoiding the runtime cost of recomputing schedules.

A naïve way to compute the precondition is to collect constraints from all input-dependent branches in an execution; for instance, if a branch instruction inspects input variable X and goes down the true branch, TERN adds a constraint that X must be nonzero to the precondition. A precondition computed this way is sufficient but contains many unnecessary constraints concerning only thread-local computations. Since an over-constraining precondition decreases schedule-reuse rates, TERN removes these unneces-

sary constraints from the precondition.

We illustrate how TERN works through a simple program based on the aforementioned parallel compression utility `PBZip2` (see Figure 2). Its input includes all command-line arguments in `argv` and input file data. To compress a file, it spawns `nthread` worker threads, splits the file accordingly, and compresses the file blocks in parallel by calling function `compress`. To coordinate the worker threads, it uses a synchronized work list. (Here we use work-list synchronization for clarity; in practice, it handles `Pthread` synchronizations.) However, the example also has a bug: Because it is missing `pthread_join` operations at line 7, the work list may be used by function `worker` after it is cleared at line 8, causing potential program crashes; this bug is based on a real bug in `PBZip2`.

We first illustrate how TERN records a schedule and its precondition. Suppose we run this example with two threads, and TERN records a schedule (see Figure 3) that avoids the use-after-free bug. (Other schedules are possible.) To compute the precondition of the schedule, TERN first records the outcomes of all executed branch statements that depend on input data; Figure 4 includes the set of collected constraints. It then applies advanced program analyses to remove those that concern only local computations and have no effect on the schedule, including all constraints collected from function `compress`. The remaining ones simplify to `nthread = 2`, forming the precondition of the schedule. TERN stores the schedule and precondition into the schedule cache.

We now illustrate how TERN reuses a schedule. Suppose a user wants to compress a completely different

Figure 4. All input constraints collected for the schedule, each labeled with its line number in Figure 2. Those collected from function `compress` are later removed by TERN because they have no effect on the schedule; the remaining ones simplify to `nthread = 2`.

```

3: 0 < nthread ? true
3: 1 < nthread ? true
3: 2 < nthread ? false
5: 0 < nthread ? true
5: 1 < nthread ? true
5: 2 < nthread ? false
16: ... // constraints collected from compress()

```

file also with two threads. TERN would detect that `nthread` satisfies `nthread = 2`, so it reuses the schedule in Figure 3 to compress the file, regardless of file data. This execution is reliable because the schedule avoids the use-after-free bug. It is also efficient because the schedule orders only synchronizations and allows the compress operations to run in parallel. Suppose the user runs the program again with four threads. TERN would detect that the input does not satisfy the precondition `nthread = 2` so will record a new schedule and precondition.

Efficiently enforcing schedules. Prior work enforces schedules at two different granularities—shared memory accesses or synchronizations—forcing users to trade off efficiency and determinism. Specifically,

memory access schedules make data races deterministic but are prohibitively inefficient (such as from 1.2X to 6X as slow as traditional multithreading⁴); synchronization schedules are much more efficient (such as average slowdown of only 16%¹⁹) because they are coarse grain but cannot make programs with data races deterministic (such as the second toy program discussed earlier, as well as many real-world multithreaded programs^{15,23}). This determinism vs. performance challenge has been open for decades in the areas of deterministic execution and replay. As a result, TERN, our first StableMT system, enforces only synchronization schedules.

This is the challenge that prompted us to build PEREGRINE, our second StableMT system.¹¹ The PEREGRINE

design insight is that although many programs have races, the races tend to occur only within small portions of an execution, with most of the execution still race-free. Intuitively, if a program is full of data races, most would have been caught during testing. We empirically analyzed the executions of seven real programs with races, finding, despite millions of memory accesses, up to only 10 data races per execution.

Since races are rare, PEREGRINE can schedule synchronizations for the race-free portions of an execution and resort to scheduling memory accesses only for the “racy” portions, combining both the efficiency of synchronization schedules and the determinism of memory-access schedules. These hybrid schedules are almost as coarse grain as synchronization schedules so can also be reused frequently (see Figure 5).

How is PEREGRINE able to predict where data races might occur before an execution actually begins? One idea is to use static analysis to detect them at compile time. However, static race detectors are notoriously imprecise, as the tendency is for most of their reports to be false, not true data races. Scheduling many memory accesses in the false reports would slow execution significantly. PEREGRINE leverages the record-and-reuse approach in TERN to predict races; a recorded execution can effectively foretell what could happen for executions reusing the same schedule. Specifically, when recording a synchronization schedule, PEREGRINE records a detailed memory-access trace. From it, it detects data races that occurred (with respect to the schedule), adding the memory accesses involved in the races to the schedule. This hybrid schedule can be enforced efficiently and deterministically, solving the aforementioned open challenge of determinism vs. performance. To reuse the schedule on other inputs, PEREGRINE provides new precondition computation algorithms to guarantee executions reusing the schedule will not run into any new data races. To enforce an order on memory accesses, PEREGRINE modifies a live program at runtime through a safe, efficient instrumentation framework called Loom we built in 2010.²¹

Figure 5. Hybrid schedule.

Circles represent synchronizations and triangles memory accesses. A synchronization schedule is efficient because it is coarse grain but not deterministic because data races could still cause executions to deviate from the schedule and fail. A memory access schedule is deterministic but slow because it is fine grain. A hybrid schedule combines the best of both by scheduling memory access for only the racy portion of an execution and for synchronizations otherwise.

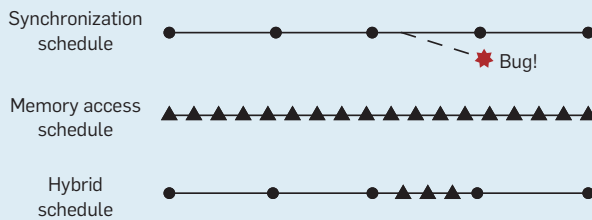
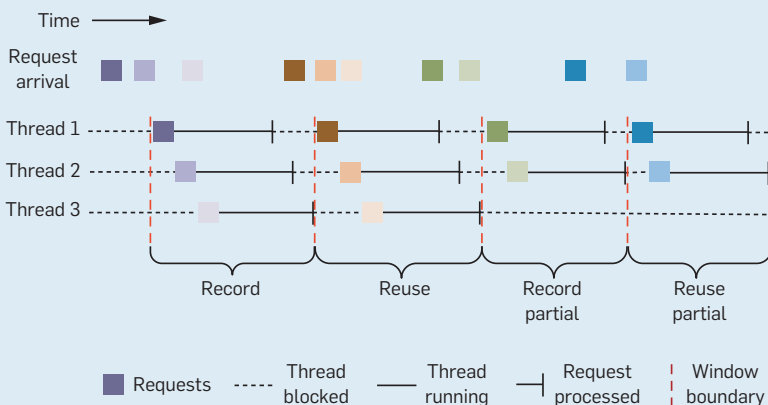


Figure 6. Recording and reusing schedules for a server program with three threads. The continuous execution stream is broken down into windows of requests, and PEREGRINE records and reuses schedules across windows.



Handling server programs. Server programs present three challenges for StableMT: First, they could run continuously, making their schedules effectively infinite and too specific to reuse; second, they often process inputs, or client requests, as soon as the requests arrive, with each request arriving at a random moment, causing a different schedule; and third, since requests do not arrive at the same time, PEREGRINE cannot check them against the precondition of a schedule up front.

Server programs tend to return to the same quiescent states, so PEREGRINE can use these states to split a continuous request stream into windows of requests (see Figure 6). Specifically, PEREGRINE buffers requests as they arrive until it gathers enough requests to keep all worker threads busy; it then runs the worker threads to process the requests while buffering newly arrived requests to avoid interference between windows. If PEREGRINE cannot gather enough requests before a predefined timeout, it proceeds with the partial window to reduce response time. By breaking a request stream into windows, PEREGRINE can record and reuse schedules across windows, stabilizing server programs. Server quiescent states may evolve; for instance, a Web server may cache requests in memory. PEREGRINE lets developers annotate the functions that query cache, treating the return values as inputs and selecting proper schedules. Windowing reduces concurrency, but the cost is moderate based on our experiments.

Stable Multithreading for Better Program Analysis

StableMT can be applied in many ways to improve reliability. Here, we describe a program analysis framework we built atop PEREGRINE to analyze multithreaded programs, an open challenge in program analysis.

At its core is the trade-off between precision and coverage. Of the two common types of program analysis, static analysis, which analyzes source code without running it, covers all schedules but is imprecise (such as in issuing many false error reports). The cause is that static analysis must over-approximate the enormous number

of schedules and thus may analyze a much larger set of schedules, including those impossible at runtime. Not surprisingly, static analysis can detect many “bugs” in the impossible schedules. Dynamic analysis, which runs code and analyzes the executions, precisely identifies bugs because it sees the code’s precise runtime effects. However, it has poor coverage due to the exponentially many schedules.

Fortunately, StableMT shrinks the set of possible schedules, enabling a new program analysis approach reflecting the best of both static analysis and dynamic analysis (see Figure 7). It statically analyzes a parallel program over a small set of schedules at compile time, then dynamically enforces these schedules at runtime. By focusing on only a small set of schedules, StableMT greatly improves the precision of static analysis and reduces false error reports; by dynamically enforcing the analyzed schedules, StableMT guarantees high coverage. Dynamic analysis benefits, too, because StableMT greatly increases its coverage, as defined by the ratio of checked schedules over all schedules.

A key challenge in implementing this approach is how to statically analyze a program with respect to a schedule. A static tool typically invokes many analyses to compute the final results. A naïve method for modifying the tool for improved precision is to modify every analysis involved, though

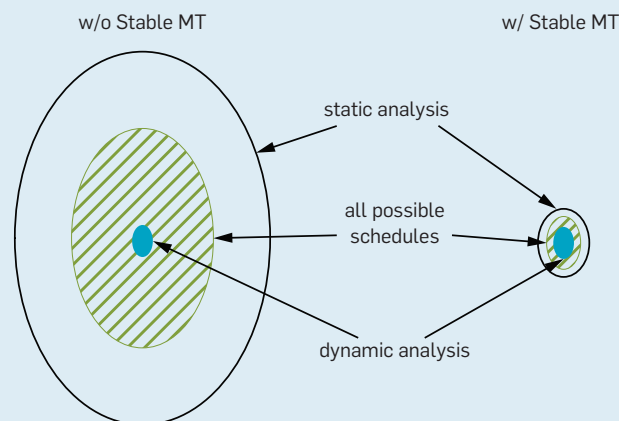
it would be quite labor intensive and error prone. It may also be fragile; that is, if a crucial analysis is unaware of the schedule, it could easily render the final results as imprecise.

We thus created a new program-analysis framework and algorithms to specialize, or intelligently transform, a program according to a schedule. The resulting program has simpler control and data flow than the original program and can be analyzed with stock analyses (such as constant folding and dead-code elimination) for significantly improved precision. In addition, our framework provides a precise “def-use” analysis that computes how values are defined and used in a program. Its results are much more precise than those of regular def-use analyses because it reports only the facts that may occur when the given schedule is enforced at runtime. This precision can be the foundation of many powerful tools, including race detectors.

Here, we illustrate the high-level idea of our framework recalling the example in Figure 2. Suppose a tool developer wants to build a static race detector that flags when different threads write the same shared memory location concurrently. Although different worker threads access disjoint file blocks, existing static analysis may be unable to determine this fact; for instance, since `nthread`, the number of threads, is determined at runtime, static analysis often has to approximate the

Figure 7. Program analysis with and without StableMT.

Without StableMT, static analysis tends to analyze many more schedules than all possible schedules; dynamic analysis tends to analyze a tiny fraction of all possible schedules. StableMT shrinks the set of schedules, improving both static analysis and dynamic analysis.



dynamic threads as one or two abstract thread instances. It may thus collapse different threads' accesses to distinct blocks as the same access, generating false race reports.

StableMT simplifies these problems. Suppose whenever `nthread` is 2, StableMT always enforces the schedule in Figure 3. Since the number of threads is fixed, our program-analysis framework rewrites the example program to replace `nthread` with 2. It then unrolls the loops and clones function `worker` to give each worker thread its own copy of `worker`, so distinguishing different worker threads becomes automatic.

Our framework also enables construction of many high-coverage and highly precise analyses; for instance, our static race detector found seven

previously unknown harmful races in programs that had been extensively checked by previous tools. It generates extremely few false reports, none for 10 of 18 programs, a huge reduction compared to other static race detectors.

Evaluation

Here, we describe the main results of PEREGRINE, focusing on two evaluation questions:

Can it reuse schedules frequently?

The higher the reuse rate, the more stable program behaviors become, and the more efficient PEREGRINE is; and

Can it enforce schedules efficiently?

Low overhead is crucial for programs that reuse schedules frequently.

We chose a diverse set of 18 programs as our evaluation benchmarks, either widely used real-world paral-

lel programs (such as Apache and PBZip2) or parallel implementations of computation-intensive algorithms in standard benchmark suites.

Stability. To evaluate PEREGRINE's stability, or how frequently it is able to reuse schedules, we compare the preconditions it computes to the best possible preconditions derived from manual inspection. Table 1 includes some of the manually derived preconditions; for nine of the 18 programs, the preconditions it computes are as good as or close to the best preconditions, allowing frequent reuses, while for the others, the preconditions are more restrictive.

We also evaluate stability by measuring the schedule reuse rates under given workloads (see Table 2 for results obtained from TERN and replicable in PEREGRINE). The four workloads are either real workloads collected by us or synthetic workloads used by developers.¹⁰ For three of the four workloads, TERN reuses a small number of schedules to process over 90% of the workloads. For MySQL-tx, TERN has a lower reuse rate largely because the workload is too random to reuse schedules. Nonetheless, it still processes 44.2% of the workloads.

Efficiency. The overhead of enforcing schedules is crucial for programs that frequently reuse schedules. Figure 8 shows this overhead for both TERN and PEREGRINE; each bar represents execution time, with TERN and PEREGRINE normalized to traditional multithreading, averaged over more than 500 runs; Figure 8 reports throughput (TPUT) and response time (RESP) for Apache.

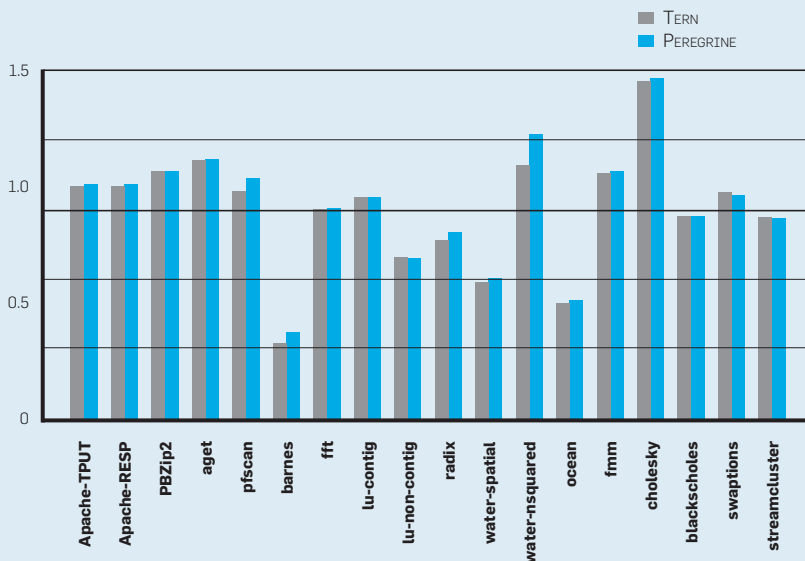
Two observations can be made about the figure: First, for most programs, overhead is less than 15%, demonstrating that StableMT can be efficient. For two programs—`water-nsquared` and `cholesky`—the overhead is relatively large because they do a large number of mutex operations within tight loops. However, overhead is still less than 50%, much less than the 1.1X–10X overhead of a prior DMT system.⁴ Some programs enjoy a speedup, as TERN and PEREGRINE safely skip some blocking operations.^{10,11} Second, PEREGRINE is only slightly slower than TERN, demonstrating full determinism can be efficient. (Recall TERN schedules only synchroniza-

Table 2. Schedule reuse rates under four workloads; the “Schedules” column lists number of schedules in the schedule cache.

Program Workload	Reuse Rates (%)	Schedules
Apache-trace	90.3%	100
MySQL-simple	94.0%	50
MySQL-tx	44.2%	109
PBZip2-usr	96.2%	90

Figure 8. Normalized execution time when reusing schedules.

A bar with value greater (or smaller) than 1 indicates slowdown (or speedup) compared to traditional multithreading. The overhead of reusing schedules is smaller than 15% of the total execution time of traditional multithreading for most programs and up to 50% for the other two programs. Five programs run faster because TERN or PEREGRINE safely skips some blocking operations.



tions, whereas PEREGRINE additionally schedules memory accesses to make data races deterministic.)

Conclusion

By conceiving, building, applying, and evaluating StableMT systems, we have demonstrated StableMT can stabilize program behaviors for better reliability, so it works efficiently and deterministically while greatly improving precision of static analysis. Moreover, it promises to help solve the grand challenge of making parallel programming easy. However, TERN and PEREGRINE are still research prototypes, not quite ready for general adoption. Moreover, the ideas we have explored are just the start of this direction in StableMT; the bulk of the work is ahead:

System. At the system level, can we build efficient, lightweight StableMT systems that work automatically with all multithreaded programs? TERN and PEREGRINE require recording executions and analyzing source code that can be computationally demanding. As the number of cores increases, can researchers build StableMT systems that scale to hundreds of cores?

Application. At the application level, we have only scratched the surface. Improving program analysis is just one possible application; others include improving testing coverage, verifying a program with respect to a small set of dynamically enforced schedules, and optimizing thread scheduling and placement based on a schedule because it effectively predicts the future. Moreover, the idea of stabilizing schedules could apply to other parallel programming methods (such as MPI, OpenMP, and Cilk-like tasks).

Conceptual. At the conceptual level, can we reinvent parallel programming to greatly reduce the set of schedules? For instance, a multithreading system might disallow schedules by default, allowing only the schedules' developers to explicitly write code to enable. Since developers are characterized by a range of skills, we may let only the best ones decide what schedules to use, reducing the likelihood of programming errors.

All are thus invited to explore with us this fertile and exciting direction in stable multithreading and reliable parallelism.

Acknowledgments

We thank all who helped with this work.^{10,11,22,25} In particular, John Gallagher built the alias analysis in PEREGRINE, Chia-Che Tsai conducted the evaluation of TERN, and Huayang Guo helped evaluate PEREGRINE. We also thank Al Aho, Remzi Arpaci-Dusseau, Tom Bergan, Emery Berger, Luis Ceze, Xi Chen, Jason Flinn, Roxana Geambasu, Gail Kaiser, Jim Larus, Jinyang Li, Jiri Simsa, Ying Xu, and the anonymous reviewers for their many helpful comments. This work is supported in part by the National Science Foundation, including an NFS Career Award, Air Force Research Laboratory, an Air Force Office of Scientific Research Young Investigator Research Program Award, Office of Naval Research, and a Sloan Research Fellowship. □

References

- Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiawicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., Wessel, D., and Yelick, K. A view of the parallel computing landscape. *Commun. ACM* 52, 10 (Oct. 2009), 56–67.
- Aviram, A., Weng, S.-C., Hu, S., and Ford, B. Efficient system-enforced deterministic parallelism. *Commun. ACM* 55, 5 (May 2012), 111–119.
- Ball, T. and Rajamani, S.K. Automatically validating temporal safety properties of interfaces. In *Proceedings of the Eighth International SPIN Workshop on Model Checking of Software* (Toronto, May 19–20, 2001), 103–122.
- Bergan, T., Anderson, O., Devietti, J., Ceze, L., and Grossman, D. CoreDet: A compiler and runtime system for deterministic multithreaded execution. In *Proceedings of the 15th International Conference on Architecture Support for Programming Languages and Operating Systems* (Pittsburgh, PA, Mar. 13–17). ACM Press, New York, 2010, 53–64.
- Berger, E., Yang, T., Liu, T., Krishnan, D., and Novark, A. Grace: Safe and efficient concurrent programming. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications* (Orlando, FL, Oct. 25–29). ACM Press, New York, 2009, 81–96.
- Bessey, A., Block, K., Chelf, B., Chou, A., Fulton, B., Hallem, S., Henri-Gros, C., Kamsky, A., McPeak, S., and Engler, D. A few billion lines of code later: Using static analysis to find bugs in the real world. *Commun. ACM* 53, 2 (Feb. 2010), 66–75.
- Bocchino, Jr., R.L., Adve, V.S., Dig, D., Adve, S.V., Heumann, S., Komuravelli, R., Overbey, J., Simmons, P., Sung, H., and Vakilian, M. A type and effect system for deterministic parallel Java. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications* (Orlando, FL, Oct. 25–29). ACM Press, New York, 2009, 97–116.
- Cantrill, B. and Bonwick, J. Real-world concurrency. *Commun. ACM* 51, 11 (Nov. 2008), 34–39.
- Clarke, E., Grumberg, O., and Peled, D. *Model Checking*. MIT Press, Cambridge, MA, 1999.
- Cui, H., Wu, J., Tsai, C.-C., and Yang, J. Stable deterministic multithreading through schedule memorization. In *Proceedings of the Ninth Symposium on Operating Systems Design and Implementation*. USENIX Association, Berkeley, CA, 2010.
- Cui, H., Wu, J., Gallagher, J., Guo, H., and Yang, J. Efficient deterministic multithreading through schedule relaxation. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*. ACM Press, New York, 2011, 337–351.
- Devietti, J., Lucia, B., Ceze, L., and Oskin, M. DMP: Deterministic shared memory multiprocessing. In *Proceedings of the 14th International Conference on Architecture Support for Programming Languages and Operating Systems* (Washington, D.C., Mar. 7–11). ACM Press, New York, 2009, 85–96.
- Lee, E.A. The problem with threads. *Computer* 39, 5 (May 2006), 33–42.
- Liu, T., Curtsinger, C., and Berger, E.D. DTHREADS: Efficient deterministic multithreading. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles* (Cascais, Portugal, Oct. 23–26). ACM Press, New York, 2011, 327–336.
- Lu, S., Park, S., Seo, E., and Zhou, Y. Learning from mistakes: A comprehensive study on real-world concurrency bug characteristics. In *Proceedings of the 13th International Conference on Architecture Support for Programming Languages and Operating Systems* (Seattle, Mar. 1–5). ACM Press, New York, 2008, 329–339.
- Musuvathi, M., Qadeer, S., Ball, T., Basler, G., Nainar, P.A., and Neamtii, I. Finding and reproducing Heisenbugs in concurrent programs. In *Proceedings of the Eighth Symposium on Operating Systems Design and Implementation* (San Diego, Dec. 8–10). USENIX Association, Berkeley, CA, 2008, 267–280.
- Nethercote, N. and Seward, J. Valgrind: A framework for heavyweight dynamic binary instrumentation. In *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation* (San Diego, June 11–13). ACM Press, New York, 2007, 89–100.
- O'Hanlon, C. A conversation with John Hennessy and David Patterson. *Queue* 4, 10 (Dec. 2006), 14–22.
- Olszewski, M., Ansel, J., and Amarasinghe, S. Kendo: Efficient deterministic multithreading in software. In *Proceedings of the 14th International Conference on Architecture Support for Programming Languages and Operating Systems* (Washington, D.C., Mar. 9–11). ACM Press, New York, 2009, 97–108.
- Shao, Z. Certified software. *Commun. ACM* 53, 12 (Dec. 2010), 56–66.
- Wu, J., Cui, H., and Yang, J. Bypassing races in live applications with execution filters. In *Proceedings of the Ninth Symposium on Operating Systems Design and Implementation* (Vancouver, Canada, Oct. 4–6). USENIX Association, Berkeley, CA, 2010.
- Wu, J., Tang, Y., Hu, G., Cui, H., and Yang, J. Sound and precise analysis of parallel programs through schedule specialization. In *Proceedings of the ACM SIGPLAN 2012 Conference on Programming Language Design and Implementation* (Beijing, June 11–16). ACM Press, New York, 2012, 205–216.
- Xiong, W., Park, S., Zhang, J., Zhou, Y., and Ma, Z. Ad hoc synchronization considered harmful. In *Proceedings of the Ninth Symposium on Operating Systems Design and Implementation* (Vancouver, Canada, Oct. 4–6). USENIX Association, Berkeley, CA, 2010.
- Yang, J., Cui, A., Stolfo, S., and Sethumadhavan, S. Concurrency attacks. In *Proceedings of the Fourth USENIX Workshop on Hot Topics in Parallelism* (Berkeley, CA, June 7–8). USENIX Association, Berkeley, CA, 2012, 15.
- Yang, J., Cui, H., and Wu, J. Determinism is overrated: What really makes multithreaded programs hard to get right and what can be done about it? In *Proceedings of the Fifth USENIX Workshop on Hot Topics in Parallelism* (San Jose, CA, June 24–25). USENIX Association, Berkeley, CA, 2013.

Junfeng Yang (<http://www.cs.columbia.edu/~junfeng>) is an associate professor and co-director of the Software Systems Lab in the Department of Computer Science at Columbia University, New York.

Heming Cui (<http://www.cs.columbia.edu/~heming>) is a Ph.D. student and a member of the Software Systems Lab in the Department of Computer Science at Columbia University, New York.

Jingyue Wu (<http://www.cs.columbia.edu/~jingyue>) is a Ph.D. student and a member of the Software Systems Lab in the Department of Computer Science at Columbia University, New York.

Gang Hu (<http://www.cs.columbia.edu/~ganghu>) is a Ph.D. student and a member of the Software Systems Lab in the Department of Computer Science at Columbia University, New York.

Yang Tang (<http://ytang.com>) is a Ph.D. student and a member of the Software Systems Lab in the Department of Computer Science at Columbia University, New York.

DOI:10.1145/2500883

To inspire women to major in CS, take them to the Grace Hopper Celebration of Women in Computing.

