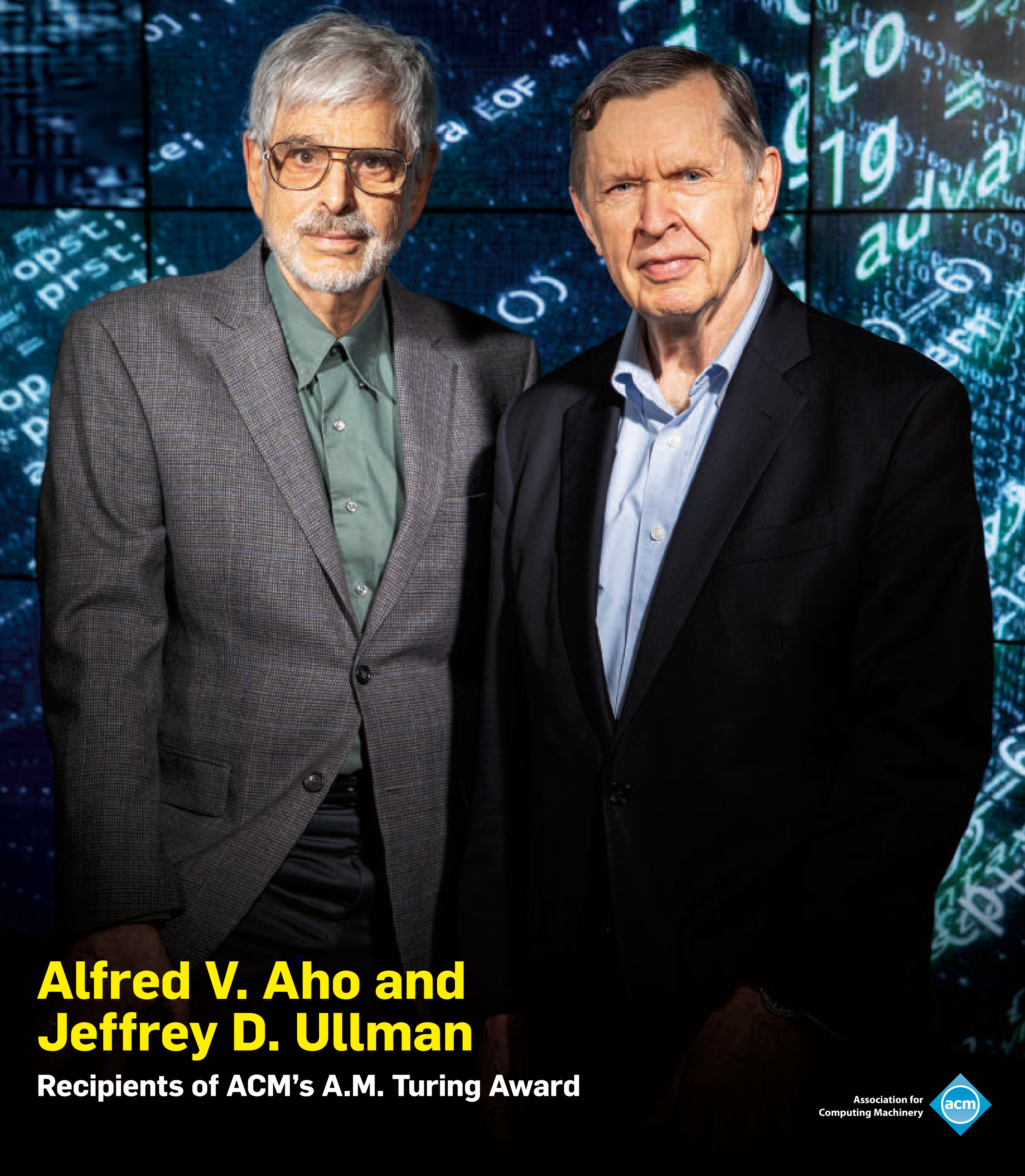
COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

06/2021 VOL.64 NO.06

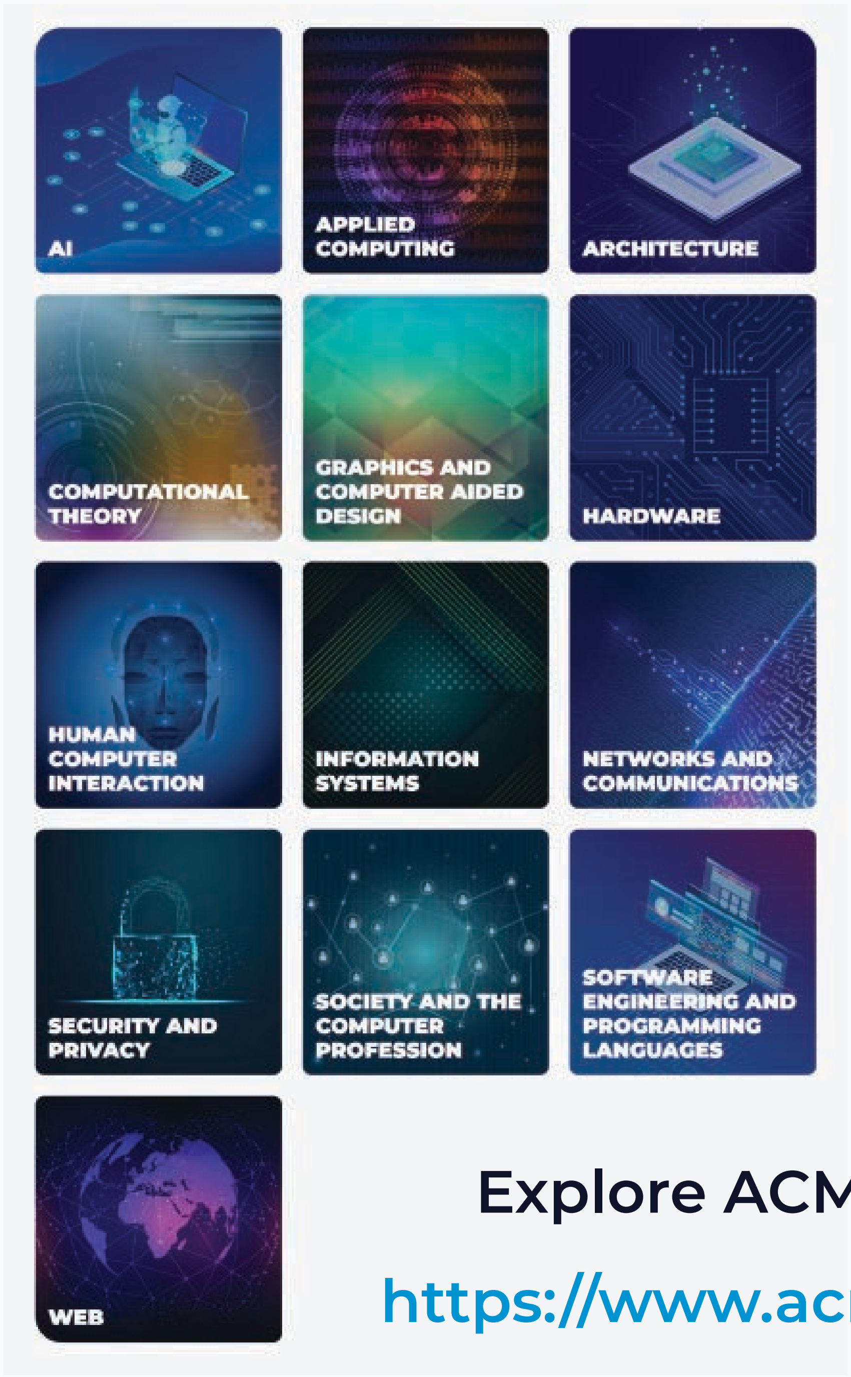


Alfred V. Aho and Jeffrey D. Ullman

Recipients of ACM's A.M. Turing Award

Association for Computing Machinery

Introducing ACM Focus


AI
APPLIED COMPUTING
ARCHITECTURE
COMPUTATIONAL THEORY
GRAPHICS AND COMPUTER AIDED DESIGN
HARDWARE
HUMAN COMPUTER INTERACTION
INFORMATION SYSTEMS
NETWORKS AND COMMUNICATIONS
SECURITY AND PRIVACY
SOCIETY AND THE COMPUTER PROFESSION
SOFTWARE ENGINEERING AND PROGRAMMING LANGUAGES
WEB

A New Way to Experience the Breadth and Variety of ACM Content

ACM Focus consists of a set of AI-curated custom feeds by subject, each serving up a tailored set of the latest relevant ACM content from papers to blog posts to proceedings to tweets to videos and more. The feeds are built in an automated fashion and are refined as you interact with them.

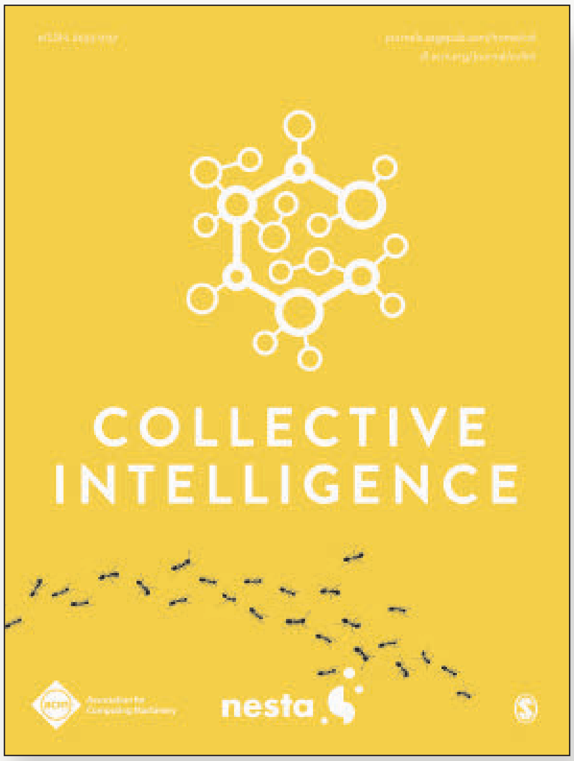
Explore ACM Focus today!

https://www.acm.org/acm-focus

acm Association for Computing Machinery

SCITRUS

A New Journal from ACM Co-published with SAGE



Collective Intelligence, co-published by ACM and SAGE, with the collaboration of Nesta, is a global, peer-reviewed, open access journal devoted to advancing the theoretical and empirical understanding of collective performance in diverse systems, such as:

- human organizations
- hybrid AI-human teams
- computer networks
- adaptive matter
- cellular systems
- neural circuits
- animal societies
- nanobot swarms

The journal embraces a policy of creative rigor and encourages a broad-minded approach to collective performance. It welcomes perspectives that emphasize traditional views of intelligence as well as optimality, satisficing, robustness, adaptability, and wisdom.

Accepted articles will be available for free online under a Creative Commons license. Thanks to a generous sponsorship from Nesta, Article Processing Charges will be waived in the first year of publication.

For more information and to submit your work, please visit https://colint.acm.org

Association for Computing Machinery

SAGE Publishing

nesta

COMMUNICATIONS OF THE ACM

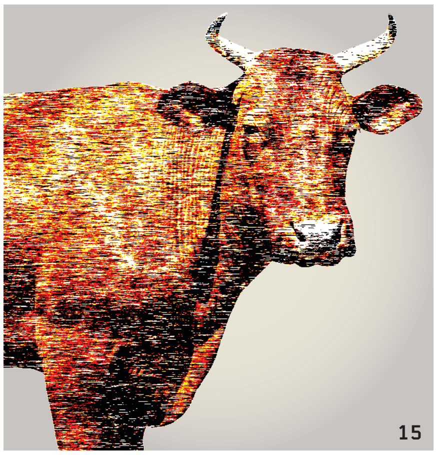
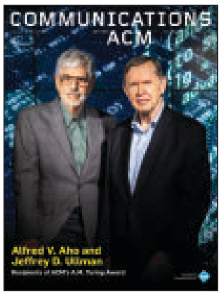


IMAGE BY ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

Association for Computing Machinery
Advancing Computing as a Science & Profession

06/2021
VOL. 64 NO. 06

About the Cover:
Jeffrey D. Ullman (left) and Alfred V. Aho, recipients of the 2020 ACM A.M. Turing Award, photographed in the Anomaly Room at Nokia Bell Labs in Murray Hill, NJ, on April 22, 2021. The location is where their professional collaborations began in 1967; the photos within this issue find the pair revisiting former office spaces and the renowned Unix Room at what was then Bell Labs. Cover photo by Alexander Berg.



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.

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.

CONTACT POINTS
Copyright permission
permissions@hq.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhelp@acm.org
Letters to the Editor
letters@cacm.acm.org

WEBSITE
http://cacm.acm.org

AUTHOR GUIDELINES
http://cacm.acm.org/about-communications/author-center

ACM ADVERTISING DEPARTMENT
1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 626-0686
F (212) 869-0481

Advertising Sales Account Manager
Ilia Rodriguez
ilia.rodriguez@hq.acm.org

Media Kit acmmediasales@acm.org

Association for Computing Machinery (ACM)
1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 869-7440; F (212) 869-0481

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 $269.

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 acmhelp@acm.org.

COMMUNICATIONS OF THE ACM
(ISSN 0001-0782) is published monthly by ACM Media, 1601 Broadway, 10th Floor New York, NY 10019-7434 USA. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER
Please send address changes to *Communications of the ACM*
1601 Broadway, 10th Floor
New York, NY 10019-7434 USA

Printed in the USA.

Association for Computing Machinery

PLEASE RECYCLE THIS MAGAZINE

DOI:10.1145/3462459

Andrew A. Chien

Time for *Two* Annual Turing Awards

It is the season when we recognize an extraordinary research contribution to computing with the ACM A.M. Turing Award. I propose a change.

ACM should bestow two Turing Awards each year, starting immediately. I do not mean to two people; a number of Turing Awards have been shared. But rather, my position is that the ACM should award several Turing Awards every year. These awards could be staggered by six months, perhaps in spring and fall, to ensure each winner receives the focused attention of the computing community and the world they richly deserve. This attention properly focuses on their research accomplishments and impact on the world.

Why give two Turing Awards?

Since the Turing Award was first given in 1966, the scope and impact of computing has grown immeasurably. At that time, more than 50 years ago, there were dozens of computers sold each year, with a few hundred computers in the world. By 2000, as the Internet transformed the world, computer sales had increased 10 million-fold, with PCs exceeding 120 million per year, with half a billion computers in the world. With the advent of smartphones, annual sales have grown by another 10-fold, exceeding 1.5 billion each year, with over five million "apps." Perhaps invention has not grown linearly with the number of computers, but as scholars of innovation would expect at least $n^{0.5}$ rates (10,000-fold) or even at $\log_2 n$ rates (25-fold). These represent phenomenal growth in innovation in computing.

But there are more measures of growth. The number of programmers worldwide (25 million software developers) and computer science researchers (over 24,000 published authors in the ACM Digital Library) are extraordinary, driving an incredible rate of invention and innovation. And in recent decades, the computing community has grown dramatically in China, India, and throughout Asia, South America, Arabia, and Africa. Among that dramatic breadth of activity, how can it be that only one is worthy of a Turing Award each year?

We know that it cannot be. It's the right thing for ACM to broaden its recognition by granting two Turing Awards each year. But not only is it the right thing to recognize extraordinary leaders, but it's good to promote the field of computing. A Turing Award is a recognition, empowering more computing champions and leaders!

It's the right thing for ACM to broaden its recognition by granting two Turing Awards each year.

Why now? Well to speak frankly, it's already long overdue. We should have expanded to multiple Turing Awards in the 1980s when computers exploded into the lives of ordinary people with the spread of the PC, and leading to an explosion of business, entertainment and personal applications. We should have expanded to multiple Turing Awards in the late 1990s when the explosive growth of the Internet transformed information access, music sharing, and connected the world at a speed and breadth unimaginable in human history. And yes, we should have expanded to multiple Turing Awards in the 2010s as smartphones transformed the personal and business lives of billions around the world.

Let's do it now! Starting now, the ACM must increase the recognition of extraordinary contributions to computing being made by leading researchers today by increasing the number of Turing Awards granted each year to two!

Please lend your voice by writing to me (eic@cacm.acm.org) or commenting on the Web.

Andrew A. Chien, EDITOR-IN-CHIEF

Andrew A. Chien is the William Eckhardt Distinguished Service Professor in the Department of Computer Science at the University of Chicago, Director of the CERES Center for Unstoppable Computing, and a Senior Scientist at Argonne National Laboratory.

Congratulations!

ALFRED V. AHO

LAWRENCE GUSSMAN PROFESSOR EMERITUS OF COMPUTER SCIENCE

2020 ACM A.M. TURING AWARD

"I am honored and humbled to receive this prestigious award. And I am delighted that with this award, the ACM recognizes the foundational importance of abstractions and algorithms in the design and implementation of programming languages." — *Alfred V. Aho*

CS@CU COMPUTER SCIENCE

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

DOI:10.1145/3462461

Vinton G. Cerf

It Came From Outer Space!

I write these words on Earth Day, April 22, 2021. Today, many words will be spoken and written, raising legitimate alarms of the risk we face with increasing temperature caused by greenhouse gases. Among others, former Vice President Al Gore raised this as a primary concern in his books, *Earth in the Balance* and *An Inconvenient Truth*. Bill Gates has just published his perspective in his new book, *How to Avoid a Climate Disaster*. That we are seeing significant climate change seems to me incontrovertible. The correlation of so many thermal observations with increased greenhouse gas in the atmosphere and with the historical record of temperature strongly implicate carbon dioxide and methane as primary causes.

Efforts to move away from reliance on greenhouse gas production for electrical power, heating, cooking, transportation, manufacturing, and a host of other activities of modern society strike me as inescapably important. The sluggish response to this threat is partly a consequence of its long-term character. The most serious problems still lie in the future even though we can document early evidence of their presence including shrinking polar ice and glaciers, rising average and peak temperatures, increasing desertification, acidification of the oceans, and increase in absolute ppm of carbon dioxide.

But there is another, less predictable but potentially equally devastating threat to our modern dependence on electricity. It goes by the name of *coronal mass ejection* (*CME*) and occurs sporadically as the sun burns its hydrogen into helium and heavier elements. A CME consists mostly of masses of ionized protons in a plasma that are ejected at high speed from the sun's corona and propagated outward by the solar wind. These ionized ejections create power magnetic fields, which themselves can induce large currents in electrical conductors. One of the best documented CMEs occurred on Sept. 1, 1859 and is known as the Carrington Event after the English astronomer, R.C. Carrington who, along with R. Hodgson, observed and reported the event. Telegraph equipment and wires were overheated and damaged over a wide area. Not all CMEs threaten the Earth; they must intercept Earth in its orbit to do damage.

There have been subsequent events in 1921 and 1989, but of lower intensity and causing less damage than the 1859 event. We are, however, far more deeply dependent on electricity and its transport than ever in human history. In the parts of the planet that are most heavily electrified, we find increased dependence on electrical equipment, especially including computing and communications. Along with an enormous number of devices dependent on electricity, our communication systems, including satellites, ground radio transceivers, and optical fiber systems are at risk. Subsea optical fibers are especially vulnerable. The CME does not affect the optical fiber but could severely damage the electrically powered repeaters needed to reinforce optical signals for long haul cables.

The resilience of the Internet with alternative paths … might play a critical part in overcoming the side effects of a major CME event.

Because the ionized particles travel more slowly than photons, there could be on the order of 13 hours or more warning from a major flare and CME but the key question is what should have been considered well ahead of the warning to protect electrical and electronic equipment from damage as well as the electrical power grid itself. The concept of a microgrid that supplies power to smaller communities or even portions of large cities and can be disarticulated at need might be one place to start. The resilience of the Internet with alternative paths, including satellite and undersea and land cable as well as high speed laser links, might play a critical part of overcoming the side effects of a major CME event.

It seems to me that now would be a good time to work through the many potential hazards of a CME and to draw conclusions for grid and net design for connectivity and resilience in the face of certain failures for power and for Internet continuity.

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

ACM ON A MISSION TO SOLVE TOMORROW.

Dear Colleague,

Without computing professionals like you, the world might not know the modern operating system, digital cryptography, or smartphone technology to name an obvious few.

For over 70 years, ACM has helped computing professionals be their most creative, connect to peers, and see what's next, and inspired them to advance the profession and make a positive impact.

We believe in constantly redefining what computing can and should do.

ACM offers the resources, access and tools to invent the future. No one has a larger global network of professional peers. No one has more exclusive content. No one presents more forward-looking events. Or confers more prestigious awards. Or provides a more comprehensive learning center.

Here are just some of the ways ACM Membership will support your professional growth and keep you informed of emerging trends and technologies:

- Subscription to ACM's flagship publication ***Communications of the ACM***
- Online books, courses, and videos through the **ACM Learning Center**
- Discounts on registration fees to ACM Special Interest Group conferences
- Subscription savings on specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

Joining ACM means you dare to be the best computing professional you can be. It means you believe in advancing the computing profession as a force for good. And it means joining your peers in your commitment to solving tomorrow's challenges.

Sincerely,

[illegible]

Gabriele Kotsis
President
Association for Computing Machinery

Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING.
JOIN ACM TODAY.

www.acm.org/join/CAPP

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

❑ Professional Membership: $99 USD

❑ Professional Membership plus ACM Digital Library: $198 USD ($99 dues + $99 DL)

ACM STUDENT MEMBERSHIP:

❑ Student Membership: $19 USD

❑ Student Membership plus ACM Digital Library: $42 USD

❑ Student Membership plus Print *CACM* Magazine: $42 USD

❑ Student Membership with ACM 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 computing. Membership in ACM-W is open to all ACM members and is free of charge.

PAYMENT INFORMATION

Name

Mailing Address

City/State/Province

ZIP/Postal Code/Country

❑ Please do not release my postal address to third parties

Email Address

❑ Yes, please send me ACM Announcements via email

❑ No, please do not send me ACM Announcements via email

❑ AMEX ❑ VISA/MasterCard ❑ Check/money order

Credit Card #

Exp. Date

Signature

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

By joining ACM, I agree to abide by ACM's Code of Ethics (www.acm.org/code-of-ethics) and ACM's Policy Against Harassment (www.acm.org/about-acm/policy-against-harassment).

I acknowledge ACM's Policy Against Harassment and agree that behavior such as the following will constitute grounds for actions against me:

- Abusive action directed at an individual, such as threats, intimidation, or bullying
- Racism, homophobia, or other behavior that discriminates against a group or class of people
- Sexual harassment of any kind, such as unwelcome sexual advances or words/actions of a sexual nature

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.

acm Association for Computing Machinery

ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)
Hours: 8:30AM - 4:30PM (US EST)

Fax: 212-944-1318
acmhelp@acm.org
acm.org/join/CAPP

BLOG@CACM

The *Communications* website, 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.

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

DOI:10.1145/3460212 http://cacm.acm.org/blogs/blog-cacm

The Search for Unlimited Productivity

Doug Meil considers the continuing search for the next great software development methodology.

Doug Meil
Anna Karenina On Development Methodologies
https://bit.ly/3vRxVPB
March 23, 2021

I am old enough to have lived through multiple technical cycles. When I started professionally, CASE tools were the rage, and OS/2 was still a thing. I even worked on a mainframe for a bit. Then came client-server development (as popularized by Windows applications, as strictly speaking the 'client-server' pattern was not new). Then Java, the first-generation Web applications with application servers and a myriad of Model View Controller frameworks. Then second-generation Web development with more responsive JavaScript-based Web applications that replicated many of the things that people liked about thick-client applications, and then some. Then big data broke on the scene, and then cloud architectures.

Concurrent to these technical advancements, new development methodologies kept appearing, such as Lean, Safe, Scrum, Agile, Extreme Programming, RAD, JAD, Spiral, and so forth, to better organize software efforts. As much as each new methodology heaped scorn upon each other, they all reserved the most contempt for Waterfall, the original methodology.

Planning, Analysis, Design, Construction, and Maintenance were for Luddites, so the chorus went. Waterfall was the Hive 0.10 of its day—everybody loved hating it, everybody loved measuring itself against it. I not only remember Waterfall, but I remember E&Y's Navigator Methodology, which was a particular expression of Waterfall which contained even more liquid and fell from even greater heights. Navigator was delivered with yards of binders. Even as a career newbie, I thought that was a bit heavyweight.

Yet the fundamental weakness was that each new methodology tried to claim that it was possible to extract 10 pounds of output from a five-pound bag of requirements, assuming that anyone had remembered to fill the bag in the first place, and that the contents hadn't shifted or expired since packing. And when the methodology *du jour* eventually lost its luster, it invariably triggered a search for the next one. The secret to unlimited productivity had to be somewhere, right?

Happy Development Teams and Practices

Happy families are all alike, every unhappy family is unhappy each in its own way. —Leo Tolstoy (1828–1910)

Even those not intimately familiar with Russian literature have probably heard that opening line from *Anna Karenina*. One can replace "families" with "development teams" and the quote still applies. For example, imagine a development team with the following attributes:

- Stakeholders identified
- Priorities understood and documented
- Effective development team
- Effective development tooling
- Effective end-to-end testing patterns
- High code quality
- Effective code deployment patterns
- High deployment velocity
- Effective operational tooling
- Happy users

It's a good list, right? So, do it. Seriously, just **do that**.

The hardest part of development is prioritization, and that is because humans are irrational and unpredictable. No development methodology will solve that. It is even more complex with multiple stakeholders, and especially multiple *groups* of stakeholders. The only way through is documenting needs with a forced-ranked priority and continually reminding people about what they are and what the corresponding statuses are. Tools are helpful, but it is more about the communication and understanding around the tooling. The fanciest tool doesn't help if nobody believes what it's telling them.

Someone inevitably will cry, "*but we need to be flexible!*" And that's fine. It's **okay** to change priorities from time to time. But the relative priorities have to change too, and likewise the previous expectations that have been set. **That** is the hard part. No fair getting the previous expectations *and* the revised expectations; that's crazy talk. And again, that's a human problem. Don't forget that pulling the emergency brake on a software-release train arbitrarily in the middle of a trip can land a development effort in a frozen, desolate location. Metaphorically speaking, of course.