BY CHRISTINE ALVARADO AND EUGENE JUDSON

Using Targeted Conferences to Recruit Women into Computer Science

THE SHORTAGE OF women in computer science (CS) is well documented. Since 2001, the Computing Research Association Taulbee Survey reports the percentage of women obtaining a bachelor's degree in CS dropped from 18.8% in 2001 to 13.8% in 2010.⁷ A 2011 study by Baumann et al.⁴ found that although some schools saw increases in the percentage of women receiving Ph.D.'s and earning faculty

positions, the percentage of women receiving bachelor's degrees continued to drop. Increasing the number of qualified women (along with other underrepresented populations) choosing to study CS is critical in combating the shortage of CS graduates in the U.S.^{16,22,23}

Harvey Mudd College (HMC) is recognized for dramatically increasing the percentage (and number) of women majoring in CS, from 12% historically to approximately 40%, where it has held steady since 2008.^{13,18} This percentage is well above the U.S. average, even among elite private schools, which, at approximately 16%, tend to have a slightly higher average than elite public schools.⁴

HMC made three changes that contributed to this gender rebalance: a revised introductory CS course, CS research opportunities for women students after their first year, and trips for first-year students to the Grace Hopper Celebration of Women in Computing (GHC; <http://gracehopper.org/>); all three are covered to some degree in Alvarado and Dodds.³ Here, we present new data and analysis that help illustrate the specific effect of attending GHC on first-year undergraduate female students' interest in CS and interest in staying in the field.

These changes are among a growing number of efforts to increase women's participation in CS. Existing practices with proven success include new introductory college CS courses (such as explored in Dodds et al.,⁸ Rich et al.,²⁰ and Summet et

» key insights

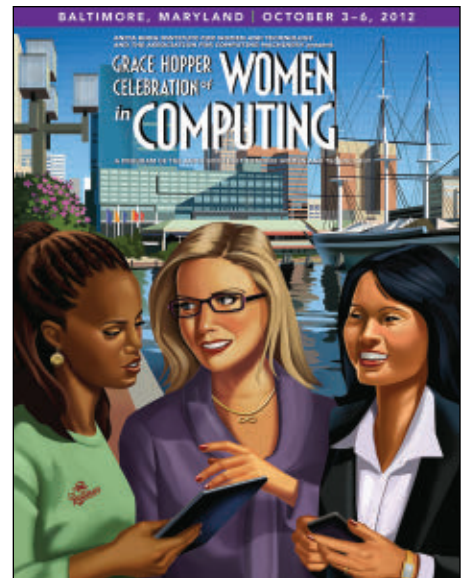
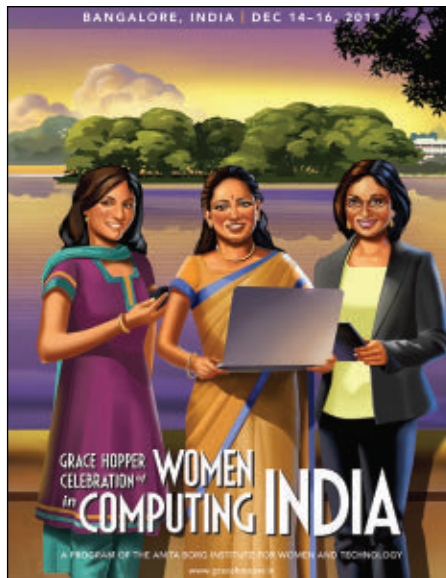
- **Attending GHC is a positive experience for female college students with little or no CS background.**
- **Attending GHC can inspire first-year female college students to take computer science classes and choose CS as their major.**
- **GHC's influence is strongest on first-year students considering CS as a possible major, though many not considering a CS major report attending GHC increased their desire to major in CS.**



GHC of Women in Computing aims to increase women's participation in CS worldwide.

al.²¹), improved high school and middle school CS education and standards,^{9,12} and many summer camps and after-school programs at all education levels (such as those described by Ericson and McKlin¹⁰ and the Intel Computer Clubhouse¹⁴). The National Center for Women & Information Technology (<http://www.ncwit.org>) provides information on dozens of practices, large and small, that have been shown to support recruitment and/or retention of women in CS. However, many of them are either expensive, particularly in terms of time investment, or have not resulted in the kind of dramatic change in students' major choices seen at HMC.

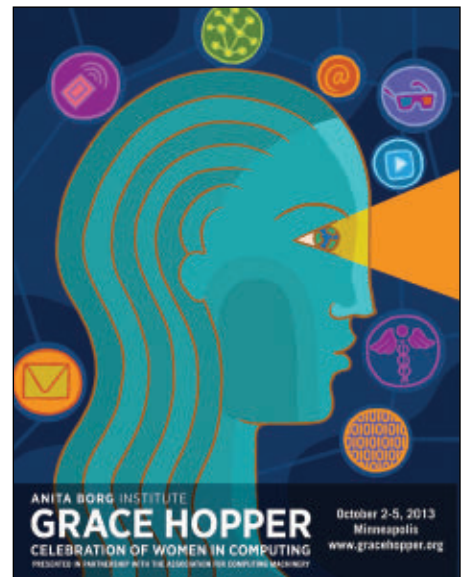
The novel contribution of the work we present here is to show how short-term intervention—attending GHC—at a critical juncture in a student's decision process (three to 18 months before declaring a major) can dramatically effect a student's major, and hence likely career choice. This approach leverages existing resources and is extremely low cost from a time-investment perspective. Moreover, it can be adopted, albeit perhaps on a smaller scale, in many contexts beyond HMC, with similar benefits. Finally, the success we report should inspire companies and universities alike to devote more resources to



sending first-year students to GHC and similar programs, making the approach financially feasible for a much larger population.

Grace Hopper Celebration

GHC is an annual conference celebrating the accomplishments of women in CS, combining technical talks, targeted workshops, panels focused on issues facing women in the field, and networking events. Since its inception in 1994, it has had a tremendous positive effect on thousands of women, including hundreds of students. Results from the GHC Evaluation and Impact Survey¹¹ indicate students feel less isolated, more committed to CS, and more inspired



as a result of attending. Analysis of data from the survey also highlights the particular importance of GHC in helping retain students through critical junctions in their educational and working careers. In 2010, 73% of the students who responded to the survey “agreed or strongly agreed that attending GHC 2010 has increased their commitment to complete their current degree program,” while 64% “agreed or strongly agreed that attending GHC 2010 has increased their intention to pursue a graduate degree in a technology field.”¹¹ The conference is so successful there is now also an Indian version (<http://gracehopper.org.in>), as well as several small “regional” celebrations of women in CS throughout the U.S.

Table 1. Number of first-year women students and upper-class student mentors attending GHC each year 2006–2011.

Year	Number of first-year students	Number of mentors
2006	8	3
2007	12	2
2008	22	6
2009	25	6
2010	35	9
2011	35	7

Most reported GHC success has involved retaining women who have already entered the field of CS. While this goal is important, what is less well understood is the effect GHC has on students who have not yet selected a major, including many first-year college students. In 2010, 46% of the students who responded to the GHC survey were undergraduates. While the survey did not report how many of them had already chosen a major, 81% of all students surveyed listed CS or a closely related field like human-computer interaction and information science as their chosen field of study. The aggregation of survey responses does not permit deciphering how the conference affects the undecideds or how it might affect the thousands of undecideds who do not attend GHC because they are not encouraged or lack the means.

School Context and Trip Goals

HMC is a relatively small (740 students) science-and-engineering-focused liberal arts college in Claremont, CA, where students declare their major in their sophomore year. Since 2006, the CS department has taken groups of first-year women students to GHC (see Table 1). Almost all were undecided about their field of study, and almost half were not even considering majoring in CS before the trip.

Table 2. Post-survey results from first-year students attending GHC 2009–2010; 1=strongly disagree, 4=neutral, 7=strongly agree.

Question	Percent of respondents (N=60)						
	1	2	3	4	5	6	7
Attending GHC gave me a better understanding of the field of CS.	2	3	2	5	20	43	25
Attending GHC changed my perception of the culture of CS.	2	2	3	8	29	39	17
Attending GHC strengthened my network of friends at HMC.	2	2	8	10	28	25	25
I met new people from other institutions at GHC.	3	7	5	7	37	17	25
I plan to keep in touch with someone from another school I met at GHC.	25	23	5	25	8	7	7
Attending GHC increased my desire to take another CS class.	3	0	0	25	23	17	32
Attending GHC increased my desire to major in CS.	3	2	7	27	30	12	20
Attending GHC was positive.	0	0	0	3	5	33	58
Given the opportunity, I would attend GHC again.	2	2	5	5	3	24	59
I would recommend other first-year students attend GHC.	2	5	0	7	8	17	62

The department recruits students by email during the summer before their freshman year. As of summer 2013 it has been able to take all students who apply, though in some years it has had to create a temporary wait list until additional funding was secured. Detailed logistics of how the department funds and organizes these trips is beyond our scope here; see Alvarado et al.¹ for more logistical detail and Alvarado² for practical resources developed to help others run similar trips.

Since the early 1990s all students at HMC have been required to take a version of CS1 (Introduction to CS) in their first semester. Yet, until 2006, the CS department struggled to recruit women to the CS major. Despite HMC’s technical focus, 59% of women and 31% of men who entered HMC in 2010 had not taken a single computing or CS class.

A primary objective of the annual GHC trip is to recruit (and retain) first-year women into CS by combating some of the documented factors that keep women from pursuing CS. Previous work has shown a lack of confidence, (mis)perception of a geeky or hostile culture, misunderstanding of the field, and lack of mentors and support networks as barriers to women’s entry into CS.^{5,6,15,17,19} Students gain confidence at GHC by hearing female role models talk about their own paths to success. They see a CS culture that is very different from the stereotypical nerd culture they might expect. They are exposed to real-world CS applications, interact with CS practitioners, and learn first-hand about dynamic CS-related companies. Finally, the very experience of attending GHC gives them a chance to increase their professional network of friends and colleagues.

Student Attitudes


Since 2007, we have conducted an annual survey of HMC’s GHC attendees to try to understand the effect the trip has on the students who attend. In our own previous work³ we reported aggregate results from these surveys, showing, overall, students report positive feedback about GHC. Here, we look more closely at the effect it has on students from diverse backgrounds. We find that while the trips

have a positive effect on almost all students who attend, those with some interest in a CS major before the trip or experience with CS before college are generally affected more positively by their experience. Nonetheless, we also see the trip has a strong positive influence on a subset of students with no CS experience and no interest in a CS major before the conference.


In 2007 and 2008 we informally surveyed HMC's GHC attendees regarding their thoughts about GHC following their return from the conference. From 2009 onward, we more formally assessed HMC GHC attendees' attitudes about CS and the GHC effect with our own pre- and post-surveys. We administered the pre-survey just prior to students' attendance to gather specific student profile and attitudinal information. The pre-survey prompted students to report their CS experience (measured by which required first-year CS class they had attended) and which major(s) they were currently interested in. The pre-survey also required them to rate on a Likert scale of 1–7 (1=not at all interested, 7=extremely interested) how interested they were in taking another CS course and in majoring in CS. Following attendance at GHC in 2009 and later, we administered the post-survey to gather information about attendees' sentiments regarding the degree to which GHC participation had affected their views of CS; see Table 2 for Likert-item post-survey questions. The post-survey also asked them to elaborate on their answers to the Likert-scale items and list the most meaningful part(s) of the conference for them. Our GHC trip website includes the complete survey instruments.²

Here, we report results from the 2009 and 2010 surveys, each involving a 100% response rate. Aggregate results from 2007 and 2008 are in Alvarado and Dodds,³ reflecting the aggregate results covered here.

Survey results (see Table 2) indicate GHC is indeed effective in addressing the barriers that keep women from choosing to study CS. The conference gave a majority of participants a better understanding of CS and changed their perception of the CS culture. The majority of participants agreed



The trip has a strong positive influence on a subset of students with no CS experience and no interest in a CS major before the conference.



(responding 5, 6, or 7) that GHC increased their desire to take another CS class and major in CS. The one area where the conference was (slightly) weaker was in network building. While a particular annual trip allowed students to build their network within their home institutions, many responded that they did not plan to keep in touch with anyone else they met at the conference.

Results by interest in a CS major.

While the aggregate data indicates overall success, we were also interested in how different groups were individually affected by the conference. To answer, we grouped participants according to their responses in the pre-survey. We first categorized participants according to whether or not they listed CS as a possible major in the pre-survey. If a student listed CS (either alone or as one of several possible majors), we say the student was considering a CS major. If a student did not list CS, we say the student was not considering a CS major. We then ran a series of non-parametric Mann-Whitney U tests to identify differences between the groups. The women who were considering a CS major had significantly higher mean rank scores than those not considering a CS major in three categories:

Take another CS class. Increased desire to take another CS class (CS mean rank=35.28, non-CS mean rank = 26.04, $z = 2.344$, $p < 0.05$);

Major in CS. Increased desire to major in CS (CS mean rank=35.50, non-CS mean rank = 24.79, $z = 2.344$, $p < 0.05$); and

Attend again. If given the opportunity would attend GHC again (CS mean rank=35.50, non-CS mean rank=24.79, $z = 2.219$, $p < 0.05$).

Figure 1 indicates both groups felt GHC increased their desire to take another CS class. The students considering pursuing a CS major included a greater proportion of those whose course plans were strongly affected by the conference. However, a substantial portion of those not considering a CS major before the conference (18%) were equally inspired by the conference, and a large portion of them (36%) agreed the conference increased their desire to take another CS course.

Figure 1. Distribution of responses to the statement “Attending GHC increased my desire to take another CS course” broken down by whether or not the student was considering a CS major before the conference.

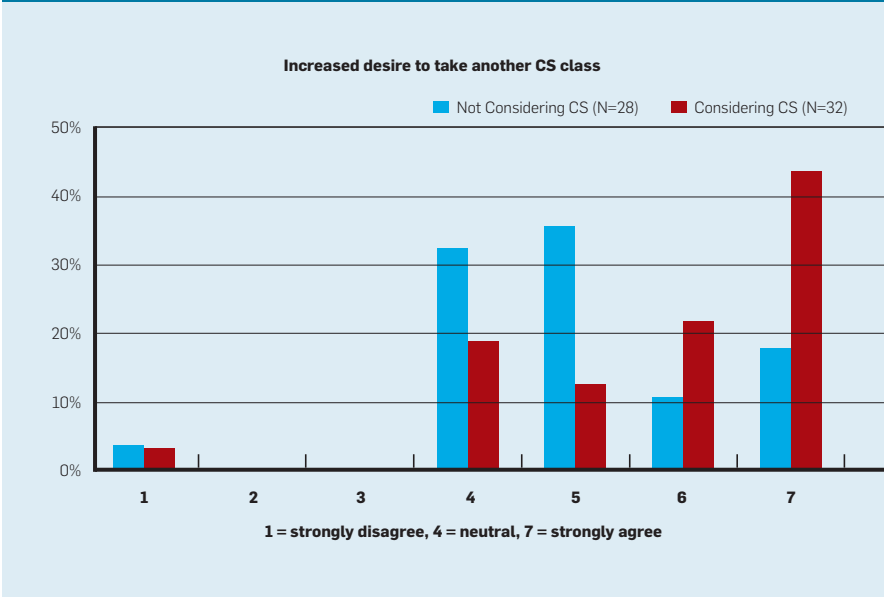


Figure 2. Distribution of responses to the statement “Attending GHC increased my desire to major in CS” broken down by whether or not the student was considering a CS major before the conference.

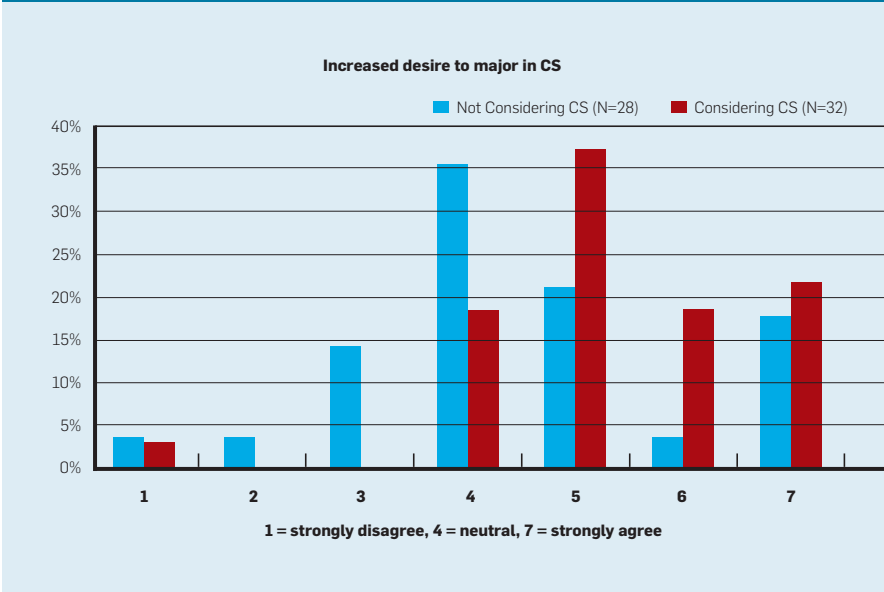


Figure 2 includes two distributions of responses to the statement “GHC increased my desire to major in CS.” They are similar, but students considering a CS major prior to the conference responded slightly more positively to the statement. Again, the conference positively affected a large number of the women not already considering a CS major. While the largest portion of respondents (36%) from the group not considering a CS major were neutral, more than twice as many respondents from this group

agreed (43%) with the statement than disagreed (21%). In addition, 18% of respondents from the non-CS-major group strongly agreed with the statement.

Although the groups’ responses differed significantly regarding whether or not they would attend again, Figure 3 shows differences mainly concerned the level of peak enthusiasm of the responses. The difference in enthusiasm over attending GHC again is likely related to a difference in desire between the groups to

continue with CS. Those listing CS as a possible major before the conference became more interested in pursuing it as a result of the conference. They would therefore be expected to be more enthusiastic about future opportunities in CS.

Results by introductory CS course.

We next categorized participants based on which introductory CS class they were enrolled in at the time of the conference they attended. At HMC in 2009 and 2010 all students were required to take introductory CS in their first semester. The CS department placed them into one of four courses based on their interests and experience: CS5 “Gold” (general introductory CS, for students with no experience); CS5 “Green” (biology-themed introductory CS, all experience levels); CS5 “Black” (general introductory CS, some experience); and CS42 (general advanced introductory CS, lots of experience). CS5 and CS42 were HMC course numbers corresponding roughly to CS1- and CS2-level courses, respectively.¹

We performed an ANOVA test that indicated a statistically significant difference among the four groups regarding the degree to which attendance at GHC increased their desire to pursue a CS major ($F(2, 31) = 9.910, p < 0.01$). A post-hoc least-square difference test indicated a statistically significant difference between the CS5 “Black” (some experience) group and the CS 5 “Gold” (no experience) group ($p < 0.05$) and between the CS5 “Black” group and the CS5 “Green” (bio-themed) group ($p < 0.01$). In each case, the CS5 “Black” group’s agreement with the statement “Attending GHC increased my desire to major in CS” was significantly stronger than either of the two other groups. There was also a statistically significant difference between the CS42 (lots of experience) group and the CS5 “Green” group, with the CS42 group rating this item significantly higher (see Figure 4). These results indicate more experience in CS may lead to greater propensity to be influenced by the GHC experience and consequently a desire to major in CS.

Enrollment Patterns

Survey results from 2009 and 2010

indicate GHC had a positive effect on attendees, especially those with some CS experience or interest. By examining enrollment patterns, we also found attendees were much more likely to take another course and major in CS than female non-attendees.

Consistent with these results, we also found those attending GHC take a second CS course at a significantly higher rate than those who do not attend GHC. From 2006 to 2010, 52% (53 of 102) of first-year female GHC attendees went on to take another CS course at some point in their college careers, compared to only 31% (80 of 261) of female students who did not attend. This difference is statistically significant (Fisher's test, $p < 0.001$), at least part of which can be accounted for by self-selection; those more interested in CS more often opted to go on the trip. However, in the context of our survey results, at least some of this difference was likely due to the trip itself.

Next, we found GHC attendees also majored in CS at a much higher rate than female students who did not attend. For this comparison we restricted our focus to only the incoming classes 2006–2009 or juniors and above in fall 2011, as HMC students were not required (or even encouraged) to declare their major until the end of their sophomore year. In this time frame, 37% (24 of 65) of first-year attendees went on to major in CS, while only 10% (20 of 196) of the women who did not attend majored in CS, a difference that is statistically significant ($p < 0.0001$).

Again, these results were influenced by the fact that students interested in majoring in CS chose to go on the trip at a higher rate than those not interested in majoring in CS. However, by considering students' expressed major interests when enrolling at HMC it appears GHC indeed influenced the choice of major for those who came to HMC intending to major in CS, as well as for those who did not.

Over the same time frame (incoming classes 2006–2009), incoming students who at enrollment time listed CS as their desired major ended up majoring in CS at a higher rate than those who listed CS as their desired major and did not attend GHC. Of

the 31 female students who declared a desire to major in CS when they enrolled, 11 went to GHC in their first year, and 20 did not. Of the 11, 91% (10) went on to declare a CS major vs. only 60% (12 of 20) of those who did not attend, though this difference is not statistically significant, possibly due to small sample sizes ($p=0.11$).

Incoming students who did list CS as their desired major when they enrolled at HMC and who attended GHC in their first year majored in CS at a significantly higher rate than

students who did not list CS as their desired major and did not attend GHC. Of the 55 students who did not declare an interest in majoring in CS and attended GHC as first-year students 2006–2009, 25% (14) went on to be CS majors. Of the 175 first-year female students 2006–2009 who did not declare an interest in a CS major and did not attend GHC, only 10% (17) went on to be CS majors, a difference that is statistically significant (Fisher's test, $p < 0.01$). This difference is likely still biased by the fact that those

Figure 3. Distribution of responses to the statement "Given the opportunity, I would attend GHC again" broken down by whether or not the student was considering a CS major before the conference.

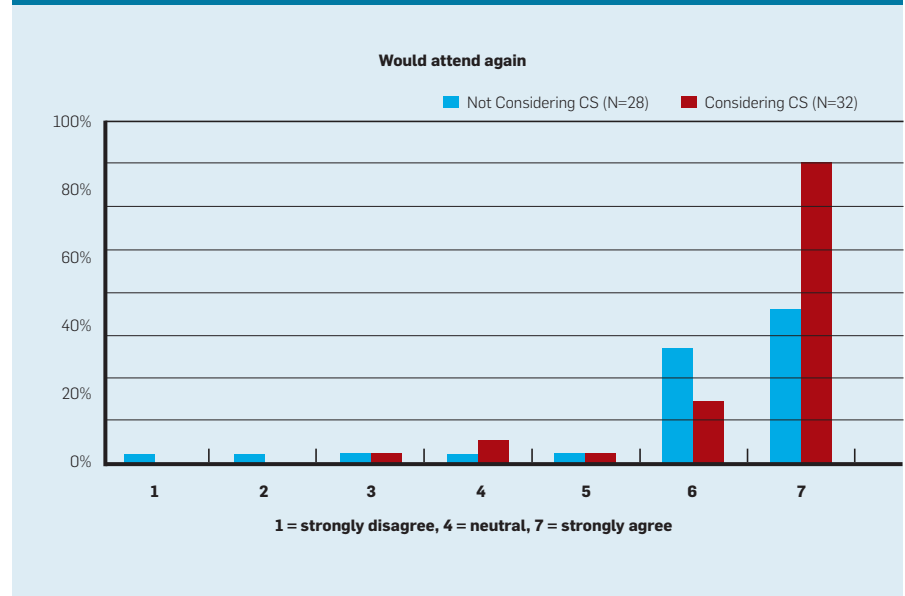
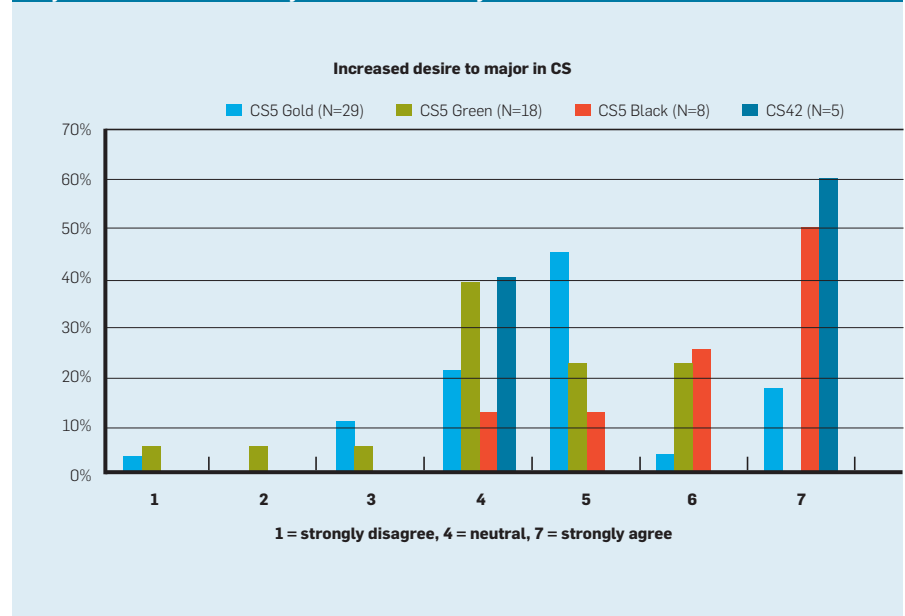


Figure 4. Distribution of responses to the statement "Attending GHC increased my desire to major in CS" broken down by which introductory CS course the student was enrolled in.



who were more receptive to CS were the ones who attended the conference. But the fact remains that 25% of those not considering a CS major and attended GHC ended up majoring in CS after attending the conference.


Beyond HMC

As effective as we found GHC to be as a recruiting tool, we usually meet skepticism as to whether these results might be replicated elsewhere. It is not possible for every school to take every interested first-year female student to GHC. In fall 2013, we began experimenting with arranging the trips at the University of California, San Diego, but it will be several years before we have concrete results, and disseminating the results will be integral to making the trips available to many more deserving students. Here, we address some of the biggest concerns about the scalability of this approach.


Finances. The first question many people ask about a GHC trip is: “Who pays for all this?” Since 2006, the HMC trip organizers, including one of the authors, have received a combination of internal and external funding to cover the total trip cost for 20 to 50 students per year. They have worked with HMC’s advancement office to identify potential donors; when approaching them they cite the positive effect GHC has on students and pitch the trip concept as an opportunity to expand the overall CS talent pool. They have also had success with local companies that desire to increase the number of qualified potential employees in the greater Los Angeles area and alumni and philanthropists who support the issue in the interest of social justice. Individual gifts have ranged from \$1,000 (approximate cost of one student) to \$25,000.