Next, design software for the immediate prioritized use-cases for the targeted timelines. There are a lot of important qualifiers in the prior sentence, such as "immediate," "prioritized," and "targeted timelines." Software designs always have context because software engineering is the art of **trade-offs**. There are *always* trade-offs. A tank isn't a sedan which isn't a racecar which isn't a truck, but they are all valid and effective forms of vehicles. We know from the classics such as Design Patterns (Gamma, et al., https://amzn.to/39PqP4H) and Refactoring (Fowler, https://amzn.to/3dxmRi4) that clean software design, configurability, and extensibility are important, but we also frequently forget that utilizing every single design pattern is not the point. *Guessing* on use-cases—both functional and technical—almost always makes the outcome worse. Remember Knuth's famous quote: "*premature optimization is the root of all evil.*" He figured that out a long time ago, and the software world has only gotten more complicated since then. Temptations and distractions are everywhere.

Pentagon Wars (https://bit.ly/3sVb4AP) is an underrated 1998 dramedy which depicted a fictionalized and satirized development of the Bradley Fighting Vehicle that was trying to be everything to everybody. This is loosely based on the very real and serious book *Pentagon Wars* by James Burton (https://amzn.to/3dAaIsS) about weapons design and procurement. It's completely reasonable and expected to make design trade-offs *within* what I would call "targeted use-cases." Just be aware of when the trade-offs become so big, they compromise the integrity and efficacy of said case. Engineering is both science and art, and no development methodology can solve that.

I was once at a company where somebody was a huge proponent of the Inversion Of Control (https://bit.ly/3uqMbxd) pattern. This person was so in love with the pattern that the resulting code was difficult to understand and impossible to test. In fact, the unit tests were all configured to mock implementations to the point where nothing but fake code was being tested. This seems too ridiculous to be true, but it actually happened, and once the actual code was deployed to real customers, it was a disaster. I got pulled in to clean this up because each release required scads of emergency hotfixes. Don't become so enamored with a technical pattern or framework that it obscures one's vision from the real problems that were supposed to be addressed in the first place.

Then automate, automate, automate, everything from the lowly build process, to unit tests, to environment setup, to deployment, to all forms of monitoring and metric collection. This is as obvious as the "*the code should always work and be in a deployable state*" mantra. Obvious, but not necessarily easy because it requires discipline to dedicate resources to automation—again, a human factor. The good news is that there have been many improvements and advances in frameworks and techniques in this area in the past few decades that can help. Pick up the Google SRE books (https://sre.google/books/) for starters.

Figure out the most frequent release cycle that works with the user-base. This can vary by industry, and often technical aspects are not the gating factor. Be able to release as quickly and easily as possible, with high quality.

Lastly, iterate. **Keep going.** Delivery velocity is life. The worst thing that can happen to any software product is a feast and famine cycle, and no development methodology can solve that. Maintaining momentum is arguably the hardest aspect, as continuous forward progress is utterly dependent on human factors such as portfolio priorities, resources, budgets, and other things such as how well your software is doing in the market. The difficulties of managing humans and money are as old as time. But it is not hopeless, just hard. So get on it.

I heard this at Strata Data Conference years ago: "*if your software is successful, you are never done*." Sage advice. Similarly, one can practically *ensure* a software solution **not** being successful with erratic delivery.

I've seen too many people effectively throw themselves in front of metaphorical trains for the love of a particular development methodology at the cost of overall project success. But a development methodology is just a "how"—it's not the *goal*. While no software success can ever be guaranteed, the best chances for development team happiness are to focus on common human factors such as stakeholder alignment, priorities, communication, and the discipline to keep driving forward. And the humility to accept that mistakes will inevitably be made and addressed in a future release, as soon as humanly possible.

As Tolstoy might say, *Ни пуха, Ни пера* (https://bit.ly/2Q5iXoz) comrade.

Postscript

Apache Hive was actually pretty good at what it did. SQL is just so darn useful, and to be able to use it over extremely large datasets was a breakthrough of its time. Hive also kept improving, but as I said earlier, most folks unfortunately only seem to want to talk about Hive 0.10, which was the version that most people were using when Hadoop went mainstream.

Doug Meil is a software architect at Ontada, https://www.ontada.com/. He also founded the Cleveland Big Data Meetup (https://bit.ly/2OoMAAQ) in 2010.

Profile | DOI:10.1145/3460734

Neil Savage

Getting Down to Basics

2020 ACM A.M. Turing Award recipients Alfred Aho and Jeffrey Ullman helped develop formal language theory, invented efficient algorithms to drive the tasks of a compiler, and put them all together in 'The Dragon Book.'

WRITING THE CODE to make a computer perform a particular job could be a Herculean task, back in the 1950s and 60s.

"In the early 1950s, people did numerical computation by writing assembly language programs," says Alfred V. Aho, professor emeritus of computer science at Columbia University. "Assembly language is a language very close to the operations of a computer, and it's a deadly way to program. It's slow, tedious, and expensive."

Of course, people can program at higher levels of abstraction, but that requires translating the higher-level language into a more basic set of instructions the machine can understand. Compilers that efficiently perform that translation exist nowadays in large part due to the work of Aho and Jeffrey D. Ullman, professor emeritus of computer science at Stanford University. Their contribution to both the theory and practice of computer languages has earned them the 2020 ACM A.M. Turing Award.

"Compilers are responsible for generating the software that the world uses today, these trillion lines of software," Aho says. "They've been around for a long time, and the modern world wouldn't exist without programming language translators."

The two men helped develop formal language theory. They invented efficient algorithms for performing lexical analysis, syntax analysis, code generation, and code optimization, all essential tasks for a compiler. And they put that all together into *Compilers: Principles, Techniques, and Tools*, the definitive textbook on the subject, now known simply as "The Dragon Book" for its colorful cover illustration.

"No matter what you do in life, you need to understand how to use computers, which usually means being able to write at least simple code in some appropriate language."

The two met in 1963 while waiting in line to register for the Ph.D. program at Princeton University. Ullman, now 78, received his B.S. from Columbia that year, and Aho, 79, had just graduated from the University of Toronto with a B.S. in engineering physics. After each received his doctorate from Princeton a few months apart, in 1966 and 1967, they both joined Bell Laboratories, a hotbed for science and innovation.

"Academia has sort of always been my game plan," Ullman says, adding that he didn't feel assistant professors were particularly well treated at the time. "Today, assistant professors are really treated well in academia, especially in the field of computer science, where we recognize it's often the young people who have the greatest ideas," he says. When he was offered a job at Columbia he jumped at the chance, but kept traveling to Bell Labs about once a week to continue collaborating with Aho, until he moved to Stanford University in 1979. Aho, meanwhile, stayed at Bell for nearly 30 years, until he joined the Columbia faculty in 1995.

The researchers laid out a generic framework that provided the roadmap for how a compiler works. First comes

PHOTO BY ALEXANDER BERG

Alfred Aho and Jeffrey Ullman
in the corridor of their
former office spaces at Bell Labs.

lexical analysis, in which the compiler scans the program to identify tokens, which are analogous to the individual words in a sentence. Next comes a parser, which analyzes the syntax of the program, and checks whether the tokens fit together. "Not every sequence of tokens is a well-constructed program," Aho says. Next comes semantic analysis, which checks whether the sequences make sense as a whole.

The two shrunk the size and increased the speed of a left-right parser, which aims to read the program in sequence from left to right without doubling back, but which is allowed to peek ahead to determine meaning. For instance, if the program wants to perform the operation A+BxC, the parser can see the A+ and know it hasn't reached the end of that particular element, so it keeps going. It eventually sees that it has to multiply B and C first, then add A, just because the creators of the compiler defined that order of precedent for arithmetic operations.

The trio of lexical, syntax, and semantic analysis makes up the front end of the compiler, which produces an intermediate representation of the program, which can then be optimized. If it turns out, for instance, that C equals 1, so that BxC=B, the program can be simplified by changing it to merely A+B. "That is a kind of optimization that can be performed to make the ultimate target program run faster," Aho says. The last step is to map the optimized intermediate to an assembly language, which can translate it directly into machine language for the particular type of computer.

Enter the Dragon

Aho and Ullman spelled all this out in the Dragon Book, which has since had two more editions and added Ravi Sethi, and later Monica Lam, as coauthors. It was Ullman's idea to depict a knight battling a dragon on the cover. The dragon represents "complexity of compiler design," Ullman explains, and "many of the weapons of the knight are the tools that we advocated people use," such as data flow analysis and syntax-directed translation. The first version showed a green dragon, the second was red, and the third, drawn by Ullman's son Scott, is purple.

The dragon was red when Krysta Svore, now head of the quantum computing group at Microsoft Research, was a student in Aho's class at Columbia. Even as advanced an application as a quantum program relies on the same ideas the two developed in the 1960s and 1970s. "What they're winning for is part of the foundation of what I and my team are working on for this next type of computer," she says. "What they developed is still critical for writing programs across smartphones, across quantum computers, across your laptop. These technologies are very much forever critical."

"What they developed is still critical for writing programs across smartphones, across quantum computers, across your laptop. These technologies are very much forever critical."

In Aho's class, he asked students to write their own domain-specific language. Svore wrote a quantum computer language, and enjoyed the experience so much that she asked Aho to become one of her thesis advisers.

The men streamlined the process of creating compilers even further by developing tools such as the Lex lexical analyzer generator, and the yacc parser generator, both of which automate the building of compiler components. Aho, along with Peter Weinberger and Brian Kernighan, also developed AWK, a "little language" to simplify some common data processing tasks. "To do these operations in C would require you to write 100 or 200 lines of C," Aho says. "We could specify these routine data processing tasks with one or two lines of AWK."

Aho and Ullman each have written several other books, both together and with other authors, including *The Design and Analysis of Computing Algorithms* with previous Turing Award recipient John Hopcroft, which laid down a framework for analyzing and teaching algorithms. They've received other awards for their work, including IEEE's John von Neumann Medal in 2010 for outstanding contributions to computer science.

The Turing Award comes with a $1-million cash prize that Aho and Ullman will split. Neither is sure yet what he'll do with the money. After a year of laying low due to the coronavirus pandemic, Ullman says, "I wouldn't mind going out to a restaurant and having a nice meal."

Broader Applications

The men say young people studying computer science, and even those who are not, could benefit from considering how thinking algorithmically fits into the wider world. "The future of computer science lies in applying the ideas more broadly," Aho says. "They should study not just computer science, but another discipline like biology, because there are fascinating opportunities for the application of computer science ideas to biological problems like how does the body work, how does the disease work?" The most intriguing question computer scientists might address, he suggests, is "how does the brain work?"

Ullman says everybody should be able to think about the world computationally, no matter what they do, just as people who are not professional writers ought to know how to write. Similarly, he says, everyone should be able to describe things precisely, even at such a high level of abstraction that it may not look like programming. "You've got to say what you mean and mean what you say," he says. "No matter what you do in life, you need to understand how to use computers, which usually means being able to write at least simple code in some appropriate language." C

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

Watch the recipients discuss their work in the exclusive *Communications* video. https://cacm.acm.org/videos/2020-acm-turing-award

Technology | DOI:10.1145/3460218

Don Monroe

Deceiving AI

The uncanny power of machine learning can be turned against it.

OVER THE LAST decade, deep learning systems have shown an astonishing ability to classify images, translate languages, and perform other tasks that once seemed uniquely human. However, these systems work opaquely and sometimes make elementary mistakes, and this fragility could be intentionally exploited to threaten security or safety.

In 2018, for example, a group of undergraduates at the Massachusetts Institute of Technology (MIT) three-dimensionally (3D) printed a toy turtle that Google's Cloud Vision system consistently classified as a rifle, even when viewed from various directions. Other researchers have tweaked an ordinary-sounding speech segment to direct a smart speaker to a malicious website. These misclassifications sound amusing, but they could also represent a serious vulnerability as machine learning is widely deployed in medical, legal, and financial systems.

The potential vulnerabilities extend to military systems, said Hava Siegelman of the University of Massachusetts, Amherst. Siegelman initiated a program called Guaranteed AI Robustness against Deception (GARD) while she was on assignment to the U.S. Defense Advanced Research Projects Agency (DARPA). To illustrate the issue to colleagues there, she said, "I showed them an example that I did, and they all started screaming that the room was not secure enough." The examples she shares publicly are worrisome enough, though, such as a tank adorned with tiny pictures of cows that cause an artificial intelligence (AI)-based vision system to perceive it to be as a herd of cows because, she said, AI "works on the surfaces."

The current program manager for GARD at DARPA, Bruce Draper of Colorado State University, is more sanguine. "We have not yet gotten to that point where there's something out there that has happened that has given me nightmares," he said, adding, "We're trying to head that off."

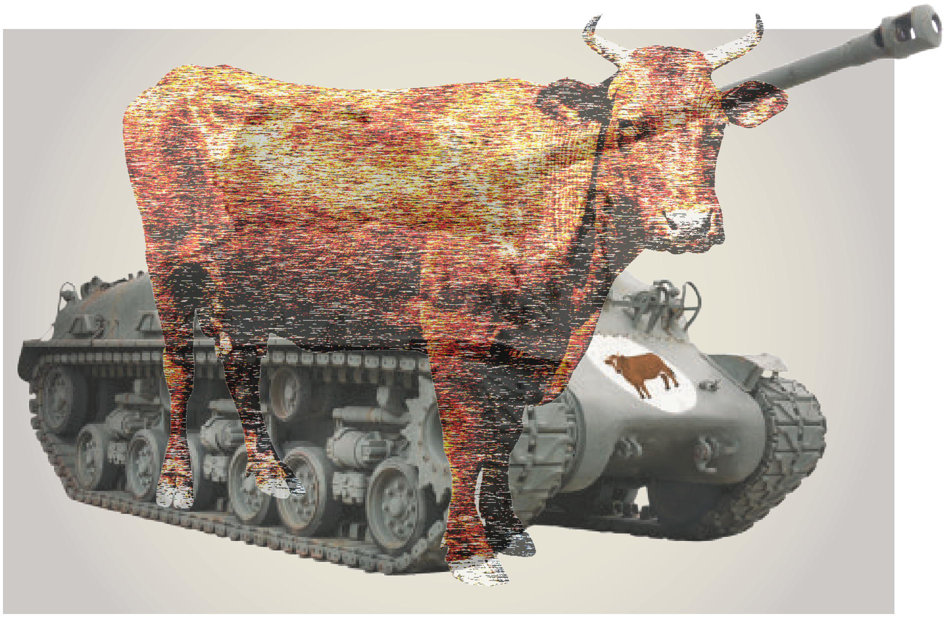


IMAGE BY ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

Researchers, some with funding from DARPA, are actively exploring ways to make machine learning more robust against adversarial attacks, and to understand the principles and limitations of these approaches. In the real world, these techniques are likely to be one piece of an ongoing, multilayered security strategy that will slow attackers but not stop them entirely. "It's an AI problem, but it's also a security problem," Draper said.

Worth a Thousand Words

All machine learning tools can be deceived, but image classification is the most intuitively compelling. Altering a small number of pixels in an input image in a way that may be trivial or even unnoticeable to people, for example, can fool a classifier into declaring a stop sign to be a speed limit sign. This would clearly be a disaster for a self-driving car.

Such examples clearly reveal that deep learning systems base their decisions on features that may be completely different from what people regard as important. "Because these systems are using visual input, we tend to assume that they see the same way we do," said Draper. "That's not a good assumption." Troublingly, this vulnerability likely extends to other applications of machine learning, where the results are harder to appreciate or visualize.

Image manipulation is particularly challenging to defend against, said Battista Biggio of the University of Cagliari in Sardinia, Italy, "because the space of pixels is so large that the attacker can do basically whatever he wants in terms of manipulating the images." In contrast, he said, for malware, "you have instructions or bytes, which have a specific meaning, so they cannot be altered in a trivial matter."

In 2013, Biggio and his coworkers showed how to fool a machine learning system by exploiting knowledge of the internal "gradients" it uses for training to design adversarial examples. "The basic idea is to use the same algorithm that is used for learning to actually bypass the classifier," Biggio said. "It's fighting machine learning with machine learning."

Around the same time, Google's Christian Szegedy and collaborators used similar techniques to train a network to make erroneous identifications. They then added imperceptible patterns to an image of a school bus until the classifier abruptly—and confidently—declared it to be an ostrich.

At first, it was unclear whether these pixel-level manipulations "were real problems or just theoretical constructions that … couldn't actually cause a problem to real vision systems," said Andrew Ilyas. Back in 2018, this uncertainty motivated him and his fellow MIT undergraduates to concoct their rifle-mimicking turtle, explicitly insuring that the subtle details they decorated it with would be recognized from different viewpoints. "It worked better than even I thought," said Ilyas, now a graduate student at MIT.

Pick Your Poison

Deception "seems to be a particular issue with AI systems that use sensory input, whether it's visual or audio or whatever," said Draper. "It's still more of an open question whether it's also an issue for an AI system that's working on network security."

Whether it involves real-world sensors or later manipulation of the resulting digital data, alteration of the input to an existing classifier is called "evasion." Another type of vulnerability occurs if an attacker can insert doctored data into the training set, which is known as "poisoning."

Mislabeled training data can move the decision boundary that separates different classifications. Siegelmann cites a poisoning example where inserting images of Wonder Woman with distinctive eyeglasses, even when made almost imperceptible, can induce a system to classify anyone wearing such glasses as the superhero.

For an attacker to poison training data, they obviously must have access to that data. Similarly, an attacker can be most successful if they know the internal design details of a machine learning system, in what is called a "white-box" scenario.

"The white-box case is interesting when you want to understand the worst-case scenario," said Biggio. "We expect then that in practice these systems remain more secure also under more restrictive models of attack." Effective security protocols can obscure both the design and the data to create more challenging "black-box" scenario, although a "gray-box" scenario, with some kinds of partial information known or inferred, is more realistic.

Even if the system details are hidden, however, researchers have found that attacks that work against one system frequently work against others that have a different—possibly unknown—internal structure. This initially surprising observation reflects the power of deep learning systems to find patterns in data, Biggio said. "In many cases, different classifiers tend to learn the same correlations from the data."

This "transferability" of attacks highlights the risk of a common training set like ImageNet, which contains a huge set of annotated images that is widely used for training vision systems. Although this common training corpus makes it easy to compare the performance of different classifiers, it makes them all potentially vulnerable to the same biases, whether malicious or not.

A Security Challenge

Siegelmann argues this vulnerability is intrinsic to deep learning systems, which all contain many hidden layers but vary in how they are interconnected. "All the deep networks are the same," she said, and for decades they have always been trained by "backpropagation" into the network of the errors in their outputs, she said. "It doesn't really matter exactly how it's connected."

Although Siegelmann worries about losing the power of machine learning in security-sensitive applications, she has no such qualms about eschewing the deep learning versions of it. "The point is how you teach it," she said. For example, she advocates an active learning process in which the system chooses which questions to ask, rather than letting users pick examples that can intentionally lead it astray.

Of course, human observers can be misled by camouflage and other techniques, too. "It's not really a shock that it's also possible to fool artificial systems," Draper said. "What is just always a shock to people is that what fools them is different."

Some researchers hope errors can be avoided by designing systems that "explain" their reasoning. For deliberate attacks, however, "Explainable AI is the easiest thing to trick," Siegelmann cautioned, "because we teach the network to say what will convince us."

In his own recent work, Ilyas has been exploring "adversarial training," which involves using especially problematic training examples to guide the network to more reliable features for classification. This strategy may help avoid fragility in non-security operations as well, he said, if it "gets classifiers to use features that are closer to what humans use, so that they don't fail in other, unexpected ways."

DARPA's GARD program will test a variety of approaches in regular challenges. In one strategy, for example, spoofing of sensory input is made harder by using multiple sensors simultaneously, and checking that they are compatible with each other and with what is known about the real world.

In the end, however, robust machine learning will just be one layer in the never-ending arms race to keep computer systems secure. In this context, what is currently missing is not so much air-tight defenses, but theoretical guarantees about how long a system can hold out, similar to those available in encryption. Such guarantees can inform the design of a secure, layered defense, Biggio said. "This is very far from what we have in the AI field."

"It is very likely that there may be no silver bullet," agreed Draper. "What we want to do is at least make it very difficult for someone to defeat or spoof one of these systems." C

Further Reading

Athalye, A, Engstrom, L., Ilyas, A., and Kwok, K.
Synthesizing Robust Adversarial Examples, *Intl. Conf. on Machine Learning* https://arxiv.org/abs/1707.07397 (2018).

Biggio, B. and Roli, F.
Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning, *Pattern Recognition 84*, 317 (2018), http://bit.ly/3kC97WF.

Szegedy, C., et al.,
"Intriguing properties of neural networks," https://arxiv.org/abs/1312.6199 (2014).

Wallace, E., Zhao, T.Z., Feng, S, and Singh, S.
Customizing Triggers with Concealed Data Poisoning, https://arxiv.org/abs/2010.12563 (2020)

Don Monroe is a science and technology writer based in Boston, MA, USA.

In Memoriam | DOI:10.1145/3462465 Simson Garfinkel and Eugene H. Spafford

Jack Minker (1927–2021)

ACM FELLOW JACK MINKER passed away on April 9, 2021, at the age of 93.

Minker was a leader in the development of automating logistic reasoning, including deductive databases, logic programming, and artificial intelligence, but he is perhaps best known for his efforts to promote the social responsibility of scientists and human rights.

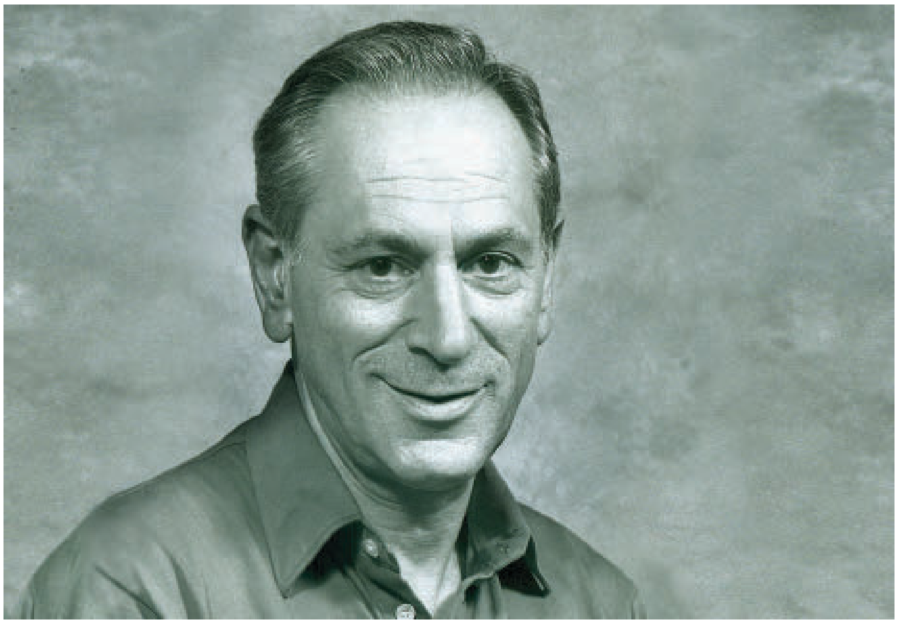


PHOTO COURTESY OF THE UNIVERSITY OF MARYLAND

In 1972, Minker was invited to join the newly constituted Committee of Concerned Scientists. He was asked to help identify Soviet computer scientists whose human rights were under attack by their government, frequently because of their career choices or because they had requested permission to emigrate from the Soviet Union. "It was something I could not refuse to do," said Jack in 2011. The following year, he became the organization's Vice-Chair, Computer Science, a position he held until his death.

Minker also served as Vice-Chair of the ACM Committee on Scientific Freedom and Human Rights from 1980 until 1989. He authored the Committee's 1981, 1982, 1984, and 1989 reports that named hundreds of people who had been unjustly imprisoned, tortured, subjected to internal exile, or denied exit visas by their governments. "Because of non-scientific considerations, some of our colleagues have been murdered, some are in jail, others are in exile in their own country, some are prevented from publishing papers in journals or attending conferences in their own country, some have been dismissed from their scientific jobs and others are trying to emigrate to countries where their human rights will not be violated," Minker wrote in a SIGMOD article about the project.

The first three reports provoked some controversy, as nearly all of those named were Jewish "refuseniks" in the Soviet Union. The 1989 report attempted to address this imbalance, and named computer professionals in Argentina, Chile, China, Czechoslovakia, Iran, Israel, Kenya, Poland, South Africa, and Turkey who had all allegedly been mistreated by their governments within the context of their work in computing. "Almost every individual on Minker's lists has had his or her human rights restored," said Ambassador Richard Schifter, then head of human rights at the U.S. State Department, at a talk given in honor of Minker's 65th birthday.

Minker graduated from Brooklyn College in 1949 with a Bachelor of Arts, received a Master of Arts degree from the University of Wisconsin in 1950, and a Ph.D. from the University of Pennsylvania in 1959. He worked in industry before joining the University of Maryland in 1967. He became a Professor of Computer Science in 1971 and was named the first chair of the CS department in 1974. He became Professor Emeritus in 1998.

Minker had over 150 refereed publications and edited or co-edited five books. He chaired the U.S. National Science Foundation's Advisory Committee on Computing from 1980 to 1982 and was a member of the U.S. National Air and Space Administration's Robotics Study Group.

In addition to being named a Fellow of the ACM, Minker was also a Fellow of the American Association for Advancement of Science (1989), a founding Fellow of the American Association for Artificial Intelligence AAAI (1990), and a Fellow of the Institute of Electronic Engineers (1991). He was the 1985 recipient of ACM's Outstanding Contribution Award and the ACM Allen Newell Award in 2005. He received the 2011 Heinz R. Pagels Human Rights Award from the NY Academy of Scientists Human Rights Committee. Following his retirement, he wrote *Scientific Freedom and Human Rights, Scientists of Conscience During the Cold War* (2011), published by IEEE Press.

Simson Garfinkel is a part-time faculty member at George Washington University in Washington, D.C., USA. He is an ACM Fellow.

Eugene H. Spafford is a professor of computer science and the founder and executive director emeritus of the Center for Education and Research in Information Assurance and Security at Purdue University, W. Lafayette, IN, USA. He is an ACM Fellow.

Science | DOI:10.1145/3460214

Marina Krakovsky

Taking the Heat

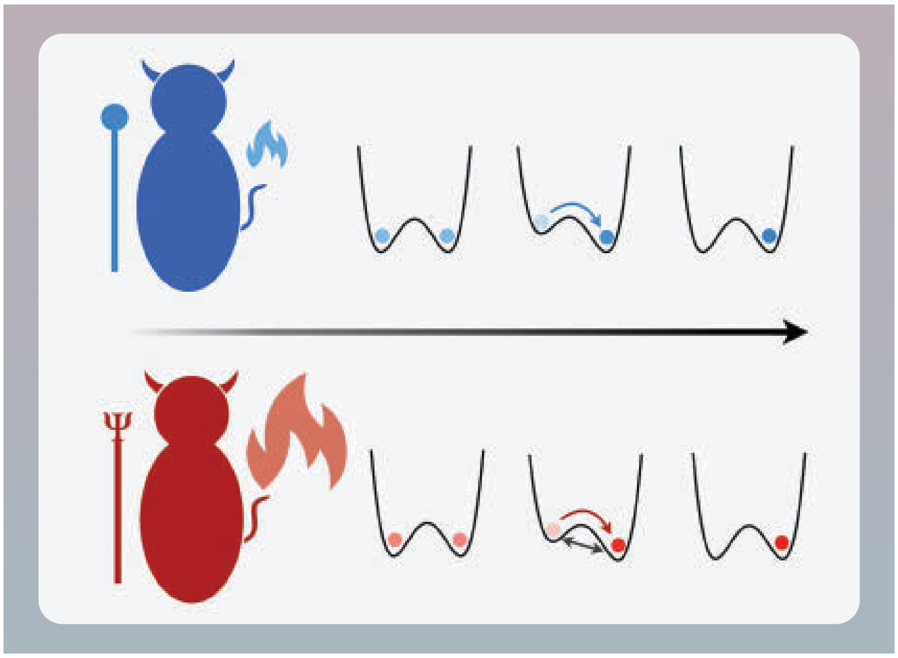
Maxwell's demon and the high cost of erasure.

QUANTUM COMPUTING, WHICH promises to harness the special properties of quantum mechanics to dramatically speed up calculations and thus help solve currently intractable problems, has attracted considerable investment from tech giants like Google, Microsoft, and IBM. Yet there is still no commercially available quantum computer because of the immense challenges in creating and running such a machine.

One of the major challenges is heat management. As with classical computing, quantum computing uses physical hardware and, therefore, is subject to the laws of physics, particularly the thermodynamics of computation. However, quantum computers are far more fragile than classical computers, requiring far lower temperatures to work properly. Also, as shown in a theoretical paper published in October 2020, the thermodynamics of quantum information processing can create highly unusual and potentially damaging effects for these delicate machines.

The paper, describing a study by researchers at the U.K.'s University of Manchester and Trinity College Dublin, models mathematically what happens when information gets erased in a quantum regime. All erasures cause some heat dissipation, as expected—but to their surprise, the researchers found that every once in a while, the heat dissipated from an erasure is extremely high. In uncovering the potential for such rare but high-impact events, the study suggested something about the future of quantum computing hardware: that it may have to use reversible logic, which by definition does not require erasure of information.

The research also brought back to the fore a famous paradox in thermodynamics that has intrigued scientists since the 19th century—and also a 20th-century insight into the relationship between information and heat, which not only resolved that paradox, but also has practical implications today.


A bit of information can be encoded in the position of a particle (left or right). A demon can erase a classical bit (blue) by raising the left side until the particle is definitely on the right. A quantum particle (red) also can tunnel under the barrier, which generates more heat.

GRAPHIC BY PROFESSOR GOOLD, TRINITY COLLEGE DUBLIN/COURTESY OF EUREKALERT.ORG

A Famous Paradox

The paradox, laid out by physicist James Clerk Maxwell, appeared to show a violation of the Second Law of Thermodynamics, which states that the entropy of a closed system is always increasing. Physicists of Maxwell's day already knew the Second Law: they understood heat cannot spontaneously flow from a cold region to a hot one, any more than a cooked egg can return to a raw state. Yet Maxwell, writing to a colleague in 1867, described a hypothetical scenario in which this fundamental law did not seem to hold.

In Maxwell's thought-experiment, a vessel was separated into two halves by a diaphragm, each half containing gas molecules moving at varying speeds; an intelligent being—later called "Maxwell's demon" by Lord Kelvin—seeing the paths and velocities of all the molecules, selectively opened and closed a door in the diaphragm to let slow (cold) molecules move through to one side, and the faster (hotter) ones gather on the other side.

The result of this selection process: instead of the vessel reaching thermal equilibrium, "The hot system has got hotter and the cold colder and yet no work has been done," as Maxwell put it. "Only the intelligence of a very observant and neat-fingered being has been employed." Entropy seemed to decrease on its own, an apparent contradiction of the Second Law that has fascinated scientists ever since.

Many brilliant minds have tried to resolve this paradox. In the 1920s, physicist Leo Szilard, devising a single-particle version of Maxwell's demon that reduced the setup to a binary decision problem, suggested that what accounted for the unseen energy costs was the demon's measurements (though Szilard turned out to be wrong about that). Later, Claude Shannon, the father of information theory, hinted at a possible connection between information and thermodynamics through his use of the term "entropy," but the connection was merely metaphorical. "He used 'entropy,' but he was talking about information—bits—not energy," explains Janet Anders, a theoretical physicist working in quantum information science at the U.K.'s University of Exeter.

Landauer and Logical Irreversibility

Though these and other scientists did not quite resolve the paradox, they planted the seeds for a set of ideas that eventually would. In a seminal 1961 paper produced as part of a large research program at IBM to understand the physics of computation, Rolf Landauer put forth the erasure principle that now bears his name. Landauer's principle, now considered the basic principle of the thermodynamics of information processing, states that erasing information always dissipates energy.

Destruction of information through erasure, analogous to irreversible processes in thermodynamics (like cooking an egg or letting a hot liquid reach room temperature), is what Landauer termed "logically irreversible."

This means that once you've turned a series of 1s and 0s into all zeroes, which is what erasure accomplishes, you cannot know which state these bits were in before. (In this way, erasure is fundamentally different from a logically reversible process like swapping every 0 for a 1 and vice versa.) What's more, Landauer argued, the logical irreversibility of erasure entailed physical irreversibility as well, because information is always represented on physical devices. "Whether that's an abacus, or information written on a page, or electronics in a transistor; ultimately, the abstract notion of information is always encoded on physical hardware," says physicist John Goold of Trinity College Dublin, one of the investigators on the quantum fluctuations study.

Or, as Landauer himself has put it, "Information is physical."

One upshot of Landauer's principle is that erasure of even one bit of information cannot be done for free; it always increases entropy. This minimum amount of heat released into the environment (a computable amount known as Landauer's limit) is not only a fundamental limit on the efficiency of any irreversible computer; it is also a way to tackle Maxwell's demon. As Landauer's IBM colleague Charles Bennett would later explain, unless the demon has an infinite memory, making the observations necessary to separate hot from cold particles requires erasing bits of memory, something Landauer showed can't be done without expending energy and thus increasing entropy.

Destruction of information through erasure, analogous to irreversible processes in thermodynamics, is what Landauer termed "logically irreversible."

Besides helping resolve the famous paradox, Landauer's ideas finally related information theory to thermodynamics. The ideas also reconciled another duality: an identity crisis within the emerging field of computer science. In an era when very few universities were offering degrees in computer science, "Computer scientists were really debating this question of what is this field about," says Aaron Wright, a historian of science at Dalhousie University in Canada who has studied IBM's research program on the physics of forgetting. "One of the big dividing lines was: is [the field] about machinery, or is it about mathematics? Landauer's work blurred that distinction, Wright says: "Landauer's principle is precisely connecting those two ostensibly separate worlds." Thanks to Landauer, it is no longer a category error to ask, for example, "how much space does the Pythagorean Theorem take up?"

Engineering Implications

On a practical level, though, Landauer's ideas have not had implications for engineers until recently. "Nobody designing a classical computer has had to

ACM Member News

ATTRACTED TO PROBLEMS THAT SEEM UNSOLVABLE

"I used to be perplexed by movies from the 1950s and 1960s, where a punch card would be put into a computer, the computer would beep and lights would blink, and then an answer would pop out," recalls Emery Berger, a professor in the College of Information and Computer Sciences at the University of Massachusetts, Amherst.

"I just couldn't understand how it could possibly do that, what on Earth was going on?"

Berger attended a high school with a roomful of new Apple II Plus computers, which captured his interest and he never looked back. He went on to earn his undergraduate degree in computer science from the University of Miami in Florida, and obtained both his master's and Ph.D. degrees in computer science from the University of Texas at Austin.

On completing his Ph.D. in 2002, Berger joined the faculty of the University of Massachusetts, Amherst, where he has remained ever since.

Berger's research interests span programming languages, systems, security, and human-computer interaction. At present, he said, "My research focuses on making programs run faster, better, more securely and safely, and more correctly."

Berger spent a significant amount of time as a visiting researcher at Microsoft over the years. During a recent sabbatical, he wrote two papers on finding errors in spreadsheets; some of that technology is being added to Excel.

He also developed and maintains the CSRankings.org website, a portal that makes it easy for students to find prospective faculty advisers by identifying the projects and research areas professors are working on at computer science institutions around the world.

—John Delaney

worry about Landauer's principle," says Christopher Jarzynski, a physicist at the University of Maryland whose expertise includes the thermodynamics of small systems. "When we're erasing information in our computers, we're actually dissipating way more energy than Landauer's limit." That's certainly a waste of energy, but with proper cooling, the dissipated heat poses no threat to the computer's accuracy—and even if it did, it wouldn't be because of proximity to Landauer's limit.

With quantum computers, though, Landauer's principle will become more relevant, and the paper by Goold and his colleagues clearly shows why. Even in classical mechanics, Goold explains, a key feature of the thermodynamics of microscopic systems—whether these small-scale systems are protein molecules or single-atom transistors—is fluctuation. In other words, "I have to perform the experiment many, many times, and I get different outcomes." (In fact, because it takes a full probability distribution to show all these possible quantities of heat and work, another term for the thermodynamics of small systems is "stochastic thermodynamics.") Microscopic particles obey the laws of thermodynamics on average, but dissipation events fluctuate greatly. At one extreme, some individual events can even appear to violate the Second Law.

This much has been known for years—but Goold and his colleagues wanted to understand some of the stochastic thermodynamics of quantum systems. So, using equations that model such things, they looked at erasing a bit of information in a quantum mechanical way. "We found that if you look at the fluctuations"—rare events—"you get very large, rare deviations when you get a quantum mechanical superposition," Goold says. Since superposition (or the appearance of a particle in two positions simultaneously) is crucial to the way quantum computers do their magic, any extreme event that interferes with this behavior could be disastrous.

"In quantum computing, everyone is working on limiting errors because it's already error-prone," says Anders, the University of Exeter physicist, "so when you put in heat, you raise the chance of errors much more." That's why quantum computers must operate at extremely cold temperatures—which means, as Anders puts it, not just "you-have-to-put-your-socks-on cold," but close to absolute zero.

What the study suggests to her is that in quantum computing, "you don't want to do Landauer erasure, since that creates high heat events." The alternative: "Whenever possible, you want to do reversible operations, so nothing is erased." C

Further Reading

Landauer, R.
Irreversibility and Heat Generation in the Computing Process
***IBM Journal*, July 1961**
https://www.informationphilosopher.com/solutions/scientists/landauer/Landauer-1961.pdf

Maddox, J.
Maxwell's demon: Slamming the door.
***Nature 417*, 903 (2002).**
https://doi.org/10.1038/417903a

Miller, H.J.D., Guarnieri, G., Mitchison, M.T., and Goold, J.
Quantum Fluctuations Hinder Finite-Time Information Erasure near the Landauer Limit
Phys. Rev. Lett. 125, 160602 – Published 15 October 2020
https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.125.160602

Rex, A.
Maxwell's Demon—A Historical Review
***Entropy 2017, 19*(6), 240**
https://www.mdpi.com/1099-4300/19/6/240/htm

Szilard, L.
On The Decrease Of Entropy In A Thermodynamic System By The Intervention Of Intelligent Beings
English translation of the classical paper ober die Enfropieuerminderung in einem thermodynamischen System bei Eingrifen intelligenter Wesen, which appeared in the *Zeitschrift fur Physik*, 1929,53,840-856.
http://fab.cba.mit.edu/classes/863.18/notes/computation/Szilard-1929.pdf

Wright, A.S.
The Physics of Forgetting: Thermodynamics of Information at IBM 1959–1982
Perspectives on Science 2016, vol. 24, no. 1
https://www.mitpressjournals.org/doi/pdf/10.1162/POSC_a_00194

Based in San Francisco, **Marina Krakovsky** is the author of *The Middleman Economy: How Brokers, Agents, Dealers, and Everyday Matchmakers Create Value and Profit* (Palgrave Macmillan).

Milestones

Aaronson Awarded ACM Prize in Computing

ACM named Scott Aaronson the recipient of the 2020 ACM Prize in Computing for his groundbreaking contributions to quantum computing.

Aaronson, the David J. Bruton Jr. Centennial Professor of Computer Science at the University of Texas at Austin, helped develop the concept of quantum supremacy, the point at which a quantum computer accomplishes something that no classical computer could in a reasonable amount of time. He also established many of the theoretical foundations for quantum supremacy experiments, which allow scientists to develop convincing evidence that quantum computers provide exponential speedups without having to first build a fully fault-tolerant quantum computer.

In addition, Aaronson contributed significantly to the areas of boson sampling and classical complexity theory, as well as writing a respected book on quantum computing, and several articles for a popular science audience.

"Few areas of technology have as much potential as quantum computation," said ACM President Gabriele Kotsis, adding that Aaronson. "is esteemed by his colleagues for the breadth and depth of his contributions. He has helped guide the development of this new field, while clarifying its possibilities as a leading educator and superb communicator. Importantly, his contributions have not been confined to quantum computation, but have had significant impact in areas such as computational complexity theory and physics."

Pravin Rao, chief operating officer of Infosys, which funds the ACM Prize in Computing's $250,000 prize, congratulated Aaronson on his accomplishments. "The successful quantum hardware experiments by Google and others have been a marvel to many who are following these developments. Scott Aaronson has been a leading figure in this area of research and his contributions will continue to focus and guide the field as it reaches its remarkable potential."

Society | DOI:10.1145/3460216

Chris Edwards

Let the Algorithm Decide?

Algorithms fail their first test to replace student exams.

STUDENTS GENERALLY REGARD exams with trepidation for good reason: a bad day can easily trip them up, leaving their grades wanting. Concern over the accuracy of high-stakes exams and standardized tests have led the U.S. and other countries to look at alternatives that are less vulnerable to a poor performance on a single day and which, in principle, offer more accurate ways of determining a student's ability in a particular subject.

In 2019, a team based at King's College London in the U.K. used a long-term study of twins to determine how well teachers' assessments fare against exams in predicting overall performance once compulsory education has finished. They found teachers' assessments are as reliable as test scores at every stage, and recommended this approach to grading could replace some, if not all, high-stakes exams.

Amid the chaos caused by lockdowns in the spring of 2020 to try to limit the spread of the COVID-19 virus, it seemed school students in the U.K., as well as those taking the International Baccalaureate who were due to complete their courses late last year, might for once be able to take advantage of assessments and no longer have to fear the exam for which they hadn't crammed by taking advantage of the reliability of classroom assessments. At least, that was how it seemed before the results were published.

Days after receiving their grades, crowds of students protested outside the Department of Education in central London, claiming their results fell so short of expectations, they must be flawed. It was a similar situation for students around the world who were expecting to graduate from the International Baccalaureate course: many students found their results differed markedly from what they thought their body of coursework indicated.

One possible reason for the backlash was simply that teachers, parents, and

PHOTO BY DOMINIKA ZARZYCKA/NURPHOTO VIA GETTY IMAGES

pupils had overly optimistic opinions of their abilities that, under normal circumstances, would be tempered by exam results. Stripped of the element of chance inherent in testing, the replacement system seemed to need more convincing explanations of why expectations were not met.

"I think it is hard to think of any process that would not suffer problems," said Imperial College London professor of statistics Guy Nason.

One issue that U.K. politicians in particular wanted to address with the exam-replacement scheme devised by the nation's education regulator Ofqual was one of grade inflation, or at least the perception of it. Since 1992, U.K. schools have had to publish their results for compilation into national league tables. Many parents use those tables to help choose where they try to send their children. This was seen as providing a clear incentive for teachers to deliver optimistic assessments.

France, which last year also opted for grading of coursework by teachers instead of independently marked exam papers, saw almost 96% of candidates obtain a pass for its form of the baccalaureate qualification. The pass rate was 88% the previous year. This, in turn, led to universities in the country opening close to 10,000 additional places for the 2020 intake.

For their respective systems, the U.K. and the International Baccalaureate Organization (IBO) decided to address the potential for grade inflation through semi-automated moderation schemes. These were used to fit the distribution of grades awarded in 2020 to the pattern of prior years. In addition to its moderation scheme, the IBO had a sampling of coursework marked by independent examiners, stating this would help "maximize the confidence that every student will receive a fair mark for their coursework." Rather than attempt to organize a mass marking of coursework by independent examiners, the U.K. education authorities opted to rely purely on algorithms to redistribute the grades.

Once they saw their results, IBO students pointed to apparent anomalies in their awarded grades, which often seemed several points below those indicated by either teacher- or examiner-marked coursework. Students taking U.K. courses encountered more extreme examples. It was possible in some cases for a student expected by their school to obtain an A grade to wind up with one as bad as a U (Unsatisfactory), a grade normally reserved for an exam paper so bad it cannot be marked.

"Part of the issue in this case is that the evidence base that regulators in the U.K. relied on in their algorithms was extremely weak," Nason said, adding that the complexity of the system, which was accompanied by several hundred pages of description, makes it hard to identify the root cause of anomalies. The urgency of the situation brought on by the pandemic meant there was no time for testing, and Ofqual relied on statistical analysis that observers believe was flawed.

"It is possible to get a better understanding of uncertainty in these situations, but gathering and quantifying such evidence is expensive, time-consuming, and complex, and would, in particular, not have been possible in the time frame afforded by this year's pandemic situation," Nason said.

One apparent source of anomalies lies in a decision to try to match the distribution of grades from prior years on a school-by-school basis. That contrasts with what happens with conventional exams: test scores for a particular subject are assigned to grade boundaries across the entire student base. In the U.K., teachers were told to rank the students in their classes in order of ability; the rank ordering would determine who was to have their grades adjusted in order to fit the average distribution of prior years.

The problem that emerged was that schools that had traditionally seen large numbers of poor performers with maybe just a few high-fliers in a given year, encountered large downgrades for those who were unlucky in their ranking. Nason said such rankings have been shown to be subject to high degrees of uncertainty and subjectivity.

Confusion as to how appeals would be dealt with further dented confidence in the approach taken by the U.K. regulator. Helen Smith, a post-graduate researcher working on the use of artificial intelligence in decision-making at the University of Bristol's Center for Ethics in Medicine, claims the process presented by Ofqual contravened the U.K. government's own guidance on algorithmic decision-making: that unexpected results, and not just those due to bias, should be subject to appeal.

Ofqual's senior executives later argued in a hearing held by members of Parliament that they expected the appeals process did allow for unexpected results, and not just those who could claim evidence of bias. However, they said they anticipated the complaints to come from schools, rather than from students or parents directly. Daan Kolkman, a research fellow in computational sociology at the Eindhoven University of Technology in the Netherlands, noted, "Although there was a procedure for redress, it was unclear to many how to appeal their grade."

As the complaints built up, a decision was made in the U.K. in the early autumn to reverse the regrading process and use the unmoderated assessments to award grades. This resulted in a large reduction in complaints, as many of the students who risked being denied university places were able to take up those places. The IBO came to a similar decision for its qualifications.

In the wake of those reverses of policy, Ed Humpherson, director-general of regulation at the U.K.'s Office for Statistics Regulation (OSR), announced a review of the procedures used by Ofqual. "The resulting negative backlash, specifically on the role that algorithms played, threatens to undermine public confidence in statistical models more broadly," he said.

A case like this presents opportunities for governments to learn how to create better processes for algorithms that are used to support decisions affecting individuals. Kolkman argued there is, in general, a clear role for an algorithm watchdog that can prevent data from being used in a way that is difficult to justify. For example, though rank ordering was relatively easy to collect, its high level of uncertainty should probably have led to it being replaced by another, though possibly more expensive to collect, statistic.

"It may be tempting for algorithm developers to work with the data at hand, not the data that is most suited for the purpose of their analysis. This does not necessarily present a problem, if the limitations of the analysis are well-enough understood. I feel the problem does not necessarily lie with the algorithms we use, but with our lack of robust quality assurance and clear redress procedures," Kolkman said.

The need to test algorithms before deployment was one tackled before the COVID pandemic hit by Sir David Spiegelhalter, chairman of the University of Cambridge's Winton Center of Risk and Evidence Communications. He would later join the OSR's panel investigating Ofqual's decisions. In a paper published early last year, Spiegelhalter proposed borrowing auditing techniques from the pharmaceutical world that would put big-data algorithms into trials before they could be approved. Kolkman said the pharmaceutical-trials model would likely be too onerous for all systems, while other areas, such as food safety, might provide models for less-impactful algorithms.

Kolkman takes the view that algorithmic analysis should be conducted not just by subject-matter experts, but by laypeople as well, in order to ensure the decisions being made by the systems are as comprehensible as possible. Sometimes the question is bigger than which algorithm provides the best replacement for an existing system that may itself be flawed.

Rebecca Cairns of the Deakin School of Education in Australia points to the work by Kings College London on the value of teacher assessment versus exams. She and others see the post-pandemic environment as an opportunity to take the time to re-examine how the evaluation of students' work should take place, and the kinds of data needed to deliver it. However, it may take more upheaval, such as another year of canceled exams, before that debate begins. C

Further Reading

Rimfeld, K., Malanchini, M., Hannigan, L.J., Dale, P.S., Allen, R., Hart, S.A., and Plomin, R.
Teacher assessments during compulsory education are as reliable, stable and heritable as standardized test scores
***The Journal of Child Psychology and Psychiatry* (2019). Volume 60, Issue 12. DOI: 10.1111/jcpp.13070**

Smith, H.
Algorithmic bias: should students pay the price?
***AI and Society* (2020). DOI: 10.1007/s00146-020-01054-3**

Kolkman, D.
The (in)credibility of algorithmic models to non-experts
***Information, Communication & Society* (2020). DOI: 10.1080/1369118X.2020.1761860**

Spiegelhalter, D.
Should we trust algorithms?
***Harvard Data Science Review*, 2.1 (2020). DOI: 10.1162/99608f92.cb91a35a**

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

DOI:10.1145/3461464

Rebecca T. Mercuri and Peter G. Neumann

Inside Risks

The Risks of Election Believability (or Lack Thereof)

With 90% of the 2020 U.S. general election ballot contents verifiable by paper, why do only 65% of voters trust the results?

Despite or perhaps because of COVID-19 health concerns, a record 155 million ballots were cast for President in the 2020 general election, with both the winner and runner-up each individually getting more votes than any candidate in U.S. history. Yet, according to post-election polling,[11] only two in three voters felt confident the election was free and fair. Even the voter-verified paper ballot[9] for direct-recording electronic voting machines (which enables hand-counting via an audit or 100% tally) does not in and of itself create sufficient credibility in the election results.

Trustworthiness in elections is inherently a total-system problem (as considered more generally[2]). Every part of the overall process (for example, voter registration, ballot layouts, casting and counting, audits, and recounts) provides potential points of compromise. Problems may result from human errors, intentional manipulations (such as ballot tampering, creative disinformation, and insider fraud), imbalanced redistricting (for example, local and state gerrymandering), Electoral College issues, unlimited funding and targeted advertising (for example, Citizens United, Cambridge Analytica), delays at the polls due to malfunctioning equipment, and even the effects of environmental conditions (such as weather, power outages, and pandemics). Education is also a vital consideration, especially in the ability of voters to cope with changing conditions, new instructions, politically biased reporting, and a host of other incidental or even disingenuous factors.

For the last four decades, computer scientists have been among the most outspoken advocates for trustworthiness, security, and reliability in election systems.

As technologists, we feel it is important to assert that while recognizing the plethora of other potential risks that can affect election outcomes, a good starting place for reform must focus on the systems used to enable the casting, counting, and reporting of votes. Here, we highlight the fundamental importance of incorporating transparency and trust methods from other confidence-building processes into these specialized computational systems, including some novel approaches that have not previously been applied to elections. While we recognize much of this column is highly U.S.-centric, many of the issues raised also translate to election concerns around the globe.

Hack the Vote

For the last four decades, computer scientists have been among the most

outspoken advocates for trustworthiness, security, and reliability in election systems." Early conferences, such as those sponsored by the Electronic Frontier Foundation and Computer Professionals for Social Responsibility, provided opportunities for discussion of election issues and potential solutions. Peter Neumann's ACM Risks Forum has logged several hundred reports of election and voting-related human and computer errors, evident fraud, and other problems, from the 1980s to date. Additionally, 11 previous *Communications* Inside Risks columns have been devoted specifically to this topic, with several other articles pointing out relevance to election integrity.