They have been pleased to find fundraising for women in computing and specifically this conference is not as difficult as they thought it would be. Furthermore, as investors value concrete results, the results presented here should make it even easier to motivate university departments and companies alike to fund such efforts.

Still, fundraising can be a challenge. Fortunately, there are opportunities to reduce the cost while main-



Those attending GHC take a second CS course at a significantly higher rate than those who do not attend GHC.



taining the benefits. First, conference organizers have historically charged students nothing to attend, but many would be willing and able to pay a small fee. When they have had to create a temporary wait list for a trip, as in 2009 and 2010, many wait-listed students offered to pay their own way if funding was inadequate. Even a small amount of money from each one could make the available funding go further.

Second, beyond the main GHC conference, several regional Celebrations of Women in Computing have emerged (<http://women.acm.org/celebrations>). Although no studies have yet shown these celebrations provide a similar effect on students’ major decisions, the important aspects of community, confidence building, and real-world CS context are still available. They may, in fact, be a more comfortable and less-overwhelming venue for students who know nothing about CS.

Early vs. late major declarations. Some people say, “This would not work at a school where students have to declare their major before they enroll.” At such schools, early GHC attendance might still serve as a retention mechanism for women who might otherwise drop out before establishing themselves in the major. Furthermore, preliminary evidence suggests GHC trips might increase the number of women who enroll at the school even before they attend GHC. Of the 35 incoming HMC women students who signed up for the GHC trip in 2010, eight had heard about HMC’s trips to GHC before deciding to come to HMC, and six of those eight reported the trips increased their desire to attend HMC.

But “Why,” others say, “would students not interested in CS want to attend this conference anyway?” The answer comes from our pre-survey. Many students simply said they want an experience at an academic conference or are just interested in learning about a field they had never heard of before. These students were just beginning college, and many were eager to take advantage of as many new opportunities as they could.

Finally, targeting the trip to students in their last year of high school

may have an even stronger effect on their major choices; we encourage others to explore this idea.

Scaling to very large schools. Those from large schools sometimes say, “This would never work at my school; there are way too many students.” HMC was able to take approximately 33% of the first-year female students to GHC. Even approaching this level of participation would be impossible at a large school with, say, 5,000 first-year women, instead of 100, no matter how much funding is available. However, even if the percentage of affected students is smaller, recruiting 10 female students to CS increases the overall population of women in CS the same, whether they are in a department of 100 or 1,000. Moreover, these students can have a big effect on the culture of a department, likewise no matter how big.

Based on our results, we recommend that as the number (or percentage) of students scales down, a department should aim for a mix of CS experience, interest, and major plans. It is important to take some students who are interested in CS, because for them the conference will likely have an important effect. However, it is also important to take some students who are not considering a CS major, as the conference could affect dispositions.

GHC size limitations. Finally, perhaps the most limiting factor to the widespread implementation of such trips is the limitations on GHC itself, which sells out every year. While we agree, we also strongly believe GHC will continue to expand and evolve to meet the related demands. As evidence consider the GHC India conference, which emerged precisely to serve more of the population that could not attend GHC North America. We also point again to the regional celebrations now in more than a dozen regions, with the intention of serving a bigger and more geographically dispersed audience GHC alone could not serve.

Conclusion

This study supports the idea that having first-year female college students participate in GHC is an effective tool for recruiting and retaining first-year

women students to CS. In particular, our data suggests students attending GHC are more likely to both major in CS and take more CS courses. This result holds for students who intended to major in CS when they started college and for those who did not.

As much as we believe GHC can be an effective tool for recruiting and retaining women students in CS, alone it is not enough. Established best practices (such as those outlined by the National Center of Women & Information Technology), including mentoring, research opportunities, pair programming, and engaging introductory curricula, are essential for not only recruiting more women students but retaining them once they choose CS as a major and a career. At HMC, GHC trips complement two other major initiatives: an engaging introductory CS course and research experiences for women after their first year.

Diversifying and expanding CS education to meet the needs of the 21st century work force requires a deliberate plan that includes many initiatives. Taking first-year women students to GHC or similar celebrations should be a key part of it.

Acknowledgments

We would like to thank Alan and Kathy Eustace, the IBM Almaden Research Center, Qualcomm, and Boeing Corp., as well as Maria Klawe and the HMC President’s Office and Department of Computer Science, for providing generous funding for first-year HMC students to attend GHC. ■

References

- Alvarado, C., Dodds, Z., and Libeskind-Hadas, R. Increasing women’s participation in computing at Harvey Mudd College. *ACM Inroads* 3, 4 (Dec. 2012), 55–64.
- Alvarado, C. First Year Women @ GHC; <https://sites.google.com/a/eng.ucsd.edu/alvarado/projects/first-year-women-ghc>
- Alvarado, C. and Dodds, Z. Women in CS: An evaluation of three promising practices. In *Proceedings of the 41st SIGCSE Technical Symposium on Computer Science Education* (Milwaukee, Mar. 10–13). ACM Press, New York, 2010, 57–61.
- Baumann, D., Hambrusch, S., and Neville, J. Gender demographics trends and changes in U.S. CS departments. *Commun. ACM* 54, 11 (Nov. 2011), 38–42.
- Beyer, S., Rynes, K., Perrault, J., Hay, K., and Haller, S. Gender differences in computer science students. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (Reno, NV, Feb. 19–23). ACM Press, New York, 2003, 49–53.
- Cohoon, J.M. Toward improving female retention in the computer science major. *Commun. ACM* 44, 5 (May 2001), 108–114.
- Computing Research Association. Taulbee Survey data

- from 2001 and 2010; <http://www.cra.org/resources/taulbee>
- Dodds, Z., Libeskind-Hadas, R., Alvarado, C., and Kuening, G. Evaluating a breadth-first CS 1 for scientists. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (Portland, OR, Mar. 12–15). ACM Press, New York, 2008, 266–270.
- Ericson, B., Guzdial, M., and Biggers, M. Improving secondary CS education: Progress and problems. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, KY, Mar. 7–10). ACM Press, New York, 2007, 298–301.
- Ericson, B. and McKlin, T. Sustainable and effective computing summer camps. In *Proceedings of the 43rd SIGCSE Technical Symposium on Computer Science Education* (Raleigh, NC, Feb. 29–Mar. 3). ACM Press, New York, 2012, 289–294.
- Gilmartin, S. and Simard, C. *Grace Hopper Celebration of Women in Computing 2010 Evaluation and Impact Report*. Technical Report, Anita Borg Institute for Women in Technology, Palo Alto, CA, 2010.
- Goode, J. Reprogramming high school computer science. *Commun. ACM* 51, 11 (Nov. 2008), 31–33.
- Haines, A. How one college president is breaking down barriers for women in tech. *Forbes* (Dec. 12, 2013); <http://www.forbes.com/sites/85broads/2011/12/12/how-one-college-president-is-breaking-down-barriers-for-women-in-tech/>
- Intel Computer Clubhouse; <http://www.computerclubhouse.org>
- Katz, S., Allbritton, D., Aronis, J., Wilson, C., and Soffa, M.L. Gender, achievement, and persistence in an undergraduate computer science program. *SIGMIS Database* 37, 4 (Fall 2006), 42–57.
- Klawe, M. and Shneiderman, B. Crisis and opportunity in computer science. *Commun. ACM* 48, 11 (Nov. 2005), 27–28.
- Klawe, M., Whitney, T., and Simard, C. Women in computing: Take 2. *Commun. ACM* 52, 2 (Feb. 2009), 68–76.
- Levi, A. A campus champion for women in computer science. *Bloomberg Businessweek* (Sept. 22, 2011); <http://www.businessweek.com/magazine/a-campus-champion-for-women-in-computer-science-09222011.html>
- Margolis, J. and Fisher, A. *Unlocking the Clubhouse: Women in Computing*. MIT Press, Cambridge, MA, 2003.
- Rich, L., Perry, H., and Guzdial, M. A CS1 course designed to address interests of women. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, VA, Mar. 3–7). ACM Press, New York, 2004, 190–194.
- Summet, J., Kumar, D., O’Hara, K., Walker, D., Ni, L., Blank, D., and Balch, T. Personalizing CS1 with robots. In *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education* (Chattanooga, TN, Mar. 4–7). ACM Press, New York, 2009, 433–437.
- U.S. Department of Homeland Security. *U.S. Faces Critical Shortage of Computer Scientists*, Jan. 2010; <http://www.homelandsecuritynewswire.com/us-faces-critical-shortage-computer-scientists>
- Worthington, D. Computer science lacks women, minorities. *Software Development Times on the Web* (Sept. 4, 2009); <http://sdt.bz/33742>

Christine Alvarado (alvarado@cs.ucsd.edu) is a lecturer with security of employment in the Computer Science and Engineering Department at the University of California, San Diego.

Eugene Judson (eugene.judson@asu.edu) is an assistant professor of science education in the Mary Lou Fulton Teachers College at Arizona State University, Tempe, AZ.

DOI:10.1145/2500873

In the same way businesses use big data to pursue profits, governments use it to promote the public good.

BY GANG-HOON KIM, SILVANA TRIMI, AND JI-HYONG CHUNG

Big-Data Applications in the Government Sector

BIG DATA, A general term for the massive amount of digital data being collected from all sorts of sources, is too large, raw, or unstructured for analysis through conventional relational database techniques. Almost 90% of the world's data today was generated during the past two years, with 2.5 quintillion bytes of data added each day.⁷ Moreover, approximately 90% of it is unstructured. Still, the overwhelming amount of big data from the Web and the cloud offers new opportunities for discovery, value creation, and rich business intelligence for decision support in any organization. Big data also means new challenges involving complexity, security, and risks to privacy, as well as a need for new technology and human skills.

Big data is redefining the landscape of data management, from extract, transform, and load, or ETL, processes to new technologies (such as Hadoop) for cleansing and organizing unstructured data in big-data applications.

Although the business sector is leading big-data-application development, the public sector has begun to derive insight to help support decision making in real time from fast-growing in-motion data from multiple sources, including the Web, biological and industrial sensors, video, email, and social communications.³ Many white papers, journal articles, and business reports have proposed ways governments can use big data to help them serve their citizens and overcome national challenges (such as rising health care costs, job creation, natural disasters, and terrorism).⁹ There is also some skepticism as to whether it can actually improve government operations, as governments must develop new capabilities and adopt new technologies (such as Hadoop and NoSQL) to transform it into information through data organization and analytics.⁴

Here, we ask whether governments are able to implement some of today's big-data applications associated with the business sector. We first compare the two sectors in terms of goals, missions, decision-making processes, decision actors, organizational structure, and strategies (see the table here), then turn to several current applications in technologically advanced

» key insights

- **Businesses, governments, and the research community can all derive value from the massive amounts of digital data they collect.**
- **Governments of leading ICT countries have initiated big-data application projects to enhance operational efficiency, transparency, citizens' well-being and engagement in public affairs, economic growth, and national security.**
- **Analyzing big-data application projects by governments offers guidance for follower countries for their own future big-data initiatives.**



countries, including Australia, Japan, Singapore, South Korea, the U.K., and the U.S. Also examined are some business-sector big-data applications and initiatives that can be implemented by governments. Finally, we suggest ways for governments of follower countries to pursue their own future big-data strategies and implementations.

Business and Government Compared

Although the primary missions of businesses and governments are not in conflict, they do reflect different goals and values. In business, the main goal is to earn profits by providing goods and services, developing/sustaining a competitive edge, and satisfying customers and other stakeholders by providing value. In government, the main goal is to maintain

domestic tranquility, achieve sustainable development, secure citizens' basic rights, and promote the general welfare and economic growth.

Most businesses aim to make short-term decisions with a limited number of actors in a competitive market environment. Decision making in government usually takes much longer and is conducted through consultation and mutual consent of a large number of diverse actors, including officials, interest groups, and ordinary citizens. Many well-defined steps are therefore required to reduce risk and increase the efficiency and effectiveness of government decision making.¹⁸ It follows that big-data applications likewise differ between public and private sectors.

Dataset Attributes Compared

The big-data environment reflects

the evolution of IT-enabled decision-support systems: data processing in the 1960s, information applications in the 1970s–1980s, decision-support models in the 1990s, data warehousing and mining in the 2000s, and big data today. The big-data era is at an early stage, as most related technology and analytics applications were first introduced only around 2010.⁴

The attributes and challenges of big data have been described in terms of “three Vs”: volume, velocity, and variety (see Figure 1). Volume is big data's primary attribute, as terabytes or even petabytes of it are generated by organizations in the course of doing business while also complying with government regulations. Velocity is the speed data is generated, delivered, and processed; that is, big data is so large and difficult to manage and to

extract value from that conventional information technologies are not effective for its management.¹³ Variety is that data comes in all forms: structured (traditional databases like SQL); semi-structured (with tags and markers but without formal structure like a database); and unstructured (unorganized data with no business intelligence behind it).

The concept of big data has evolved to imply not only a vast amount of the data but also the process through which organizations derive value from it. Big data, synonymous today with business intelligence, business analytics, and data mining, has shifted business intelligence from reporting and decision support to prediction and next-move decision making.¹³ New data-management systems aim to meet the challenges of big data; for example, Hadoop, an open-source platform, is the most widely applied technology for managing storage and access, overhead associated with large datasets, and high-speed parallel processing.²² However, Hadoop is a challenge for many businesses, especially small- and mid-size ones, as applications require expertise and experience not widely available and may thus need outsourced help. Finding the right talent to analyze big data is perhaps the greatest challenge for business organizations, as required skills are neither simple nor solely technology-oriented. Searching for and finding competent data scientists (in data

mining, visualization, analysis, manipulation, and discovery) is difficult and expensive for most organizations.

Other commercial big-data technologies include the Cassandra database, a Dynamo-based tool that can store two million columns in a single row, allowing inclusion of a large amount of data without prior knowledge of how it is formatted.¹³ Another challenge for businesses is deciding which technology is best for them: open source technology (such as Hadoop) or commercial implementations (such as Cassandra, Cloudera, Hortonworks, and MapR).

Governments deal not only with general issues of big-data integration from multiple sources and in different formats and cost but also with some special challenges. The biggest is collecting data; governments have difficulty, as the data not only comes from multiple channels (such as social networks, the Web, and crowdsourcing) but from different sources (such as countries, institutions, agencies, and departments). Sharing data and information between countries is a special challenge, as shown by the terrorist bombing attack on the Boston Marathon in April 2013. National governments must be prepared and willing to share data and build systems for crime prevention and fighting. As reported in the public media, the Boston Marathon tragedy might have been prevented if the Russian secret services had shared critical information about the

terror suspects with U.S. intelligence agencies. In addition, sharing information across national boundaries involves language translation and interpretation of text semantics (meaning of content) and sentiment (emotional content) so the true meaning is not lost. Dealing with language requires sophisticated and costly tools.

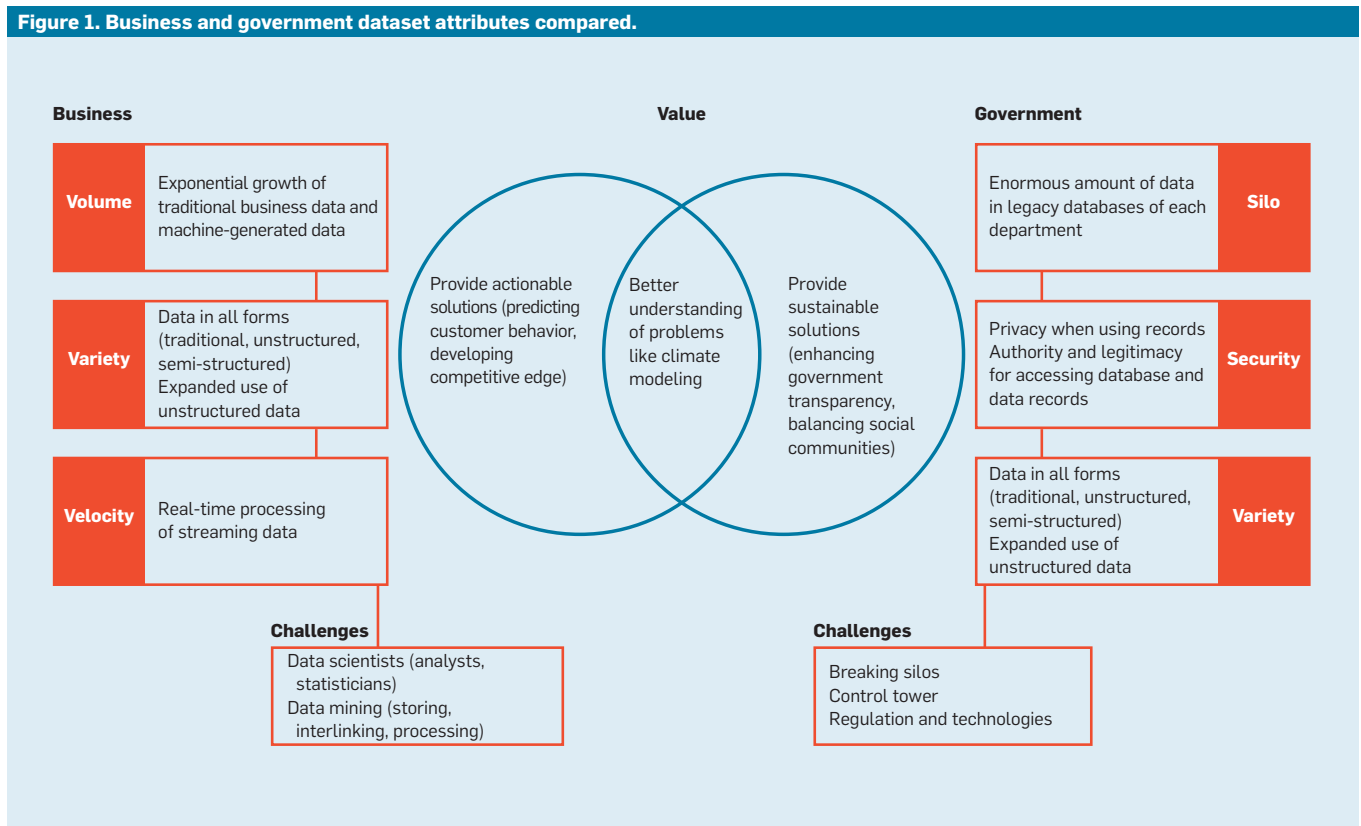
Data sharing within a country among different government departments and agencies is another challenge. The most important difference of government data vs. business data is scale and scope, both growing steadily for years. Governments, both local and national, in the process of implementing laws and regulations and performing public services and financial transactions accumulate an enormous amount of data with attributes, values, and challenges that differ from their counterparts in the business sector.

Government big-data issues can be categorized as silo, security, and variety. Each government agency or department typically has its own warehouse, or silo, of confidential or public information, with agencies often reluctant to share what they might consider proprietary data. The “tower of Babel” in which each system keeps its data isolated from other systems complicates trying to integrate complementary data among government agencies and departments. Communication failure is sometimes the issue for data integration;¹⁹ for example, in the U.K., a coalition of police departments and hospitals intended to share data on violent crimes has been reported as a failure due to a lack of communication among participating organizations.¹⁹ Another challenge for sharing and organizing government data involves finding a cohesive format that would allow for analytics in the legacy systems of different agencies. Even though most government data is structured, rather than semi-structured or unstructured, collecting it from multiple channels and sources is a further challenge. Then there is the lack of standardized solutions, software, and cross-agency solutions for extracting useful information from discrete datasets in multiple government agencies and insufficient funding due to government austerity measures to develop and implement these solutions.

Attributes of business- and government-sector projects.

Attribute	Business Firm	Government
Goal	Profit to stakeholders	Domestic tranquility, sustainable development
Mission	Development of competitive edge, customer satisfaction	Security of basic rights (equality, liberty, justice), promotion of general welfare, economic growth
Decision Making	Short-term decision-making processes for maximizing self-interest and minimizing cost	Long-term decision-making processes for maximizing self-interest and promoting the public interest
Decision Actors	Limited number of decision actors	Diverse decision actors
Organizational Structure	Hierarchical	Governance
Financial Resources	Revenue	Taxes
Nature of Collective Activity	Competition and engagement	Cooperation and checking

Figure 1. Business and government dataset attributes compared.



Governments must also address related legality, security, and compliance requirements when using data. There is a fine line between collecting and using big data for predictive analysis and ensuring citizens' rights of privacy. In the U.S., the USA PATRIOT Act allows legal monitoring and sometimes spying on citizens; the Electronic Communication Privacy Act allows email access without warrant; the proposed Cyber Intelligence Sharing and Protection Act (CISPA) (not enacted as of February 2014) raises concern, as it might position the U.S. government toward the ultimate big-data end game—access to all data for all entities in the U.S.¹⁴ Even though the intent is to prevent attacks from both domestic and foreign sources against networks and systems, CISPA raises concerns of misconstrued profiling and/or inappropriate use of information.

Data security is the primary attribute of government big data, as collecting, storing, and using it requires special care. However, most big-data technologies today, including Cassandra and Hadoop, lack sufficient security tools, making security another challenge for governments.

Compliance in highly regulated industries (such as financial services and health care) is yet another obstacle for gathering data for big-data government projects; for example, U.S. health-care regulations must be addressed when extracting knowledge from health-related big data. The two U.S. laws posing perhaps the greatest obstacle to big-data analytics in health care are the Health Insurance Portability and Accountability Act (HIPAA) and the Health Information Technology for Economic and Clinical Health Act (HITECH). HIPAA protects the privacy of individually identifiable health information, provides national standards for securing electronic data and patient records, and sets rules for protecting patient identity and information in analyzing patient safety events. HITECH expanded HIPAA in 2009 to protect the health records and electronic use of health information by various institutions. Together, these laws limit the amount and types of health records used for big-data analytics in health care. Because big data by definition involves large-scale data, these laws complicate collecting data and performing analysis on such a scale. As

of February 2014, health-care information in the U.S. intended for big-data analytics is collected only from volunteers willing to share their own.

Businesses use big data to address customer needs and behavior, develop unique core competencies, and create innovative products and services. Governments use it, along with predictive analytics to enhance transparency, increase citizen engagement in public affairs, prevent fraud and crime, improve national security, and support the well-being of people through better education and health care.

Choosing and implementing technology to extract value and finding skilled personnel are constant challenges for businesses and governments alike. However, the challenges for governments are more acute, as they must look to break down departmental silos for data integration, implement regulations for security and compliance, and establish sufficient control towers (such as the Federal Data Center in the U.S.).

Big-Data Applications

Comparing the big-data applications of leading e-government countries can reveal where current and future appli-

cations are focused and serve as a guide for follower countries looking to initiate their own big-data applications:

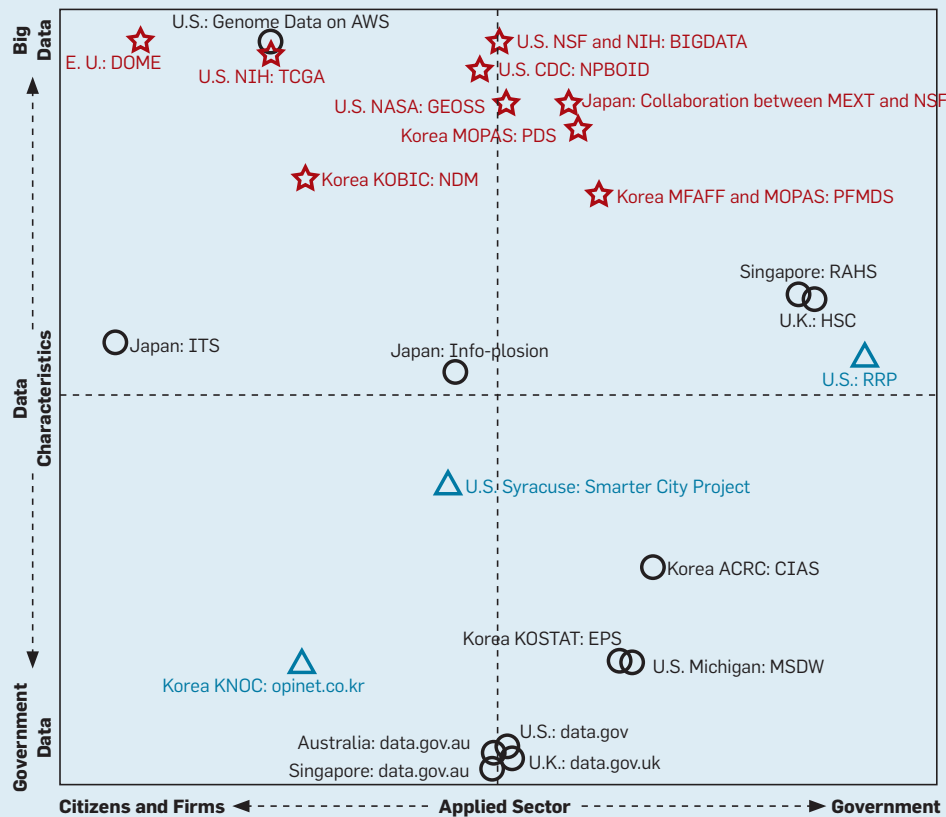
U.S. To manage real-time analysis of high-volume streaming data, the U.S. government and IBM collaborated in 2002 to develop a massively scalable, clustered infrastructure.¹ The result, IBM InfoSphere Stream and IBM Big Data, both widely used by government agencies and business orga-

nizations, are platforms for discovery and visualization of information from thousands of real-time sources, encompassing application development and systems management built on Hadoop, stream computing, and data warehousing.

In 2009, the U.S. government launched Data.gov as a step toward government transparency and accountability. It is a warehouse con-

taining 420,894 datasets (as of August 2012) covering transportation, economy, health care, education, and human services and the data source for multiple applications: 1,279 by governments, 236 by citizens, and 103 mobile-oriented.²¹ In 2010, the President's Council of Advisors on Science and Technology (the primary mechanism the federal government uses to coordinate its unclassified

Figure 2. Government data and big-data practices and initiatives.