One of the earliest researchers to raise serious concerns related to hacking was Roy Saltman. His 1988 treatise[14] for the National Bureau of Standards (now NIST) was cited frequently in testimonies about the Florida 2000 presidential election. Since 1975, Saltman's position has been that reduced public confidence in the election process can be related to the lack of assurances that software modifications have not occurred, as well as to the vulnerabilities inherent in commercial-off-the-shelf (COTS) products used in voting systems. These and many other issues he had flagged continue to fail to be sufficiently addressed by the Voluntary Voting System Guidelines (VVSG) established by the U.S. Election Assistance Commission (EAC). Actually, the Federal Voting System Guidelines are not mandatory, and are not applied nationwide—due to the U.S. Constitution's preservation of states' rights.

Some states may leave the choice of voting method and system up to their counties. Even within a single county, the voting and tabulation systems may differ—depending on whether the ballot is cast or counted at a polling location, at a voting center, at the election office, or via in-person or remote accessibility. Thus, there is a total lack of uniformity among the states and also within many states. For example, the November 2020 election in California used 21 different voting systems or versions from seven different vendors: seven products from Dominion, five from ES&S, five from Hart, and one each from Democracy Live, Interactive, Runbeck, and VSAP.[15] Ensuring all of these systems are operating in accordance with whatever guidelines the state has decided to impose for each particular election is a daunting logistical task.

After the 2000 election, the IEEE (well-respected for its 802.11 family of communications standards) launched a project intended to establish a performance standard for the evaluation of voting equipment (P1583). Unfortunately, the multiple-year effort bogged down when "attempts to insert adequate security into the standard [were] thwarted by vendors attempting to protect legacy systems, software, and proprietary trade-secret products that produce no independent method for auditing the election."[12] These issues (and others related to Mean Time Between Failures, accuracy thresholds, and COTS) were vigorously argued between election integrity advocates and system vendors, eventually leading to an unresolvable stalemate—resulting in the non-issuance of the draft as an accepted IEEE standard.

The EAC's VVSG 1.0 (2005), 2.0 (2015) and 2.1 (2021) each pertain to new products; no guidance is provided as to obsolescence or with respect to vulnerabilities later discovered in systems previously certified. Older systems are grandfathered and continue to be used. Also, due to the length of time needed to update and recertify voting systems under the new VVSG, purchases of election equipment during 2021, 2022, and likely even into 2023, will have been designed under the prior guidelines, and will not adequately address more recent concerns. Communities should be advised to wait until VVSG 2.1-certified products become available.

As an example of such built-in obsolescence, in the early and mid-2000s (despite strong efforts by knowledgeable citizens to educate county and state officials about the dangers of paperless electronic voting systems), most New Jersey counties replaced their legacy lever machines with AVC Advantage DREs that were certified only to the then-obsolete FEC 1990 standards. Anomalous situations were reported early on, with observations of vote flipping, where a press for one candidate selects another instead. In the Super Tuesday Presidential Primary of February 2008, some 37 voting machines in eight New Jersey counties showed the Republican candidates to have received more votes than the number of voters who had signed in at the polls. This tabulation error was eventually attributed to a software bug. Andrew Appel demonstrated in open court that he could pick the lock on these machines, replace the ROM containing the software, and relock the door in less than seven minutes.[1] To date, most of New Jersey continues to use these vulnerable and unauditable systems.

DEFCON25, held in 2017, featured the first-ever Voting Machine Hacking Village "to highlight cyber-vulnerabilities in U.S. election infrastructure—including voting machines, voter registration databases, and election office networks."[3] Lessons learned indicated: the systems could be hacked even with limited time, information, and resources; foreign-made parts introduce supply-chain concerns; the exercise was not merely a stunt, and demonstrated that a diverse community of stakeholders should be engaged; and "affirmed what election security advocates have been arguing for years: There is urgent need for election officials to implement measures to secure U.S. election infrastructure."

The 2018 and 2019 DEFCON Voting Villages continued to report similar vulnerabilities, also finding that equipment was sometimes shipped with security features turned off, previously studied equipment showed new vulnerabilities, ballot-marking devices posed new systemic risks, remote attacks were possible even with air-gapped equipment, some hacks could occur in two minutes (less than the time it takes to vote), and earlier hacked equipment models were not remediated by their vendors, even though they had been informed about the known risks.

Election Forensics

Forensics is the process by which evidence is examined and described for presentation in a legal setting (for example, at a trial, hearing, or mediation), in order to allow for adjudication or resolution of a dispute. Some key aspects of forensics include these: each side is allowed independent access to the evidence; the evidence has been preserved in a traceable and pristine fashion; and the forensic review occurs using standardized tools and approaches that can be replicated.

What is immediately evident in comparing forensics to elections is this: typically, only the losing candidates are permitted to challenge the results, and often they are not given an opportunity to directly examine the evidence; some of the evidence may not be properly secured (for example, it may sit at voting locations for days before being transported to warehouse facilities that may also be insecure); and while there may be methods in place for performing recounts, there is a complete lack of specified procedures and protocols for conducting a thorough forensic review of any aspects of elections. In fact, critical features of such examinations are typically prohibited by restrictive trade-secret agreements forced on the municipalities by the election equipment vendors.

One would not inspect a few vehicles for emissions and then provide a pass for all others of the same model and type, but this is what is done with election equipment.

There is the general assumption that election administrators are unbiased, and that the voting and vote-counting processes have been honest. Most do abide by the rules of the offices in which they serve, but notably, some do not. Still, all are given powers that, unlike in a legal trial, enable them to prevent (intentionally or by invocation of laws and procedures) a full triage of the equipment, software, ballots, and other evidence by the forensic examiners for any or all candidates in the election (whether winners or losers). This was illustrated in Georgia's 2020 election, due to the need to prepare the voting equipment quickly to allow early voting for the state's run-off Senate races. Such preparations would necessarily reset the equipment, which eliminates the possibility of a forensic investigation. Imagine a murder scene where no one is allowed to examine the dead body, while the murderer has access to the evidence and is allowed to eradicate all traces of it—before the forensic examiners arrive. That is the situation we have been confronting in election investigations, and it must change.

Voting System Testing

In addition to maintaining the VVSG, the EAC also arranges for certification and testing (paid for by the equipment vendors and with results shielded by trade secrecy) for a small number of sample machines of each version and style, intended to provide an assurance of compliance with the guidelines. Analogously, one would not inspect a few vehicles for emissions and then provide a pass for all others of the same model and type, but this is what is done with election equipment. Lacking individual acceptance testing for each voting system, there is no way to know whether the procured units all actually conform to the VVSG.

The only U.S. state that ever claimed to perform such scrutiny was Georgia, which for many years had contracted with Kennesaw State University's Center for Election Systems to provide services that involved checking each voting system on procurement, before delivery to the counties. This contract was terminated by Secretary of State Brian Kemp in October 2017, following an investigation that shockingly revealed that critical vulnerabilities to the systems were known by the Center to have existed prior to the 2016 general election.[5] This information was never properly reported, and the evidence was deliberately deleted—leaving the accuracy of the election results in doubt.

With such a diversity of complex systems, amplified by the necessity of correct recognition of hundreds of different ballot layouts with thousands of candidates, it is difficult—indeed impractical—for state and county officials to fully verify that proper functionality existed before, during, and after each election. Instead, they typically rely on the use of sample ballot sets during pre-election setup. But it has been demonstrated that sample ballots could act to circumvent or even install malware in vulnerable systems. Nor are officials typically provided with the in-depth knowledge needed to establish believability that these products are in compliance with the state's voting system standards. On the other hand, those states with a single type of voting system, or products from only one manufacturer, may produce a monoculture risk such that an attack could potentially affect their entire election's results.

In recent flurries of legislation, we are now seeing that states can and do establish their own rules for how elections are conducted. Variations may include the dates and times for voting, the manner in which ballots are laid out (some states use the party of the current governor to determine which candidates appear first—above or to the left of others), the types of voting equipment that will be used, absentee and mail-in ballot rules, and so on. One complex problem is that there has been no reconciliation of the various state laws pertaining to what ballot mark (X, dot, check, circle, arrow, and so forth) constitutes a legal vote, so many voting systems fail to recognize legally valid ballot choices. This can be remediated with hand counting of the full set of ballots (as long as the people doing the counting have been properly instructed), but this is customarily not performed. Partial audits may not pick up enough of these misreported votes to make a difference in the election outcome.

One of the myths of having trustworthy computer systems for elections is that everything depends only on the quality of the software.

Audits vs. Recounts

Some states (including Florida, Georgia, and Michigan) have instituted policies and laws that actually prohibit the hand counting of paper ballots and allow only rescanning. Many other states prescribe only a partial statistical audit. The only reason that Georgia was able to conduct a 100% manual recount (in violation of its state law) for the 2020 presidential race was that wording in the state's Risk-Limiting Audit (RLA) legislation allowed for such, if the disparity between the candidates was close enough that nearly a full count would have had to be performed for results to be above the threshold for sufficient assurance of correctness. Had Georgia's law been worded differently, a tedious process of randomization for ballot selection would have had to occur, possibly not providing results in the time needed to certify the election, and with less believability than the multiple full counts that actually were performed throughout the state.

As of 2017, approximately half of the U.S. states required some form of post-election audit, typically only of 1% of the ballots cast. The RLA method has been gaining in popularity, because the number of ballots counted depends on the margin of votes between the leading candidates. If there's a wide difference, counting can stop sooner. Actually, the formula that determines when to stop counting is fairly sophisticated,[13] such that most election officials (beyond a rare few with deep knowledge of statistics and probability) would not be able to conduct an RLA without computer assistance. So, while the method may seem to be transparent, the calculations and their correctness are not necessarily obvious or comprehensible.

Some have suggested referring to the RLA as a Recount-Limiting Audit, since a primary intention of this method is to speed up validation of the election results by preventing full recounts. In addition, since the number of ballots to audit is determined by the initially computer-generated vote tallies (which have not yet been certified), there is a believability problem with regard to whether the RLA will be sufficient to reveal ballot tabulation anomalies.

Another issue is that since the selection of ballots to hand-count can be based on precinct groups, RLA is inherently better at detecting localized problems (such as with a particular voting machine or scanner) than it is with dispersed issues (a few votes added or subtracted here and there). Localized problems are usually more immediately evident in the vote tallies anyway. Dispersed problems are harder to detect, and are also less likely to be caught via RLAs. For example, major cities often show consistent voting patterns for particular party candidates in large numbers, year after year. A hacker might siphon off some votes from the winning candidate in these cities, adding them to the runner-up, or even to a third-party candidate, without detection. This would not affect the outcome of local races, but could be sufficient to alter the results of statewide races (such as for president, senator, and governor) without detection. Note that audit discrepancies may provide little or no clues as to how the tally anomalies occurred, so a forensic review of the voting systems should be (but typically is not) performed.

Given these flaws and concerns for audits, and including the time that it takes to conduct them, a better alternative might be to publicly count paper ballots on election night (as is done in the U.K. and Canada). These proceedings should be live-streamed and recorded for later scrutiny. One-of-*N* races (where a single candidate is selected by the voter out of *N* choices) are especially easy to perform using the bin-sort method well known to computer scientists.

CISA Oversight

For the 2020 election cycle, the U.S. Cybersecurity and Infrastructure Security Agency (CISA) took a proactive role in trying to thwart election disinformation via its cisa.gov/rumorcontrol website. The agency's October 20 pre-election statement asserted "We remain confident that no foreign cyber actor can change your vote, and we still believe that it would be incredibly difficult for them to change the outcome of an election at the national level." Noted is that this seemingly positive press release does not encompass the full gamut of election shenanigans, including those that are capable of being performed by U.S. citizens.

Following the 2020 general election, on November 12th, CISA issued a 10-person joint statement that included leaders from the Election Assistance Commission, the National Association of Secretaries of State, the National Association of State Election Directors, and numerous representatives from election service companies (Unisyn, Hart InterCivic, ES&S, ERIC, and DemocracyWorks). The statement included the assertion: *There is no evidence that any voting system deleted or lost votes, changed votes, or was in any way compromised.*[6]

Four days later, Matt Blaze issued a letter signed by 59 computer scientists[a] (many of whom are well known to the election integrity community), which echoed the CISA joint statement asserting that, "To our collective knowledge, no credible evidence has been put forth that supports a conclusion that the 2020 election outcome in any state has been altered through technical compromise."

What is curious about both the CISA and joint computer scientists' statements is that they were premature. The secretaries of states would not be certifying their election results for many weeks. Most of the nation's voting systems were still on lockdown for pending election challenges and recounts, and could not be physically examined. Some vote counting was still ongoing. Absolutely none of the 10 people on the CISA Joint Statement, or the 59 people on the computer scientists' letter, had performed any post-election forensics or triage that would support these conclusions. In fact, many of the scientists (including Andrew Appel, Richard DeMillo, Alex Halderman, Harri Hursti, and Philip Stark) had, a few months prior to the November election, provided testimony in a Georgia U.S. District Court matter on behalf of plaintiffs against Brad Raffensperger, et al., objecting to the use of the electronic Ballot Marking Devices on the grounds that they might not be sufficiently accurate and that the scanners may incorrectly report results.

The closing pages of U.S. District Judge Amy Totenberg's ruling in that case[b] included the following pertinent statements: "The stealth vote alteration or operational interference risks

a Scientists say no credible evidence of computer fraud in the 2020 election outcome, but policymakers must work with experts to improve confidence (Nov. 16, 2020); https://bit.ly/3mZeG2u

b Totenberg, Honorable Amy, Opinion and Order in *Curling, et al. v. Raffensperger, et al.*, Civil Action No. 1:17-cf-2989-AT (Oct. 11, 2020); https://bit.ly/2QxtysN posed by malware that can be effectively invisible to detection, whether intentionally seeded or not, are high once implanted, if equipment and software systems are not properly protected, implemented, and audited. ... Given the masking nature of malware and the current systems described here, if the State and Dominion simply stand by and say 'we have never seen it' the future does not bode well."

Not heeding this warning, the CISA and computer scientists' statements effectively said "we have never seen it" without conducting any actual investigation to determine if a cybersecurity breach affecting the election results had happened or not, in Georgia or anywhere else in the country. This is not reassuring to the voters. Far better to have said "we don't know" or "we are investigating" than to prematurely issue statements intended to convey that everything was copacetic.

Ironically, we should note that after the election concluded, it was later learned that CISA and other security-related government agencies had been breached with the SolarWinds attack, suffering an as yet fully unknown extent of damages and loss of information to foreign agents. This unprecedented hack remained undetected for about nine months, spanning the time of the 2020 election.

If those charged with protecting elections cannot defend their own assets, their claims of "no compromise" of the election systems must be considered with a skeptical eye. On the other hand, research-based claims that voting systems and election results have been or could be compromised, may begin to be suppressed. This is now being put to the test by the defamation lawsuits seeking $4.3B in restitution by the voting system vendors Dominion and Smartmatic. Whether free speech and freedom of the press will prevail with regard to exposure of election security concerns is left to be seen.

Paper Ballots, Anonymity, and Cryptography

While much of the controversy in the 2020 presidential election focused on absentee vs. mail-in ballots, structurally these paper ballots are the same, and may even be identical to the scanned paper ballots used at the polling locations. Their differentiation is legal in nature—an absentee ballot is issued to a registered voter who has requested one because they will not be able to visit a polling place on Election Day (or during early voting) for a legitimate reason. A mail-in ballot is one that is issued by the locality or state without having been specifically requested by the voter. In 2000, Oregon became the first state to eliminate precinct voting, replacing it with all mail-in ballots. More recently, and especially due to postal delays, local drop-boxes have become a convenient way of depositing absentee or mail-in ballots, but some states are now trying to roll back their use.

Elections must be constructed and conducted such that everyone can, with extremely high confidence, rationally believe the results reflect the will of the voters.

As for the paper ballots themselves, very little has evolved to make them more trustworthy over the past quarter-century. By comparison, paper money and checks have changed dramatically during this time. One can obtain 10,000 checks that have 20 security features (including a foil hologram, prismatic multicolor background, microprinting, heat-sensitive icons, watermarks, invisible fluorescent fibers, red/blue visible fibers, toner adhesion, and chemical sensitivity) for a little less than 13 cents each. Still, the printing companies and voting system vendors have yet to implement any of these types of document authentication methods for paper ballots. Certainly the paper stock that contains the authentication would need to be strictly inventoried and controlled, to prevent spoofed ballots from being subversively created. However, there has been no demand for such ballot features, not even by the individuals and groups who have been complaining about the possibility of dead people voting (which turns out to be exceedingly rare).

With the greater availability of non-precinct voting comes the increased risk of vote selling or coercion. But it may also be the case that as people have become more willing to share personal details via social media, and as photographing and publishing things (such as your filled-out ballot) has also become commonplace, there may be less concern about the secrecy or privacy of one's voting choices. A major finding of Rebecca Mercuri's Ph.D. dissertation (partly synopsized in Mercuri[10]) established that "the need for anonymity precludes the use of transaction logging for providing access assurances" in direct-recording electronic voting systems. In other words, it is not possible in fully anonymous voting to ensure individuals (such as voters or precinct workers) have not tampered with the voting system during the election in order to alter the vote totals, without the availability of voter-verified paper ballots to use in performing a cross-check. This is not just a theoretical speculation but rather is based on NP-completeness proofs.

With the elimination of full anonymity (such as happens in a stock shareholder election—one casts ballots that are tracked to the owner of particular shares)—the voter can be contacted to verify that they did cast their votes as recorded. Actually, the U.K. does use ballot numbers to track votes, and under very constrained circumstances can require a voter to later validate that their ballot was cast as intended.

Various cryptographers have devised methods for generating encrypted ballots. Some of these schemes (notably Chaum's[4]) enable the voter to decode their votes or track them in order to confirm that not only was the ballot received for counting, but also that their vote choices have been entered into the tallies correctly. Unfortunately, as with Risk-Limiting Audits, these crypto algorithms are too complex for most people to understand, which limits the believability of correctness of the implementing software (which may also be insecure) to an elite intelligent few. Some have considered the use of blockchain as a voting method, but this would require extensive reliance on trustees and software to properly maintain the cryptographic records. Quantum voting has also been suggested, but this technology is not yet mature—and likely to be overkill. These techniques could someday show promise for elections but are not yet well-enough understood by the general public to ensure believable results.

Open Source Software and Open Architecture Hardware

One of the myths of having trustworthy computer systems for elections—and indeed for trustworthy applications in general—is that everything depends only on the quality of the software. This myth is finally being debunked by exploits that take direct advantage of already existing hardware risks. Examples include speculative execution vulnerabilities introduced by the Spectre exploits,[7] and the Thunderclap vulnerabilities[8] whereby a USB-C stick could take over most systems with or without IOMMUs. Neither of those cases can be resolved only in software. In addition, hardware supply-chain compromises have long been a concern, most recently exemplified by the hacked water-treatment facility in Florida.

Ideally, total-system hardware-software architectures should be deployed to eliminate vulnerabilities as well as simplify the programming process that is presently riddled with potential flaws. An example of such an architecture is provided by the emerging Capability Hardware Enhanced (CHERI) RISC Instructions hardware instruction-set architecture and its operating systems.[c] Existing versions that could be used to develop voting systems include the open sourced CHERI-RISC-V hardware ISA and Arm's prototype Morello board of the CHERI spec integrated into their Version 8 hardware, each with appropriate open sourced software.

A combination of CHERI's least-privilege access controls, fine-grained and course-grained compartmentalization, with highly principled design, has the potential of dramatically increasing the trustworthiness of the computer aspects of future elections. In addition, formal proofs that critical security properties are satisfied by the CHERI instruction-set architecture could further increase the believability of that claim.

Still, the admonition by Ken Thompson in his classic 1984 ACM A.M. Turing Award Lecture—"Reflections on Trusting Trust"—holds true today: "You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.)" Indeed, many issues continue to persist (including some noted in this column) that can impact the believability of election results from voting systems whose vendors deliberately prevent open software and hardware reviews.

But neither COTS nor open source hardware and software are inherently immune to exploits. Consequently, election integrity must also involve protections against insider misuse, including accountability in reliably recording all changes in hardware, software, data, and system configurations. This is a non-trivial problem that posting source code on GitHub cannot solve entirely. The very real possibility of exposure (via an open review) of a major flaw or breach shortly before, during, or after an election, stands to wreak havoc on the believability of the vote totals, but this risk is not reduced by refusing to make the code open to review.

Conclusion

Elections must be constructed and conducted such that everyone (all of the winning and losing candidates, as well as those who have supported them) can, with extremely high confidence, rationally believe the results reflect the will of the voters. Technology is only one part of the end-to-end voting process. Repeatability and transparency can provide critical assurances that enhance trust—but voter-verified paper ballots and open software and hardware do not ensure correctness if scrutiny is thwarted or lacking. Risk-Limiting Audits and cryptographic methods, while perhaps mathematically sound, are not believable if the general public lacks the intellectual sophistication to understand how they work, or the capability to independently demonstrate correctness proofs for themselves. In short, the risks of seeking election believability are that the cure can be worse than the disease. As computer scientists, we must bear responsibility for warning about election vulnerabilities and proposing solutions, while also being careful not to make unfounded statements of assurance or promote voting and auditing methods that are incomprehensible by the preponderance of the electorate. C

c See the CHERI website for published papers and reports, including the hardware instruction-set architecture report, the compartmentalization paper, the Thunderclap paper, and so on: https://bit.ly/2RNa5Fb

References
1. Appel, A. et al. The New Jersey Voting-machine Lawsuit and the AVC Advantage DRE Voting Machine, EVT/WOTE'09 (Aug. 2009); https://bit.ly/32tWlRS
2. Bellovin, S.M. and Neumann, P.G. The big picture: A systems-oriented view of trustworthiness. *Commun. ACM 61*, 11 (Nov. 2018); https://bit.ly/3mYVpOR
3. Blaze, M. et al. Voting Machine Hacking Village Report on Cyber Vulnerabilities in U.S. Election Equipment, Databases, and Infrastructure, (Sept. 2017); https://bit.ly/3suT6UH
4. Chaum, D. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy.* (Jan. 2004); https://bit.ly/3eh7nzm
5. Gordon, G., Condon, C. and Dunlap, S. Georgia election officials knew system had 'critical vulnerabilities' before 2016 vote. McClatchy DC Associated Press (Aug. 6, 2018); https://bit.ly/3mZGQKN
6. Joint Statement from Elections Infrastructure Government Coordinating Council and The Election Infrastructure Sector Coordinating Executive Committees (Nov. 12, 2020); https://bit.ly/3v3I9Lo
7. Kocher, P. et al. Spectre Attacks: Exploiting Speculative Execution. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy* (May 2019); https://bit.ly/32qhH2o
8. Markettos, A.T. et al. Thunderclap: Exploring Vulnerabilities in Operating-System (IOMMU) Protection via DMA from Untrustworthy Peripherals, Network and Distributed System Security Symposium. NDSS 2019 (Feb. 24–27, 2019).
9. Mercuri, R. A better ballot box. *IEEE Spectrum* (Oct. 1, 2002); https://bit.ly/3xao8oz
10. Mercuri, R. Uncommon criteria. *Commun. ACM 45*, 1 (Jan. 2002); https://bit.ly/3gjSJtP
11. Monmouth University Poll. More Americans happy about Trump loss than Biden win. Press Release (Nov. 18, 2020); https://bit.ly/3swuuej
12. Rein, L. The IEEE P1583 Voting Machine Standard. *IEEE Internet Computing 8*, 1 (Jan.–Feb. 2004); https://bit.ly/3edEwM9
13. Risk-Limiting Audits Working Group, Risk-Limiting Post-Election Audits: Why and How (Oct. 2012); https://bit.ly/3sr0qRe
14. Saltman, R. Accuracy, Integrity, and Security in Computerized Vote-Tallying. NBS/NIST (Aug. 1, 1988); https://bit.ly/3sx8y2z
15. Weber, S.N. California Secretary of State, Voting Technologies In Use By County, as of 15 October 2020; https://bit.ly/32tPfg0

Rebecca T. Mercuri (mercuri@acm.org) has been researching, writing, and testifying about election system integrity since the late 1980s. She also provides digital investigations and expert testimony on a variety of criminal and civil matters through her company, Notable Software, Inc.

Peter G. Neumann (neumann@csl.sri.com) is Chief Scientist of the SRI International Computer Science Lab, and has moderated the ACM Risks Forum since its beginning in 1985.

The authors are grateful to Steven M. Bellovin, E. John Sebes, Vanessa Teague, and Brent Turner for their constructive comments during the development of this column.

DOI:10.1145/3461468

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
Aversion to Versions

Code needs to run anywhere as long as the necessary dependencies can be resolved.

Dear KV,
Recently I have been trying to piece together a set of software packages that are supposedly intended to work together but seem very fragile. The main source of their fragility comes from how the developers "resolved" the dependencies between packages, libraries, and their own software. Their solution to making all the pieces work together was to encode the version of the library or package into the file system, as in, /opt/pkg-v2.87/lib/.... As you might imagine, this causes no end of trouble for us consuming this software when a library or package is upgraded. I have counted no fewer than 30 locations where this was done. You cannot tell me that this is the right way to handle this particular problem, but these people are paid professionals, and we paid for their software. What would KV do?

Aversion to Versions

Dear Aversion,
The problems of dependency analysis and resolution—as well as versioning—have been with us since the earliest days of the software library, and some of the solutions, such as SAT solvers for package systems, are clever and elegant and mostly work. Build dependencies are usually handled by systems such as `automake` and `autoconf`, which, so long as you never look inside them, are quite useful. If you look inside, you will not only see how the sausage is made so much as how the ingredients were processed and eventually spat back out as a makefile. All of which is to say that these problems are solved, and the solutions are often complex and tortuous, but we all admire those who undertake to solve them.

Then there are those who, either through ignorance or stupidity, decide just to take a stab in the dark and solve the problem in their own, inimitable style. It is definitely these types you are dealing with today. I guess you could file a bug against the software and see if someone fixes it. But given the quality of what they have already given you, I think that is a long shot.