- Japan** Collaboration of Ministry of Education, Culture, Sports, Science, and Technology and National Science Foundation
- Japan** Intelligent Traffic System
- Korea** Anti-Corruption and Civil Rights Commission of Korea: Complaints Information Analysis Center
- Korea** Statistics Korea: Employment Position Statistics
- Korea** Ministry for Food, Agriculture, Forestry, and Fisheries and Ministry of Public Administration and Security: Preventing Foot and Mouth Disease Syndrome System
- Korea** Ministry of Public Administration and Security: Preventing Disasters System
- Singapore** Risk Assessment and Horizon Scanning
- U.K.** Horizon Scanning Center
- U.S.** Centers for Disease Control and Prevention: Networked Phylogenomics for Bacteria and Outbreak ID
- U.S.** Genome Data on Amazon Web Services
- U.S.** Michigan: Michigan Statewide Data Warehouse
- U.S.** National Aeronautics and Space Administration: Global Earth Observation System of Systems
- U.S.** National Science Foundation and National Institutes of Health: BIGDATA
- U.S.** Return in Review

networking and information technology research investments) spelled out a big-data strategy in its report *Designing a Digital Future: Federally Funded Research and Development in Networking and Information Technology*.¹⁵ In 2012, the Obama Administration announced the Big Data Research and Development Initiative,¹² a \$200 million investment involving multiple federal departments and agencies, including the White House Office of Science and Technology Policy, National Science Foundation (NSF), National Institutes of Health (NIH), Department of Defense (DoD), Defense Advanced Research Projects Agency, Department of Energy, Health and Human Services, and U.S. Geological Survey. The main objectives were to advance state-of-the-art core big-data technologies; accelerate discovery in science and engineering; strengthen national security; transform teaching and learning and expand the work force needed to develop and use big-data technologies.¹¹

As of February 2014, NIH has accumulated hundreds terabytes of data for human genetic variations on Amazon Web Services, enabling researchers to access and analyze huge amounts of data without having to develop their own supercomputing capability. In 2012, NSF joined NIH to launch the Core Techniques and Technologies for Advancing Big Data Science & Engineering program, aiming to advance core scientific and technological means of managing, analyzing, visualizing and extracting useful information from large, diverse, distributed, heterogeneous datasets. Several federal agencies have launched their own big-data programs. The Internal Revenue Service has been integrating big data-analytic capabilities into its Return Review Program (RRP), which by analyzing massive amounts of data allows it to detect, prevent, and resolve tax-evasion and fraud cases.¹⁰ DoD is also spending millions of dollars on big-data-related projects; one goal is developing autonomous robotic systems (learning machines) by harnessing big data.

Local governments have also initiated big-data projects; for example, in 2011, Syracuse, NY, in collaboration with IBM, launched a Smarter City



Governments expect big data to enhance their ability to serve their citizens and address major national challenges involving the economy, health care, job creation, natural disasters, and terrorism.



project to use big data to help predict and prevent vacant residential properties.⁷ Michigan's Department of Information Technology constructed a data warehouse to provide a single source of information about the citizens of Michigan to multiple government agencies and organizations to help provide better services.

European Union. In 2010, The European Commission initiated its "Digital Agenda for Europe" to address how to deliver sustainable economic and social benefits to EU citizens from a single digital market through fast and ultra-fast interoperable Internet applications.⁵ In 2012, in its "Digital Agenda for Europe and Challenges for 2012," the European Commission made big-data strategy part of the effort, emphasizing the economic potential of public data locked in filing cabinets and data centers of public agencies; ensuring data protection and increasing individuals' trust; developing the Internet of things, or communication between devices without direct human intervention; and assuring Internet security and secure treatment of data and online exchanges.⁵

U.K. The U.K. government was one of the earliest implementer EU countries of big-data programs, establishing the U.K. Horizon Scanning Centre (HSC) in 2004 to improve the government's ability to deal with cross-departmental and multi-disciplinary challenges.¹⁷ In 2011, the HSC's Foresight International Dimensions of Climate Change effort addressed climate change and its effect on the availability of food and water, regional tensions, and international stability and security by performing an in-depth analysis on multiple data channels. Another U.K. government initiative was the creation of the public website <http://data.gov.uk> in 2009, opening to the public more than 1,000 existing datasets from seven government departments initially, later increased to 8,633 datasets.

The Netherlands, Switzerland, the U.K., and 17 other countries launched a collaborative project with IBM called DOME to develop a supercomputing system able to handle a dataset in excess of one exabyte per day derived from the Square Kilometer Array (SKA) radio telescope.³ The project aims to

investigate emerging technologies for exascale computing, data transport and storage, and streaming analytics required to read, store, and analyze all the raw data collected daily. This big-data project, headquartered at Manchester's Jodrell Bank Observatory in England, aims to address a range of scientific questions about the observable universe.

Asia. The United Nations' 2012 E-Government Survey gave high marks to several Asian countries, notably South Korea, Singapore, and Japan.²⁰ Australia also ranked. These leaders have launched diverse initiatives on big data and deployed numerous projects:

South Korea. The Big Data Initiative, launched in 2011 by the President's Council on National ICT Strategies (the highest-level coordinating body for government ICT policy),¹⁶ aims to converge knowledge and administrative analytics through big data. Its Big Data Task Force was created to play the lead role in building the necessary infrastructure. The Big Data Initiative aims to establish pan-government big-data-network-and-analysis systems; promote data convergence between the government and the private sectors; build a public data-diagnosis system; produce and train talented professionals; guarantee privacy and security of personal information and improve relevant laws; develop big-data infrastructure technologies; and develop big-data management and analytical technologies.

Many South Korean ministries and agencies have proposed related action plans; for example, the Ministry of Health and Welfare initiated the Social Welfare Integrated Management Network to analyze 385 different types of public data from 35 agencies, comprehensively managing welfare benefits and services provided by the central government, as well as by local governments, to deserving recipients. The Ministry of Food, Agriculture, Forestry, and Fisheries and the Ministry of Public Administration and Security, or MOPAS, plan to launch the Preventing Foot and Mouth Disease Syndrome system, harnessing big data related to animal disease overseas, customs/immigration records, breeding-farm surveys, livestock migration, and workers in the livestock industry. Another sys-

tem MOPAS is planning is the Preventing Disasters System to forecast disasters based on past damage records and automatic and real-time forecasts of weather and/or seismic conditions. Moreover, the Korean Bioinformatics Center plans to develop and operate the National DNA Management System to integrate massive DNA and medical patient information to provide customized diagnosis and medical treatment to individuals.

Singapore. In 2004, to address national security, infectious diseases, and other national concerns, the Singapore government launched the Risk Assessment and Horizon Scanning (RAHS) program within the National Security Coordination Centre.⁶ Collecting and analyzing large-scale datasets, it proactively manages national threats, including terrorist attacks, infectious diseases, and financial crises. The RAHS Experimentation Center (REC), which opened in 2007, focuses on new technological tools to support policy making for RAHS and enhance and maintain RAHS through systematic upgrades of the big-data infrastructure. A notable REC application is exploration of possible scenarios involving importation of avian influenza into Singapore and assessment of the threat of outbreaks occurring throughout southeast Asia.

Aiming to create value through big-data research, analysis, and applications, the Singapore government also launched the portal site <http://data.gov.sg/> to provide access to publicly available government data gathered from more than 5,000 datasets from 50 ministries and agencies.

Japan. The Japanese government has initiated several programs to use accumulated large-scale data. From 2005 to 2011, the Ministry of Education, Sports, Culture, Science, and Technology (MEXT), in association with universities and research institutes, operated the New IT Infrastructure for the Information-explosion Era project (the so-called Info-plosion). Since 2011, the government's top priority has been to address the consequences of the Fukushima earthquake, tsunami, and nuclear-power-plant disaster and the reconstruction and rehabilitation of affected areas, as well as relief of related social and

economic consequences. MEXT has been collaborating with the country's National Science Foundation to enhance research and leverage big-data technologies for preventing, mitigating, and managing natural disasters.

The Council of Information and Communications and the ICT Strategy Committee, both branches of the Ministry of Internal Affairs and Communications, designated "big data applications" as a crucial mission for 2020 Japan. A big-data expert group was formed to search for technical solutions and manage institutional issues in deploying big data.

Australia. The Australian Government Information Management Office (AGIMO) provides public access to government data through the Government 2.0 program, which runs the <http://data.gov.au/> website to support repository and search tools for government big data. The government expects to save time and resources by using automated tools that let users search, analyze, and reuse enormous amounts of data.

Implementations and Initiatives Compared

Reviewing big-data projects and initiatives in leading countries (see Figure 2) identifies three notable big-data trends: First, most projects operated or implemented today can only marginally be classified as big-data applications, as outlined in the figure's upper-left quadrant. The majority of government data projects in these countries appears to share structured databases of stored data; they do not use real-time, in-motion, and unstructured or semi-structured data. Second, large and complex datasets are becoming the norm for public-sector organizations. Governments expect big data to enhance their ability to serve their citizens and address major national challenges involving the economy, health care, job creation, natural disasters, and terrorism. However, the majority of big-data applications are in the citizen (participation in public affairs) and business sectors, rather than in the government sector. And third, most big-data initiatives in the government sector, especially in the U.S. (such as the National Science Foundation's

and National Institutes of Health's Big Data program); are just getting under way or being planned for future implementation. This means big-data application projects in the government sector are still at an early stage of development, with only a handful of projects in operation (such as the U.S.'s RRP, Singapore's RAHS, and the U.K.'s HSC).

Conclusion

Elected officials, administrators, and citizens all seem to recognize that being able to manage and create value from large streams of data from different sources and in many forms (structured/stored, semi-structured/tagged, and unstructured/in-motion) represents a new form of competitive differentiation. Most governments operating or planning big-data projects need to take a step-by-step approach for setting the right goals and realistic expectations. Success depends on their ability to integrate and analyze information (through new technologies like Hadoop), develop supporting systems (such as big-data control towers), and support decision making through analytics.⁴

Here, we have explored the challenges governments face and the opportunities they find in big data. Such insights can also help follower countries in trying to deploy their own big-data systems. Moreover, follower countries may be able to leapfrog the leaders' applications through careful analysis of their successes and failures, as well as exploit future opportunities in mobile services.

Follower countries should therefore be cognizant of several insights regarding big-data applications in the public sector:

National priorities. All big-data projects in leading countries' governments share similar goals (such as easy and equal access to public services, better citizen participation in public affairs, and transparency). The main concerns with big-data applications converge on security, speed, interoperability, analytics capabilities, and lack of competent professionals. However, each government has its own priorities, opportunities, and threats based on its unique environment (such as terrorism and health care in the U.S., natu-

ral disasters in Japan, and national defense in South Korea).¹

Analytics agency. For data that cuts across departmental boundaries, a top-down approach is needed to manage and integrate big data. Governments should look to establish big-data control towers to integrate accumulated datasets, structured or unstructured, from departmental silos. Moreover, governments need to establish an advanced analytics agency responsible for developing strategies for how big data can be managed through new technology platforms and analytics and how to secure skilled professional staff.

Real-time analysis. They need to manage real-time analysis of in-motion big data while protecting individual citizens' privacy and security. They should also explore new technological playgrounds (such as cloud computing, advanced analytics, security technologies, and legislation).

Global collaboration. Much government data is global in nature and can be used to prevent and solve global issues; for example, the Group on Earth Observations (GEO) is a collaborative international intergovernmental effort to integrate and share Earth-observation data. Its Global Earth Observation System of Systems (GEOSS), a global public infrastructure that generates comprehensive, near-real-time environmental data, intends to provide information and analyses for a wide range of global users and decision makers. Governments also need to share data related to security threats, fraud, and illegal activities. Such big data needs not only translation technologies but an international collaborative effort to share and integrate data,

ICT big brothers. Finally, governments should collaborate with "ICT big brothers" like EMC, IBM, and SAS; for example, Amazon Web Services hosts many public datasets, including Japanese and U.S. census data, and many genomic and medical databases. □

References

1. Accenture. *Build It and They Will Come?* Chicago, 2012; <http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Digital-Citizen-FullSurvey.pdf>
2. Braham Group Inc. *Maximizing the Value Provided By a Big Data Platform.* Salt Lake City, UT, June 2012; <http://public.dhe.ibm.com/common/ssi/ecm/en/iml14324usen/IML14324USEN.PDF>

3. Broekema, C.P. et al. DOME: Towards the ASTRON and IBM Center for Exascale Technology. In *Proceedings of the 2012 Workshop on High-Performance Computing for Astronomy Data*, 2012, 1–4.
4. Chen, H., Chiang, R.H.L., and Storey, V.C. Business intelligence and analytics: From big data to big impact. *MIS Quarterly* 36, 4 (Dec. 2012), 1165–1188.
5. European Commission. *A Digital Agenda for Europe.* Brussels, Aug. 26, 2010; <http://ec.europa.eu/digital-agenda/>
6. Habegger, B. Strategic foresight in public policy: Reviewing the experiences of the U.K., Singapore, and the Netherlands. *Futures* 42, 1 (Feb. 2010), 49–58.
7. IBM. *IBM's Smarter Cities Challenge.* Syracuse, Dec. 2011; http://smartercitieschallenge.org/city_syracuse_ny.html
8. McAfee, A. and Brynjolfsson, E. Big data: The management revolution. *Harvard Business Review* (Oct. 2012), 61–68.
9. McKinsey Global Institute. *Big Data: The Next Frontier for Innovation, Competition, and Productivity.* New York, May 2011; http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
10. National Information Society Agency. *Evolving World on Big Data: Global Practices.* May 2012; <http://www.koreaninformationsociety.com/2013/11/korean-national-information-society.html>
11. Office of Science and Technology Policy, Executive Office of the President. *Fact Sheet: Big Data Across the Federal Government.* Washington, D.C., Mar. 29, 2012; <http://www.whitehouse.gov/administration/eop/ostp>
12. Office of Science and Technology Policy, Executive Office of the President. *Obama Administration Unveils 'Big Data' Initiative: Announces \$200 Million in New R&D Investments.* Washington, D.C., Mar. 29, 2012; <http://www.whitehouse.gov/administration/eop/ostp>
13. Ohlhorst, F.J. *Big Data Analytics: Turning Big Data Into Big Money.* John Wiley & Sons, Hoboken, NJ, 2013.
14. Plant, R. CISPA: Information without representation? Big Data Republic, Apr. 24, 2013; http://www.bigdatarepublic.com/author.asp?section_id=2635&doc_id=262480
15. President's Council of Advisors on Science and Technology. *Designing a Digital Future: Federally Funded Research and Development in Networking and Information Technology.* Washington, D.C., Dec. 2010; <http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-nitrd-report-2010.pdf>
16. President's Council on National ICT Strategies. *Establishing a Smart Government by Using Big Data.* Washington, D.C., Nov. 7, 2011.
17. Sherry, S. 33B pounds drive U.K. government big data agenda. Big Data Republic, Nov. 16, 2012; http://www.bigdatarepublic.com/author.asp?section_id=2642&doc_id=254471
18. Stone, D.A. *Policy Paradox: The Art of Political Decision Making.* W.W. Norton & Company, Inc., New York, 2002.
19. Stonebraker, M. What does 'big data' mean? Blog@CACM, Sept. 21, 2012; <http://cacm.acm.org/blogs/blog-cacm/155468-what-does-big-data-mean/fulltext>
20. United Nations. *E-government Survey 2012: E-government for the People, 2012;* <http://www.un.org/en/development/desa/publications/connecting-governments-to-citizens.html>
21. U.S. Government. Data.gov; <http://www.data.gov>
22. Zikopoulos, P.C., Eaton, C., DeRoos, D., Deutsch, T., and Lapis, G. *Understanding Big Data: Analytics for Enterprise-Class Hadoop and Streaming Data.* McGraw-Hill, New York, 2012.

Gang-Hoon Kim (ironhoon@etri.re.kr) is a researcher in the Creative Future Research Laboratory at the Electronics and Telecommunications Research Institute, Daejeon, South Korea.

Silvana Trimi (strimi@unl.edu) is an associate professor of management information systems in the College of Business Administration at the University of Nebraska-Lincoln.

Ji-Hyong Chung (jhc123@etri.re.kr) is a researcher in the Creative Future Research Laboratory at the Electronics and Telecommunications Research Institute, Daejeon, South Korea.

Methods for embedding secret data are more sophisticated than their ancient predecessors, but the basic principles remain unchanged.

BY ELŻBIETA ZIELIŃSKA, WOJCIECH MAZURCZYK, AND KRZYSZTOF SZCZYPIORSKI

Trends in Steganography

THE MASS MEDIA anointed 2011 as the "year of the hack"²³ due to the numerous accounts of data security breaches in private companies and governments. Indeed, the sheer volume of stolen data was estimated in petabytes (that is, millions of gigabytes).²

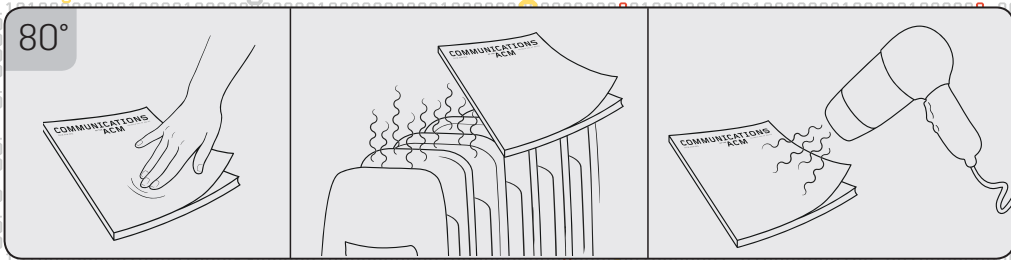
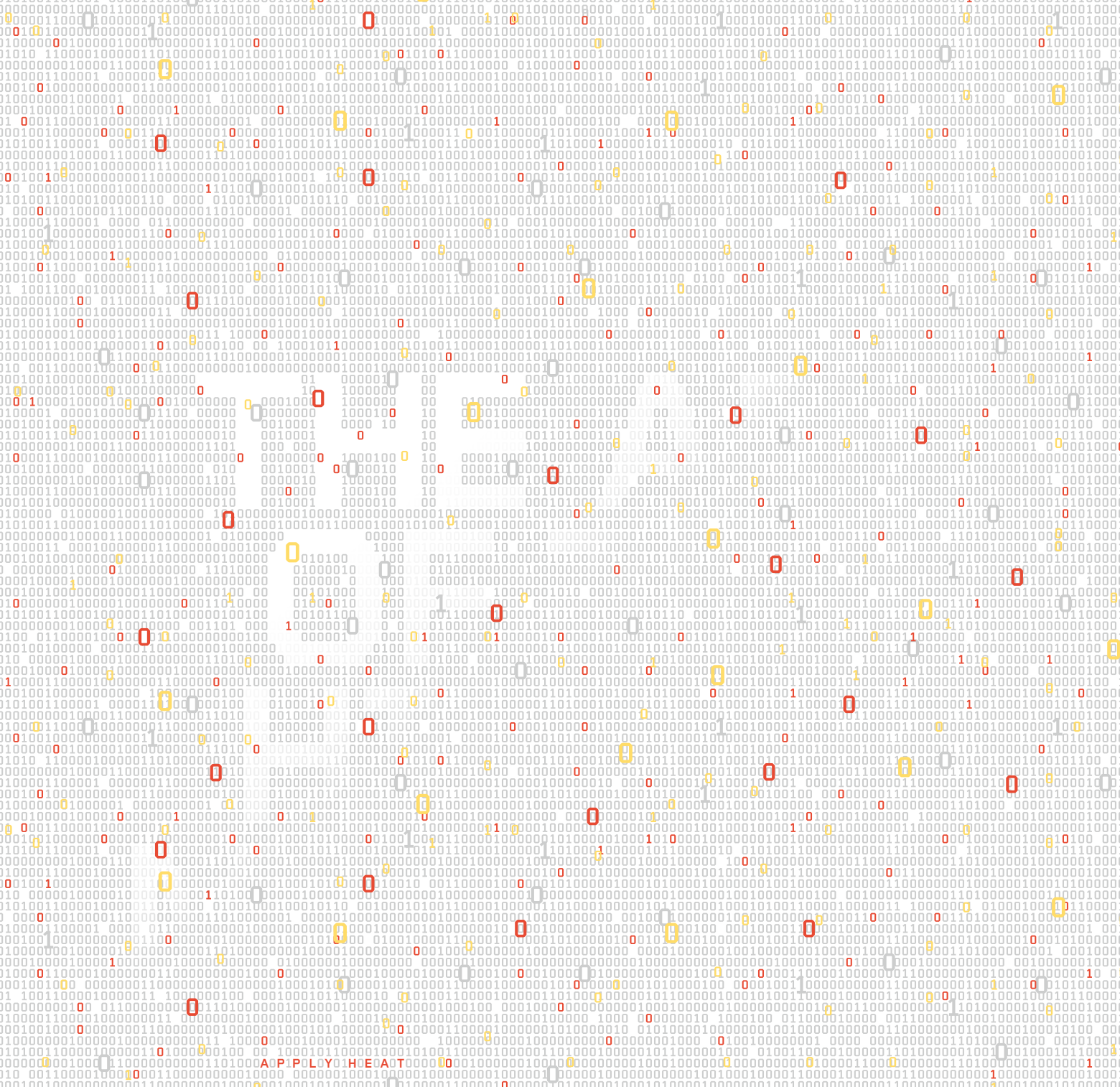
A large fraction of the security breaches that year could be attributed to the so-called Operation Shady RAT.⁴⁹ These actions were targeted at numerous institutions around the world and the inflicted damage lasted, in many cases, for months. The mechanism of infection was mainly by means of conning an unaware user to open a specially crafted email message (phishing) and implanting a back door on the victim's computer. The next step was to connect to a website and download files that only seemed to be legitimate HTML or JPEG files. What cybercriminals had actually done was encode commands into pictures or crafted Web pages so they were invisible to unaware third parties, and smuggled them through firewalls into the system under attack. These control commands then ordered a victim's computer to obtain executable code

from remote servers, which in turn permitted an outsider to gain access to local files on the compromised host.³² In numerous cases, the side channel to the confidential resources remained accessible for months, thus deeming the security breach severe. The villains were so daring they did not even put much effort into obscuring the fact that information hiding techniques were involved in the attack. One of the pictures used as a vector for control commands was the famous "Lena," a cropped picture of a Playboy model, which is the standard test image for any digital image processing or steganographic algorithm.

It is very likely we are witnessing the birth of a whole new breed of malware. It all started with the discovery of the well-known Stuxnet¹⁸ computer worm in June 2010 that stirred increased attention because it had been targeted specifically to affect Iranian nuclear power plants.¹⁴ In September 2011 a new worm, called Duqu, was discovered, and it seems to be closely related to Stuxnet.⁷ The general characteristics of the malware's structure are the same in both cases, however, unlike its predecessor, Duqu is oriented at gathering information on the infected system. The most stunning intricacy in Duqu's functioning is its employment of special means for transferring the obtained data to the command and con-

» key insights

- **Steganography, or camouflaging the presence of hidden messages in legitimate carriers, has recently become a tool of the trade for malware suppliers, as proven by the recent attacks on major global targets.**
- **Despite the fact steganography has been known for centuries, it recently has proliferated new grounds—digital media, computer networks, and popular telecommunications services.**
- **There is no miracle solution for the abuse of steganography, other than the meticulous search for any loophole that might be exploited for the purpose of embedding of illicit information, or any means of altering the potential carrier in a manner escaping human perception.**



trol centers of the malware's authors. The captured information is hidden in seemingly innocent pictures and traverses the global network as ordinary files, without raising any suspicion.^{21,51} A similar functioning mechanism was found in a new variant of the Alureon malware,³ which was also discovered around the same time.

These facts indicate that in today's world of digital technologies, it is easily imaginable that the carrier, in which secret data is embedded, was not necessarily an image or Web page source code, but may have been any other file type or organizational unit of data—for example, a packet or a frame—that naturally occurs in computer networks. However, we emphasize the process of embedding secret information into an innocent-looking carrier is not some recent invention—it has been known and used for ages by humankind. This process is called *steganography* and its origins can be traced back to ancient times. Moreover, its importance has not decreased since its birth.

Among steganography's applications is providing means for conducting clandestine communication. The purpose of establishing such information exchange may vary; possible uses can fall into the category of legal or illicit activity. Frequently, the illegal aspect of steganography is accentuated, starting from the obvious criminal communication, through information leakage from guarded systems, cyber weapon exchange, up to industrial espionage. On the other side of the spectrum lie legitimate uses, which include circumvention of Web censorship and surveillance,⁵⁵ computer forensics (tracing and identification), and copyright protection (watermarking images, broadcast monitoring).

Stuxnet, Duqu, Alureon, and Shady RAT are merely examples of what is becoming a daily routine for security experts. What should ring the alarm bells is the incorporation of steganography into the already versatile armory of rouge hackers. It can be concluded that steganography is becoming the new black among Black Hats.

The inverse of steganography—steganalysis, which concentrates on the detection of covert communication—started to surface fairly recently, what is reflected in the proportion of

available software tools concerning information hiding. Programs for the embedding of data considerably outnumber those dedicated to the detection and extraction of embedded content. The largest commercial database of steganographic tools contains 1,025 applications (as of February 2012),⁴ up from 111 tools mentioned by Hayati et al.²⁴ as of year 2007.

Let us take a closer look at the evolution of this technique, with special attention directed toward the class of methods falling in the category of network steganography.

Recent Cases of Steganography Usage

Besides the previously noted cases of steganographic methods' utilization, in the last decade, one can observe intensive research effort related to steganography and its detection methods (steganalysis). This has been caused by two facts: first, industry's and business's interest in DRM (Digital Rights Management) and second, the alleged utilization of the steganographic methods by terrorists while planning the attacks on U.S. on September 11, 2001.^{29,45} It is claimed that the rouge organization used images to conceal instructions regarding the plot, which were then posted on publicly available websites.⁴⁵ It seems that such communication could have passed unnoticed for as long as three years.²⁸

Recent findings suggest that steganography is presently exploited, mainly for illicit purposes.^{29,44,50,57} Robin Bryant¹¹ recollects the case of "Operation Twins," which culminated in 2002 with the capture of criminals associated with the "Shadowz Brotherhood," a pedophile organization responsible for distribution of child pornography with the aid of steganography.

The mushrooming incidents involving the use of information hiding had triggered an official recognition of the problem. In the 2006 Federal Report,³⁸ steganography had been named among the major threats of the present-day networks, whose significance is predicted to increase. One of the solutions to alleviate the risks connected with this technique is to become acquainted with the evolution of steganography and, consequentially, predict its further development. This need has

been recognized by the academic world in the early 1980s, when steganography started to gain popularity.

Steganographic methods have also proven to be useful tools for data exfiltration, for example, in 2008 it was reported¹ that someone at the U.S. Department of Justice smuggled sensitive financial data out of the agency by embedding the data in several image files.