Since you have been able to count the number of these sins committed in software (and you number them at 30), I am assuming you have some amount of the source code; perhaps it was even all delivered as source. One quick and very dirty solution is to use the inimitable sed (stream editor) program[a] to update the version numbers as necessary. A manual page can be found here, but I am sure Stack Exchange or some other cheater site will give you code to "swap version numbers throughout my code" or some such thing. Just slap the code into a repo somewhere, find the right incantation of sed(1), sacrifice a live animal of your choice, and voilà, you will be able to update the versions to match the latest library. Purists will scream this is not the right way to solve the problem, and they are correct, but then purists have plenty of time to do the right thing, while the rest of us are trying to do the thing right.

Of course, the better way—and again you will need the source to do this—is to update the code to actually take a path argument or inquire after some sort of environment variable (MYLIBPATH) that can be used to point the software to the right place, no matter what version you want it to use. If you go this route, be sure to tell the developers you will send them the patch.

The higher-level point here is that one should never hardcode a version or a path inside the code itself. Code needs to be flexible so that it can be installed anywhere (the hardcoding of /usr/local is blatantly foolish and yet persists) and run anywhere so long as the necessary dependencies can be resolved, either at build time for statically compiled code or at runtime for interpreted code or code with dynamically linked libraries. There are, as KV has just pointed out, current, good ways to get this right, so it is a shame so many people continue to get it wrong.

KV

a See https://www.freebsd.org/cgi/man.cgi?query=sed&sektion=&n=1

Related articles on queue.acm.org

Kode Vicious Unleashed
https://queue.acm.org/detail.cfm?id=1046939

Understanding Software Patching
Joseph Dadzie, Microsoft
https://queue.acm.org/detail.cfm?id=1053343

Immutability Changes Everything
Pat Helland
https://queue.acm.org/detail.cfm?id=2884038

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

DOI:10.1145/3461466

Peter J. Denning

The Profession of IT
Locality and Professional Life

The locality principle extends beyond computer memories. It teaches us something about being human.

ONE SUNDAY MORNING nearly three decades ago, my wife Dorothy and I were walking along the Potomac River in Washington, D.C. I was considering a job change and was concerned about whether my new responsibilities would divert me from my aspiration that my work "make a mark." She asked what I meant by making a mark. That meant, I confided, that people would long remember my contribution by name. She said that if that is my philosophy of life, I am likely to be disappointed. She explained her philosophy, in which she does not have that concern. She sees herself as a cell in the large body of humanity past, present, and future. Her life purpose is to be a good cell. She embraces every project with care and excellence—to do the best possible job. In this way she will contribute to the health of the whole and have impact on the whole. It is not her purpose that her name be attached to anything she has contributed. When she is gone, her job is done and other cells will continue to serve the well-being of the whole. I asked her about the awards and recognitions she received for her work. She said she appreciated the honors, but it was never her objective or interest to win awards or be recognized. This conversation forever altered my thinking about the contributions I could make.

Think about this. If each of us is doing our job, being a good cell in the large body of all humanity, we keep our neighboring cells healthy and thereby contribute to the health of the whole. Our contribution flows through to the whole like a ripple in the river of humanity. Over a period of time, the ripple remains but any memory of me as the author is likely to disappear.

I could see her insight firsthand in my walks through the Arlington Cemetery. In every direction, fading into the distance, are lines of headstones, each labeled simply with the name and dates of a soldier. With only a few exceptions, that is all we know about them any more. There would never be an answer to my question, "Who was that soldier?" Yet we know that collectively they made our country safe. Their combined ripples were a torrent etching an indelible mark on history.

Locality and Human Identities

The ideas of strong interactions with neighbors and loss of identity over time sound like locality in computing. Locality is the idea that, as a program executes, it confines its accesses to a small subset of its data for an extended period. In our professional work we do the same. For extended periods, we confine our interactions mainly to people in our immediate teams or communities, a small subset of all the people we could possibly interact with. This restriction to local neighborhoods is the spatial aspect of locality.

IMAGE BY ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

Our contributions begin in our local networks (neighborhoods) and spread out like ripples across many local networks in the conversations people have with people in other networks. Over time, our identity as the origin of ripples disappears. Only in rare cases does a name survive. This dying out of identity is the temporal aspect of locality.

In our profession, we are brought up with stories of great scientists, engineers, and leaders who are held up as role models. We develop desires to leave our own mark, meaning that our contribution, like theirs, is remembered with our name attached. The urge to have fame seems to have grown stronger in recent times as the Internet reveals how tiny each of us is compared to all of humanity. More people recoil from a sense of insignificance by tracking followers in Twitter, likes of their social media posts, or citations of their published papers. Achieving some sort of fame seems to be a way of demonstrating that their life has had meaning and impact. Yet locality tells us that any memory with our name on it is likely to be ephemeral. Most of us have a professional concern to have an impact on the world. The locality principle offers some insights into how to do this.

To drive home the points about the similarities of the development and preservation of our identities I would like to review the idea of locality as it evolved in computer science.

Locality and Computer Memories

Locality grew up in computer science beginning in the middle 1960s. It is the idea that as a program computes, it confines its memory accesses to relatively small localities—subsets of its data objects—for extended periods of time. We design operating systems to arrange the workspace so that the current locality is accessible in nearby local memory. Other items can be farther away. When this is done well, the operating system and all its processes will operate at peak efficiency.

Locality has two aspects, spatial and temporal. The spatial aspect is the localities themselves and possible connections between them. The temporal aspect is that the actions within a locality are good predictors of actions immediately in the future and poor predictors of actions far in the future.

Locality teaches us that being human is to serve our communities and look for reward from our contribution rather than long-term recognition.

The first glimmers of locality were seen by OS engineers trying to figure out how to control virtual memory, which was introduced in 1962 by Tom Kilburn at University of Manchester. Virtual memory was seen as a tremendous breakthrough because, by automating data transfers between main and secondary memory, it doubled or tripled programmer productivity, and it significantly reduced errors resulting from manually planning page moves. The paging algorithm was the virtual memory component that determined what pages to evict from main memory. Early virtual memory was hampered by poor performance, traceable to poor paging algorithms. The first careful scientific study of paging algorithms was published by Les Belady of IBM in 1966. It revealed that paging algorithms that employed "use bits" to protect recently used pages from being evicted performed better than other algorithms. The LRU (least recently used) algorithm emerged as the most robust and the most likely to give good performance. Belady speculated that LRU worked because of a "locality principle"—programs were likely to reuse pages they had used in the recent past.

Contemporaneous with Belady I was studying how an operating system running a virtual memory ought to determine what pages to load—its working set. I had the insight that a program's working set could be measured by observing which pages it used in the immediate past. I defined the working set policy, which loaded process' working sets and protected them from swapping. Unlike the previous paging policies, which were restricted to deciding the contents of the fixed memory space allocated to each process, working set defined how to establish a dynamically varying multiprogramming partition and prevent it from thrashing.

By 1970, locality was accepted as a key determinant for good performance of virtual memory. It was also—mistakenly—seen as an artifact how compilers arranged code and data on pages.

Over the next 40 years designs to take advantage of locality spread well beyond virtual memory. These included caches in CPUs, devices, and Internet; buffers between memory and I/O devices and networks; video cards; information encapsulation in program modules; accounting and event logs; most recently used lists of applications; Web browsers; search engines; video streaming; and edge caches in the Internet. Designs to harness locality are everywhere.[1]

Locality and Human Thinking

The success of locality prompted investigations into what might be causing locality to appear in program behavior. In 1976, Wayne Madison and Alan Batson demonstrated that locality was measurable in source codes of programs. They and many others attributed this to the way we humans think about problem solving. Techniques such as iterative loops, divide and conquer, and modularity all generated locality behavior. Locality is a reflection of human thought.

The algorithm is another reflection of human thought, intended to capture a procedure so that it can be followed by any other person or, in modern cases, by a computing machine. In 2010 Yuri Gurevich published a report seeking to answer the question, "What is an algorithm?"[3] He was trying to find the smallest set of essential assumptions that make a procedure an algorithm. One of his findings was a bounded domain principle: an operation can only alter a finite, bounded region of the data structure. In other words, to qualify as an algorithm, a computational method must necessarily obey a locality principle.

Because machines are also the product of human thought, we might ask if there is a locality principle embedded into the design of computers. There is. Each component of a machine interacts locally with a few other components by receiving inputs and generating outputs. This enables components to be very fast because they can fire without waiting for signals from distant components to propagate through the circuit. This unleashes the amazing speeds that enable computers to do many tasks humans cannot. The locality principle enables computing machines to be fast. It also deprives machines of the human capability to sense context. The brain with its many intricate folds can bring into physical proximity neurons that are neuronally distant from each other. This may be a structural reason why the brain can recognize context but our computing machines cannot.

Not only does locality reflect human thought, it shapes human thought. Consider the brain's working memory, which holds memories of recent events. To be recalled later, short term memories must be moved to the longer-term memory areas of the brain. Working memory is like a computer cache. This structure explains why multitasking is difficult for many people and inhibits their productivity. To switch to a new task, you need to purge the local working memory of the old task and load it for the new.[2,4] This context-switch takes time and introduces errors when fragments of short-term memories are lost in the process. As in a computer, too much context switching causes performance loss because the cache must be purged and reloaded. If the demand for context switches is too high compared to your brain's capacity, your brain can thrash like a computer. You experience this as a state of overwhelm, in which your capacity can be greatly diminished.

Thus it seems that locality is bedrock for everything we know and think about computation. We cannot have algorithms without it. We cannot manage memory well without it. We do not multitask well because of it.

Conclusion

Give the limitations on our brains and interactions with other people, how can we make a difference? Start with our neighbors in our communities. We do not accomplish anything alone. A contribution spreads in the conversations, stories, and practices we share with each other. A contribution becomes greater when we mobilize our communities around it—they adopt its practice and become voices advocating it in new networks. This helps explain why individual names often disappear from contributions—they were actually done by communities.

Locality teaches us that being human is to serve our communities and look for reward from our contribution rather than from long-term recognition. Instead of thinking we are the serfs of technologies that drive our conditions, consider that technologies can enable us to take care of our community's well being and to learn more about the human condition. The idea that technology reveals and enables the human condition is not the current common sense, although it was a commonly held view in previous eras. Perhaps the fears that computers will take away our humanity are overblown. Perhaps our pursuit of more effective human-centered computing—for example with data science, machine learning, and AI—will help us to become more fully human.

We do our best when we focus on what we can do together in our neighborhoods. If we are concerned about credit and recognition of our work propagating through the human network, we are chasing the chimera. Our human identities are mostly local. We have no control over what happens at large distances (in space and time) from where we are in the human network. Locality empowers our neighborhoods. The ripples we create with our neighbors will likely travel far, but our names will not. And that is how we fulfill our purpose. **C**

References
1. Denning, P. The locality principle. *Commun. ACM 48*, 7 (July 2005), 19–24.
2. Denning, P. Multitasking without thrashing. *Commun. ACM 60*, 9 (Sept. 2017), 32–34.
3. Gurevich, Y. What is an algorithm? In *Proceedings of the 38th International Conference on Current Trends in Theory and Practice of Computer Science* (SOFSEM'12). M. Bieliková et al., Eds. Springer-Verlag, Berlin, Heidelberg, (2012), 31–42.
4. Van der Stigchel, S. Dangers of divided attention. *American Scientist 109* (Jan.–Feb. 2021), 46–53.

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.

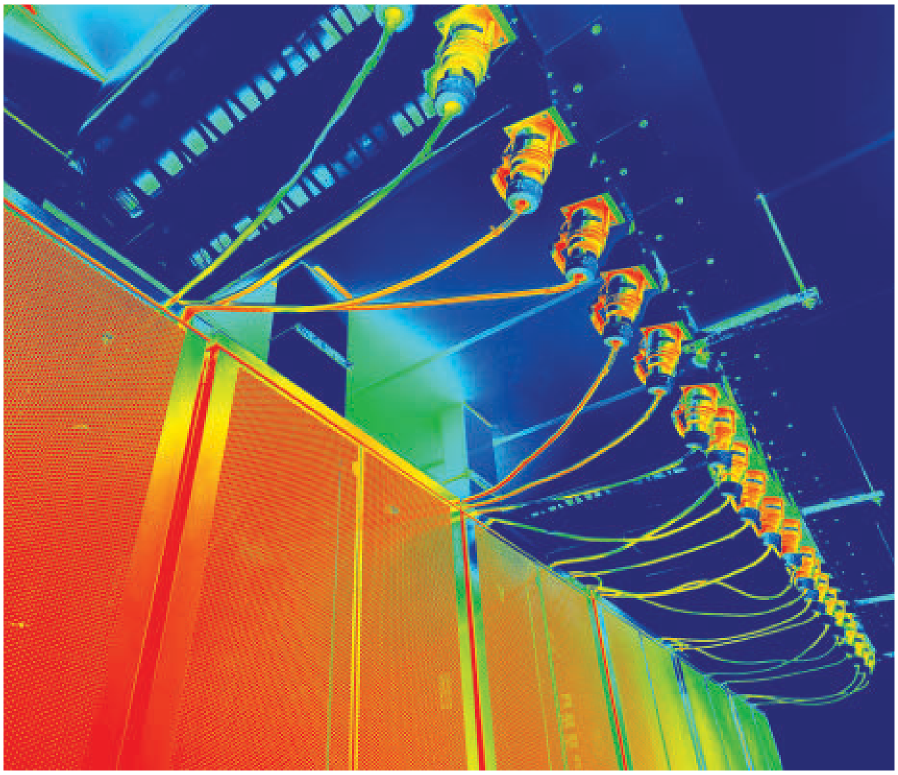
DOI:10.1145/3431282

Todd Hylton, Thomas M. Conte, and Mark D. Hill

Viewpoint

A Vision to Compute Like Nature: Thermodynamically

Advocating a new, physically grounded, computational paradigm centered on thermodynamics and an emerging understanding of using thermodynamics to solve problems.



CLASSICAL COMPUTING USING digital symbols—equivalent to a Turing Machine—is reaching its limits. It is undeniable that computing's historic exponential performance increases have improved the human condition. Yet such increases are a thing of the past due in large part to the constraints of physics and how today's systems are constructed. Hardware device designers struggle to eliminate the effects of nanometer-scale thermodynamic fluctuations, and the soaring cost of fabrication plants has eliminated all but a few companies as a source of future chips. Software developers' ability to imagine and program effective computational abstractions and implementations are clearly challenged in complex domains like economic systems, ecological systems, medicine, social systems, warfare, and autonomous vehicles. Machine learning techniques, such as deep neural networks, can help but their capabilities are limited and they are implemented on top of the digital hardware with the aforementioned challenges.

Yet, even as we encounter these limits, we recognize that living systems evolve energy-efficient, universal, self-healing, and complex computational capabilities that dramatically transcend our current technologies. Animals, plants, bacteria, and proteins solve problems by spontaneously finding energy-efficient configurations that enable them to thrive in complex, resource-constrained environments. For example, proteins fold naturally into a low-energy state in response to their environment.[a] In fact, all matter evolves toward low-energy configurations in accord with the Laws of Thermodynamics. For near-equilibrium systems (see the sidebar) these ideas are well known and have been used extensively in the analysis of computational efficiency and in machine learning techniques.

Since the time of Carnot and Babbage in the early 19th century, the fields of computation and thermodynamics have co-evolved with seminal contributions from Maxwell, Boltzmann, Gibbs, von Neumann, Turing, Shannon and many others.

a Even this relatively simple system is still too compute intensive to model effectively on our most powerful supercomputers—what costs nature a few electronvolts currently costs a few terajoules on a supercomputer.

Lately, we see this trend continuing as our understanding of thermodynamics expands to include non-equilibrium systems. For example, a new generation of "fluctuation theorems" (for example, Jarzynski,[7] Crooks[3]) suggests a path toward a computing technology grounded in an understanding of open, non-equilibrium systems. To continue technological progress—and acquire its corresponding social and economic benefits—we advocate a new, physically grounded, computational paradigm centered on thermodynamics and an emerging understanding of using thermodynamics to solve problems that we call "Thermodynamic Computing" or TC. Like quantum computers, TCs are distinguished by their ability to employ the underlying physics of the computing substrate to accomplish a task.

As illustrated in Figure 1, we envision a thermodynamic computing system (TCS) as a combination of a conventional computing system and novel TC hardware. The conventional computer is a "host" through which users can access the TC and define a problem for the TC to solve. The TC, on the other hand, is an open thermodynamic system directly connected to real-world input potentials (for example, voltages), which drive the adaptation of its internal organization via the transport of charge through it to relieve those potentials. Qualitatively, better or poorer organization results in smaller or larger internal dissipation and fluctuation, which drives the network to stabilize existing configurations or sample new ones. The TC itself may comprise, for example, a network[b] of "cores" and connections that spontaneously and collectively adapt to achieve low energy network states/high charge transport efficiency in response to changing external potentials.

Figure 1. Conceptual schematic for a Thermodynamic Computing system.

The top half of the figure represents a conventional computing system that "hosts" the TC. The host computer is entirely prescribed by humans, who are its interface to the real world. The TC, illustrated in the lower half of the figure, has independent interfaces to raw potential in its environment and complex, multiscale, recurrent adaptive internal evolution that communicate and connect environmental potentials. Humans can direct the evolution of TCs by programming constraints that influence the connections to the environment and the evolution of the system. The TC can also provide feedback to the conventional computing system; for example, it may evolve a representation of its environment that can be used as input. The thermal reservoir plays an active role in the evolution of the TC by providing fluctuations.

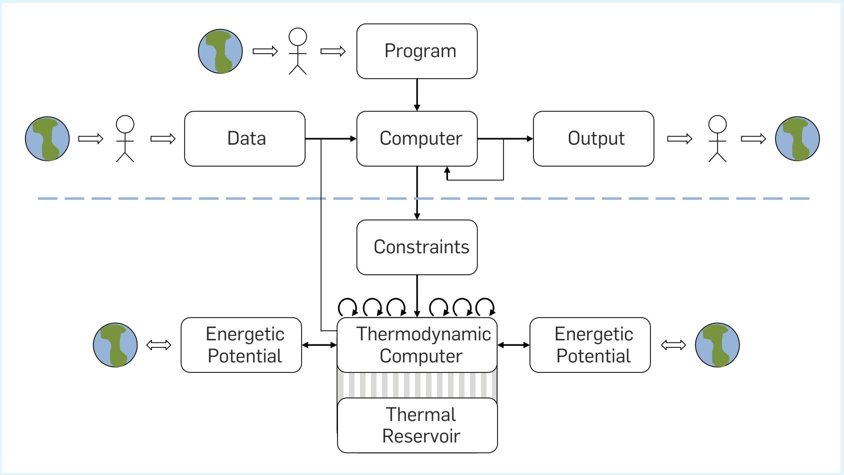
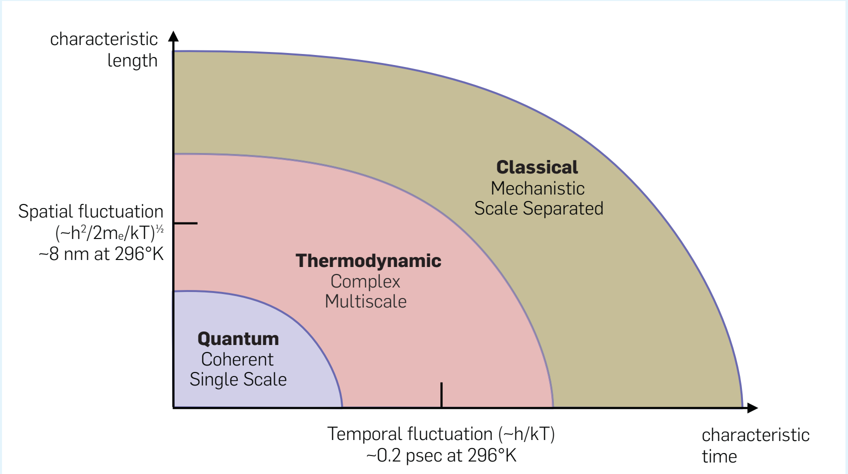


Figure 2. The three major domains of computing.



What problems can these systems solve? We see TC as particularly well-suited for searching complex energy landscapes that leverage both rapid device fluctuations and the ability to search a large space in parallel, and addressing NP-complete combinatorial optimization problems or sampling many-variable probability distributions. For example, Borders et al.[1] have shown the ability to solve optimization problems such as integer factorization using a network of stochastic binary neurons constructed from thermally fluctuating nanoscale magnetic tunnel junctions configured as stochastic bits. Thermodynamic computing systems make such ideas both more general and more accessible, giving a TCS user the ability to easily define optimization problems that receive feedback not only from their programmed constraints, but also from direct interaction with an external environment.

To put this vision in a larger context, Figure 2 divides computing into domains according to their relationship to fluctuation scales. Spatial and temporal fluctuation scales are esti-

b Among existing computing systems, TC is perhaps most similar to neuromorphic computing, except that it replaces rule-driven adaptation and neuro-biological emulation with thermo-physical evolution.

mated in terms of thermal energy (kT, that is, Boltzmann constant times temperature) and corresponding electronic quantum coherence times and lengths. We divide the computing paradigm into three qualitatively different domains that we label as "Classical," "Thermodynamic," and "Quantum." In the Classical domain, fluctuations are small compared to the smallest devices in a computing system (for example, transistors, gates, memory elements), thereby separating the scales of "computation" and "fluctuation" and enabling abstractions such as device state and the mechanization of state transformation that underpin the current computing paradigm. In the Quantum domain, fluctuations in space and time are large compared to the computing system. While the Classical domain avoids fluctuations by "averaging them away," the Quantum domain avoids them by "freezing them out." In the Thermodynamic domain, fluctuations in space and time are comparable to the scale of the computing system and its devices. This is the domain of non-equilibrium, mesoscale thermodynamics, cellular operations, neuronal plasticity, genetic evolution, and so forth—that is, it is the domain of self-organization and the evolution of life. This is the domain that we need to understand in order to build technologies that operate near the thermodynamic limits of efficiency and spontaneously self-organize, but paradoxically it is also the domain that we assiduously avoid in our current classical and quantum computing efforts.

We note that the idea of using the physics of self-organizing electronic or ionic devices to solve computational problems has shown dramatic progress in recent years. For example, networks of oscillators built from devices exhibiting metal-insulator transitions have been shown to solve computational problems in the NP-hard class.[9] Memristive devices have internal state dynamics driven by complex electronic, ionic, and thermodynamic considerations,[4] which, when integrated into networks, result in large-scale complex dynamics that can be employed in applications such as reservoir computing.[10] Other systems of memristive devices have been shown to implement computational models such as Hopfield networks[8] and to build neural networks capable of unsupervised learning. Today we see opportunity to couple these recent experimental results[11] with the new theories of non-equilibrium systems through both existing (for example, Boltzmann Machines[5]) and newer (for example, Thermodynamic Neural Network[6]) model systems. Given these experimental, theoretical, and modeling components, we envision a roadmap for developing TCs with three complementary foci:

- Using classical computing to model and simulate potential TC advances and, conversely, focusing the lens of TC back on classical systems in order to improve them.
- Developing nearer-term hybrid computer systems with both classical and thermodynamically-augmented components—for example, thermodynamic "bits," "neurons," "synapses," "gates," and "noise generators"—and evolving these systems toward greater TC exploitation.
- Creating systems using complex thermodynamics networks wherein a classical computing system provides an interface to and scaffolding for mesoscale assemblies of interacting, self-organizing components exhibiting complex dynamics and multiscale, continuously evolving structure, either at room temperature or—if quantum effects are key—at very low temperature (milli-Kelvin).

At least initially, we expect that TC will enable new computing opportunities rather than replace Classical Computing at what Classical Computing does well (enough), following the disruption path articulated by Christensen.[2] These new opportunities will likely enable orders of magnitude more energy efficiency and the ability to self-organize across scales as an intrinsic part of their operation. These may include self-organizing neuromorphic systems and the simulation of complex physical or biological domains, but the history of technology shows that compelling new applications often emerge after the technology is available.

As TCs will co-exist with classical computing, future research will devel-

What Is Equilibrium Thermodynamics?

Equilibrium thermodynamics is the study of matter and/or energy transfer in systems as they pass from one state of thermodynamic equilibrium to another, where "thermodynamic equilibrium" indicates a state with no unbalanced potentials, or driving forces, between macroscopically distinct parts of the system. An important goal of equilibrium thermodynamics is to determine how the equilibrium state of a given system changes as its surroundings change.

Laws of Thermodynamics:

- **Zeroth Law—If two systems are each in thermal equilibrium with a third, then they are in thermal equilibrium with each other (or 'there is a game');**
- **First Law—Energy cannot be created nor destroyed, but only change forms (or 'you can't win');**
- **Second Law—The entropy of an isolated system not in equilibrium will tend to increase over time, approaching a maximum value at equilibrium (or 'you can't break even'); and,**
- **Third Law—As temperature approaches absolute zero, the entropy of a system approaches a constant minimum (or 'you can't quit the game').**

The idea of using the physics of self-organizing electronic or ionic devices to solve computational problems has shown dramatic progress in recent years.

Thermodynamic computing can extend computing's transformational effect on society well into the 21st century.

op interaction abstractions. For example, the classical host might configure TC elements by viewing them as a fixed edge-weighted directed graph of thermodynamic elements. In the longer term, as more complex thermodynamic devices and networks become available, the TC might be presented as a continuously evolving neural network processor with networks and external inputs defined by the architect.

Our viewpoint is that developing TC can extend computing's transformational effect on society well into the 21st century.

- TC's potential societal impacts include sustaining U.S. computing leadership, improving outcomes in many areas of human enterprise (for example, medicine, business, agriculture, defense, security, and leisure), and new workforce and business opportunities.
- TC's potential technical impacts include enabling ultra-low energy systems (for example, with perceptual capabilities that rival those of animal sensory systems), fundamental increases in battery life, and a potential reduction in the national computer energy consumption.
- TC's potential scientific impacts include probing the fundamental efficiency limits of computing, exploring self-organization to reduce human programming effort, repurposing the extraordinary capabilities of living systems for human-engineered systems, and creating more intellectual synergy among diverse fields in engineering and physical sciences.