In 2010, the revealing of a Russian spy ring of the so-called 'illegals,' proved that steganography can pass unnoticed for much longer. The compromised group used digital image steganography to leak classified information from U.S. to Moscow.⁴⁴

Steganography and Its Relationship to Cryptography

Steganography, frequently interchangeably and incorrectly referred to as information hiding, is the art of embedding secret messages (steganograms) in a certain carrier, possibly to communicate them in a covert manner. The border between the two fields cannot be visibly demarcated as their definitions are elusive, and there is a lack of a coherent classification of the invented clandestine communication methods, attributing them to specific domains of steganography or information hiding. The arising misconceptions may be attributed to the recent surge of interest in steganography observed in the mass media, which had only shed light on a small fraction of the available techniques. The reports of espionage and terrorist activities emerging during the last decade^{29,44,50,57} have mainly promoted these information hiding techniques, which are associated with the Internet and digital image steganography.

The question remains: How to provide rules to distinguish what belongs to the spectrum of steganographic methods? This can be done by means of providing certain conditions that must be fulfilled to consider something steganography. These may be expressed as follows:

- ▶ The information undergoing such hidden transmission is embedded in a seemingly innocent carrier, serving as camouflage for the hidden content.
- ▶ The purpose of applying a steganographic technique is to communicate information in a covert manner.

► The secrecy of the communication is guaranteed primarily by the camouflaging capability of the algorithm applied to the utilized carrier, and how well the processed data blends in with the whole bulk of legitimate entities of the cover (without embedding); this is understood as the capability to withstand detection attempts, which may rely on statistical analysis of the captured traffic or perceptual analysis of a suspicious message.

The best carrier for secret messages must possess two features. Firstly, it should be popular, that is, the usage of such a carrier should not itself be considered an anomaly. Secondly, the steganogram insertion-related modifications of the carrier should not be “visible” to the third party not aware of the steganographic procedure. Thus, if the embedding of additional information causes degradation of the carrier, then their severity should be limited to a level that would not cause suspicion.

Steganography is not only limited to concealing the fact that a message is being sent, and if not detected, make the sender and receiver “invisible.” It should also provide anonymity and privacy, which become understandable desires in modern societies. Obviously, the anonymity potential of steganography, while it can be considered as beneficial in the context of protecting privacy, poses a new type of threat to individuals, societies, and states. The trade-off between the benefits and threats involves many complex ethical, legal, and technological issues. In this article, we only consider the latter.

Steganography is often confused with cryptography, due to their common purpose of providing confidentiality. The difference becomes visible once the etymology of these words is known. Steganography is derived from the Greek: “covered writing,” whereas cryptography stands for “secret writing.” While the first describes the techniques to create a hidden communication channel, the latter is a designation of ongoing overt message exchange, where the informative content is unintelligible to unauthorized parties. To summarize, it is either the method to establish a communication channel that is kept confidential, or the message itself. Either way, the goal of protecting information from disclosure remains

common for both techniques—it is the means that permits us to differentiate between the two. Table 1 summarizes differences between cryptography and steganography, and Table 2 summarizes the relationship between steganography and watermarking.

In spite of the common historical background of the stated communication protection methods, only cryptography has managed to sustain an invariably strong position. Steganography experienced its golden age in the times of ancient Greece and Rome, to be gradually marginalized as time passed. Intuitively, any method of protection that relies on its own confidentiality to provide the secrecy of com-

munication, should not be publicized. Therefore, very few accounts proving contemporary exploitation of steganography can be found, which does not point to the conclusion that it is a neglected scientific discipline.

The Origins of Steganography

The inspiration of steganography is strongly related to phenomena observable in the animal and plant kingdoms. Evolution proved long ago that impersonation is good protection and capacitates survival for numerous species. The ability to camouflage one’s presence by means of adopting the characteristics of another living organism is referred to as mimicry. This

Table 1. Comparison of characteristics of steganography and cryptography.

	Cryptography	Steganography	
Goal	Obfuscate the content of communication	Hide the fact of communication	
Secrecy	Ciphertext is illegible	Embedded information is “invisible” to an unaware observer	
Security of communication	Relies on the confidentiality of the key	Relies on the confidentiality of the method of embedding	
Characteristics	Warranty of robustness	Complexity of the ciphering algorithm	Perceptual invisibility/statistical invisibility/compliance with protocol specification
Attacks	Detection is easy/extraction is complex	Detection is complex/extraction is complex	
Countermeasures	Technical	Reverse engineering	Constant monitoring and analysis of exchanged data
	Legal	Cryptography export laws	Rigid device/protocol specification

Table 2. Comparison of characteristics of steganography and watermarking.


	Watermarking	Steganography	
Goal	Protect the carrier	Protect secret information from disclosure	
Secrecy	Invisibility or perceptual visibility depending on the requirements	Embedded information is “invisible” to an unaware onlooker	
Type of robustness	Robustness against tampering or removal	Robustness against detection	
Characteristics	Effect of signal processing/random errors/compression	Must not lead to the loss of the watermark	May lead to the loss of hidden data
Type of carrier	Digital files—audio, video, text, or images	Any service, protocol, file, environment employing digital representation of data	

capability permits certain organisms to improve their chances for survival. The ancient Greeks, who sought inspiration in nature, had considered the ability to simulate its ways as a measure of craftsmanship. Inherently, the ancient people picked ordinary objects as a carrier for the secret message. The vector physical object, possibly even a living organism, had to be transported from one participant of communication to the other without raising suspicion on the way.


It should not be surprising that the first written report of the use of steganography is attributed to the Greek historian Herodotus. The reported method involved camouflaging a secret message within a hare corpse.¹⁶ The animal was meant to imitate a game trophy and was carried by a man disguised as a huntsman. In this way, a message could be passed without raising unnecessary suspicion.

The most notable method quoted in the historian's works is the communication on wooden tablets—these were usually coated with a thin layer of wax, on which text would be embossed. Clandestine passing of information with the aid of such medium could be achieved if the text was carved permanently on the wood (the carrier of the steganograms), and then coated with wax. Such object would then be passed as an unused tablet, and only an aware recipient would know that the letters would become visible if the wax coat was melted.

The Greek methods were fairly easy to implement as they relied on common patterns—the messages that were passed utilized a carrier cover that could be considered common at the time. Alongside the progress of human civilization and of the way people communicated, new opportunities arose. The popularization of parchment, which substituted papyrus, brought about a new cover for steganograms. Its popularity led to the development of complementary steganographic algorithms, capable of exploiting the new cover's properties. Pliny the Elder is considered the inventor of sympathetic inks,⁴⁷ as he postulated the use of thithymallus plant's sap to write text, which would become invisible upon drying. A subtle heating process would lead to the charring of the or-



The best carrier for secret messages must possess two features. Firstly, the carrier should be popular. Secondly, the steganogram insertion-related modifications of the carrier should not be “visible” to the third party not aware of the steganographic procedure.



ganic substances contained in the ink, which would then turn brown.

The common factor of all of the aforementioned techniques is the operation of adding surplus content (additional features) to a carrier, which otherwise would not physically contain the inserted elements.

A different type of steganography invented in ancient Rome is the sema-gram, or a secret message that does not take written form. Tacitus, the historiographer of the ancient world, became interested in the Astragali,⁴⁸ which were small dice made of bone. Such objects could be threaded onto a string, where the placement of the holes could be attributed meaning. A properly crafted object would pass unnoticed as a toy.

The Medieval Ages had brought about major progress in the art of information hiding. The Chinese invention of paper, upon its introduction to Europe in the Middle Ages, had brought forth the necessity of differentiating between different manufacturers' products. This is how paper watermarking was born.⁴³ Today, digital watermarking and digital image steganography are based on the same principle. It should be stressed that file watermarking is now considered a separate branch of the information hiding techniques. In their 1999 survey paper, Petitcolas, Anderson, and Kuhn⁴⁰ derived a whole field of copyright marking, of which watermarking is a subclass. The current notion refrains from classifying digital watermarking as steganography, due to the lack of an explicit communication aspect and the inferior role of providing “invisibility” to the participants of communication and a larger importance of robustness of such embedded watermarks.

The popularization of paper had further consequences. The steganographic vector was no longer necessarily a physical object, but could take written form, where the carrier text itself would conceal the privileged information. Among the inventions that achieved popularity during medieval times are the textual steganographic methods, particularly the acrostic. This term refers to pieces of writing, whose first letters or syllables spell out a message. The most famous example

of such textual steganography is attributed to a Dominican priest named Francesco Colonna, who, in 1499 hid in his book, *Hypnerotomachia Poliphili*, a love confession which could be spelled out from the first letters of subsequent chapters.¹⁵

A more sublime carrier is the language itself, as the medieval people had discovered. Here, the embedding process occurs in the linguistic syntax and semantics. Linguistic steganography may be derived from the aforementioned technique of textual steganography, as it relies on the manipulations on the written (possibly even spoken) language with the aim of tricking the perception of an unaware dupe. Following the postulates of Richard Bergmair,¹⁰ linguistic steganography covers within its scope any technique that involves intentional mimicry of typical structures of words, characteristic to a specific language. This may concern the deliberate tampering with grammar, syntax, and the semantics of a natural language. Any action involving modification of those aspects should capacitate maintaining of the innocent appearance of the cover text.

The Renaissance brought about an invention by an Italian scientist Giambattista della Porta who, in the 16th century, detailed how to hide a message inside a hard-boiled egg: write on the shell using ink made from a mixture of alum and vinegar. The solution penetrated the eggshell, leaving no trace on the surface, but a discoloration occurred on the white, leaving the message on its surface, which was only readable once the shell was removed.

Gaspar Schott, a German Jesuit from the Age of Enlightenment, followed the trail marked by his Renaissance predecessors. His work, published in 1680, entitled *Schola Steganographica*, explained how to utilize music scores as a hidden data carrier. Each note corresponded to a letter, which appeared innocent as long as nobody attempted to play the odd-sounding melodies.

The Industrial Revolution, which followed the Age of Enlightenment, brought about new means of communication. Newspapers became a popular and reliable source of the latest information. At some point it became obvious that a newspaper could serve as a perfect steganographic carrier.

Since daily papers could be sent free of charge, it was convenient to poke holes over selected letters and thus craft a secret message. This is how “newspaper codes” were born.

The first symptoms of the growing interest in steganography may be traced back to the period of the World War I and II and then the Cold War. These events had brought about such steganographic techniques as microdots—punctuation marks with inserted microscopic negatives of images or texts.⁵⁶

The period during the two World Wars was a true bonanza of hidden communication schemes. World War I witnessed the spectacular return of all sorts of invisible inks.²⁷ World War II was marked by Hedy Lamarr and George Antheil’s patent for spread spectrum communication.³⁴ They devised a method for guiding torpedoes with a special, multifrequency set of signals, resistant to jamming attempts. The control information was dispersed over a wide-frequency bandwidth that provided cover. The idea of embedding information in a number of different frequencies later found use in the fields of digital image and audio steganography.

The technological development in the 20th century had also accelerated the development of more sophisticated techniques. Among these inventions were the so-called “subliminal channels” based on cryptographic ciphers for the embedding of steganograms. The main principle was to insert content into digital signatures. Gustavus Simmons introduced this concept in 1984, despite the U.S. government’s prohibition on publishing of materials on steganography. Simmons proposed the overt and monitored communication conducted between two participants be supplemented with a steganographic channel. This channel would be based on a number of dedicated bits of the message authentication. These, at the cost of reducing the message authentication capability of the digital signature, would serve as the steganographic channel capacity.⁴⁶ The steganographic channel established in this way would be visible, yet undetectable. Subliminal channels utilized the cryptographic protocol as the carrier for steganograms.

Contemporary Trends of Development

Modern steganographic techniques utilize the 20th century’s inventions—computers and networking. Four main trends of development of the so-called digital steganography can be distinguished: digital media steganography; linguistic steganography; file system steganography; and network steganography.

These four main branches of digital steganography are explained and described here. It must be also emphasized that most of the current research in this area is devoted to digital media and network steganography. The prior is a mature research area with significant achievements, thus the exploration of this field is presently not as dynamic as in the case of the recently sprouted group of techniques falling into the category of network steganography.

Digital media steganography dates back to the 1970s, when researchers focused on developing methods to secretly embed a signature in a digital picture. Many different methods were proposed, including patchwork, least significant bit modifications, and texture block coding.⁸ These techniques were intended for both types of images: undergone lossy or lossless compression, like JPEG or BMP, which are the most common image formats. The variety of algorithms for embedding in digital pictures can be grouped according to the type of alterations that were induced. Following Johnson and Jajodia, the modifications are either bit-wise—influencing the spatial domain characteristics of the image, or affect the frequency domain characteristics. Thirdly, specific file format intricacies may be exploited, indeed, a mix of all these techniques is possible. The transform domain provides for the most versatile medium of embedding. Affecting of the image processing algorithms may involve, among others, discrete cosine transform (DCT), discrete wavelet transform (DWT), Fourier transform that may result in alterations of for example, luminance or other measurable property of an image.^{20,26} Digital image steganography’s position is unflinching—the survey paper by Cheddad et al.¹³ points to the current interest con-

centrated on employing digital media steganography and watermarking for embedding confidential, patient-related information in medical imagery. Another application of digital image steganography predicted to become popular is the implantation of additional data in printed matter, which, invisible to the naked eye, becomes decodable, when photographed and processed by a cellphone.¹³

Notably, digital image steganography is mostly oriented toward tricking the human visual system into believing the perception of the image has not been manipulated in any way.⁸ Similar rules apply to the whole field of digital media steganography, whose primary function is to trick the observer to believe the crafted “forgery” is indeed genuine. The communication aspect of the whole steganographic algorithm is secondary to the process of embedding of the secret data.

Alongside the development of digital image steganography, it appeared the human auditory system is equally prone to delusion as the visual perception. The research focus moved to audio files like MPEGs. The developed techniques included, among others, frequency masking, echo hiding, phase coding, patchwork, and spread spectrum. It also became apparent that error correction coding is a good supplemental carrier for audio steganography—any redundant data can be used to convey the steganogram at the cost of losing some robustness to random errors.⁸ This idea later found use in network-protocol based steganography.

Next, steganographers took video files as target carrier. Most of the proposed methods were adaptations of the algorithms proposed for audio and image files. Video-specific solutions involved using either videos I-frames’ color space⁵⁴ as a steganographic carrier or motion vectors for P-frames and B-frames.⁵⁸ Currently, steganography in video files either takes advantage of the existing methods for audio and image files, or makes use of the intrinsic properties of the video transmission, like movement encoding.

Parallel to digital image and audio steganography, information hiding in text was developed—the available methods exploited various aspects of the written word. The first set of tech-

niques altered word spacing, which was even claimed to have been used at the times of Margaret Thatcher to track leakages of cabinet documents.⁵ More advanced steganographic methods used syntactic and semantic structure of the text as a carrier. The methods introduced permitted for such displacement of punctuation marks, word order, or alterations of the choice of synonyms, that could be attributed certain meaning. Today, some suggest even SPAM messages may be a carrier of steganography, due to the large amounts of such mail emitted every day.¹² According to work by Bennet,⁹ the possible techniques can either rely on the generation of text with a cohesive linguistic structure or the use of natural language text as a carrier. We should note the first technique does not completely fulfill the definition of steganography, where the existence of the carrier should be independent of the existence of the injected hidden content. Thus, a text-lacking rhetorical structure cannot be considered a proper carrier.

Specialists also differentiate between textual steganography and linguistic steganography.⁹ The “SPAM method” is a linguistic method, and the embedding occurs with the aid of Context Free Grammars. CFGs have a tree structure, therefore the selection of proper words, or branches, provides encoding for binary data. An example of a textual method would be a substitution technique, where a message’s carrier is the set of white spaces and punctuation marks undergoing shifting, repetition, or other modifications.

Parallel to this research, it was revealed that x86 machine code can also be subject to embedding.¹⁷ Some amount of information can be placed in the carrier code, with the aid of careful selection of functionally equivalent instructions. This method exploits the same principle as linguistic steganography, where the choice of words from the set of synonyms can be attributed steganographic meaning.

The invention of a steganographic file system by Anderson, Needham, and Shamir was an eye-opener.⁶ It became apparent that information can be steganographically embedded even in isolated computing environments. The main principle of steganogram

preparation was similar to invisible inks—one that knew how to search could reveal the encrypted files from a disk. The utilized mechanism relied on the fact that ciphered data resembles random bits naturally present on the disk and only the ability to extract the vectors marking the file boundaries permitted the location process. Another example of a steganographic file system can be found in Pang et al.,³⁹ whose authors created a steganographic file system implementation on Linux. Their invention preserves the integrity of the stored files and employs a hiding scheme in the disk space with camouflaging with the aid of Dummy Hidden Files and Abandoned Blocks.

Alongside the above-mentioned types of digital steganography, currently the target of increased interest is network steganography. This modern family of methods stems from “covert channels”—a number of techniques intended for monolithic systems, like mainframes. This term was first introduced by Lampson, who identified the problem of information leakage in non-confined programs.³¹ The expression “network steganography” was coined by Szczypiorski.³² Currently, the terms network steganography and covert channels are used interchangeably (and incorrectly), but historically they are sovereign of each other.

A summary of the evolution of the steganographic data carrier is presented in the accompanying figure.

Network Steganography: The Youngest in the Spotlight

Network steganography is the youngest branch of information hiding. It is a fast-developing field: recent years have resulted in multiple new information hiding methods, which can be exploited in various types of networks. The exploitation of protocols belonging to the Open Systems Interconnection (OSI) reference model⁵⁹ is the essence of network steganography. This family of methods may utilize one or more protocols simultaneously or the relationships between them—relying on the modification of their intrinsic properties for the embedding of steganograms.

Network steganography is on the rise because embedding secret data into digital media files has been found

to possess two serious drawbacks: it permits hiding only a limited amount of data per one file and the modified picture may be accessible for forensics experts (for example, because it was uploaded to some kind of server). Network-level embedding changes the state of things diametrically; it allows for leakage of information (even very slow) during long periods of time and, if all the exchanged traffic is not captured, then there is nothing left for forensics experts to analyze. As a result, such methods are more difficult to detect and eliminate from networks.

Today, network steganography relies on certain loopholes to conceal its presence. The first is the perceptual inability of the end user to sense minor differences between seemingly identical objects. For example, upon hearing a real-time audio recording transferred through a public network, a person almost certainly will not notice slight alterations of the transmitted voice, especially that he or she will lack any reference for that particular VoIP call. The second loophole permits the passage of steganograms through a network, without raising any alarms in the intermediate nodes. This typically relates to the statistical invisibility—that is, the induced anomalies do not exceed a reasonable threshold typical for network functioning. Typically, three characteristics of communications are utilized for steganographic purposes:

- The communication channel is not perfect—errors are a natural phenomenon and thus it is possible to embed information in a pattern mim-

icking an ordinary distribution of damaged Protocol Data Units.

- Most protocols bear some quantity of redundant information. The surplus fields can be used for embedding, if this does not induce malfunctioning of the carrier information flow.

- Not every protocol is completely defined. Most of the specifications permit some amount of freedom in implementation, which can be abused.

Network steganography methods, following Jankowski et al.,²⁵ can be broadly classified according to the number of protocols used for steganographic purposes. The modification of the properties of a single protocol from the OSI model is called intra-protocol steganography, whereas exploitation of relationships between multiple protocols is classified as inter-protocol steganography. Once a protocol or number of protocols are chosen as a carrier for secret data, it is decided how the embedding should be performed. The first possibility is to inject the covert information into the Protocol Data Unit (PDU)—this can be done by means of modification of protocol specific fields or by means of insertion into the payload, or both. Alternatively, or complementary to the previous technique, it is possible to modify the time relations between the PDUs. These changes may impact the order of PDUs, their losses or their relative delays. Hybrid methods utilize both—modification of PDUs and their time relations.

The predecessor of current, more sophisticated network steganography methods, was the utilization of

different fields of TCP/IP stack's protocols⁴² as a hidden data carrier. The majority of early methods concentrated on embedding in the unused or reserved fields of protocols to convey secret data. Then, more advanced methods were invented, which were targeted toward specific environments or toward specific services. Recent solutions exploit:

- Multimedia, real-time services like IP telephony;³³

- Popular peer-to-peer services: Skype³⁵ or P2P file-sharing systems like BitTorrent;³⁰

- Social media sites like Facebook;⁷

- Wireless network environments: for example, Wireless Local Area Networks (WLAN),⁵³ or Long Term Evolution (LTE);²²

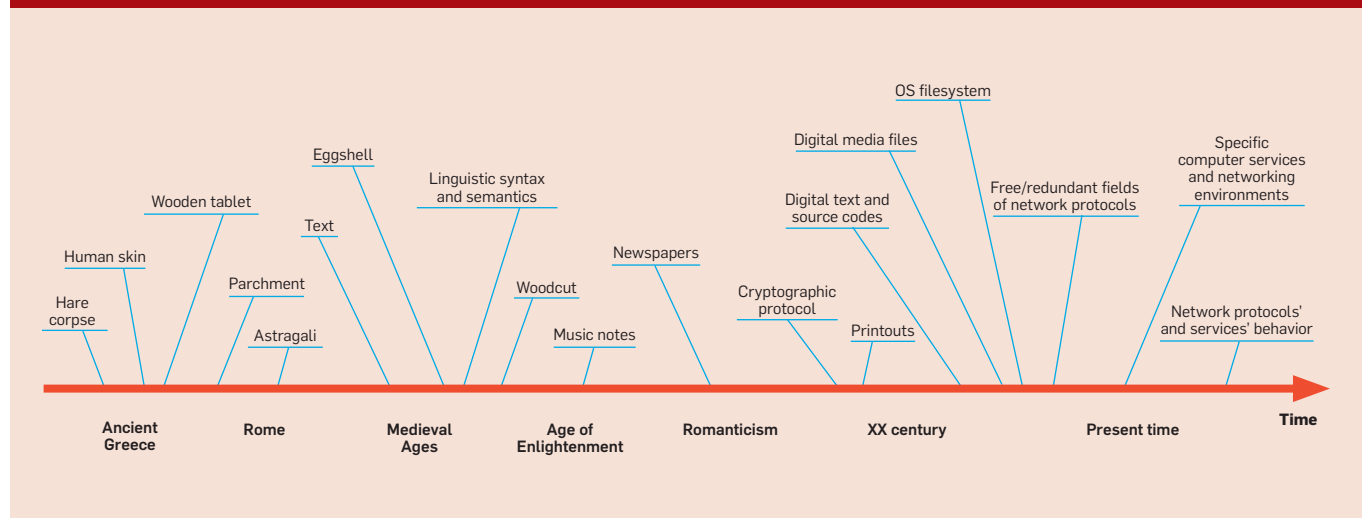
- Cloud computing environments;⁴¹ and

- New network protocols, like Stream Control Transmission Protocol (SCTP).¹⁹

Although the use of IP telephony service as a hidden data carrier can be considered a fairly recent discovery,³³ the existing VoIP steganographic methods stem from two distinct research origins. The first is the aforementioned, well-established digital media steganography, which has given rise to methods that target the digital representation of the transmitted voice as the carrier for hidden data. The second sphere of solutions target specific VoIP protocol (for example, signaling, transport, or control protocols) fields, or the protocol's behavior.

More recently, Transcoding Steg-

Timeline of the evolution of hidden data carrier.




anography (TranSteg)—intended for a broad class of multimedia and real-time applications like IP telephony—has been proposed.³⁶ TranSteg is based on the general idea of transcoding (lossy compression) of the voice data from a higher bit rate codec, and thus greater voice payload size, to a lower bit rate codec with smaller voice payload size. This occurs with the least possible degradation in voice quality; compression of the overt data is utilized to make space for the steganogram in the payload field. The achieved steganographic bandwidth is as high as 32kbit/s.

Looking into the P2P services' steganographic applicability, one may encounter a steganographic method named SkyDe (Skype Hide), proposed for Skype by Mazurczyk et al.³⁵ It utilizes encrypted Skype voice packets as a hidden data carrier. By taking advantage of the high correlation between speech activity and packet size, packets without voice signals can be identified and used to carry secret data. This is achieved by replacing the encrypted silence with secret data bits. The resulting steganographic bandwidth, or hidden-data rate (amount of secret data that can be sent per unit of time, when using a particular method) is about 2kbit/s.


Another recent invention for an Internet P2P service, the StegTorrent, has been introduced for the BitTorrent application.³⁰ StegTorrent takes advantage of the fact that there are usually many-to-one transmissions in BitTorrent, and that for one of its specific protocols— μ TP—the header provides a means for numbering packets and retrieving their original sequence. This allows for sending hidden data with a rate of about 270b/s.

For social media sites like Facebook, Nagaraja et al.³⁷ proposed creating a botnet communicating over unobservable communication channels. The bots exchanged information with their botmaster by embedding information in images and using the image sharing capabilities to route the secret data to the recipient.

When it comes to the “wireless environment,” different standards are targeted by steganographers. For example, for WLANs, Szczypiorski and Mazurczyk have introduced a method called WiPad (Wireless Padding).⁵³ The



Today, network steganography relies on certain loopholes to conceal its presence.



technique is based on the insertion of hidden data into the padding of frames at the physical layer of WLANs. It allows data to pass in a covert way with a significantly high data rate of about 1.5Mbit/s. A similar concept was utilized in Grabska and Szczypiorski²² for LTE and the resulting data rate was about 1.2Mbit/s.

The cloud computing environment, which Ristenpart et al.⁴¹ views as vulnerable to cross-Virtual Machine information leakage, is a great playground for exercising steganography. They proposed a range of techniques for obtaining classified information by probing the values of shared-cache load, CPU load, keystroke activity, or similar methods.

Other promising future-network protocols, like the SCTP, which is a candidate for new transport layer protocol and might replace TCP (Transmission Control Protocol), and UDP, (User Datagram Protocol) protocols, are also prone to steganography. Detailed analysis in Frączek et al.¹⁹ reveals the most likely places in SCTP transmissions to be utilized for information hiding. Special attention is directed toward steganographic methods that utilize new features, characteristic to SCTP, such as multihoming and multistreaming.

To summarize, various network services and applications can and will become targets of embedding, and the larger the proliferation of a certain service or application, the more attractive it is to piggyback secret data by means of network steganography.


Conclusion

Information hiding covers within its scope various techniques intended for the communication of messages with the aim of keeping some aspect of such exchange secret. This may involve providing security by obscurity for the participants of the dialogue (anonymity), secrecy of the messages (steganography), or protection of the carrier (copyright marking).

The roots of these methods stem from historic times. The need for sending messages that cannot be compromised in case of interception had motivated people to create codes or symbols that appeared innocent, but in fact had different significance than the apparent.