This Viewpoint has been developed from the output of the Computing Community Consortium's[c] Thermodynamic Computing workshop that brought together computer scientists, mathematicians, physicists, and computation biologists. To learn more about the Thermodynamic Computing workshop, visit the workshop website[d] or read the workshop report.[e] The authors thank all workshop participants and CCC/CRA staff. C

c The Computing Community Consortium (CCC) is a programmatic committee of the Computing Research Association (CRA). The mission of Computing Research Association's Computing Community Consortium (CCC) is to enable the pursuit of innovative, high-impact computing research that aligns with pressing national and global challenges. Learn more about the CCC here: https://cra.org/ccc/about/

d See https://bit.ly/32tmEre

e See https://bit.ly/3tAq0V7

References

1. Borders, W.A. et al. Integer factorization using stochastic magnetic tunnel junctions. *Nature 573*, 7774 (2019), 390–393.
2. Christensen, C.M. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail.* Harvard Business Review Press, 2013
3. Crooks, G.E. Entropy production fluctuation theorem and the nonequilibrium work relation for free energy differences. *Physical Review. E, Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics 60*, 3 (1999), 2721–2726.
4. Dittmann, R. and Strachan, J.P. Redox-based memristive devices for new computing paradigm. *APL Materials 7*, 11 (2019).
5. Hinton, G.E. et al. Learning and relearning in Boltzmann Machines. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1*, 282–317 (1986), 2.
6. Hylton, T. Thermodynamic neural network. *Entropy 22*, 3 (2020), 256-279.
7. Jarzynski, C. Nonequilibrium equality for free energy differences. *Physical Review Letters 78*, 14 (1997), 2690–2693.
8. Kumar, S., Strachan, J.P. and Williams, R.S. Chaotic dynamics in nanoscale NbO2 mott memristors for analogue computing. *Nature 548*, 7667 (2017), 318–321.
9. Raychowdhury, A. et al. Computing with networks of oscillatory dynamical systems. In *Proceedings of the IEEE 107*, 1 (2018), 73–89.
10. Sillin, H.O. et al. A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology 24*, 38 (2013).
11. Wang, Z. et al. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics 1*, 2 (2018), 137–145.

Todd Hylton (thylton@ucsd.edu) is Professor of Practice in Mechanical and Aerospace Engineering at UC San Diego, San Diego, CA, USA.

Thomas M. Conte (tom@conte.us) is Professor joint appointed in the Schools of CS and ECE at the Georgia Institute of Technology, Atlanta, GA, USA.

Mark D. Hill (markhill@cs.wisc.edu) is a Partner Hardware Architect at Microsoft and the Gene M. Amdahl and John P. Morgridge Professor Emeritus of Computer Sciences at the University of Wisconsin-Madison, WI, USA.

Mark D. Hill performed this work prior to joining Microsoft; the views expressed in this Viewpoint represent only those of the coauthors.

DOI:10.1145/3437255

Danfeng (Daphne) Yao

Viewpoint

Depth and Persistence: What Researchers Need to Know About Impostor Syndrome

Understanding impostor syndrome's complexity and its effect on research persistence.

IMPOSTOR SYNDROME IS the unfortunate psychological state where one—who could be rather successful careerwise—feels like a fraud and feels that he or she does not fit in. The phenomenon was first reported by Clance and Imes in 1978 after studying more than 150 high-achieving women.[4] Some suggest that a more accurate name should be self-underappreciation phenomenon or self-depreciation.[7] Unlike devastating physical states such as cancer, impostor syndrome is much more benign. It cannot kill anyone. What it can kill, however, is one's career. Impostor syndrome sabotages it, silently and internally, appearing innocuous and almost trivial. For researchers, impostor syndrome can be deadly. It can sap the energy out of sufferers and erode their scientific pursuits and career dreams. Victims may not even realize they have been suffering from this career-ending cancer, before they lose their zest for work, their passion for research, and stop trying.

I feel extremely compelled to share my deeply intimate research story. It is my obligation as an educator, as a senior female researcher of the community—but most importantly—as someone who now sees the other side and realizes how badly impostor syndrome was holding me back. I am left with no choice.

Public Key Infrastructure

On a cold spring day, I was in a research meeting with a friendly collaborator from Sun Microsystems. I had not published any papers, but was making promising progress on an identity-based encryption project and an access-control project. I had also passed the dreaded week-long programming Ph.D. qualifier exam with flying colors. My collaborator, whom I met for the first time in that meeting, suddenly asked if I was familiar with public key infrastructure (PKI). Caught off guard, I quickly shook my head and said "no" with an awkward smile. But I knew PKI, quite well actually. I did not know why I played dumb.

Fast forward to 2014, I was tenured after moving to a different university in 2009 to solve a two-body problem. I was burned out. Luckily, I stepped out of a mild form of post-tenure depression[14] fairly quickly. But I could not help noticing that I consistently did horribly in most of my TV, news, and radio station interviews for my work. I could not look like a bigger fraud in a high-profile 2017 interview. Understandably, CBS never aired that segment. Much later, I learned that the brains of individuals with impostor syndrome tend to overreact to potential threat signals, which inhibits prefrontal-cortex-based brain pathways—a phenomenon referred to as amygdala hijack.[7] Plainly speaking, when I freaked out, I could not think, which happened quite often. Intellectual inauthenticity—downplaying of knowledge, skills, or abilities—is another symptom of impostor syndrome,[13] which explained my earlier PKI episode.

Depth vs. Volume

When Barbara Liskov gave her inspiring Turing lecture, "The Power of Abstraction"[11] at our department, I asked her at the end of our group lunch—how important the volume of work is. The field of computer science has many different successful styles. I was particularly confused by the pure-number-driven assessment approaches (a.k.a. bean counting), which force researchers to simply pump up the volume of papers and grants. Infinity, of course, is the ultimate winner.

In a stern voice, Barbara told me it is all about depth. Being able to work toward a direction and continuously design better solutions to address bigger challenges is much more important than the volume. Gaining research experiences from exploring other problems is useful, which helps one identify the most interesting and suitable area. But ultimately, what matters is depth.

However, pursuing research depth requires a tremendous amount of courage, persistence, and faith. I once did a number of short-lived ad hoc projects, because deep down I was afraid: afraid of committing to my own vision and afraid of asking my students to commit to my vision. Feeling like a fraud also made my research vision blurrier and blurrier.

During her talk, Barbara also calmly showed a black-and-white photo of her sitting in front of a computer wearing an elegant long dress, in response to an earlier question during her visit ("What was it like when you were young?"). First of all, it was a very odd question. But I was more surprised by Barbara Liskov showing her dress in a Turing lecture. Would that not make her appear unprofessional in this male-dominated technical field?

Rock Bottom

One day during my much-needed sabbatical in San Diego, my 9-year-old daughter and I walked to the Torrey Hills Park on a beautiful Southern California afternoon. While my daughter was pounding a handball against the outer wall of a public restroom, I vividly remember feeling like a loser, a big one. I was in a trance, standing under a tree, staring at spotty shadows of leaves on the ground.

My sense of failure was profound and overwhelming. Even though I had many important accomplishments (three U.S. patents on system anomaly detection and document integrity; DARPA, ONR, ARO, and NSF projects; degrees from Peking University, Princeton, Indiana, and Brown; multiple best paper awards, NSF CAREER, and ARO YIP; technical news about my work; and many cybersecurity publications in respectable venues), at that moment, I felt that I knew nothing; I had done nothing useful, nothing that mattered.

People struggling with impostor syndrome do not like to talk about how they feel,[1] as they are ashamed by their self-perceived incompetence. I thought of quitting. I thought of becoming a full-time stay-at-home mom and wife. While I still managed to get papers published and proposals funded, I gradually lost my internal flame as a researcher.

In that brief yet extremely dark period of my career, I lost faith in myself as a researcher. I had no vision about what I could become, how I could contribute to the technological world, and whether or not I had the strength to continue to invent and create. No matter how I compared my work with others in the field, I fell short. I was very certain that I had zero strength in research. I was ready to give up. What is the point of continuing as a researcher?

In a rare circumstance, I was pressed

by an extremely sharp female professor from another institution about the specific plans I could think of to improve their cybersecurity program. “Daphne, what should we do?,” she asked anxiously. That question snapped me out of the impostor syndrome fog. Wait, people need me? I have value as a researcher?

The Chicken vs. Egg Problem

Does the experiment not working happen first? Or, does the scientist not believing it would work happen first? If the scientist does not do anything, then it inevitably results in the experiment not working. Thus, the person starts it.

However, the following scenario is also possible. The scientist might have seen some negative indications, for example, someone with authority told the scientist this experiment will fail, or someone repetitively told the scientist he or she is incompetent. Then, the person draws the conclusion that the experiment will never work and it would be both futile and unwise to make any attempts. In that scenario, the environment is the culprit.

Numerous episodes contributed to me feeling like a fraud. A Ph.D. student asked me to change her thesis topic the day after I enthusiastically described my vision, because her husband thought it was obsolete. Another Ph.D. student decided not to join my group, because her cousin told her the thesis topic I chose for her will make her jobless after graduation. A senior Ph.D. student refused to revise our rejected manuscript and demanded to work with a new male professor. A star student of mine with—what I thought and still strongly believe—an outstanding publication record adamantly decided not to apply for any faculty positions, because he thought his publication record with me was too weak.

For many years, people advised me to smile less and to look serious and authoritative. I guess eventually I did manage to achieve that, as my family now urges me to smile more. I can never get it right on the smile spectrum!

Know Your Research Strengths

Overcoming impostor syndrome unlocked my full research potential. I now take a lot more risks in research, specifically in selecting problems. I enjoy working much more. I speak up whenever I deem necessary. I think much more clearly when being interviewed or criticized. My research passion meter now reads fierce, what it should be. There will always be people who think I am weak at research. It is impossible to please everyone. Just like in cybersecurity, it is impossible to achieve perfect security and there will always be attacks that evade detection.

Feeling like a fraud or an outsider has a detrimental long-term effect on research, regardless of one’s gender.

Know Your Strengths

Not truly recognizing one’s strengths and achievements is not modest. It is stupidity. It denies one from fully capitalizing those strengths. For years, I did not realize that my styles of designing algorithms, conducting quantitative measurements, and developing generalizations and abstractions were my research strengths. My tastes are understandably different, because of my unique training in both natural science and computing. For quite some time, I thought of them as my weaknesses, which I tried to get rid of—unsuccessfully. I felt ashamed of them. I felt ashamed of me always thinking differently, of me always asking my students to work hard on things that mainstream cybersecurity reviewers do not appreciate. Those feelings all stopped when I started to embrace who I am.

I take time to celebrate the research achievements of myself, my students, and my collaborators. Instead of saying “Boy, I got lucky this time,” I discuss our research strategies that led to the successful outcomes, as well as mistakes made. Celebrating every achievement helps solidify the process of knowing your research styles and strengths. Embracing your own research strengths, like embracing one’s strengths (such as kindness) as a human (for example, Clance,[4] Hunt[7]), needs practice.

I also understand now why Barbara Liskov showed that dress picture—one’s outfit has nothing to do with one’s research competence!

In the meantime, I make efforts to improve my weaknesses. I do not recommend anyone to hide their weakness, if it is a critical research skill. Also, do not be intimidated by others just because they seem assertive.[22]

Self Help vs. Support System

Earlier studies proposed psychotherapy treatment, focusing on correcting the deeply flawed thought patterns of impostor syndrome sufferers.[4,10] However, I occasionally found myself at the receiving end of impostor-syndrome-inducing microaggression or sexual harassment that caused me to feel like I did not belong. I had to let some minor ones slide, in order to stay fiercely focused on research. I could not control what others said or did, but I could try consciously creating a supportive environment for myself.

Growing a thick skin is insufficient. Lean in alone is not enough. We do not know whether the chicken or the egg comes first, whether the person gives up first or the computing culture—being imperfect as it is[2,3]—pushes the person to quit. Therefore, we need to improve both the chicken and the egg, both the individual and the system.

Raising the awareness of impostor syndrome and creating an honest, open, and judgment-free environment for people to share experiences are excellent starting points. As an executive committee member of ACM SIGSAC, I organize multiple inclusive excellence programs,[8,19,20] where impostor syndrome is a frequently discussed topic. I actively share recordings of my lectures on impostor syndrome.[21] We welcome all people to attend these events, regardless of their gender and race. Great minds do not think alike.

Impostor-Syndrome-Inducing Sexual Harassment

What do one-off hand rubbing, leg touching, thigh grabbing, objectification remarks, or second-class status comments have anything to do with research? Speaking from experiences, sexual harassments of all forms significantly dampen, if not completely kill, the research flame of victims. Sexual harassment incidents are well documented (for example, in scientific research[12,15] and in Silicon Valley.[2] Oftentimes, victims choose not to report for fear of workplace retaliation, which widely exists.[3,17] They simply quit, dropping out of graduate school, avoiding meetings where perpetrators may attend, not going to conferences or professional social events. Needless to say, the person's research passion meter gradually points to the dropout zone.

Organizational climate for sexual harassment—a perceived organizational-level tolerance of sexual harassment[6]—is the most potent predictor of sexual harassment.[12,18] I am happy to see that ACM SIGSAC and IEEE Security & Privacy TC recently started to require sponsored conferences to have code of conduct on sexual harassment. I strongly recommend all leaders to read the 2018 landmark report by the National Academies of Sciences, Engineering, and Medicine on sexual harassment[12] and begin understanding the extent of devastation.

Conclusion

Feeling like a fraud or an outsider has a detrimental long-term effect on research, regardless of one's gender. For researchers who are prone to impostor syndrome, the biggest takeaway is to identify and reduce impostor-syndrome-inducing factors. May it be your own thinking habits, may it be the society's implicit bias against you, may it be weekly microaggressions, or may it be sexual harassment. Know that these seemingly unrelated factors can seriously damage one's research career.

How do you know you have it? When self-doubt keeps you from pursuing new opportunities, you are probably experiencing impostor syndrome.[7] But do not be afraid—Maria Klawe has it too.[9]

For community, organization, and research leaders, educate yourself about impostor syndrome too. Be aware of the negative impacts and provide the necessary changes and support for potentially vulnerable underrepresented groups. As our computing community starts to understand the impact of impostor syndrome on undergraduate students (for example, Rosenstein[16]), let's also begin to discuss its broad impact on research. C

References

1. Bravata, D.M. et al. Prevalence, predictors, and treatment of impostor syndrome: A systematic review. *J Gen Intern Med. 35*, 4 (Apr. 2010), 1252–1275.
2. Chang, E. Brotopia: Breaking Up the Boys's Club of Silicon Valley. Penguin Random House. 2019.
3. Cheryan, S. et al. Why are some STEM fields more gender balanced than others? *Psychological Bulletin 143*, 1 (2017).
4. Clance, P.R. and Imes, S. The imposter phenomenon in high achieving women: Dynamics and therapeutic intervention. *Psychotherapy Theory, Research and Practice 15*, 3 (1978).
5. Hart, C.G. The penalties for self-reporting sexual harassment. *Gender & Society. 33*, 4 (2019).
6. Hulin, C.L., Fitzgerald, L.F. and Drasgow, F. Organizational Influences on Sexual Harassment. In *Sexual Harassment in the Workplace: Perspectives, Frontiers, and Response Strategies.* Sage Publications. 1996.
7. Hunt, J. *Unlocking Your Authentic Self: Overcoming Impostor Syndrome, Enhancing Self-Confidence, and Banishing Self-Doubt.* 2020.
8. Individualized Cybersecurity Research Mentoring (iMentor) Workshop. Nov. 2020; https://bit.ly/3ar68wf
9. Klawe, M. Impostoritis: A Lifelong, but Treatable, Condition; https://bit.ly/3v79mga
10. Langford, J. and Clance, P.R. The imposter phenomenon: Recent research findings regarding dynamics, personality and family patterns and their implications for treatment. *Psychotherapy: Theory, Research, Practice, Training 30*, 3 (1993).
11. Liskov, B. The Power of Abstraction. ACM A.M. Turing Award Lecture. 2009; https://bit.ly/3vjiC12
12. National Academies of Sciences, Engineering, and Medicine. *Sexual Harassment of Women: Climate, Culture, and Consequences in Academic Sciences, Engineering, and Medicine.* The National Academies Press. 2018.
13. Orbé-Austin, L. and Orbé-Austin, R. *Own Your Greatness: Overcome Impostor Syndrome, Beat Self-Doubt, and Succeed in Life.* Ulysses Press. 2020.
14. Perlmutter, D.D. Avoiding PTDS: Post-Tenure Depression Syndrome. *The Chronicle of Higher Education.* (Feb. 2, 2015); https://bit.ly/3an3ebK
15. Picture a Scientist. Documentary. Directed by Ian Cheney and Sharon Shattuck. 2020; https://bit.ly/3n5nqnY
16. Rosenstein, A., Raghu, A., and Porter, L. Identifying the Prevalence of the Impostor Phenomenon Among Computer Science Students. ACM Technical Symposium on Computer Science Education (SIGCSE), Mar. 2020, Portland, OR, USA.
17. Tucker, J. and Mondino, J. Coming Forward: Key Trends and Data. TIME'S UP Legal Defense Fund. (Oct. 2020).; https://bit.ly/3dvCTdC
18. Willness, C., Steel, P. and Lee, K. A meta-analysis of the antecedents and consequences of workplace sexual harassment. *Personnel Psychology 60*, 1 (2007).
19. Women in Cybersecurity Research (CyberW) Workshop e-meeting. Mar. 2020; https://bit.ly/3u0Lx9P
20. Women in Cybersecurity Research (CyberW) Workshop, co-located with ACM CCS. Dallas, TX. Oct. 2017; https://bit.ly/3aIGinD
21. Yao, D. Research style diversity, impostor syndrome, and know your strengths. ACM CCS iMentor Workshop. Nov. 2020; https://bit.ly/32RyxaA
22. Young, V. *The Secret Thoughts of Successful Women: Why Capable People Suffer from the Impostor Syndrome and How to Thrive in Spite of It.* Crown Publishing Group. 2011.

Danfeng (Daphne) Yao (danfeng@vt.edu) is a Professor, Elizabeth and James E. Turner Jr. '56 Faculty Fellow, and CACI Faculty Fellow in the Department of Computer Science at Virginia Tech in Blacksburg, VA, USA.

DOI:10.1145/3429776

Michael L. Littman

Viewpoint
Collusion Rings Threaten the Integrity of Computer Science Research

Experiences discovering attempts to subvert the peer-review process.

THE DISCIPLINE OF computer science has historically made effective use of peer-reviewed conference publications as an important mechanism for disseminating timely and impactful research results. Recent attempts to "game" the reviewing system could undermine this mechanism, damaging our ability to share research effectively.

I want to alert the community to a growing problem that attacks the fundamental assumptions that the review process has depended upon. My hope is that exposing the behavior of a community of unethical individuals will encourage others to exert social pressure that will help bring colluders into line, invite a broader set of people to engage in problem solving, and provide some encouragement for people trapped into collusion by more senior researchers to extricate themselves and make common cause with the rest of the community. My motivation for writing this Viewpoint is because I became aware of an example in the computer-architecture community where a junior researcher may have taken his own life instead of continuing to engage in a possible collusion ring.[a]

a See https://bit.ly/3duc9tY

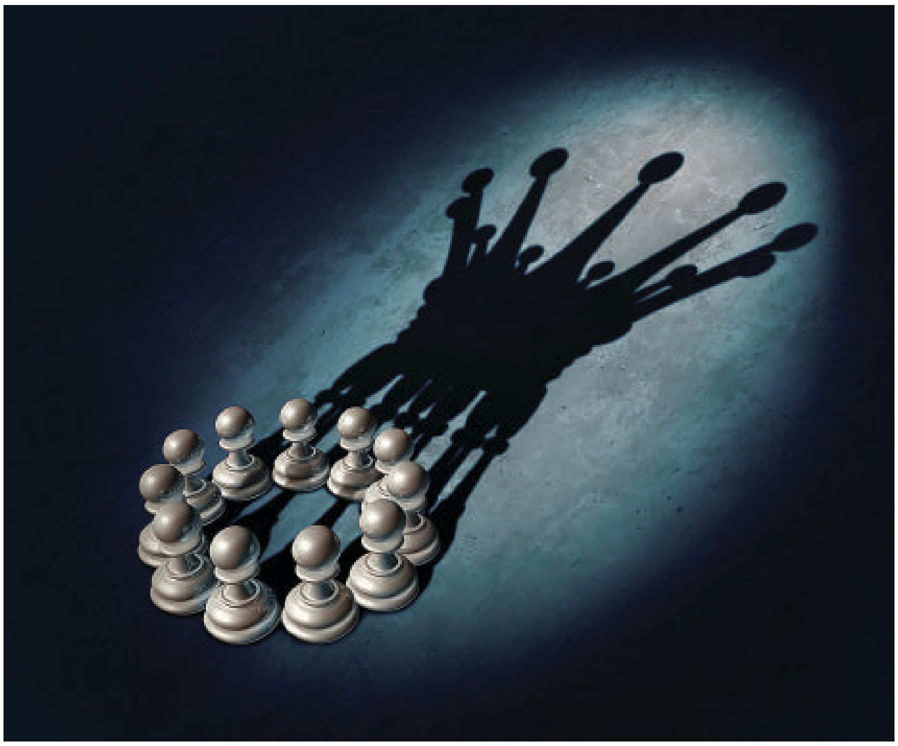


IMAGE BY LIGHTSPRING

Collusion rings extend far beyond the field of computer architecture. I will share another data point, from artificial intelligence and machine learning. I will keep some of the details (like the identity of the specific conference) vague because I think naming names could do more harm than good. Since my goal is to raise awareness of the issue and help people understand how widespread it is, I do not think such details are essential.

Let me start with a reminder about several salient attributes of the review process. What I describe is not precisely what is used by any specific conference but it matches well with the three or four big conferences I have been involved in organizing.

▸ The peer-review process is carried out by a program committee consisting of one or two program chairs, several-hundred area chairs, and approximately 5,000 reviewers. Reviewers are asked to declare conflicts of interest so they are not assigned to review papers that would compromise their partiality.

▸ Authors submit papers with their names withheld for reviewing ("blind"). One notable conference received 10,000 submissions last year, up from an all-time high of 1,000 only six years earlier.

▸ Reviewers "bid" on specific submitted papers based on the paper titles/abstracts to indicate those they are qualified to review.

▸ Reviewers are assigned papers by the program chair(s), attempting to respect their bids while avoiding disclosed conflicts of interest.

▸ Reviewers read their assigned papers and submit reviews. They share their reviews with one another and try to reach a consensus recommendation (accept/reject) for each paper, which the area chairs and program chairs use to build the conference's technical program.

Overall, stakes are high because acceptance rates are low (15%–25%), opportunities for publishing at any given conference are limited to once a year, and publications play a central role in building a researcher's reputation and ultimate professional success. Academic positions are highly competitive, so each paper rejection—especially for graduate students—has a real impact on future job prospects. Some countries correlate promotion and salary decisions to the number of papers accepted at a specific set of high-profile conferences (and journals).

Given the intensity of the process, researchers push themselves very hard to do the best work that they can. The week or two leading up to a conference deadline is exceptionally stressful, with researchers neglecting other responsibilities, running their computers at capacity, and getting very little sleep. Even so, hard work does not appear to be enough to guarantee success—the review process is notoriously random. In a well-publicized case in 2014, organizers of the Neural Information Processing Systems Conference formed two independent program committees and had 10% of submissions reviewed by both. The result was that almost 60% of papers accepted by one program committee were rejected by the other, suggesting that the fate of many papers is determined by the specifics of the reviewers selected and not just the inherent value of the work itself.

Without better investigative tools, we may never be able to hold the colluders to account.

In response, some authors have adopted paper-quality-independent interventions to increase their odds of getting papers accepted. That is, they are cheating.

Here is an account of one type of cheating that I am aware of: a collusion ring. Although the details of this particular case have not been publicly disclosed, the program chairs who discovered and documented the behavior spent countless hours on their analysis. The issues are complicated, but I have no reason to doubt their conclusions. Here is how a collusion ring works:

▸ A group of colluding authors writes and submits papers to the conference.

▸ The colluders share, amongst themselves, the titles of each other's papers, violating the tenet of blind reviewing and creating a significant undisclosed conflict of interest.

▸ The colluders hide conflicts of interest, then bid to review these papers, sometimes from duplicate accounts, in an attempt to be assigned to these papers as reviewers.

▸ The colluders write very positive reviews of these papers, perhaps even lobbying area chairs through back channels outside the view of the other reviewers.

▸ Colluders occasionally send threatening email messages to non-colluding reviewers if the colluders discover their names and believe the non-colluding reviewers can be influenced.

▸ Some colluding reviewers temporarily change their names on the online conference management system during the discussion process, perhaps to avoid getting a reputation for supporting weak papers.

The outcome of this attack, if undetected and successful, is that some authors are rewarded with paper acceptances for very unethical behavior. Given that many conferences have to cap the number of accepted papers due to limits on the number of papers that can be presented at the conference, that means other deserving papers are being rejected to make room. The quality, and perhaps even more importantly, the overall integrity, of the conference suffers as a result.

The research community must respond forcefully to collusion rings, sending a clear message to misbehaving authors and reviewers that what they are doing is unacceptable. Beyond unambiguous messaging, however, it is not yet clear what interventions should be adopted to squelch collusion rings. Conference organizers behind the scenes are weighing dozens of proposals, all of which have potential pitfalls. Better paper-assignment technology would help close one loophole that is being exploited. But, without better investigative tools, we may never be able to hold the colluders to account.