Modern information hiding employs

various embedding techniques—many of these are the result of the transfer of some previously known method into the digital domain. An interesting exception to this notion is network steganography, a family of methods that emerged with the popularization of networked environments. The appearance of new secret data carriers in steganography can be treated as evolutionary steps in the development of information hiding techniques. The growing number of communication protocols, services, and computing environments offers almost unlimited opportunities for displaying a whole spectrum of steganographic methods. It is noteworthy that it is the carrier's properties as well as its popularity that predestine or limit its capability to serve as an efficient medium for clandestine communication, and the emergence of a new technology will likely bring about new information embedding opportunities.

Illicit activities conducted in the virtual world pose a tangible threat to society, as recent cyberwarfare events show. Indisputably, information hiding has joined the arsenal of the utilized weapons, and thus it should be recognized that it poses a large threat to the security of information systems. More importantly, the matter is pressing, because steganalysis techniques are still one step behind the newest steganography methods. There is no “one size fits all” solution available and ready to detect covert communication in our current network security defense systems. Thus, we urge the research community to focus its efforts to discover steganalysis methods that can be practically and promptly deployed in networking environments. 

References

- Adee, S. *Spy vs. spy*. *IEEE Spectrum*, (Aug. 2008); <http://spectrum.ieee.org/computing/software/spy-vs-spy/1>.
- Alperovitch, D. *Revealed: Operation Shady RAT*. McAfee, 2011; <http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf>.
- Alureon trojan uses steganography to receive commands. (Sept. 2011); http://www.virusbtn.com/news/2011/09_26.
- Analysis, S. and Center, R. World's largest digital steganography database expands again. *SARC Press Release* (Feb. 2012); http://www.sarc-wv.com/news/press_releases/-2012/safdb_v312.aspx.
- Anderson, R. *Stretching the limits of steganography*. *Information Hiding*. Springer, 1996, 39–48.
- Anderson, R., Needham, R. and Shamir, A. The steganographic file system. *Information Hiding*. Springer, 1998, 73–82.
- Bencsáth, B., Pék, G., Buttyán, L. and Félégyházi, M. Duqu: A Stuxnet-like malware found in the wild. (2011); <http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf>.
- Bender, W., Gruhl, D., Morimoto, N. and Lu, A. Techniques for data hiding. *IBM Systems Journal* 35, 3&4 (1996), 313–336.
- Bennett, K. Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text (2004).
- Bergmair, R. A comprehensive bibliography of linguistic steganography. In *Proceedings of the SPIE Intl Conf. on Security, Steganography, and Watermarking of Multimedia Contents*, 2007.
- Bryant, R., Ed. *Investigating Digital Crime*. John Wiley & Sons, 2008, 1–24.
- Castiglione, A. De Santis, A., Fiore, U. and Palmieri, F. An asynchronous covert channel using SPAM. *Computers & Mathematics with Applications*, 2011.
- Cheddad, A., Condell, J., Curran, K. and Mc Kevitt, P. Digital image steganography: Survey and analysis of current methods. *Signal Processing* 90, 3 (2010), 727–752.
- Chen, T. Stuxnet, the real start of cyber warfare? *IEEE Network* 24, 6 (2010), 2–3.
- Cox, I. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2008.
- De Sélincourt, A. *Herodotus: The Histories*. Penguin, 1954.
- El-Khalil, R. and Keromytis, A. Hydan: Hiding information in program binaries. *Information and Communications Security*, (2004), 287–291.
- Falliere, N., Murchu, L. and Chien, E. W32.Stuxnet dossier. White paper. Symantec Corp., Security Response, 2011.
- Frączek, W., Mazurczyk, W. and Szczypiorski, K. Hiding information in Stream Control Transmission Protocol. *Computer Communications* 35, 2 (2012), 159–169.
- Fridrich, J. *Steganography in Digital Media—Principles, Algorithms, and Applications*. Cambridge University Press, 2010.
- Goodin, D. Duqu spawned by 'well-funded team of competent coders': World's first known modular rootkit does steganography, too. *The Register*, Nov. 2011.
- Grabska, I. and Szczypiorski, K. Steganography in LTE. In *Proc. of Intl. Workshop on Cyber Crime*, (San Jose, May 2014).
- Gross, M.J. Exclusive: Operation shady RAT—Unprecedented cyber-espionage campaign and intellectual-property bonanza. *Vanity Fair* (Aug. 2011).
- Hayati, P., Potdar, V. and Chang, E. A survey of steganographic and steganalytic tools for the digital forensic investigator. In *Workshop of Information Hiding and Digital Watermarking*, (Canada, 2007).
- Jankowski, B., Mazurczyk, W., and Szczypiorski, K. PadSteg: Introducing inter-protocol steganography. *Telecommunication Systems: Modeling, Analysis, Design and Management* 52, 2 (2013), 1101–1111.
- Johnson, N. and Jajodia, S. Steganalysis of images created using current steganography software. *Information Hiding*. Springer, 1998, 273–289.
- Kahn, D. The history of steganography. *Information Hiding*. Springer, 1996, 1–5.
- Kellen, T. Hiding in plain view: Could steganography be a terrorist tool? *SANS Institute InfoSec Reading Room*, 2001; http://www.sans.org/reading_room/whitepapers/-steganography/hiding-plain-view-steganography-terrorist-tool_551.
- Kelley, J. Terror groups hide behind Web encryption. *USA Today* (May 2001); <http://www.usatoday.com/tech/news/2001-02-05-binladen.htm>.
- Kopiczko, P., Mazurczyk, W. and Szczypiorski, K. StegTorrent: A steganographic method for P2P files sharing service. In *Proc. of Intl. Workshop on Cyber Crime*, (San Francisco, CA, May 2013).
- Lampson, B. A note on the confinement problem. *Commun. ACM* 16, 10 (Oct. 1973), 613–615.
- Lau, H. The truth behind the Shady RAT. *McAfee report*, (Aug. 2011); <http://www.symantec.com/connect/blogs/truth-behind-shady-rat>.
- Lubacz, J., Mazurczyk, W. and Szczypiorski, K. Vice over IP. *IEEE Spectrum* 47, 2 (2010), 42–47.
- Marker, H. and Anthell, G. Secret communication system. Aug. 11 1942, US Patent 2,292,387.
- Mazurczyk, W., Karaś, M. and Szczypiorski, K. SkyDe: A Skype-based steganographic method. *International J. Computers, Communications & Control* 8, 3 (June 2013), 389–400.
- Mazurczyk, W., Szaga, P. and Szczypiorski, K. Using transcoding for hidden communication in IP telephony. *Multimedia Tools and Applications* (2011), 1–27; DOI 10.1007/s11042-012-1224-8.
- Nagaraja, S., Houmansadr, A., Piyawongwisal, P., Singh, V., Agarwal, and Borisov, N. Stegobot: A covert social network botnet. *Information Hiding*. Springer, 2011, 299–313.
- Networking and Information Technology Research and Development Program, I.W.G. on Cyber Security and I. Assurance. *Federal Plan for Cyber Security and Information Assurance Research and Development*, (Apr. 2006); http://www.nitrd.gov/pubs/csia/-csia_federal_plan.pdf.
- Pang, H., Tan, K. and Zhou, X. StegFS: A steganographic file system. In *Proceedings of the 19th Intl. Conf. on Data Engineering*. IEEE, 2003, 657–667.
- Petitcolas, F., Anderson, R. and Kuhn, M. Information hiding—A survey. In *Proceedings of the IEEE* 87, 7 (1999), 1062–1078.
- Ristenpart, T., Tromer, E., Shacham, H. and Savage, S. 'Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds'. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*. ACM, 2009, 199–212.
- Rowland, C. Covert channels in the TCP/IP protocol suite. *First Monday* 2, 5 (1997).
- Rudin, B. and Tanner, R. *Making Paper: A Look into the History of an Ancient Craft*. Rudin, 1990.
- Shachtman, N. FBI: Spies hid secret messages on public websites. *Wired* (June 2010); <http://www.wired.com/dangerroom/2010/06/alleged-spies-hid-secret-messages-on-public-websites/>.
- Sieberg, D. Bin Laden exploits technology to suit his needs. *CNN* (Sept. 2001); <http://edition.cnn.com/2001/US/09/20/inv.terrorist.search/>.
- Simmons, G. The prisoners' problem and the subliminal channel. In *Proceedings of Crypto '83: Advances in Cryptology* (1984), 51–67.
- Singh, S. *The Code Book: The Secret History of Codes and Codebreaking*. Fourth Estate, 2000.
- Smith, D. Number games and number rhymes: The great number game of dice. *The Teachers College Record* 13, 5 (1912), 39–53.
- Srivastava, K. Congress wants answers on world's largest security breach. Aug. 2011; <http://www.mobiledia.com/news/102480.html>.
- Stier, C. *Russian spy ring hid secret messages on the Web*. (July 2010); <http://www.newscientist.com/article/dn19126-russian-spy-ring-hid-secret-messages-on-the-web.html>.
- Symantec. W32.Duqu—the precursor to the next Stuxnet. (Nov. 2011); http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet_research.pdf.
- Szczypiorski, K. Steganography in TCP/IP networks. In *Proceedings of State of the Art and a Proposal of a New System—HICCUPS*. Institute of Telecommunications' seminar, Warsaw University of Technology, Poland, 2003.
- Szczypiorski, K. and Mazurczyk, W. Steganography in IEEE 802.11 OFDM symbols. *Security and Communication Networks* 3 (2011), 1–12.
- Wang, Y. and Izquierdo, E. High-capacity data hiding in MPEG-2 compressed video. In *Proceeding of the 9th Intl. Workshop on Systems, Signals and Image Processing* (Manchester, U.K., 2002), 212–218.
- Wayner, P. *Disappearing Cryptography—Information Hiding: Steganography & Watermarking*. Morgan Kaufmann, 2009.
- White, W. *The Microdot: History and Application*. Phillips Publications, 1992.
- Williams, C. Russian spy ring bust uncovers tech toolkit. *The Register* (June 2010); http://www.theregister.co.uk/2010/06/29/spy_ring_tech/.
- Xu, C., Ping, X. and Zhang, T. Steganography in compressed video stream. In *Proceedings of the First International Conference on Innovative Computing, Information and Control*. IEEE, 2006, 269–272.
- Zimmermann, H. OSI reference model—the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications* 28, 4 (1980), 425–432.

Elżbieta Zielińska (ezielinska@tele.pw.edu.pl) is a research assistant at Warsaw University of Technology, Institute of Telecommunications, Warsaw, Poland.

Wojciech Mazurczyk (wmazurczyk@cygnus.tele.pw.edu.pl) is an assistant professor at Warsaw University of Technology, Institute of Telecommunications, Warsaw, Poland.

Krzysztof Szczypiorski (kszc@tele.pw.edu.pl) is a professor at Warsaw University of Technology, Institute of Telecommunications, Warsaw, Poland.



Distinguished Speakers Program

talks by and with technology leaders and innovators

Chapters • Colleges and Universities • Corporations • Agencies • Event Planners

A great speaker can make the difference between a good event and a WOW event!

The Association for Computing Machinery (ACM), the world's largest educational and scientific computing society, now provides colleges and universities, corporations, event and conference planners, and agencies – in addition to ACM local Chapters – with direct access to top technology leaders and innovators from nearly every sector of the computing industry.

Book the speaker for your next event through the ACM Distinguished Speakers Program (DSP) and deliver compelling and insightful content to your audience. **ACM will cover the cost of transportation for the speaker to travel to your event.** Our program features renowned thought leaders in academia, industry and government speaking about the most important topics in the computing and IT world today. Our booking process is simple and convenient. Please visit us at: www.dsp.acm.org. If you have questions, please send them to acmdsp@acm.org.

The ACM Distinguished Speakers Program is an excellent solution for:

Corporations Educate your technical staff, ramp up the knowledge of your team, and give your employees the opportunity to have their questions answered by experts in their field.

Colleges and Universities Expand the knowledge base of your students with exciting lectures and the chance to engage with a computing professional in their desired field of expertise.

Event and Conference Planners Use the ACM DSP to help find compelling speakers for your next conference and reduce your costs in the process.

ACM Local Chapters Boost attendance at your meetings with live talks by DSP speakers and keep your chapter members informed of the latest industry findings.

Captivating Speakers from Exceptional Companies, Colleges and Universities

DSP speakers represent a broad range of companies, colleges and universities, including:

IBM	Sony Pictures	Georgia Tech	University of British Columbia
Microsoft	McGill University	Carnegie Mellon University	Siemens Information Systems Bangalore
BBN Technologies	Tsinghua University	Stanford University	Lawrence Livermore National Laboratory
Raytheon	UCLA	University of Pennsylvania	National Institute of Standards and Technology

Topics for Every Interest

Over 400 lectures are available from 120 different speakers with topics covering:

Software	Web Topics	Career-Related Topics	Computer Graphics, Visualization
Cloud and Delivery Methods	Computer Systems	Science and Computing	and Interactive Techniques
Emerging Technologies	Open Source	Artificial Intelligence	High Performance Computing
Engineering	Game Development	Mobile Computing	Human Computer Interaction

Exceptional Quality Is Our Standard

The same ACM you know from our world-class Digital Library, magazines and journals is now putting the affordable and flexible Distinguished Speaker Program within reach of the computing community.

Microsoft
Research
The DSP is sponsored
in part by Microsoft Europe



**Association for
Computing Machinery**

Advancing Computing as a Science & Profession

research highlights

P. 98

Technical Perspective Smartphone Security 'Taint' What It Used to Be

By Dan Wallach

P. 99

TaintDroid: An Information Flow Tracking System for Real-Time Privacy Monitoring on Smartphones

By William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox,
Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth

Technical Perspective

Smartphone Security

'Taint' What It Used to Be

By Dan Wallach

THERE IS SOMETHING SEDUCTIVE about information flow as a security policy. You can state a very clear and concise policy (for example, “forbid my GPS location information from flowing to the network”), which seems to more closely capture our intuition for right and wrong than the sorts of policies that smartphone operating systems like iOS and Android seek to enforce today (more like “give this app your GPS location, yes or no, and you have no say over how it’s used”). Information flow research dates back to the early 1970s. Although much of the original computer science theory and systems were developed around modeling the military’s rules for handling classified, secret, and top-secret data, information flow policies and techniques are perfectly valuable today and we can benefit from this earlier work.

An excellent foundational reference is Dorothy and Peter Denning’s landmark 1977 paper, “Certifying Programs for Secure Information Flow,” which pursued a static analysis strategy and appeared in *Communications*.^a At the same time, others were pursuing hardware or runtime-based solutions. A lot of the complexity then, as now, comes when the control flow of the program depends on sensitive values (for example, “if my GPS location is in Washington D.C., then behave differently”), never mind unusual control flows (for example, interrupt handlers, exceptions, indirect branches) and ambiguous data references (pointer dereferencing, array indexing). Tracking all of this in hardware requires extra computation and state, while analyzing it statically can induce false alarms over execution paths that might never happen at runtime.


Fast-forward to the present day. What’s new? We now have insanely fast computers, even in our phones, with applications that typically spend

most of their time idle, waiting for user input or network data. We can afford extra runtime CPU overhead without wasting too much battery or slowing down the user experience. Likewise, better algorithms and faster computers have made whole program static analysis, whether for bug finding or security verification, into a billion-dollar industry.

The TaintDroid project takes a runtime taint tracking approach toward analyzing Android apps. They have the benefit that the wire format for distributing apps (bytecode for the Dalvik VM) is not raw machine code, allowing them to modify the on-phone Dalvik compiler to add runtime taint tracking, yielding them an entirely reasonable 14% performance overhead on CPU-bound workloads. They similarly added annotations to the Android filesystem and IPC layers to track tainted data. The TaintDroid team did cut some corners: they punted on dealing with native ARM code, which some Android apps might include for better performance, and they similarly do not track taint from a conditional expression through to the resulting computation. (Impact: malicious apps could be easily engineered to trick the current TaintDroid implementation, and souping up TaintDroid to deal with this would require static analysis, more runtime overhead, or some combination of both.)

Despite these limitations, the TaintDroid project managed to identify a number of Android apps that were clearly behaving in ways that users would find objectionable. (In hindsight, it should not be surprising when you install an advertising-supported free app that they are going to want to know as much about you as they can to better target their advertisements at you.) Since the publication of TaintDroid in 2010, there has been an explosion of research into every possible method of enhancing An-

droid security, whether through static analysis or dynamic runtime mechanisms. Even Google is in the game, launching its “Bouncer” system in 2012, running within its Android app store, doing some combination of static analysis along with running each app inside a virtual machine to detect undesirable behavior. Google itself has had very little to say about how Bouncer works,^b but Miller and Oberheide did some clever reverse engineering that suggests Google still has a ways yet to go.^c

Certainly, whatever Google does, malicious developers will find ways around it. For example, code obfuscators do a great job of confusing static analyzers. Likewise, runtime systems can only check code that actually executes; if a malicious app can detect that it’s being monitored, it might then avoid misbehavior. Over the long term, it’s seemingly easy to predict circumstances analogous to current pattern-matching anti-virus systems, which require continuous updates. However, consider that Google may well choose to err on the side of denying programs access to the store. Apps with ambiguous behaviors could be preemptively forbidden, working around some of the imprecision inherent in analyzing apps for malice. Information flow techniques will, inevitably, play a huge role in policing the app ecosystems on our phones and elsewhere in our computational world. 

b <http://googlemobile.blogspot.com/2012/02/android-and-security.html>

c <https://blog.duosecurity.com/2012/06/dissecting-androids-bouncer/>

Dan Wallach (dwallach@cs.rice.edu) is a professor in the systems group at Rice University’s Department of Computer Science, Houston, TX.

a <http://dl.acm.org/citation.cfm?id=359712>

Copyright Held by Author.

TaintDroid: An Information Flow Tracking System for Real-Time Privacy Monitoring on Smartphones

By William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth

Abstract

Today's smartphone operating systems frequently fail to provide users with adequate control over and visibility into how third-party applications use their privacy-sensitive data. We address these shortcomings with TaintDroid, an efficient, systemwide dynamic taint tracking and analysis system capable of simultaneously tracking multiple sources of sensitive data. TaintDroid provides real-time analysis by leveraging Android's virtualized execution environment. Using TaintDroid to monitor the behavior of 30 popular third-party Android applications, we found 68 instances of misappropriation of users' location and device identification information across 20 applications. Monitoring sensitive data with TaintDroid provides informed use of third-party applications for phone users and valuable input for smartphone security service firms seeking to identify misbehaving applications.

1. INTRODUCTION

A key feature of modern smartphone platforms is a centralized service for downloading third-party applications. The convenience to users and developers of such "app stores" has made mobile devices more fun and useful, and has led to an explosion of development. Many of these applications combine data from remote cloud services with information from local sensors such as a GPS receiver, camera, microphone, and accelerometer. Applications often have legitimate reasons for accessing this privacy-sensitive data, but users would also like assurances that their data is used properly.

Resolving the tension between the fun and utility provided by third-party applications and the privacy risks they pose is a critical challenge for smartphone platforms. Smartphone operating systems currently provide only coarse-grained controls for regulating whether an application can *access* private information, but provide little insight into how private information is actually used. For example, if a user allows an application to access her location information, she has no way of knowing if the application will send her location to a location-based service, to advertisers, to the application developer, or to any other entity. As a result, users must blindly trust that applications will properly handle their private data.

This problem inherently cannot be solved by traditional access control techniques. The naïve solution is to disallow network access once privacy-sensitive information is received by a process. However, perhaps more so than other platforms, smartphone applications are built around cloud services. This has two major implications. First, the vast majority of applications functionally require network access. Second, and perhaps more important, some applications must send privacy-sensitive information to specific network hosts to meet the needs of the user. Therefore, the problem is not simply one of determining *if* such information is sent to the network, but rather to determine *what* information is sent *where*.

Determining how an application uses and discloses privacy-sensitive information is achievable using fine-grained dynamic taint analysis, commonly known as "taint tracking." A "taint" is simply a label on a data item or variable. The label assigns a semantic type (e.g., geographic location) to the data, and may simultaneously encode multiple such types (commonly called a *taint tag*). It is the task of the taint tracking system to (1) assign taint labels at a *taint source*, (2) automatically propagate taint labels to dependent data and variables, and finally (3) take some action based on the taint label of data at a *taint sink*. For example, a taint tracking system might label the variables containing the geographic coordinates of a phone when they are returned from the location API (taint source), propagate that label to all variables that are derived from those variables (e.g., if $a = b + c$, then a is derived from b and c), and then take some action (e.g., log and drop) when a variable with a location label reaches the network API (taint sink).

Security literature has many examples of taint tracking, but proposed solutions are either coarse or slow. We show that taint tracking can be efficient for Android applications. Furthermore, we find that monitoring only a single process is insufficient in Android, as data commonly flows between applications.

The original version of this paper was published in the *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

Our goal is to create a whole-system taint tracking framework that operates in *real time* to detect sensitive data exposure with sufficient context to identify potentially misbehaving applications. The real-time constraint enables both daily use by concerned users and efficient study by external security services. As with any such practical system, the solution design requires careful trade-offs between performance and precision.

We introduce the TaintDroid extension to Android, which is the first practical system that can track the flow of privacy-sensitive data throughout a smartphone platform. To balance performance and tracking precision, TaintDroid leverages Android's virtualized architecture to integrate four granularities of taint propagation: variable-level, method-level, message-level, and file-level. Though the individual techniques are not new, our contributions lie in the integration of these techniques and in identifying appropriate trade-offs between tracking precision and performance for resource-constrained smartphones.

Our second contribution lies in our use of TaintDroid to perform the first study of smartphone applications that identifies extensive misuse of privacy-sensitive data. This study considered 30 randomly selected, popular Android applications that use location, camera, or microphone data. These applications were manually run on a phone with a TaintDroid firmware. We then collected various TaintDroid and network logs and noted user expectations of privacy-sensitive data exposure to evaluate potential data misuse. In our experiments, TaintDroid correctly flagged 105 TCP connections as containing privacy-sensitive information. After further inspection, 37 were classified as clearly legitimate. By inspecting the remaining 68 TCP connections, we discovered that 15 of the 30 applications reported users' locations to remote advertising servers. Seven applications collected the device ID and, in some cases, the phone number and the SIM card serial number. In all, two-thirds of the applications in our study used sensitive data suspiciously. These results raise strong concerns of potential widespread collection of geographic location and phone identifiers without users' knowledge.

2. DESIGN OVERVIEW

We seek a design that allows users to monitor how third-party smartphone applications handle their private data in real time. Existing static analysis techniques that require source code are not suitable as many smartphone applications are closed-source and techniques that convert bytecode to source code are still far from error-free. Even if source code is available, runtime events and configuration often dictate information use; real-time monitoring accounts for these environment-specific dependencies. Furthermore, we assume that all downloaded third-party applications are untrusted and that these applications run simultaneously.

Monitoring network disclosure of privacy-sensitive information on smartphones presents several challenges:

- *Smartphones are resource constrained.* The resource limitations of smartphones preclude the use of heavy-weight information tracking systems.
- *Third-party applications are entrusted with several types of privacy-sensitive information.* The monitoring system

must distinguish multiple information types, which requires additional computation and storage.

- *Privacy-sensitive information can be difficult to identify even when sent in the clear.* For example, geographic locations are pairs of floating point numbers that frequently change and are hard to predict.
- *Applications can share information.* Limiting the monitoring system to a single application does not account for flows via files and Inter Process Communication (IPC) between applications, including core system applications designed to disseminate privacy-sensitive information.

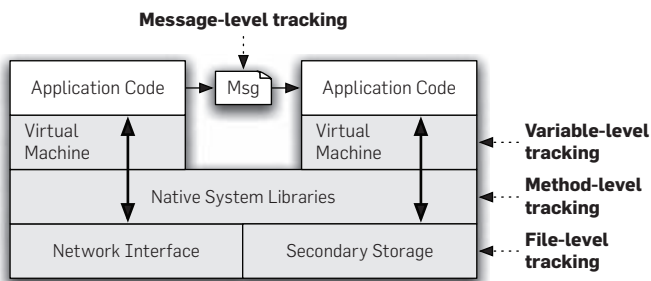
Whole-system, fine-grained dynamic taint analysis satisfies these challenges, if performance limitations and over-tainting can be overcome. Here, sensitive information is first identified at a *taint source* that assigns a *taint marking* indicating the information type. Smartphones have well-defined application programming interfaces (APIs) for retrieving privacy-sensitive information (e.g., microphone, location, and phone identifiers). The dynamic taint analysis then tracks how the labeled data impacts other data in a way that might leak the original sensitive information. This tracking is often performed at the instruction level. For example, if the instruction $a = b + c$ is executed and c has a taint t , a will have taint t after the instruction has executed. Finally, the impacted data is identified before it leaves the system at a *taint sink* (the network interface in our design).

Clearly, performing dynamic taint analysis at the instruction level incurs significant performance overhead. For example, whole-system emulation and per-process dynamic binary translation (DBT)^{2, 4, 17} commonly incur 2–20 times slowdown. This overhead occurs because for each monitored instruction, the tracking framework must (1) save the execution context, (2) perform the taint propagation, and then (3) restore the execution context.

We overcome this limitation by moving the tracking framework inside of the OS and taking advantage of the virtual machine (VM)-based architecture used by Android, BlackBerry, and Windows Phone. For these platforms, applications consist of Java-based or .NET byte-code that is executed within an interpreter. We modify the interpreter to perform taint propagation and then carefully extend propagation to the rest of the system. By modifying the interpreter, we avoid saving and restoring execution context. Furthermore, our approach focuses on tracking the data used by interpreted code, which is only a small fraction of the overall process memory. Note that our approach is not compatible with iOS, since it uses binary applications.

Figure 1 presents our tracking design. Tracking occurs in four ways. First, we instrument the VM interpreter to provide *variable-level tracking* within untrusted application code. Using variable semantics increases precision over traditional x86 tracking logics and focuses taint marking storage on data instead of code. Second, we use *message-level tracking* between applications. The message granularity minimizes IPC overhead while extending the analysis system-wide. Third, for system-provided native libraries, we use *method-level tracking*. Here, we run platform native code without instrumentation and patch the taint propagation on return. Finally, we

Figure 1. Multi-level approach for performance-efficient taint tracking within Android.



use *file-level* tracking to ensure that persistent information conservatively retains its taint markings.

While this design allows practical real-time tracking, it relies on the firmware’s integrity. We trust the virtual machine executing in user space and any native system libraries loaded by the untrusted interpreted application. Hence, we assume that only platform native libraries can be loaded. Without this, applications can not only remove taint markings, but also corrupt the tracking within the interpreter. In our target platform (Android), we modified the native library loader to only load native libraries from the firmware. To test compatibility, we surveyed the top 50 most popular free applications in each category of the Android Market (1,100 applications in total) in July 2010 and found that less than 5% of applications included a `.so` file. Therefore, we expect that TaintDroid is incompatible with only a small percentage of applications.

3. TAINTDROID

TaintDroid is a realization of our multiple granularity taint tracking for Android. Central to the design is a careful trade-off between tracking precision and performance. TaintDroid uses variable-level tracking within the VM interpreter. Multiple taint markings are stored as one *taint tag*. When applications execute native methods, variable taint tags are patched on return. Finally, taint propagation is extended to IPC and files.

This section overviews the core implementation challenges of TaintDroid. Here we discuss (a) taint tag storage, (b) interpreted code taint propagation, (c) native code taint propagation, (d) IPC taint propagation, and (e) secondary storage taint propagation. Additional details can be found in our original paper.⁹

3.1. Taint tag storage

Taint tag storage impacts both performance and memory overhead. Traditional taint tracking systems store one tag for every data byte or word.^{3, 23} Often, this tag consists of a single bit in implementations. To further reduce storage overhead, such systems only maintain tags for tainted bytes using non-adjacent shadow memory²³ or tag maps.²⁵ TaintDroid takes a different approach. Since we know which bytes are variables, we significantly reduce the scope of memory to track by only keeping track of the taint states of variables. This allows TaintDroid to store taint tags adjacent to variables in memory, which provides spatial locality

when accessing taint tags at runtime. Furthermore, it allows one to practically store a 32-bit bit vector with each variable, allowing 32 different taint markings.

TaintDroid adds taint tag storage for all scalar values in Android’s Dalvik VM interpreter. Android applications are written in Java, but compiled to a special DEX bytecode that is executed by Dalvik. Given these Java origins, TaintDroid must provide taint tag storage for method local variables, method arguments, class static fields, class instance fields, and arrays.

DEX bytecode differs from Java bytecode in that it is register based. This is important to the TaintDroid implementation. When a DEX method is called, Dalvik creates a new stack frame that allocates 32-bit register storage for all of the scalar and object reference variables used by the method. As shown in Figure 2, method arguments are also stored on the stack and are mapped to high indexed registers in the callee stack frame. TaintDroid provides taint tag storage for these variables by interleaving taint tags between the registers.

TaintDroid stores taint tags adjacent to class fields and arrays within internal data structures. Only one taint tag is stored per array to minimize storage overhead, which is often sufficient for strings. However, this loss in precision may result in false positives. For example, as soon as a tainted value is stored to an array, all values read out of the array will also be tainted. Fortunately, Java arrays frequently contain object references, which are infrequently tainted, resulting in fewer false positives in practice.

3.2. Interpreted code taint propagation

Operating on DEX bytecode provides TaintDroid several distinct advantages. First, all operations have clear semantics. Unlike x86, there is no lack of registers or strange conventions for clearing variables (e.g., `xor %eax, %eax`). Second, scalar values are distinct from pointers. This allows taint propagation to be more precise. Finally, variables that are not method local have clear taint tag storage (described above) that retains types.

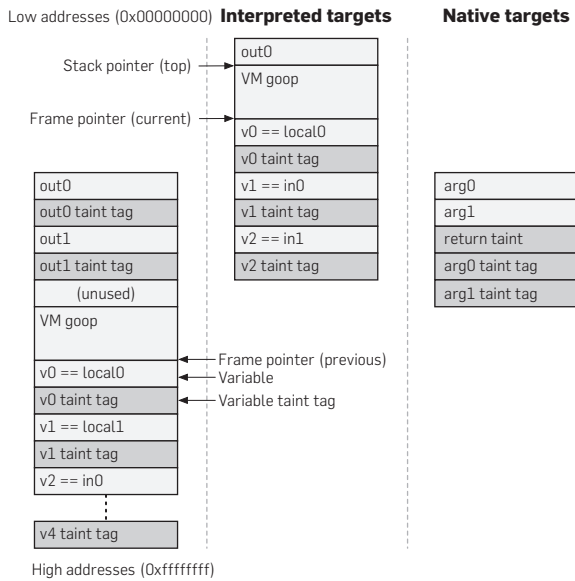
For the most part, taint tag propagation proceeds as one might expect. Instructions always overwrite the destination register; therefore, unary operations set the taint tag of the destination register to that of the source register, and binary operations (e.g., `a = b + c`) set the taint tag of the destination register to the union of the taint tags of the two source registers (e.g., $\tau(a) \leftarrow \tau(b) \cup \tau(c)$). For the implementation, the union is simply a bitwise OR of the taint tag bit vectors. However, there are several cases where the taint propagation is not straightforward (e.g., for array indexes and object references). A full propagation logic and discussion is provided in our original paper.⁹

3.3. Native code taint propagation

Native code is unmonitored in TaintDroid, as performing automated taint propagation would require heavyweight techniques such as dynamic binary translation (DBT). Instead, we synthesize the taint status after the method terminates based on a combination of source code inspection and simple heuristics.

Internal VM methods. The Dalvik VM contains a set of core methods that are called directly by interpreted code and are passed a pointer to an array of 32-bit register

Figure 2. Modified stack format. Taint tags are interleaved between registers for interpreted method targets and appended for native methods. Dark grayed boxes represent taint tags.



arguments and a pointer to a return value. TaintDroid places all the taint tags after the argument values (recall the stack in Figure 2). This ensures that methods that do not impact taint propagation require no modifications. For those that do, the respective taint tags are readily available. Of the 185 internal VM methods in Android version 2.1, only 5 required patching (e.g., for array manipulation and reflection).

JNI methods. The remaining majority of native methods use the Java Native Interface (JNI) and are invoked through a JNI call bridge. The call bridge parses Java arguments and assigns a return value, making it the ideal place to patch the tracking state after a native method executes. To do this, we define a method profile table that defines a list of (*from*, *to*) pairs indicating flows between method parameters, class variables, and return values. Completely populating the method profile table is best completed using automated static analysis tools; however, for the purposes of this work, we manually defined several methods as needed. To supplement this manual specification, we created a propagation heuristic: *assign the union of the method argument taint tags to the taint tag of the return value*. This heuristic is conservative if the method only operates on primitive and string arguments and return values. For Android version 2.1, we found this condition to hold for 913 of the 2,844 JNI methods. The remaining methods may have false negatives and potentially require explicit method profile specification. While we found these methods effective for our investigations, more thorough consideration of native code is a valuable direction for future work.

3.4. IPC taint propagation

When Android applications communicate with one another, they send *parcel* objects over the binder IPC interface. It is important for TaintDroid to propagate taint tags on parcels

to track sensitive information passed not only between downloaded third-party applications, but also between third-party applications and the system. In fact, much of Android's core functionality is implemented using the same application abstractions as third-party software.

TaintDroid assigns one taint tag per parcel message. This results in better performance and lower memory overhead than variable-level or byte-level tracking in parcels. Furthermore, variable-level tracking is subject to manipulation, because the parcel packing of different sized variables is defined by the sender and receiver. However, the disadvantage is false positives (similar to arrays). As we discuss in Section 7, this makes certain taint sources problematic for TaintDroid. Future implementations will investigate the overhead of finer-grained parcel tracking.

3.5. Secondary storage taint propagation

TaintDroid must ensure that when tainted data is written to a file, the taint tag is restored when it is later read. We currently store one taint tag per file, because finer-grained tracking would incur significant overhead. However, the drawback is false positives if the type of tracked information is frequently mixed. In our experiments, this was not a significant problem. To store taint tags, TaintDroid uses extended attributes in the file system. When TaintDroid was developed, the predominately used YAFFS2 file system did not have xattr support, which we needed to add. Official xattr support was later added to YAFFS2, and newer phones have a hardware flash translation layer that allows standard ext4 file systems. A second limitation of the Android storage architecture is the SDcard. Android uses a FAT file system for the SDcard, which does not support xattrs. We formatted the SDcard ext2 and patched the file write API to use file permissions consistent with FAT to ensure compatibility with existing applications.

4. PRIVACY HOOK PLACEMENT

Before TaintDroid can be used to monitor applications, taint sources must be added to the Android Framework. We modified the Android system code to add taint tags to various taint sources. For the most part, we chose to add the taint sources within the Java portion of system applications that retrieve the values from hardware. The following describes the most important classes of taint sources we encountered.

Low-bandwidth sensors. A variety of privacy-sensitive information types are acquired through low-bandwidth sensors, for example, location and accelerometer. Such information often changes frequently and is simultaneously used by multiple applications. Therefore, Android multiplexes access to low-bandwidth sensors using a sensor manager. This sensor manager represents an ideal point for taint source hook placement. We placed hooks in Android's LocationManager and SensorManager applications.

High-bandwidth sensors. Sources such as the microphone and camera are high-bandwidth. Each request from the sensor returns a large amount of data that is only used by one application. Therefore, the OS makes sensor information available via large data buffers, files, or both. When sensor information is shared via files, the file must be tainted

with the appropriate tag. We added hooks for both types of API abstractions provided for accessing microphone and camera interfaces.

Information databases. Shared information such as address books and SMS messages are often stored in file-based databases. By adding a taint tag to such database files, all information read from the file will be automatically tainted. We initially used this technique for tracking address book information. Later implementations modified Android's content resolver class to add an appropriate taint tag based on the name of the content provider (i.e., the "authority string") specified by the querying application.

Device identifiers. Information that uniquely identifies the phone or the user is privacy-sensitive. Not all personally identifiable information can be easily tainted. However, the phone contains several easily tainted identifiers: the phone number, SIM card identifiers (IMSI, ICC-ID), and device identifier (IMEI) are all accessed through well-defined APIs. We instrumented the APIs for the phone number, ICC-ID, and IMEI. An IMSI taint source has inherent limitations discussed in Section 7.

Network taint sink. TaintDroid identifies when tainted information is transmitted out the network interface. Our interpreter-based approach requires TaintDroid's code to detect network transmission within interpreted code. Hence, we instrumented the Java framework libraries at the point the native socket library is invoked.

5. APPLICATION STUDY

To demonstrate the utility of TaintDroid, we studied 30 popular third-party Android applications that have access to privacy-sensitive user data and the Internet. This set of applications was randomly selected from a larger set of popular applications that have access to the Internet and to at least one of location, camera, or audio data. We chose to bias our random selection toward applications with access to interesting privacy-sensitive information, because applications without access clearly cannot expose data. The details of our experimental methodology can be found in our original paper.⁹ The following describes our major findings.

Our experiments consisted of manually running and exploring the functionality of the applications. We recorded TaintDroid logs and a tcpdump packet trace for ground truth. We also took note of End User License Agreements (EULAs) and implicit expectations of data exposure. Our experiments generated 1,130 TCP connections, and TaintDroid correctly flagged 105 TCP connections as containing tainted privacy-sensitive information (i.e., TaintDroid had no false positives). The flagged TCP connections included both plaintext and binary encoded data.

Upon inspecting the 105 flagged TCP connections containing privacy-sensitive information, we found that 37 were for clearly legitimate uses. For example, several of these flagged TCP connections contained HTTP headers indicating the use of the Google Maps for Mobile (GMM) API, and the corresponding application showed a map of the user's location. However, the privacy-sensitive information disclosures in the remaining 68 flagged TCP connections were not expected. These findings are summarized in Table 1.

Location data to advertisement servers. Half of the studied applications exposed location data to third-party advertisement servers without implicit or explicit user consent. Of these fifteen applications, only two presented a EULA on first run; however, neither EULA indicated this practice. Exposure of location information occurred both in plaintext and in binary format. The latter highlights TaintDroid's advantages over simple pattern-based packet scanning. Applications sent location data in plaintext to admob.com, ad.qwapi.com, ads.mobelix.com (11 applications) and in binary format to FlurryAgent (4 applications). The plaintext location exposure to AdMob occurred in the HTTP GET string:

```
...& s=a14a4a93f1e4c68 &.& t=062A1CB1D476DE85B717D  
9195A6722A9&d%5Bcoord%5D=47.661227890000006%2C-  
122.31589477 &...
```

Investigating the AdMob SDK revealed that the `s=` parameter is an identifier unique to an application publisher, and the `coord=` parameter provides the geographic coordinates.

For binary data sent by FlurryAgent, we confirmed location exposure based on the following sequence of events. First, a component named "FlurryAgent" registers with the location manager to receive location updates. Then, TaintDroid log messages show the application receiving a tainted parcel from the location manager. Finally, the application's log to Android's logcat reports "sending report to http://data.flurry.com/aar.do," which occurs immediately after receiving the tainted parcel.

Our experiments indicate that these fifteen applications collect location data and send it to advertisement servers. In some cases, location data was transmitted to advertisement servers even when no advertisement was displayed in the application. However, we note that TaintDroid helped us verify that three of the studied applications (not included in Table 1) only transmitted location data per user's request to pull localized content from their servers. This finding demonstrates the importance of monitoring how the application *actually* uses or abuses the granted permissions.

Phone information. Of the 30 studied applications, 20 require permissions to read phone state and access the Internet. We found that 2 of the 20 applications transmitted to their server (1) the device's phone number, (2) the IMSI, which is a unique 15-digit code used to identify an individual user on a GSM network, and (3) the ICC-ID number, which is a unique SIM card serial number. We verified that messages were flagged correctly by inspecting the plaintext payload. In neither case was the user informed that this information was transmitted off the phone. Note that while we did not explicitly track the IMSI (see Section 7), it was contained in the plaintext network buffer flagged by TaintDroid.

This finding demonstrates that Android's coarse-grained access control provides insufficient protection against third-party applications seeking to collect sensitive data. Moreover, we found one application that transmits the phone information *every time* the phone boots. While this application displays a terms of use on first use, the terms of use does not specify collection of this highly sensitive data. Surprisingly, this application transmits the phone

Table 1. Application study results.

Application [package.name]	Permissions				Info sent		
	Location	Phone state	Camera	Microphone	Location	Phone info	IMEI
Babble Book [com.kalicenscy.babble]	✓						
Cestos Full [com.chickenbrickstudios.cestos_full]	✓						
Manga Browser [com.mangabrowser.main]	✓				•		
Movies and showtimes [com.stylem.movies]	✓						
Solitaire Free [com.mediafill.solitaire]	✓				•		
The Weather Channel [com.weather.Weather]	✓						
3001 Wisdom Quotes Lite [com.xim.wq_lite]	✓	✓			•		
Antivirus Free [com.antivirus]	✓	✓			•	•	•
Astrid [com.timsu.astrid]	✓	✓			•		
BBC News listen & tweet [daaps.media.bbc]	✓	✓			•		
Blackjack [spr.casino]	✓	✓			•		
Bump [com.bumptechnology.bumppga]	✓	✓					•
Children's ABC Animals (lite) [com.mob4.childrenabc.animals]	✓	✓			•		
Hearts (Free) [com.bytesequencing.hearts_ads]	✓	✓			•		
Horoscope [fr.telemaque.horoscope]	✓	✓			•		•
Mabilo Ringtones [mabilo.ringtones]	✓	✓			•		
The directory for Germany [de.dastelefonbuch.android]	✓	✓					
Traffic Jam Free [com.jiuzhangtech.rushhour]	✓	✓			•		
Wertago for Nightlife [com.wertago]	✓	✓			•		•†
Yellow Pages [com.avantar.yip]	✓	✓					•
Knocking Live Video Beta [com.pointyheadsllc.knockingvideo]	✓	✓	✓				•
Layar [com.layar]	✓	✓	✓				◦
Pro Basketball Scores [com.plusmo.probasketballscores]	✓	✓	✓		•		
Slide: Spongebob [com.mob4.slideme.qw.android.spongebob]	✓	✓	✓		•		
The coupons App [thecouponsapp.coupon]	✓	✓	✓			•	•
Trapster [com.trapster.android]	✓	✓	✓				•
Barcode Scanner [com.google.zxing.client.android]			✓				
iXmat Barcode Scanner [com.ixellence.ixmat.android.community]			✓				
Myspace [com.myspace.android]			✓				
Evernote [com.evernote]	✓		✓	✓			

✓ = Potential violation; ◦ = Clearly stated in EULA; † Sent the hash of the value.

data immediately after it is installed, which is before it is even used.

Device unique ID. The device's IMEI was also exposed by applications. The IMEI uniquely identifies a specific mobile phone and is used to prevent a stolen handset from accessing the cellular network. TaintDroid flags indicated that nine applications transmitted the IMEI. Seven out of the nine applications either do not present an EULA or do not specify IMEI collection in the EULA. One of the seven applications is a popular social networking application and another is a location-based search application. Furthermore, we found two of the seven applications include the IMEI when transmitting the device's geographic coordinates to their content server, potentially repurposing the IMEI as a client ID.

In comparison, two of the nine applications treat the IMEI with more care. One application displays a privacy statement that clearly indicates that the application collects the device ID. The other uses the hash of the IMEI instead of the number itself. We verified this practice by comparing results from two different phones. Hashing the IMEI provides more protection, because it cannot be reversed to obtain the actual IMEI. However, if all applications hash the IMEI directly, similar privacy concerns can result.

The collection of phone identifiers allows third parties to track user behavior. Phone numbers are often easy to

correlate with the owner's name, as many users post their phone number on social networking and other websites. However, collecting seemingly unidentifiable numbers such as the IMSI, ICC-ID, and the IMEI also has privacy implications. First, all applications on the phone use the same phone identifiers. If identifiers and behaviors are collected by an entity that is associated with many applications (e.g., an ad or analytics service), more accurate user profiles can be created. Second, these identifiers are fixed for the duration the user uses the phone, and potentially longer if the SIM card is moved to a new phone. This property means that users cannot simply clear the tracking cookies as they might in a Web browser. Finally, these identifiers are often collected along with personally identifiable information such as email addresses. Such collections create small databases that can be used to correlate actual users with their phone identifiers. Traditionally, this mapping is only held by cellular providers.

6. PERFORMANCE EVALUATION

During the application study, we noticed very little performance overhead. This is likely because (1) most applications are primarily in a "wait state," and (2) heavyweight operations (e.g., screen updates and Webpage rendering) occur in unmonitored native libraries.

We evaluated the performance of TaintDroid for Android

Table 2. Macrobenchmark results.

	Android (ms)	TaintDroid (ms)
App load time	63	65
Address book (create)	348	367
Address book (read)	101	119
Phone call	96	106
Take picture	1718	2216

version 2.1 using macrobenchmarks representing common smartphone activities: loading an application, accessing the address book, making a phone call, and taking a picture. As shown in Table 2, our macrobenchmarks observed negligible overhead (less than 30ms), with the exception of taking a picture, which added just over half a second. This overhead is likely due to the current method of propagating taint tags to files using `xattrs`, which could be improved with caching.

While the macrobenchmarks report the performance overhead perceived by users during common smartphone use, we also performed a microbenchmark on Java operations. For this experiment, we used an Android port of the standard CaffeineMark 3.0 benchmark for Java. TaintDroid has an average overall CPU overhead of 14%. We also measured the memory consumption of the benchmark process during the experiments. The benchmark process consumed 21.28MB on Android and 22.21MB on TaintDroid, indicating a 4.4% memory overhead.

7. DISCUSSION

Approach limitations. To minimize performance overhead, TaintDroid only tracks data flows (i.e., explicit flows) and does not track control flows (i.e., implicit flows). Section 5 shows that TaintDroid can track the flow of sensitive data and identify many applications that exfiltrate sensitive information. However, applications that are truly malicious can game our system and exfiltrate privacy-sensitive information through control flows. Fully tracking control flow requires static analysis,^{7, 14} which is challenging for third-party applications whose source code is unavailable. Direct control flows can be tracked dynamically if a taint scope can be determined²¹; however, DEX does not maintain branch structures that TaintDroid can leverage. On-demand static analysis to determine method control flow graphs (CFGs) provides this context¹⁵; however, TaintDroid does not currently perform such analysis in order to avoid false positives and significant performance overhead. Our data flow taint propagation logic is consistent with existing, well-known, taint tracking systems.^{3, 23} Finally, once information leaves the phone, it may return in a network reply. TaintDroid cannot track such information propagation once the information leaves the phone.

Implementation limitations. Android uses the Apache Harmony implementation of Java with a few custom modifications. This implementation includes support for the `PlatformAddress` class, which contains a native address and is used by `DirectBuffer` objects. The file and network IO APIs include write and read “direct” variants that consume the

native address from a `DirectBuffer`. TaintDroid does not currently track taint tags on `DirectBuffer` objects, because the data is stored in opaque native data structures. Currently, TaintDroid logs when a read or write “direct” variant is used, which anecdotally occurs with minimal frequency. Similar implementation limitations exist with the `sun.misc.Unsafe` class, which also operates on native addresses.

Taint source limitations. While TaintDroid is very effective for tracking sensitive information, it causes significant false positives when the tracked information contains configuration identifiers. For example, the IMSI numeric string consists of a Mobile Country Code (MCC), Mobile Network Code (MNC), and Mobile Station Identifier Number (MSIN), which are all tainted together. Android uses the MCC and MNC extensively as configuration parameters when communicating other data. If the IMSI is treated as tainted, this causes all information in a parcel to become tainted, eventually resulting in an explosion of tainted information. Thus, for taint sources that contain configuration parameters, tainting individual variables within parcels would be more appropriate. However, as our analysis results in Section 5 show, message-level taint tracking is effective for the majority of our taint sources.

8. RELATED WORK

Information flow tracking and control has been the basis of many operating system and programming language designs over the past several decades. For brevity, we focus on systems using dynamic taint analysis, which is primarily used to track information flows in legacy programs. It has been used to enhance system integrity (e.g., defend against software attacks^{4, 16, 17}) and confidentiality (e.g., discover privacy exposure^{8, 23, 25}), as well as track Internet worms.⁵ Dynamic tracking approaches range from whole-system analysis using hardware extensions^{6, 19, 20} and emulation environments^{3, 23} to per-process tracking using dynamic binary translation (DBT).^{2, 4, 17, 25} The performance and memory overhead associated with dynamic tracking have stimulated much research on optimizations, including optimizing context switches,¹⁷ on-demand tracking¹² based on hypervisor introspection, and function summaries for code with known information flow properties.²⁵ If source code is available, significant performance improvements can be achieved by automatically instrumenting legacy programs with dynamic tracking functionality.^{13, 22} Automatic instrumentation has also been performed on x86 binaries,¹⁸ providing a compromise between source code translation and DBT. Our TaintDroid design was inspired by these prior works, but addresses different challenges unique to mobile phones. To our knowledge, TaintDroid is the first taint tracking system for a mobile phone and is the first dynamic taint analysis system to achieve practical system-wide analysis through the integration of tracking multiple data object granularities.

Finally, dynamic taint analysis has been applied to virtual machines and interpreters. Haldar et al.¹⁰ instrument the Java String class with taint tracking to prevent SQL injection attacks. WASP¹¹ has similar motivations; however, it uses positive tainting of individual characters to ensure that the SQL query contains only high-integrity substrings. Chandra and Franz¹ propose fine-grained information flow tracking within the JVM and instrument Java byte-code to

aid control flow analysis. Similarly, Nair et al.¹⁵ instrument the Kaffe JVM. Vogt et al.²¹ instrument a Javascript interpreter to prevent cross-site scripting attacks. Xu et al.²² automatically instrument the PHP interpreter source code with dynamic information tracking to prevent SQL injection attacks. Finally, the Resin²⁴ environment for PHP and Python uses data flow tracking to prevent an assortment of Web application attacks. When data leaves the interpreted environment, Resin implements filters for files and SQL databases to serialize and de-serialize objects and policy with byte-level granularity. TaintDroid's interpreted code taint propagation bears similarity to some of these works. However, TaintDroid implements system-wide information flow tracking, seamlessly connecting interpreter taint tracking with a range of operating system sharing mechanisms.


9. CONCLUSION

While smartphone operating systems allow users to control applications' access to sensitive information, users lack visibility into how applications use their private data. To address this, we presented TaintDroid, an efficient, system-wide information flow tracking tool that can simultaneously track multiple sources of sensitive data. A key design goal of TaintDroid is efficiency, which is achieved by integrating four granularities of taint propagation (variable-level, message-level, method-level, and file-level). Our evaluation shows that TaintDroid has only a 14% performance overhead on a CPU-bound microbenchmark. Previously, most work on taint tracking was either slow (requiring multiple times performance overhead) or required source code. The source code for Android applications is not available; therefore, one might have expected TaintDroid to be very slow. TaintDroid shows this is not the case: one can track information flows of Android applications without source code, with modest overhead.

We used TaintDroid to study the behavior of 30 popular third-party applications and found that two-thirds handle sensitive data inappropriately. In particular, 15 of the 30 applications shared users' locations with remote advertising and analytics servers. Our findings demonstrate the effectiveness and value of enhancing smartphone platforms with monitoring tools such as TaintDroid.

TaintDroid is an ongoing effort that has been incorporated into further projects by both the authors and others in the research community. TaintDroid is available for Android version 2.1, version 2.3 (and adding JIT support), and version 4.1. Information for downloading and building TaintDroid can be found at <http://www.appanalysis.org>.

Acknowledgments

We thank everyone who helped with the original paper.⁹ Enck and McDaniel were partially supported by NSF Grants CNS-0905447, CNS-0721579, and CNS-0643907. Cox and Gilbert were partially supported by NSF CAREER Award CNS-0747283. 