Scientific research is a deeply cooperative endeavor. Researchers compete for attention and funding resources, but also build their ideas on top of those of their rivals. Most researchers see their work as a quest for deeper understanding, not just a way to pay the bills. At present, the peer-review process consists largely of honest participants. But, once unethical behaviors are sufficiently widespread, the incentives for continuing to engage in a community of discovery evaporate. The cheaters run the risk of destroying the very system they depend on for their professional success. It is time to take a close look at the peer-review process and to align the incentives so everyone is working toward sharing the best research work possible. C

Michael L. Littman (mlittman@cs.brown.edu) is The Royce Family Professor of Teaching Excellence in Computer Science at Brown University in Providence, RI, USA.

Publish Your Work Open Access With ACM!

ACM offers a variety of Open Access publishing options to ensure that your work is disseminated to the widest possible readership of computer scientists around the world.

Please visit ACM's website to learn more about ACM's innovative approach to Open Access at: https://www.acm.org/openaccess

acm Association for Computing Machinery

DOI:10.1145/3453928

Article development led by acmqueue
queue.acm.org

There's more to it than you think.

BY NICOLE FORSGREN, MARGARET-ANNE STOREY, CHANDRA MADDILA, THOMAS ZIMMERMANN, BRIAN HOUCK, AND JENNA BUTLER

The SPACE of Developer Productivity

DEVELOPER PRODUCTIVITY IS complex and nuanced, with important implications for software development teams. A clear understanding of defining, measuring, and predicting developer productivity could provide organizations, managers, and developers with the ability to make higher-quality software—and make it more efficiently.

Developer productivity has been studied extensively. Unfortunately, after decades of research and practical development experience, knowing how to measure productivity or even define developer productivity has remained elusive, while myths about the topic are common. Far too often teams or managers attempt to measure developer productivity with simple metrics, attempting to capture it all with "one metric that matters."

One important measure of productivity is personal perception;[1] this may resonate with those who claim to be in "a flow" on productive days.

There is also agreement that developer productivity is necessary not just to improve engineering outcomes, but also to ensure the well-being and satisfaction of developers, as productivity and satisfaction are intricately connected.[12,20]

Ensuring the efficient development of software systems and the well-being of developers has never been more important as the Covid-19 pandemic has forced the majority of software developers worldwide to work from home,[17] disconnecting developers and managers from their usual workplaces and teams. Although this was unexpected and unfortunate, this change constitutes a rare "natural experiment" that statisticians can capitalize upon to study, compare, and understand developer productivity across many different contexts. This forced disruption and the future transition to hybrid remote/colocated work expedites the need to understand developer productivity and well-being, with wide agreement that doing so in an efficient and fair way is critical.

This article explicates several common myths and misconceptions about developer productivity. *The most important takeaway from exposing these myths is that productivity cannot be reduced to a single dimension (or metric!).* The prevalence of these myths and the need to bust them motivated our work to develop a practical multidimensional framework, because only by examining a constellation of metrics in tension can we understand and influence developer productivity. This framework, called SPACE, captures the most important dimensions of developer productivity: satisfaction and well-being; performance; activity; communication and collaboration; and efficiency and flow. By recognizing and measuring productivity with more than just a single dimension, teams and organizations can better understand how people and teams work, and they can make better decisions.

The article demonstrates how this framework can be used to understand productivity in practice and why using

IMAGE BY NATTAKORN MANEERAT

it will help teams better understand developer productivity, create better measures to inform their work and teams, and may positively impact engineering outcomes and developer well-being.

Myths and Misconceptions About Developer Productivity

A number of myths about developer productivity have accumulated over the years. Awareness of these misconceptions leads to a better understanding of measuring productivity.

Myth: Productivity is all about developer activity. This is one of the most common myths, and it can cause undesirable outcomes and developer dissatisfaction. Sometimes, higher volumes of activity appear for various reasons: working longer hours may signal developers having to "brute-force" work to overcome bad systems or poor planning to meet a predefined release schedule. On the other hand, increased activity may reflect better engineering systems, providing developers with the tools they need to do their jobs effectively, or better collaboration and communication with team members in unblocking their changes and code reviews.

Activity metrics alone do not reveal which of these is the case, so they should never be used in isolation either to reward or to penalize developers. Even straightforward metrics such as number of pull requests, commits, or code reviews are prone to errors because of gaps in data and measurement errors, and systems that report these metrics will miss the benefits of collaboration seen in peer programming or brainstorming. Finally, developers often flex their hours to meet deadlines, making certain activity measures difficult to rely on in assessing productivity.

Myth: Productivity is only about individual performance. While individual performance is important, contributing to the success of the team is also critical to measuring productivity. Measures of performance that balance the developer, team, and organization are important. Similar to team sports, success is judged both by a player's personal performance as well as the success of their team. A developer who optimizes only for their own personal productivity may hurt the productivity of the team. More team-focused activities such as code reviews, on-call rotations, and developing and managing engineering systems help maintain the quality of the code base and the product/service. Finding the right balance in optimizing for individual, team, and organizational productivity, as well as understanding possible trade-offs, is key.

Myth: One productivity metric can tell us everything. One common myth about developer productivity is that it produces a universal metric, and that this "one metric that matters" can be used to score teams on their overall work and to compare teams across an organization and even an industry. This isn't true. Productivity represents several important dimensions of work and is greatly influenced by the context in which the work is done.

Myth: Productivity measures are useful only for managers. Developers often say that productivity measures aren't useful. This may come from the misuse of measures by leaders or managers, and it's true that when productivity is poorly measured and implemented, it can lead to inappropriate usage in organizations. It's disappointing that productivity has been co-opted this way, but it's important to note that developers have found value in tracking their own productivity—both for personal reasons and for communicating with others.

By remembering that developer productivity is personal,[7] developers can leverage it to gain insights into their work so they can take control of their time, energy, and days. For example, research has shown that high productivity is highly correlated with feeling satisfied and happy with work.[12,20] Finding ways to improve productivity is also about finding ways to introduce more joy, and decrease frustration, in a developer's day.

Myth: Productivity is only about engineering systems and developer tools. While developer tools and workflows have a large impact on developer productivity, human factors such as environment and work culture have substantial impact too. Often the critical work needed to keep the environment and culture healthy can be "invisible" to many members of the organization or to metrics traditionally used for measuring productivity. Work such as morale building, mentoring, and knowledge sharing are all critical to supporting a productive work environment and yet are often not measured. The "invisible" work that benefits the overall productivity of the team is just as important as other more commonly measured dimensions.[21]

SPACE: A Framework for Understanding Developer Productivity

Productivity is about more than the individual or the engineering systems; it cannot be measured by a single metric or activity data alone; and it isn't something that only managers care about. The SPACE framework was developed to capture different dimensions of productivity because without it, the myths just presented will persist. The framework provides a way to think rationally about productivity in a much bigger space and to choose metrics carefully in a way that reveals not only what those metrics mean, but also what their limitations are if used alone or in the wrong context.

Satisfaction and well-being. *Satisfaction* is how fulfilled developers feel with their work, team, tools, or culture; *well-being* is how healthy and happy they are, and how their work impacts it. Measuring satisfaction and well-being can be beneficial for understanding productivity[20] and perhaps even for predicting it.[15] For example, **productivity and satisfaction** are correlated, and it is possible that satisfaction could serve as a leading indicator for productivity; a decline in satisfaction and engagement could signal upcoming burnout and reduced productivity.[13]

For example, when many places shifted to mandatory work from home during the pandemic, an uptick occurred in some measures of productivity (for example, code commits and speed to merge pull requests).[8] Qualitative data, however, has shown that some people were struggling with their well-being.[3] This highlights the importance of balanced measures that capture several aspects of productivity: While some activity measures looked positive, additional measures of satisfaction painted a more holistic picture, showing that productivity is personal, and some developers were approaching burnout. To combat this, some software groups in large organizations implemented "mental health" days—essentially, free days off to help people avoid burnout and improve well-being.

It is clear that satisfaction and well-being are important dimensions of productivity. These qualities are often best captured with surveys. To assess the satisfaction dimension, you might measure the following:

- *Employee satisfaction.* The degree of satisfaction among employees, and whether they would recommend their team to others.
- *Developer efficacy.* Whether developers have the tools and resources they need to get their work done.
- *Burnout.* Exhaustion caused by excessive and prolonged workplace stress.

Performance is the outcome of a system or process. The performance of software developers is hard to quantify, because it can be difficult to tie individual contributions directly to product outcomes. A developer who produces a large amount of code may not be producing high-quality code. High-quality code may not deliver customer value. Features that delight customers may not always result in positive business outcomes. Even if a particular developer's contribution can be tied to business outcomes, it is not always a reflection of performance since the developer may have been assigned a less impactful task, instead of having agency to choose more impactful work. Furthermore, software is often the sum of many developers' contributions, exacerbating the difficulty in evaluating the performance of any individual developer. In many companies and organizations, software is written by teams, not individuals.

For these reasons, **performance is often best evaluated as *outcomes* instead of *output*.** The most simplified view of software developer performance could be, Did the code written by the developer reliably do what it was supposed to do? Example metrics to capture the performance dimension include:

- *Quality.* Reliability, absence of bugs, ongoing service health.
- *Impact.* Customer satisfaction, customer adoption and retention, feature usage, cost reduction.

Activity is a count of actions or outputs completed in the course of performing work. Developer activity, if measured correctly, can provide valuable but limited insights about developer productivity, engineering systems, and team efficiency. Because of the complex and diverse activities that developers perform, their activity is not easy to measure or quantify. In fact, it is *almost impossible to comprehensively measure and quantify all the facets of developer activity across engineering systems and environments*. A well-designed engineering system, however, will help in capturing activity metrics along different phases of the software development life cycle and quantify developer activity at scale. Some of the developer activities that can be measured and quantified relatively easily are:

- *Design and coding.* Volume or count of design documents and specs, work items, pull requests, commits, and code reviews.
- *Continuous integration and deployment.* Count of build, test, deployment/release, and infrastructure utilization.
- *Operational activity.* Count or volume of incidents/issues and distribution based on their severities, on-call participation, and incident mitigation.

These metrics can be used as waypoints to measure some tractable developer activities, but they should never be used in isolation to make decisions about individual or team productivity because of their known limitations. They serve as templates to start with and should be customized based on organizational needs and development environments. As mentioned earlier, many activities that are essential to developing software are intractable (such as attending team meetings, participating in brainstorming, helping other team members when they encounter issues, and providing architectural guidance, to name a few).

Communication and collaboration. *Communication and collaboration* capture how people and teams communicate and work together. Software development is a collaborative and creative task that relies on extensive and effective communication, coordination, and collaboration within and between teams.[11] Effective teams that successfully contribute to and integrate each other's work efficiently rely on high transparency[5] and awareness6 of team member activities and task priorities. In addition, how information flows within and across teams impacts the availability and discoverability of documentation that is needed for the effective alignment and integration of work. Teams that are diverse and inclusive are higher performing.[22] More effective teams work on the right problems, are more likely to be successful at brainstorming new ideas and will choose better solutions from all the alternatives.

Productivity and satisfaction are correlated, and it is possible that satisfaction could serve as a leading indicator for productivity.

Work that contributes to a team's outcomes or supports another team member's productivity may come at the expense of an individual's productivity and their own ability to get into a state of flow, potentially reducing motivation and satisfaction. Effective collaboration, however, can drive down the need for some individual activities (for example, unnecessary code reviews and rework), improve system performance (faster pull request merges may improve quality by avoiding bugs), and help sustain productivity and avoid (or conversely, if not done right, increase) burnout.

Understanding and measuring team productivity and team member expectations are, however, complicated because of items that are difficult to measure such as invisible work[21] and articulation work for coordinating and planning team tasks.[18] That said, the following are examples of metrics that may be used as proxies to measure communication, collaboration, and coordination:

- Discoverability of documentation and expertise.
- How quickly work is integrated.
- Quality of reviews of work contributed by team members.
- Network metrics that show who is connected to whom and how.
- Onboarding time for and experience of new members.

Efficiency and flow. Finally, *efficiency* and *flow* capture the ability to complete work or make progress on it with minimal interruptions or delays, whether individually or through a system. This can include how well activities within and across teams are orchestrated and whether continuous progress is being made.

Some research associates productivity with the ability to get complex tasks done with minimal distractions or interruptions.[2] *This conceptualization of productivity is echoed by many developers when they talk about "getting into the flow" when doing their work*—or the difficulty in finding and optimizing for it, with many books and discussions addressing how this positive state can be achieved in a controlled way.[4] For individual efficiency (flow), it's important to set boundaries to get productive and stay productive—for example, by blocking off time for a focus period. Individual efficiency is often measured by uninterrupted focus time or the time within value-creating apps (for example, the time a developer spends in the integrated development environment is likely to be considered "productive" time).

At the team and system level, efficiency is related to value-stream mapping, which captures the steps needed to take software from idea and creation to delivering it to the end customer. To optimize the flow in the value stream, it is important to minimize delays and handoffs. The DORA (DevOps Research and Assessment) framework introduced several metrics to monitor flow within teams[9]—for example, deployment frequency measures how often an organization successfully releases to production, and lead time for changes measures the amount of time it takes a commit to get into production.

In addition to the flow of changes through the system, the flow of knowledge and information is important. Certain aspects of efficiency and flow may be difficult to measure, but it is often possible to spot and remove inefficiencies in the value stream. Activities that produce no value for the customer or user are often referred to as software development waste[19]—for example, duplicated work, rework because the work was not done correctly, or time-consuming rote activities.

Some example metrics to capture the efficiency and flow dimension are:

▸ Number of handoffs in a process; number of handoffs across different teams in a process.

▸ Perceived ability to stay in flow and complete work.

▸ Interruptions: quantity, timing, how spaced, impact on development work and flow.

▸ Time measures through a system: total time, value-added time, wait time.

Efficiency is related to all the SPACE dimensions. Efficiency at the individual, team, and system levels has been found to be positively associated with increased satisfaction. Higher efficiency, however, may also negatively affect other factors. For example, maximizing flow and speed may decrease the quality of the system and increase the number of bugs visible to customers (performance). Optimizing for individual efficiency by reducing interruptions may decrease the ability to collaborate, block others' work, and reduce the ability of the team to brainstorm.

Framework in Action

To illustrate the SPACE framework, Figure 1 lists concrete metrics that fall into each of the five dimensions. The figure provides examples of individual-, team- or group-, and system-level measures. Three brief discussions about these metrics follow: First, an example set of metrics concerning code review is shown to cover all dimensions of the SPACE framework, depending on how they are defined and proxied. Next, additional examples are provided for two select dimensions of the framework: activity, and efficiency and flow. The section closes with a discussion of how to use the framework: combining metrics for a holistic understanding of developer productivity, as well as cautions. The accompanying sidebar shows how the framework can be used for understanding productivity in incident management.

Let's begin with code review as an example scenario that presents a set of metrics that can cover all five dimensions of the SPACE framework, depending on how it is framed and which metric is used:

▸ *Satisfaction.* Perceptual measures about code reviews can reveal whether developers view the work in a good or bad light—for example if they present learning, mentorship, or opportunities to shape the codebase. This is important, because the number of code reviews per developer may signal dissatisfaction if some developers feel they are consistently assigned a disproportionate amount of code re-

Example metrics.

Level	**Satisfaction and well-being** How fulfilled, happy, and healthy one is	**Performance** An outcome of a process	**Activity** The count of actions or outputs	**Communication and collaboration** How people talk and work together	**Efficiency and flow** Doing work with minimal delays or interruptions
Individual One person	▸ Developer satisfaction ▸ Retention† ▸ Satisfaction with code reviews assigned ▸ Perception of code reviews	▸ Code review velocity	▸ Number of code reviews completed ▸ Coding time ▸ # Commits ▸ Lines of code†	▸ Code review score (quality or thoughtfulness) ▸ PR merge times ▸ Quality of meetings† ▸ Knowledge sharing, discoverability (quality of documentation)	▸ Code review timing ▸ Produc-tivity perception ▸ Lack of inter-ruptions
Team or Group People that work together	▸ Developer satisfaction ▸ Retention†	▸ Code review velocity ▸ Story points shipped†	▸ # Story points completed†	▸ PR merge times ▸ Quality of meetings† ▸ Knowledge sharing or discoverability (quality of documentation)	▸ Code review timing ▸ Handoffs
System End-to-end work through a system (like a development pipeline)	▸ Satisfaction with engineering system (e.g., CI/CD pipeline)	▸ Code review velocity ▸ Code review (acceptance rate) ▸ Customer satisfaction ▸ Reliability (uptime)	▸ Frequency of deploy-ments	▸ Knowledge sharing, discoverability (quality of documentation)	▸ Code review timing ▸ Velocity/ flow through the system

† Use these metrics with (even more) caution — they can proxy more things.

views, leaving them with less time for other work.

▸ *Performance.* Code-review velocity captures the speed of reviews; because this can reflect both how quickly an individual completes a review and the constraints of the team, it is both an individual- and a team-level metric. (For example, an individual could complete a review within an hour of being assigned, but a team could have a policy of leaving all reviews open for 24 hours to allow all team members to see the proposed changes.)

▸ *Activity.* Number of code reviews completed is an individual metric capturing how many reviews have been completed in a given time frame and contributes to the final product.

▸ *Communication and collaboration.* Code reviews themselves are a way that developers collaborate through code, and a measure or score of the quality or thoughtfulness of code reviews is a great qualitative measure of collaboration and communication.

▸ *Efficiency and flow.* Code review is important but can cause challenges if it interrupts workflow or if delays cause constraints in the system. Similarly, having to wait for a code review can delay a developer's ability to continue working. Batching up code reviews so they don't interrupt a developer's coding time (which would impact individual measures), while also not causing delays in the throughput of the system (which impacts system measures), allows teams to deliver code efficiently (team-level measures). Therefore, measuring the effects of code-review timing on the efficiency and flow of individuals, teams, and the system is important—this can be done through perceptual or telemetry measures that capture the time to complete reviews and the characteristics of interruptions (such as timing and frequency).

Let's examine the SPACE framework in more depth by looking further at the dimensions of activity and efficiency and flow. In this example, the activity measures are individual-level metrics: number of commits, coding time (total time spent or times of day), and number of code reviews completed. These best describe work that directly contributes to the final product, understanding that work patterns and behaviors are influenced by the teams and environments in which developers work.

Efficiency and flow have a broader mix of metrics. Self-reported measures of productivity are best captured at the individual level: asking a developer whether the team is productive is subject to blind spots, while asking if that member felt productive or was able to complete work with minimal distractions is a useful signal. You can also measure the flow of work—whether code, documents, or other items—through a system, and capture metrics such as the time it takes or the number of handoffs, delays, and errors in the software delivery pipeline. These

SPACE and SRE: The Framework in Incident Management

The SPACE framework is relevant for SREs (site reliability engineers) and their work in IM (incident management). An incident occurs when a service is not available or is not performing as defined in the SLA (service-level agreement). An incident can be caused by network issues, infrastructure problems, hardware failures, code bugs, or configuration issues, to name a few.

Based on the magnitude of the impact caused by an incident, it is typically assigned a severity level (sev-1 being the highest). An outage to the entire organization's customer-facing systems is treated differently than a small subset of internal users experiencing a delay in their email delivery.

Here are some of the common myths associated with IM:

▸ MYTH: Number of incidents resolved by an individual is all that matters. Like a lot of other activities in the SDLC (software development life cycle), IM is a team activity. A service that causes a lot of outages and takes more hours to restore reflects badly on the entire team that develops and maintains the service. More team-focused activities such as knowledge sharing, preparing troubleshooting guides to aid other team members, mentoring juniors and new members of the team, doing proper handoffs and assignment/re-assignments are important aspects of IM.

▸ MYTH: Looking at one metric in isolation will tell you everything. It is important to understand the metrics in context: the number of incidents, how long they took to resolve—the volume and resolution times of sev-1 incidents compared with sev-4, and other factors relevant to understanding incidents and how to improve both the system and the team's response. So, there is no "one metric that matters."

▸ MYTH: Only management cares about incident volume and meeting SLAs. With the rise of DevOps, developers are also doing operations now. IM (a part of operations) can take away a significant chunk of developers' time and energy if the volume and severity of the incidents are high. As important as it is to management and executives to guarantee SLAs and reduce incident volume and resolution times, it is equally important to the individual developers who are part of the IM process.

▸ MYTH: Effective IM is just about improving systems and tools. Better monitoring systems, ticketing systems, case-routing systems, log-analysis systems, etc. will help make developers productive. While tools, guides, and workflows have a large impact on productivity, the human factors of the environment and work culture have substantial impact too. Mentoring new members of the team and morale building are important. If developers are constantly being paged in the night for sev-1 incidents while working from home during COVID-19, these "invisible" factors are especially helpful to make them more productive.

Incident management is a complex process that involves various stakeholders performing several individual and team activities, and it requires support from different tools and systems, so it is critical to identify metrics that can capture various dimensions of productivity:

▸ *Satisfaction:* How satisfied SREs are with the IM process, escalation and routing, and on-call rotations are key metrics to capture, especially since burnout is a significant issue among SREs.

▸ *Performance:* These measures focus on system reliability; monitoring systems' ability to detect and flag issues faster, before they hit the customer and become an incident. MTTR (mean time to repair) overall, and by severity.

▸ *Activity:* Number of issues caught by the monitoring systems, number of incidents created, number of incidents resolved—and their severity distribution.

▸ *Communication and collaboration:* People included in resolving the incident, how many teams those people came from, and how they communicate during an incident. Incident resolution documentation outlines the steps involved in resolving incidents; this can be measured by completeness (to check if any resolution data was entered) or quick quality scores (for example, thumbs up/down). Teams may also include a metric that measures the percentage of incidents resolved that reference these guides and documentation.

▸ *Efficiency and flow*: Incident handoffs, incident assignment/reassignment, number of hops an incident has to take before it is assigned to the right individual or team.

would constitute system-level metrics, because their values would capture the journey of the work item through the entire workflow, or system.

How To Use the Framework

To measure developer productivity, teams and leaders (and even individuals) should capture several metrics across multiple dimensions of the framework—at least three are recommended. For example, if you are already measuring commits (an activity measure), don't simply add the number of pull requests and coding time to your metrics dashboard, as these are both activity metrics. Adding these can help round out the way you capture the activity dimension of productivity, but to really understand productivity, add at least one metric from two different dimensions: perhaps perception of productivity and pull request merge time.

Another recommendation is that at least one of the metrics include perceptual measures such as survey data. By including perceptions about people's lived experiences, a more complete picture of productivity can be constructed. Many times, perceptual data may provide more accurate and complete information than what can be observed from instrumenting system behavior alone.[10]

Including metrics from multiple dimensions and types of measurements often creates metrics in tension; this is by design, because a balanced view provides a truer picture of what is happening in your work and systems. This more balanced view should help to reinforce smarter decisions and trade-offs among team members, who may otherwise understandably focus on one aspect of work to the detriment of the whole system.

One example is story points, a metric commonly used in Agile development processes to assess team-level progress. If a team is rated only on story points, members will focus on optimizing their own points, to the detriment of completing potentially invisible work that is important to other developers' progress and to the company if that means collaborating with other teams or onboarding future developers. And if leaders measured progress using story points without asking developers about their ability to work quickly, they wouldn't be able to identify if something wasn't working and the team was doing workarounds and burning out, or if a new innovation was working particularly well and could be used to help other teams that may be struggling.

This leads to an important point about metrics and their effect on teams and organizations: They signal what is important. One way to see indirectly what is important in an organization is to see what is measured, because that often communicates what is valued and influences the way people behave and react. For example, companies that care about employee health, well-being, and retention will likely include the satisfaction and well-being dimension in their productivity measures. As a corollary, adding to or removing metrics can nudge behavior, because that also communicates what is important.

For example, a team where "productivity = lines of code" alone is very different from a team where "productivity = lines of code AND code review quality AND customer satisfaction." In this case, you have kept a (problematic, but probably embedded) metric about productivity and output, but nudged perceptions about productivity in a direction that also values both teamwork (by valuing thoughtful code reviews) and the end user (by valuing customer satisfaction).

Metrics shape behavior, so by adding and valuing just two metrics, you've helped shape a change in your team and organization. This is why it's so important to be sure to pull from multiple dimensions of the framework: it will lead to much better outcomes at both the team and system levels. In this example, as the teams continue to improve and iterate, they could exchange the activity metric *lines of code* for something like *number of commits*.

What to Watch For

Having too many metrics may also lead to confusion and lower motivation; not all dimensions need to be included for the framework to be helpful. For example, if developers and teams are presented with an extensive list of metrics and improvement targets, meeting them may feel like an unattainable goal. With this in mind, note that a good measure of productivity consists of a handful of metrics across at least three dimensions; these can prompt a holistic view, and they can be sufficient to evoke improvement.

Any measurement paradigm should be used carefully because no metric can ever be a perfect proxy. Some metrics are poor measures because they are noisy approximations (some examples are noted in Figure 1). Retention is often used to measure employee satisfaction; however, this can capture much more than satisfaction—it can reflect compensation, promotion opportunities, issues with a team, or even a partner's move. At the team level, some managers may block transfers to protect their own retention ratings. Even if retention did reflect satisfaction, it is a lagging measure, and teams don't see shifts until it is too late to do anything about it. We have written elsewhere about the limitations inherent in the use of story points,[9] which could give teams incentive to focus on their own work at the expense of collaborating on important projects.

Teams and organizations should be cognizant of developer privacy and report only anonymized, aggregate results at the team or group level. (In some countries, reporting on individual productivity isn't legal.) Individual-level productivity analysis, however, may be insightful for developers. For example, previous research shows that typical developer work shifts depend on the phase of development, and developers may have more productive times of day.[14] Developers can opt in to these types of analyses, gaining valuable insights to optimize their days and manage their energy.

Finally, any measurement paradigm should check for biases and norms. These are external influences that may shift or influence the measures. Some examples are included here, but they aren't exhaustive, so all teams are encouraged to look for and think about external influences that may be present in their data:

▸ *Peer review and gender.* Research shows that women are more likely to receive negative comments and less likely to receive positive comments in their code reviews.[16] Any analysis of satisfaction with the review process should check for this in your environment. Understand that developers are likely influenced by the broader tech industry even if the patterns are not in your organization or team. Take these effects into account.