References

- Chandra, D., Franz, M. Fine-grained information flow analysis and enforcement in a Java virtual machine. In *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC)* (Dec. 2007).
- Cheng, W., Zhao, Q., Yu, B., Hiroshige, S. TaintTrace: efficient flow tracing with dynamic binary rewriting. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)* (Jun. 2006), 749–754.
- Chow, J., Pfaff, B., Garfinkel, T., Christopher, K., Rosenblum, M. Understanding data lifetime via whole system simulation. In *Proceedings of the 13th USENIX Security Symposium* (Aug. 2004).
- Clause, J., Li, W., Orso, A. Dytan: a generic dynamic taint analysis framework. In *Proceedings of the 2007 International Symposium on Software Testing and Analysis* (2007), 196–206.
- Costa, M., Crowcroft, J., Castro, M., Rowstron, A., Zhou, L., Zhang, L., Barham, P. Vigilante: end-to-end containment of internet worms. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Oct. 2005), 133–147.
- Crandall, J.R., Chong, F.T. Minos: control data attack prevention orthogonal to memory model. In *Proceedings of the International Symposium on Microarchitecture* (Dec. 2004), 221–232.
- Denning, D.E., Denning, P.J. Certification of Programs for Secure Information Flow. *Commun. ACM* 20, 7 (Jul. 1977).
- Egele, M., Kruegel, C., Kirda, E., Yin, H., Song, D. Dynamic spyware analysis. In *Proceedings of the USENIX Annual Technical Conference* (Jun. 2007), 233–246.
- Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (Oct. 2010).
- Haldar, V., Chandra, D., Franz, M. Dynamic taint propagation for Java. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)* (Dec. 2005), 303–311.
- Halfond, W.G., Orso, A., Manolios, P. WASP: protecting web applications using positive tainting and syntax-aware evaluation. *IEEE Trans. Softw. Eng.* 34, 1 (2008), 65–81.
- Ho, A., Fetterman, M., Clark, C., Warfield, A., Hand, S. Practical taint-based protection using demand emulation. In *Proceedings of the European Conference on Computer Systems (EuroSys)* (Apr. 2006), 29–41.
- Lam, L.C., Cker Chiueh, T. A general dynamic information flow tracking framework for security applications. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)* (Dec. 2006), 463–472.
- Myers, A.C. JFlow: practical mostly-static information flow control. In *Proceedings of the ACM Symposium on Principles of Programming Languages (POPL)* (Jan. 1999).
- Nair, S.K., Simpson, P.N., Crispo, B., Tanenbaum, A.S. A virtual machine based information flow control system for policy enforcement. In *The 1st International Workshop on Run Time Enforcement for Mobile and Distributed Systems (REM)* (2007).
- Newsome, J., Song, D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of the 12th Network and Distributed System Security Symposium (NDSS)* (2005).
- Qin, F., Wang, C., Li, Z., seop Kim, H., Zhou, Y., Wu, Y. LIFT: a low-overhead practical information flow tracking system for detecting security attacks. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture* (2006), 135–148.
- Saxena, P., Sekar, R., Puranik, V. Efficient fine-grained binary instrumentation with applications to taint-tracking. In *Proceedings of the IEEE/ACM symposium on Code Generation and Optimization (CGO)* (Apr. 2008), 74–83.
- Suh, G.E., Lee, J.W., Zhang, D., Devadas, S. Secure program execution via dynamic information flow tracking. In *Proceedings of the Conference on Architectural Support for Programming Languages and Operating Systems (ASPLoS)* (Oct. 2004), 85–96.
- Vachharajani, N., Bridges, M.J., Chang, J., Rangan, R., Ottoni, G., Blome, J.A., Reis, G.A., Vachharajani, M., August, D.I. RIFLE: an architectural framework for user-centric information-flow security. In *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture* (2004), 243–254.
- Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., Vigna, G. Cross-site scripting prevention with dynamic data tainting and static analysis. In *Proceedings of the 14th Network and Distributed System Security Symposium* (2007).
- Xu, W., Bhatkar, S., Sekar, R. Taint-enhanced policy enforcement: a practical approach to defeat a wide range of attacks. In *Proceedings of the USENIX Security Symposium* (Aug. 2006), 121–136.
- Yin, H., Song, D., Egele, M., Kruegel, C., Kirda, E. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM Conference on Computer and Communications Security* (2007), 116–127.
- Yip, A., Wang, X., Zeldovich, N., Kaashoek, M.F. Improving application security with data flow assertions. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Oct. 2009).
- Zhu, D.Y., Jung, J., Song, D., Kohno, T., Wetherall, D. Tainteraser: protecting sensitive data leaks using application-level taint tracking. *Operating Sys. Rev.* 45, 1 (2011), 142–154.

William Enck (enck@cs.ncsu.edu), Department of Computer Science, North Carolina State University.

Peter Gilbert and Landon P. Cox ([gilbert, lpcox]@cs.duke.edu), Department of Computer Science, Duke University.

Byung-Gon Chun (bgchun@snu.ac.kr), Seoul National University.

Jaeyeon Jung (jjung@microsoft.com), Microsoft Research.

Patrick McDaniel (mcdaniel@cse.psu.edu), Department of Computer Science and Engineering, Pennsylvania State University.

Anmol N. Sheth (anmol.sheth@technicolor.com), Technicolor Research.

Masdar Institute of Science and Technology, Abu Dhabi Faculty Positions

*The Electrical Engineering and
Computer Science Department
Computing & Information Science*

Masdar Institute of Science and Technology, located in Abu Dhabi, U.A.E., is a private, not-for-profit, independent, graduate-level, research-driven institute developed with the support and cooperation of Massachusetts Institute of Technology (MIT). The goal of the Institute is to develop, over a period of years, indigenous R&D capacity in Abu Dhabi, addressing issues of importance to the region in critical areas such as: renewable energy, sustainability, environment, water resources and microelectronics. The Institute offers graduate degree programs (MSc & PhD) in science and engineering disciplines with a focus on advanced energy and sustainable technologies (www.masdar.ac.ae and <http://web.mit.edu/mit-tdp/www/>). Several industrial collaborative research projects are on-going, such as, with Siemens (Smart Grids/ Smart Buildings), ATIC (Advanced Technology Investment Company), Boing/Etihad/UOP Honeywell, and ADCO (Abu Dhabi Company for On-shore Oil Operations).

The Electrical Engineering and Computer Science department program (EECS) at the Masdar Institute invites applications for a full-time faculty position at the Assistant, Associate or Full Professor level in all areas of Computing and Information Sciences, to complement its growing and highly successful faculty body (for more information see <http://cis.masdar.ac.ae/>)

Candidates must hold an earned doctorate in Computer Science, Information Science, or related field; have a solid publication record; show ability and potential to demonstrate a commitment to excel in both research and teaching. The successful candidate is expected to develop a research program, supervise graduate students, and show outstanding teaching at graduate levels. Successful candidates will also be required to seek external funding, engage with both local and international stakeholders, and participate in the Institute's service and outreach activities.

Application submittal information: Massachusetts Institute of Technology is assisting Masdar Institute in the search. Initial screening of applications will begin immediately and the positions will remain open until filled. Application materials should include applicant name and contact information, a curriculum vitae, statements of research and teaching interests, an application letter describing the applicant's current position and how his/her experience matches the position requirements, and e-mail contact information for at least three references. Materials must be submitted electronically to: <https://academicjobsonline.org/ajo/MasdarInstitute>.

Max Planck Institute for Software Systems Senior Faculty position in Software Systems

Applications are invited for a senior faculty position in the Max Planck Institute for Software Systems (MPI-SWS). The position is that of a director and scientific member of the Max Planck Society, and is comparable to an endowed chair position at a leading university. Directors lead their individual research groups, and also provide strategic direction for the institute, mentor junior faculty, and take turn in chairing the faculty. A successful candidate is an internationally recognized leader in the research community, and pursues a compelling and far-reaching research vision.

All areas related to the study, design, and engineering of software systems are considered. These areas include, but are not limited to, security and privacy, embedded and mobile systems, social computing, large-scale data management, programming languages and systems, software verification and analysis, parallel and distributed systems, storage systems, and networking. Preference will be given to candidates whose research

complements existing strengths.

MPI-SWS, founded in 2005, is part of a network of 82 Max Planck Institutes, Germany's premier basic research facilities. MPIs have an established record of world-class, foundational research in the fields of medicine, biology, chemistry, physics, technology and humanities. Since 1948, MPI researchers have won 17 Nobel prizes. MPI-SWS aspires to meet the highest standards of excellence and international recognition with its research in software systems.

To this end, the institute offers a unique environment that combines the best aspects of a university department and a research laboratory:

- a) Faculty independently lead a team of graduate students and post-docs. They have full academic freedom and publish their research results freely. Substantial base funding complements third-party funds.
- b) Faculty supervise doctoral theses, and have the opportunity to teach graduate and undergraduate courses.
- c) Faculty are provided with outstanding technical and administrative support facilities as well as internationally competitive compensation packages.

Faculty Search

ShanghaiTech University

The newly launched *ShanghaiTech University* invites *highly qualified* candidates to fill multiple tenure-track/tenured faculty positions as its core team in the School of Information Science and Technology (SIST). Candidates should have exceptional academic records or demonstrate strong potential in cutting-edge research areas of information science and technology. They must be fluent in English. Overseas academic connection or background is highly desired.

ShanghaiTech is built as a world-class research university for training future generations of scientists, entrepreneurs, and technological leaders. Located in Zhangjiang High-Tech Park in the cosmopolitan Shanghai, ShanghaiTech is ready to trail-blaze a new education system in China. Besides establishing and maintaining a world-class research profile, faculty candidates are also expected to contribute substantially to graduate and undergraduate education within the school.

Academic Disciplines: We seek candidates in all cutting edge areas of information science and technology. Our recruitment focus includes, but is not limited to: computer architecture and technologies, nano-scale electronics, high speed and RF circuits, intelligent and integrated signal processing systems, computational foundations, big data, data mining, visualization, computer vision, bio-computing, smart energy/power devices and systems, next-generation networking, as well as inter-disciplinary areas involving information science and technology.

Compensation and Benefits: Salary and startup funds are highly competitive, commensurate with experience and academic accomplishment. We also offer a comprehensive benefit package to employees and eligible dependents, including housing benefits. All regular ShanghaiTech faculty members will be within its new tenure-track system commensurate with international practice for performance evaluation and promotion.

Qualifications:

- A detailed research plan and demonstrated record/potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, or related field);
- A minimum relevant research experience of 4 years.

Applications: Submit (in English) a cover letter, a 2-page research plan, a CV plus copies of 3 most significant publications, and names of three referees to: sist@shanghaitech.edu.cn by March 31st, 2014 (until positions are filled). For more information, visit <http://www.shanghaitech.edu.cn>.



The Hong Kong Polytechnic University is a government-funded tertiary institution in Hong Kong. It offers programmes at various levels including Doctorate, Master's, and Bachelor's degrees. It has a full-time academic staff strength of around 1,200. The total consolidated expenditure budget of the University is close to HK\$5 billion per year.

DEPARTMENT OF COMPUTING

The Department of Computing is being ranked 49th in Computer Science and Information Systems in 2013 QS World University Ranking. The Department offers a full range of programmes leading to the awards from bachelor degrees to PhDs and currently has 32 academic staff conducting high-quality teaching and high-impact research. The Department strives to achieve academic excellence and aims to become one of the leaders in inter-disciplinary education and research through teamwork and partnership. The Department is determined to provide an environment conducive to the staff's career development and productivity through firm commitment of funding, resources and infrastructure support. For more information about the Department, please visit its website: <http://www.comp.polyu.edu.hk>.

The Department is seeking for outstanding applicants in the areas related to Big Data and Human Centered Computing. In particular, candidates with expertise and experiences in big data platforms, analytics and visualization, human behavior analysis, and human-computer interaction are strongly encouraged to apply.

(1) Associate Professor / Assistant Professor (two posts)

The appointees will be required to (a) teach at both undergraduate and postgraduate levels, and supervise research students; (b) initiate innovative research projects that lead to publications in top-tier refereed journals and awards of external research grants; (c) engage in research collaborations both internally and internationally; (d) participate in professional services to the academic community and in promotional activities; and (e) contribute to departmental activities.

Applicants should have a PhD degree with an excellent track record of research in one of the above areas relevant to the strategic development of the Department. Applicants should also show strong evidence of commitment to research that leads to top quality publications with high impact.

Candidates with higher qualification/experience will be considered for the post of Associate Professor.

(2) Research Assistant Professor (two posts)

The appointees will be required to (a) conduct innovative research projects that lead to publications in top-tier refereed journals and awards of external research grants; (b) engage in research collaborations both internally and internationally; (c) teach at both undergraduate and postgraduate levels, and supervise research students; (d) participate in professional services to the academic community and in promotional activities; and (e) contribute to departmental activities.

Applicants should have a PhD degree plus an excellent track record of research in one of the above areas relevant to the strategic development of the department. Applicants should also show strong evidence of commitment to research that leads to top quality publications with high impact.

Remuneration and Conditions of Service

A highly competitive remuneration package will be offered. Initial appointment for Assistant Professor will be on a fixed-term gratuity-bearing contract. Appointment for Research Assistant Professor (with remuneration package same as that for an Assistant Professor) will be on a fixed-term gratuity-bearing contract for up to three years. Re-engagement thereafter is subject to mutual agreement. An appropriate term will be provided for appointment at Associate Professor level. Applicants should state their current and expected salary in the application.

Application

Please submit application form via email to hrstaff@polyu.edu.hk; by fax at (852) 2364 2166; or by mail to **Human Resources Office, 13/F, Li Ka Shing Tower, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong**. If you would like to provide a separate curriculum vitae, please still complete the application form which will help speed up the recruitment process. Application forms can be obtained via the above channels or downloaded from <http://www.polyu.edu.hk/hro/job.htm>. **Recruitment will continue until the positions are filled.** Details of the University's Personal Information Collection Statement for recruitment can be found at <http://www.polyu.edu.hk/hro/jobpics.htm>.

For further details about the University, please visit www.polyu.edu.hk

MPI-SWS currently has 10 tenured and tenure-track faculty and 50 doctoral and post-doctoral researchers. The institute is funded to support 17 faculty and up to 100 doctoral and post-doctoral positions. Additional growth through outside funding is expected. We maintain an open, international and diverse work environment and seek applications from outstanding researchers regardless of national origin or citizenship. The working language is English.

The institute is located in Kaiserslautern and Saarbruecken, in the tri-border area of Germany, France and Luxembourg. The area offers a high standard of living, beautiful surroundings and easy access to major metropolitan areas in the center of Europe, as well as a stimulating, competitive and collaborative work environment. In immediate proximity are the MPI for Informatics, Saarland University, the Technical University of Kaiserslautern, the German Center for Artificial Intelligence (DFKI), and the Fraunhofer Institutes for Experimental Software Engineering and for Industrial Mathematics.

Qualified candidates are invited to send a CV and cover letter to rupak@mpi-sws.org. The review of applications will begin on Feb 1, 2014; applications will continue to be accepted until the position is filled.

The Max Planck Society is committed to increasing the representation of women and individuals with physical disabilities in Computer Science. We particularly encourage such individuals to apply.

Washington College Department of Mathematics & Computer Science Assistant Professor

The Department of Mathematics & Computer Science of **Washington College** in Chestertown Md. invites applications for a tenure-track position in computer science at the rank of **Assistant Professor** beginning in August 2014. The successful candidate will be expected to teach a variety of upper-level and introductory classes in computer science, and to supervise senior undergraduate theses. For full details visit <http://www.washcoll.edu/offices/human-resources/employment.php>



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable. Rates: \$325.00 for six lines of text, 40 characters per line. \$32.50 for each additional line after the first six. The MINIMUM is six lines.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact: acmm mediasales@acm.org Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

Ads are listed for a period of 30 days.

**For More Information Contact:
ACM Media Sales
at 212-626-0686 or
acmm mediasales@acm.org**



Peter Winkler

DOI:10.1145/2578281

Puzzled Solutions and Sources

Last month (February 2014) we posted three games in which you were asked to pick a positive integer. The question in each was: What is the highest number you should think about picking? Here, we offer solutions to all three. How did you do?

1. Found dollar.

Alice and Bob are vying for a found dollar, with lowest number the winner and a tie winning it for neither. Sadly, the only “Nash equilibrium” solution is for both players to choose “1” and the dollar to go unclaimed—a mini “prisoner’s dilemma.” Collaboration could have won each of them 50 cents.

2. Zero sum.

Here, the writer of the lower number wins \$1 from the other player, unless it is lower by only 1, in which case the player with the higher number would win \$2 from the other player. A tie would result in no money changing hands. This problem was published by Martin Gardner, appearing as Problem 11.13 in his *The Colossal Book of Short Puzzles and Problems* (W.W. Norton & Co., New York, 2006). The Nash equilibrium solution would be to write “1,” “2,” “3,” “4,” or “5” with probabilities $1/16$, $5/16$, $1/4$, $5/16$, and $1/16$, respectively. (Gardner did not provide a proof, but it is not difficult to show this is a Nash equilibrium strategy and, with a little more work, the only one.) The highest number either player should consider writing is thus “5.”

3. Swedish lottery.

This game, which I included in my book *Mathematical Puzzles: A Connoisseur’s Collection* (A K Peters Ltd., Natick, MA, 2001) as “Swedish Lottery,” has the surprising property that the equilibrium strategy calls for playing every positive integer with positive probability. There is no largest integer you should consider playing. To see this, imagine for the sake of reaching a contradiction that there is a highest number you (and the other players) should ever play; call it number k . When would you win playing k ? Only when the other players choose the same lower number. But if you played $k+1$, you would win all those times *plus* the times the other two players both play k . $k+1$ is thus a better choice than k , contradicting the assumption that the strategy is a Nash equilibrium. There is, in fact, a common Nash equilibrium strategy for Alice, Bob, and Charlie—calling for the number j to be selected with probability $(1-r)r^{j-1}$ where r is the root of a certain cubic equation and approximately 0.543689. The probabilities for choosing 1, 2, 3, and 4 are about 0.456311, 0.248091, 0.134884, and 0.073335, respectively; the probability of choosing

a number greater than 100 is teeny. As an experiment in 2010, I ran a Swedish Lottery among 40 graduate students in Dartmouth’s Computer Science Department. The winning number was 6. A much larger version—actually a game a day for 47 straight days—was run in Sweden in 2007 under the name “Limbo.” The number of players each day averaged 53,785; the lowest winning number was 162, the highest 4,465. For more, see <http://swopec.hhs.se/hastef/papers/hastef0671.pdf>.

All readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science, at Dartmouth College, Hanover, NH.

© 2014 ACM 0001-0782/14/03 \$15.00



ACM
SIGIR 2014
JULY 6 – 11, 2014

THE 37TH ANNUAL INTERNATIONAL ACM SIGIR
CONFERENCE

Planning is well underway for the 37th Annual ACM SIGIR Conference, to be held on the Gold Coast, Queensland from Sunday 6 – Friday 11 July 2014.

SIGIR is the major international forum for the presentation of new research results and for the demonstration of new systems and techniques in the broad field of information retrieval. Next year's conference will feature 6 days of papers, posters, demonstrations, tutorials and workshops focused on research and development in the area of informational retrieval, also known as search.

The Conference and Program Chairs are now inviting all those working in areas related to information retrieval to submit original papers related to any aspect of information retrieval theory and foundation, techniques and application. A list of key submission dates, relevant paper topics, submission guidelines and instructions are now available on the official SIGIR 2014 Conference website: <http://sigir.org/sigir2014/callforpapers.php>. Abstract submission closes 20 January 2014.



In addition, to a full scientific program the conference presents delegates with the perfect networking opportunity, bringing together several hundred researchers, academic faculty, students and industry leaders from around the world.



SIGIR 2014 will take place at one of Australia's premier tourist destinations, the Gold Coast. From the iconic Surfers Paradise beach, to the sophisticated dining precincts of Broadbeach and out to the lush, green Hinterland, there is a new experience waiting for you at every turn on the Gold Coast. Theme parks, world-renowned beaches, shopping and almost year-round sunshine are just a few reasons why delegates will enjoy this vibrant coastal city.

From the SIGIR Conference Organising committee we hope to see you on the Gold Coast in 2014 for the 37th Annual ACM SIGIR Conference.

[CONTINUED FROM P. 112] Hence “reduced instruction set,” and your focus on including only instructions that were actually used.

We wanted to build a fast, lean microprocessor that performed well by tapping the rapidly improving capabilities of silicon due to Moore’s Law, and do the complicated stuff with software. There was a related effort at IBM led by John Cocke that predated ours, and John Hennessy’s work at Stanford came a little later.

Take me through the project’s evolution and key concepts like register windows, which break down the register file into global and local variables.

Part of what I realized at DEC is that it is hard to get things done at universities, because universities do not have deadlines. The only deadline at universities is when classes start and classes end. So we pursued the project through a series of four courses. The first one was the architecture investigation, and two of the grad students in that class came up with the idea of register windows as a way to take advantage of the resources to get good performance. Then we learned VLSI design in the next course, and we implemented them in subsequent courses.

After RISC, you moved on to the Redundant Arrays of Inexpensive Disks, or RAID—a powerful architecture whose name morphed from “inexpensive” to “independent” once it was commercialized.

When Randy Katz and I started working, that was the premise: to put a lot of inexpensive disks together in an array to get more performance at a cheaper per-byte cost. By adding redundancy, the cheap ones could actually be made more reliable than the expensive mainframe disks IBM was making.

After it became successful, and people wanted to sell it, the word “inexpensive” was a problem. If it is inexpensive, how can we charge \$100,000 for it? So they asked Randy if they could call it “independent.” That is marketing. Fine. But when I give a retrospective talk and I use the word “inexpensive,” people think I got it wrong.

Let’s talk about your more recent work. You are currently involved in a big data

“Now that we have made so much progress, we should reach out to other fields because we need new challenges to drive our technology.”

lab called the AMPLab—which stands for “algorithms, machines, and people”—that’s taking on some very tough questions...

Someone I spoke with put it this way: in the first 50 years of computer science, we did not need to talk to people in other fields. Since software was so bad and hardware was so slow, it was obvious what to work on. But now that we have made so much progress, we should reach out to other fields because we need new challenges to drive our technology.

In the three years since AMPLab was founded, you have worked on a series of collaborative projects to do with cancer genomics.

Cancer genomics is a realm where I think computer scientists can really help. My last biology course was in high school, but my strength is that I can bring together a multidisciplinary team. In 2011, I wrote an Op-Ed in *The New York Times* arguing just that. It did not receive a universally warm reaction, because the premise is that if computer scientists learn a little biology, we may be able to make contributions that are as important as those of people who dedicate their lives to biology and learn a little computer science.

Three years on, I am standing by my editorial. Our lab is now rolling, with three interesting results coming out at about the same time. That is happening because we are collaborating with much smarter domain experts, but also because our ability to build software, understand algorithms, and leverage cloud computing means we can

create systems that can do things that weren’t possible before.

What are some of the challenges you have faced in your interdisciplinary work?

We’ve learned the hard way how to write a paper for a biology journal. They will not tell you how; they will just reject your paper. The first paper we submitted was not only rejected—it was rejected and we were told never to resubmit it again.


One of our Microsoft colleagues who had successfully made the transition said, “You don’t understand—you have to write backwards.” Step one, you write your 12-page computer science paper, which covers all topics in great depth; that’s your appendix. Then you write the part that goes into the journal; it’s a six-page section that you can think of as an extended abstract with highlights and important results. But the meat is in the appendix.

What are your results?

First is the Scalable Nucleotide Alignment Program, or SNAP. The cost of genetic sequencing has gone down faster than Moore’s Law over the past decade, but it is still around \$1,000. The first step is among the most time-consuming: you break cells into many copies of small pieces in the wet lab, and then you have to assemble them back together to figure out the DNA. It’s like a jigsaw puzzle, and if you could align the pieces together faster, you could cut the data processing cost. SNAP is by far the fastest sequence aligner, and is as accurate as others.

Our second result (ADAM) stores genetic information in a way that allows it to be processed in parallel 50 times faster using the cloud.

The third is a benchmarking project called SMaSH. Fields of computer science accelerate when they agree on benchmarks, because you can’t measure progress without them. Genetics doesn’t really have benchmarks like ours, so we are proposing one.

The bottom line is that it’s both inspiring and exciting that people like us can help fight the war on cancer. 

Leah Hoffmann is a technology writer based in Piermont, NY.

© 2014 ACM 0001-0782/14/03 \$15.00

Q&A

RISC and Reward

Having helped develop Reduced Instruction Set Computing and Redundant Arrays of Inexpensive Disks, David Patterson has set his sights on interdisciplinary research.

THOUGH HIS ENTRY into computer science was somewhat accidental—enrolling in a computing course by chance after a college math class was canceled—the University of California at Berkeley’s David Patterson has left a deep mark on the field. The Reduced Instruction Set Computer, or RISC, project that he led at Berkeley inspired the Oracle SPARC architecture, as well as the ARM architecture (the “R” in each stands for RISC) that powers most mobile phones. RAID—redundant arrays of inexpensive disks—offered a powerful new way to prevent data loss. More recently, Patterson has turned his attention to interdisciplinary research, collaborating with bioinformaticists and clinicians to better understand cancer genomics.

Tell me about your childhood and what drew you to the field.

I was the first of my family to graduate from college, and I was a math major because I did well in math in high school. In my junior year at UCLA, a math class was canceled, so I took a computing course as a lark. It was love at first sight. There was no computer science major at the time, so I just informally took all the computing courses that I could.

At the time, I was working at my dad’s company in downtown Los Angeles—an industrial job—and I casually mentioned to the instructor of one CS course that I would rather be writing software than working in a factory. On his own, he talked around and found me a job on a research proj-



For more of David Patterson’s thoughts on research challenges, see “How To Build a Bad Research Center,” on page 33.

ect. Once I was in that environment, it seemed cost-effective to get a master’s, since it was just another year. And then they just assumed I was going to get a Ph.D., so I did. Had that instructor, Jean-Loup Baer, not found me a job as an undergrad, I would not be on this path.

That led to a job at Berkeley, where you have been ever since. How did you get involved with RISC, which is likely the first very-large-scale integration (VLSI) Reduced Instruction Set microprocessor?

Before we started, I took a leave of absence at DEC, and it gave me a chance to reflect. DEC had a very

successful minicomputer called the VAX—this was 1979—and at the time, people were designing microprocessors to imitate minicomputers. But VAX had an extremely complicated instruction set, and I thought it would be a terrible idea to put that into silicon, in part because there would be so many mistakes. When I got back to Berkeley, I wrote a paper arguing that if you do that, you need to make a chip that was easy to modify. The paper was rejected, because it does not make sense to have a microprocessor that you need to modify. Those two mutually exclusive facts led to my involvement in the RISC effort. [CONTINUED ON P. 111]

Computing Reviews



BEST OF
2013

BEST REVIEWS
NOTABLE
BOOKS & ARTICLES

April 2014 Online & Print



Association for
Computing Machinery

ThinkLoud

www.computingreviews.com

Take a look at the **Internet's future.**

Find **the leaders** in data communication
and networking from industry and academia,
all in one place, once a year.



The **Annual Festival** of the **ACM** Special
Interest Group on **Data Communication**

Chicago August 17 - 22, 2014