▸ *Normalizing measures across time.* Teams should be careful about any methods used to normalize time, especially across long periods. For example, looking at metrics over a year would bias against those taking parental leave.

▸ *Perceptual measures.* Teams and organizations should be mindful of cultural norms—and embrace these. Some cultures naturally report higher, while some report lower. It doesn't mean perceptual measures can't be trusted; it just means measures from these different cultures will have a different baseline and shouldn't be compared with each other.

Why This Matters Now

Developer productivity is about more than an individual's activity levels or the efficiency of the engineering systems relied on to ship software, and it cannot be measured by a single metric or dimension. We developed the SPACE framework to capture different dimensions of productivity, because without it, pervasive and potentially harmful myths about productivity may persist.

The SPACE framework provides a way to logically and systematically think about productivity in a much bigger space and to carefully choose balanced metrics linked to goals—and how they may be limited if used alone or in the wrong context. The framework helps illuminate tradeoffs that may not be immediately obvious and to account for invisible work and knock-on effects of changes such as increased work if activity is measured at the expense of unfulfilled developers or disruptions to overall flow and efficiency.

The need to understand and measure productivity holistically has never been greater. As the Covid-19 pandemic disrupted work and brought a sudden switch to working from home, many questioned its impact on productivity and posed questions around how to understand and measure this change. As the world slowly returns to a "new normal," the SPACE framework captures the dimensions of productivity that are important to consider as future changes are proposed and made. The framework is meant to help individuals, teams, and organizations identify pertinent metrics that present a holistic picture of productivity; this will lead to more thoughtful discussions about productivity and to the design of more impactful solutions.

Acknowledgments. We are grateful for the thoughtful review and insightful comments from our reviewers and are confident that incorporating their notes and responses has strengthened the article.

Related articles on queue.acm.org

Getting What You Measure
Eric Bouwers, Joost Visser, and Arie van Deursen
https://queue.acm.org/detail.cfm?id=2229115

DevOps Metrics
Nicole Forsgren and Mik Kersten
https://queue.acm.org/detail.cfm?id=3182626

Beyond the "Fix-it" Treadmill
J. Paul Reed
https://queue.acm.org/detail.cfm?id=3380780

People and Process
James Champy
https://queue.acm.org/detail.cfm?id=1122687

References

1. Beller, M., Orgovan, V., Buja, S., Zimmermann, T. Mind the gap: on the relationship between automatically measured and self-reported productivity. *IEEE Software* (2020); https://arxiv.org/abs/2012.07428.
2. Brumby, D. P., Janssen, C. P., Mark, G. How do interruptions affect productivity? *Rethinking Productivity in Software Engineering.* C. Sadowski and T. Zimmermann, eds. Apress, Berkeley, CA, 2019, 85–107; https://link.springer.com/chapter/10.1007/978-1-4842-4221-6_9.
3. Butler, J.L., Jaffe, S. Challenges and gratitude: a diary study of software engineers working from home during Covid-19 pandemic (Aug. 2020). Microsoft; https://www.microsoft.com/en-us/research/publication/challenges-and-gratitude-a-diary-study-of-software-engineers-working-from-home-during-covid-19-pandemic/.
4. Csikszentmihalyi, M. *Flow: The Psychology of Optimal Experience.* Harper Perennial Modern Classics, 2008.
5. Dabbish, L., Stuart, C., Tsay, J., Herbsleb, J. Social coding in GitHub: Transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 Conf. Computer-supported Cooperative Work* (Feb. 2012), 1277–1286; https://dl.acm.org/doi/10.1145/2145204.2145396.
6. Dourish, P., Bellotti, V. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM Conf. Computer-supported Cooperative Work* (Dec. 1992), 107–114; https://dl.acm.org/doi/10.1145/143457.143468.
7. Ford, D. et al. A tale of two cities: software developers working from home during the Covid-19 pandemic, 2020; https://arxiv.org/abs/2008.11147.
8. Forsgren, N. Finding balance between work and play. *The 2020 State of the Octoverse.* GitHub; https://octoverse.github.com/static/github-octoverse-2020-productivity-report.pdf.
9. Forsgren, N., Humble, J. Kim, G. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations.* IT Revolution Press, 2018.
10. Forsgren, N., Kersten, M. DevOps metrics. *Commun. ACM 61*, 4 (2018), 44–48; https://dl.acm.org/doi/10.1145/3159169.
11. Fuks, H., Raposo, A., Gerosa, M. A., Pimental, M. The 3C collaboration model. *Encyclopedia of E-Collaboration.* Ned Kock, ed. IGI Global, 2008, 637–644. https://www.researchgate.net/publication/292220266_The_3C_collaboration_model.
12. Graziotin, D., Fagerholm, F. Happiness and the productivity of software engineers. *Rethinking Productivity in Software Engineering.* C. Sadowski and T. Zimmermann, eds. Apress, Berkeley, CA, 2019, 109–124; https://link.springer.com/chapter/10.1007/978-1-4842-4221-6_10.
13. Maslach, C., Leiter, M.P. Early predictors of job burnout and engagement. *J. Applied Psychology 93*, 3 (2008), 498–512; https://doi.apa.org/doiLanding?doi=10.1037%2F0021-9010.93.3.498.
14. Meyer, A. N., Barton, L. E., Murphy, G. C., Zimmermann, T., Fritz, T. The work life of developers: activities, switches and perceived productivity. *IEEE Trans. Software Engineering 43*, 12 (2017), 1178—1193; https://dl.acm.org/doi/10.1109/TSE.2017.2656886.
15. Murphy-Hill, E. et al. What predicts software developers' productivity? *IEEE Trans. Software Engineering*, 2019; https://ieeexplore.ieee.org/document/8643844/.
16. Paul, R., Bosu, A., Sultana, K.Z. Expressions of sentiments during code reviews: male vs. female. In *IEEE 26th Intern. Conf. Software Analysis, Evolution and Reengineering*, 2019, 26–37; https://ieeexplore.ieee.org/document/8667987.
17. Ralph, P. et al. Pandemic programming: How Covid-19 affects software developers and how their organizations can help. *Empirical Software Engineering 25*, 6 (2020), 4927–4961; https://www.researchgate.net/publication/344342621_Pandemic_programming_How_COVID-19_affects_software_developers_and_how_their_organizations_can_help.
18. Schmidt, K., Bannon, L. Taking CSCW seriously: Supporting articulation work. *Computer Supported Cooperative Work 1*, 1 (1992), 7–40; https://link.springer.com/article/10.1007/BF00752449.
19. Sedano, T., Ralph, P., Péraire, C. Software development waste. In *Proceedings of the 39th Inter. Conf. Software Engineering*, 2017, 130–140; https://dl.acm.org/doi/10.1109/ICSE.2017.20.
20. Storey, M. A., Zimmermann, T., Bird, C., Czerwonka, J., Murphy, B., Kalliamvakou, E. Towards a theory of software developer job satisfaction and perceived productivity. *IEEE Trans. Software Engineering*, 2019; https://ieeexplore.ieee.org/document/8851296.
21. Suchman, L. Making work visible. *Commun. ACM 38*, 9 (Sept. 1995), 56–64; https://dl.acm.org/doi/10.1145/223248.223263.
22. Vasilescu, B.et al., Filkov, V. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd Annual ACM Conf. Human Factors in Computing Systems* (Apr. 2015), 3789-3798; https://dl.acm.org/doi/abs/10.1145/2702123.2702549.

Nicole Forsgren is the VP of Research & Strategy at GitHub. She is an expert in DevOps and the author of the Shingo Publication Award-winning book *Accelerate: The Science of Lean Software and DevOps*. Her work on technical practices and development is used to guide organizational transformations around the world.

Margaret-Anne Storey is a professor of computer science at the University of Victoria and a Canada Research Chair in Human and Social Aspects of Software Engineering. She consults with Microsoft to improve developer productivity.

Chandra Maddila is a Senior Research Engineer at Microsoft Research. He developed tools and techniques that are used organization-wide at Microsoft.

Thomas Zimmermann is a Senior Principal Researcher at Microsoft Research. He is best known for his work on mining software repositories and data science in software engineering.

Brian Houck is a Principal Program Manager in the Azure Engineering Systems at Microsoft. His work focuses on improving developer productivity and satisfaction for engineers within the Azure organization.

Jenna Butler is an adjunct professor at Bellevue College in the Radiation Therapy department and is a senior software engineer at Microsoft. She is currently working with MSR's Productivity & Intelligence team to study alignment and decision making; working in the services; and the impact of remote work during this time.

DOI:10.1145/3453930

Article development led by acmqueue
queue.acm.org

Extending hardware-enforced cryptographic protection to data while in use.

BY MARK RUSSINOVICH, MANUEL COSTA, CÉDRIC FOURNET, DAVID CHISNALL, ANTOINE DELIGNAT-LAVAUD, SYLVAN CLEBSCH, KAPIL VASWANI, AND VIKAS BHATIA

Toward Confidential Cloud Computing

ALTHOUGH LARGELY DRIVEN by economies of scale, the development of the modern cloud also enables increased security. Large datacenters provide aggregate availability, reliability, and security assurances. The operational cost of ensuring that operating systems, databases, and other services have secure configurations can be amortized among all tenants, allowing the cloud provider to employ experts who are responsible for security; this is often unfeasible for smaller businesses, where the role of systems administrator is often conflated with many others.

Cloud datacenters are also subject to tight physical security: the number of people with physical access is limited, and the controls on their access are stricter in a large, cloud provider's datacenters than on premises—often vulnerable to insider threats such as disgruntled former employees leaving with a copy of sensitive data or physical media (including on-site backups).

Cloud providers systematically encrypt data in transit (on the network) and at rest (on disks and backups) using keys associated with tenants: even if attackers gain access to a datacenter, they cannot see the plaintext of tenant data unless they also manage to compromise their managed keys. This trend of increasing security in the cloud will continue; the next step is *confidential computing*, extending hardware-enforced cryptographic protection to data while in use (that is, during computation).

Trusting the Cloud

Why would tenants take security assurances from the cloud at face value? Tenants trust their providers to different extents. Some may fully trust it to keep their data secure. Some may be concerned about other tenants, software bugs, or insider attacks (for example, from datacenter technicians). Some may require compliance with strict privacy regulations. Some may also doubt the provider's willingness or ability to enforce its stated security policies—guaranteeing, for example, that their data will never be used without their consent—or even fear subpoenas and other legal attacks in the jurisdictions where the cloud operates. To address these concerns, tenants increasingly expect the following:

- Minimal hardware, software, and operational trusted computing bases (TCBs) for their sensitive workloads.
- Technical enforcement, rather than just business policies.
- Transparency about the guarantees, residual risks, and mitigations that they get.

Confidential computing meets these expectations by allowing tenants to exercise full control over the TCB used to run their cloud workloads: *Confidential computing allows tenants to precisely define all the hardware and*

IMAGE BY GORAPHOL K

software that has access to their workloads (data and code), and it provides the technical mechanisms to verifiably enforce this guarantee. In short, tenants retain full control over their secrets.

In particular, confidential computing can render workloads opaque to the cloud provider because tenants can use this precise level of control to prevent access to their secrets by the hypervisor and other cloud-hosting infrastructure. This prevents attacks from the cloud fabric and its operators and complements the more traditional security goal of protecting the cloud fabric from potentially malicious tenants.

This level of control goes beyond preventing accesses by the cloud-hosting infrastructure: it allows a tenant to specify that a particular set of secrets can be processed only by a specific code module. This capability is powerful because it can be used to design resilient systems with reduced attack surfaces. Precise control over the trust placed in confidential cloud services enables useful scenarios among multiple parties that do not fully trust one another. For example, a tenant may in turn deploy a service with strong privacy assurances for its own customers; and competing parties may jointly configure and run a multiparty cloud computation (such as data analytics or machine learning) with strong technical guarantees about the use of their pooled data.

The shift to confidential computing is part of an industrywide effort. The Confidential Computing Consortium, founded in 2019, includes Alibaba, AMD, Arm, Google, Huawei, Intel, Microsoft, nVidia, Oracle, Red Hat, and many others. The consortium exists in recognition that transitioning to a world where confidential computing is the default for all cloud services will require significant effort at all levels of the stack, starting from the hardware and including hypervisors, operating-system kernels, and cloud services.

Confidential Computing Platforms

Let's look at trusted execution environments, dynamic implementations of such, blind hypervisors, and various platform abstractions.

Hardware foundation: Trusted execution environments. At the lowest level of the stack, the hardware must be able to provide a TEE (trusted execution environment) that isolates the code and data of a given confidential workload from any other code running in a system—including code running at the highest privilege levels. The hardware must also support encryption for all of its I/O, as data flows in and out of the TEE, and be able to measure and sign the contents of the TEE to produce verifiable evidence that it is secure. This in turn requires a *hardware root of trust* to hold the platform root secrets and signing keys, and a public-key infrastructure to endorse these keys. Thankfully, these features can often be fitted into new generations of existing platforms, so they can be used in *confidential mode*, rather than requiring dedicated hardware and software.

Isolation. For confidentiality and integrity, software running in the TEE must be safe from snooping or interference by other parts of the system. This may involve fencing and locking mechanisms to prevent changes to trusted code once it has been loaded and measured, and to reserve resources such as cores or memory caches for the exclusive use of a TEE to mitigate side channels. Many TEEs use encryption for either all of their state or the portion that is stored outside of trusted tightly coupled memory. Side channels, such as the Spectre and Meltdown family of vulnerabilities or differential power analysis, have the potential to pierce isolation. The full hardware/software stack for TEEs must provide defenses against any that are in scope for the threat model—for example, speculative load hardening[3] in the compiler, or secure caches[8,13] and mechanisms for secure speculation[9,14] in the hardware. Some of these defenses may be built entirely at the software level—for example, by using data trace-oblivious data structures.[5]

Hardware root of trust. Protection must be rooted in hardware; otherwise, cloud operators could easily break isolation by emulating TEEs in software. Each TEE-capable device must have a unique identity, cryptographically secured with a hardware secret. This secret may, for example, be sampled and recorded in fuse banks within the device at the end of its manufacturing process, and the corresponding public key may be harvested by the manufacturer to issue the platform certificate. Trusted platform modules (TPMs) provide an early example of discrete roots of trust, with limited protection from physical attacks. Google's recently released OpenTitan core and Microsoft's Pluton subsystem are examples of more advanced, integrated hardware roots of trust. Pluton was originally created for the Xbox One gaming console and therefore has a long history of large-scale deployment into potentially hostile environments. It is also integrated into Azure Sphere IoT (Internet of things) devices and has been licensed to major CPU vendors (AMD, Intel, and Qualcomm), where it initially provides a TPM interface and can provide the hardware root of trust for confidential computing systems.

Although hardware roots of trust are now well established, they traditionally protect someone offering a service on a device (a game publisher or, in our case, a cloud provider) from misuse, whereas confidential computing further requires they protect third parties against attackers that have physical access to the device or logical access to the nonconfidential parts of the device. As an example, a mechanism that allows firmware signed by the cloud provider direct access to the hardware secret violates the promise of confidential computing.

Attestation. Attestation is the mechanism that builds the confidence into confidential computing. As with cryptographic messaging systems, confidentiality without integrity is insufficient. Being able to run software in such a way that the cloud provider cannot inspect or tamper with it is no use if you can't guarantee that it really *is* the software that you expected to run.

Attestation uses keys derived from the hardware secrets maintained by the hardware root of trust to sign evidence that a TEE is in a known-good state protected by a real hardware device. This evidence is similar to a secure boot signature: a set of measurements of the TEE, for example, the hash of the initial memory contents, and the state of various security-critical registers. Upon receiving and verifying this evidence, a remote user or software component can be certain of the integrity of the TEE and will then typically establish an encrypted channel to deploy secrets and control computation inside the TEE.

Hardware advancements: Dynamic TEE implementations. Physical isolation is the simplest way of guaranteeing confidentiality and integrity—for example, by using an isolated core with a simple I/O interface. This is the route taken by Apple's Secure Element (found in iOS devices and recent Macs). This is sufficient when the code running in the TEE has compute and storage requirements that are known up front and is provided by a single vendor. It is not sufficient in a cloud environment with elastic requirements and many tenants: the cloud fabric must be able to create variable-sized TEEs, to host multiple TEEs on a single system, and to dynamically allocate/deallocate resources to TEEs.

As part of the confidential computing effort, TEEs became available on most general-purpose processors over the past few years. Arm's TrustZone provided an early TEE implementation, allowing memory to be assigned at boot time to one of two worlds—secure or normal—and allowing a small, trusted kernel in the secure world to provide isolated processes. In this model, all of the secure-world components must trust the secure kernel (and secure hypervisor, if present), though they do not have to trust the normal-world's hypervisor and operating system kernel(s).

Intel's SGX (Software Guard Extensions) took this a step further and provided the ability to create isolated regions of memory in the virtual address space of user-mode processes. Any number of these regions can be created dynamically after booting, subject to resource constraints. Code and data inside the regions are protected against software attacks by access-control checks implemented by the CPU: the hypervisor and the kernel cannot see or tamper with the data inside the regions. The regions are also protected against physical attackers by memory encryption: whenever data in use leaves the CPU caches, it is encrypted before being written back to memory.

Memory encryption overhead is low if you want only to guarantee confidentiality, and not cryptographic integrity in the presence of replay attacks from an adversary with access to the memory bus. The Xbox 360 and later models have used AES (Advanced Encryption Standard) for memory encryption since their launch and provided enough bandwidth and latency for games, some of the most demanding workloads on modern CPUs. The memory controllers of mainstream CPUs are now gaining similar functionality, including support for different keys for different memory regions. Protecting large-scale memory integrity against physical attacks is more expensive; schemes to reduce this overhead are an area of active research.

SGX's isolated memory regions are

ideal for small-TCB services,[10] but using them to run full virtual machine (VM) confidentiality is challenging because they lack support for multiple address spaces and privileged and unprivileged mode separation. This led AMD to develop another type of TEE focused on VM-level isolation. AMD processors went through a series of design iterations aimed at fully removing the hypervisor from the trust boundary. Their first step, Secure Encrypted Virtualization (SEV) automatically encrypted memory in use by VMs. Next, SEV_ES (SEV with Encrypted State) added encryption of VM register state on every transition to the hypervisor. Finally, SNP (Secure Nested Paging), provided an additional guarantee that the hypervisor cannot break memory integrity by tampering with the virtual memory mappings to execute integrity or replay attacks on a confidential VM. Taken together, these features guarantee that the hypervisor cannot read or tamper with VM state (that is, the hypervisor is out of the TCB). Intel's recently announced TDX (Trust Domain Extensions) provides a similar set of security guarantees to SNP, and it targets comprehensive VM-level confidentiality.

Finally, several important workloads require specialized processors, such as GPUs, FPGAs (field-programmable gate arrays), and other accelerators. These devices can also be augmented with TEE capabilities, with a large design space.[12] For example, some accelerators may have large memories (for example, high-bandwidth memory) protected with physical packaging, making encryption less relevant. In many cases, accelerators can be allocated to a single tenant at a time, which removes attacks and further simplifies their designs.

To use these devices securely and efficiently, I/O buses also require changes. Most current systems assume the I/O bus is trusted, but this has been a problem for embedded systems: until very recently the CAN (controller-area network) bus used in most cars did not perform any end-to-end authentication, allowing compromised components such as a media player to send messages pretending to be from the engine-management system. Similarly, the PCI (peripheral component interconnect) bus specification assumes that all endpoints are trusted, enabling a malicious device to spoof the originator ID, for example, but confidential accelerators need a mechanism for establishing end-to-end secure channels between device and host TEEs and for integrating encryption between the device and the host.

Exposing confidential computing hardware requires changing the systems layer of the cloud fabric.

Virtualization advancements: Blind hypervisors. Exposing confidential computing hardware requires changing the systems layer of the cloud fabric. This includes changing the hypervisor to handle the constraint that it cannot see VM state. Mainstream hypervisor designs follow a hierarchical trust model. The hypervisor is fully trusted by the guest, is responsible for storing guest state between context switches and has full access to guest memory.

Most paravirtualized device interfaces are designed on the assumption that any guest memory can be used for the virtual equivalent of DMA (direct memory access) buffers. These assumptions stopped being true on mainstream hardware with the introduction of AMD's SEV. Memory encryption meant that the hypervisor had to provide a region for bounce buffers that the guests could use. Linux supports this mode of operation with SEV via the software IOTLB (input/output translation lookaside buffer) driver.

When performing an explicit domain transition from inside the TEE to the surrounding environment, hardware implementations typically preserve the register context. This is not usually true for asynchronous exits, which may leak sensitive information. In a model where the hypervisor is trusted, this information may be useful for handling the VM exit. When the hypervisor is untrusted, either it must be modified so it does not take advantage of the feature, or a shim layer must be added to sanitize the information.

For example, consider a page fault in second-level address translation. In a conventional system, the hypervisor receives a trap with the full register context and an exception register specifying the fault address. It can then either kill the VM, issue an upcall to ask the VM's kernel to handle it, or page in the missing page from backing store. In a confidential computing system, this would allow the hypervisor to single-step execution and see the register state at every step. At a minimum, the hypervisor must receive only an encrypted and integrity-protected version of the register state, but even knowing the fault address may leak too much information. In a more secure design, a shim layer inside the TEE would receive the asynchronous exit and decide whether to issue a hypercall to fill in the missing page or simply notify the kernel of the fault. This code can then enforce policy related to the frequency of such exits in order to mitigate possible attacks from the hypervisor.

It is worth noting that it is possible to create a form of TEE by trusting the hypervisor when hardware isolation is not available. This is the basis for Windows' virtualization-based security, introduced with Windows Server 2016 and Windows 10, where critical components run isolated from the Windows kernel by Hyper-V. This can support the same abstractions as other TEEs, including isolated memory regions similar to SGX, but with a weaker threat model: in this design, the hypervisor is still trusted. This is similar to the approach recently used by Amazon's Nitro Enclaves.[1]

Platform abstractions: Confidential VMs, confidential containers, enclaves. The systems layer exposes confidential computing hardware to developers and users through a set of platform abstractions: confidential VMs, confidential containers, and enclaves.

Confidential VMs allow tenants to have a fully backward-compatible VM experience running existing unmodified applications. In the background, systems record and check attestations to verify the security guarantees and make them auditable. Placing entire VMs in TEEs is important for fast and easy adoption, but it also causes some problems. For example, the administrator for the VM has full read/write control over the VM, which is too coarse in many cases. Another concern is that the TCB for a VM is large: a VM image is far more than just a kernel and an application; it includes a large number of system services. In the worst case, this is still likely to be more secure than running the software on premises or on existing cloud infrastructure, but there could be a better solution.

Confidential containers allow tenants to have a finer degree of control over the TCB and to run new or existing containerized applications confidentially. Over the past few years, containers have emerged as a common way of deploying software in the cloud. The exact technology varies, but a container is typically a small (often layered or virtual) file system that contains the minimum required to run a single program.

Best practices recommend running a single microservice in each container. Orchestration infrastructure then supports deploying fleets of cooperating microservice containers. This has several advantages: the container can be configured to have a smaller TCB than a complete confidential VM, and the confidential container may run in a VM without the VM administrator being able to access it. The TEE providing the isolation for the container may still be based on VM-level isolation mechanisms (SNP, TDX, and so on), or it may be based on process-level isolation. (Systems such as Haven[2] and SGX-LKL[11] [Linux Kernel Library] adopt ideas from the library operating system and Exokernel world to run a Windows or Linux library operating system inside SGX memory regions.)

Container deployments of microservices introduce complex attestation issues: orchestration frameworks need to provision these services with sufficient state so they can acquire keys, and they need to manage protocols for establishing the identity of an entire set of microservices.

In an ideal world, software written for security would focus on a minimal TCB. To give developers full control over the TCB, confidential computing platforms expose an enclave abstraction. Enclaves are fully flexible. They can hold as little or as much code as a developer wishes to put in them—for example, they can hold a single function that processes credit-card information or a secret signal-processing algorithm.

In an ideal world, software written for security would focus on a minimal TCB. To give developers full control over the TCB, confidential computing platforms expose an enclave abstraction.

As with containers, the actual hardware-level isolation mechanism for enclaves may be VM- or process-level isolation. For example, VM-level isolation can be used to create a sidecar VM that exposes a simple enclave interface to the base VM. Developers can design services that partition code with different privileges into distinct enclaves. Each enclave should be limited to access only the data necessary to perform its function. Following this principle of least-privilege requires developers to do the additional work of refactoring services, but it yields the highest security and most resilient applications.

There is still a large design space for the interfaces that enclaves will present. The OpenEnclave SDK,[6] originally from Microsoft and now part of the Confidential Computing Consortium, offers C and C++ environments, providing a relatively easy starting point for small-TCB development targeting several types of confidential hardware. Other SDKs have begun to emerge for memory-safe languages such as Rust.

Confidential Computing Applications

Confidential computing makes it possible to outsource sensitive workloads to the cloud, and it even introduces new computing patterns. For example, it enables secure and confidential multiparty computation in which groups of users, who may be mutually distrusting, can run joint computations and share their results without revealing their private inputs to one another or to anyone with physical or logical access to the hardware on which the computations execute. This should lead to the development of a broad range of confidential-computing applications—in fact, these properties are already resulting in advances in multiple application domains.

Confidential AI. Machine-learning algorithms are rapidly increasing their demand for more computation and larger datasets. At the same time, their applications raise significant security and privacy concerns. The elastic and scalable nature of the cloud is already a natural choice for this type of computation, but confidential computing makes it feasible to leverage the cloud with strong security assurances.

In medical and pharmaceutical applications, for example, training data may include individuals' private health-care records, and the resulting models may be used to make clinical decisions. Running the training process inside an enclave ensures that the data cannot be viewed or modified by anyone else. It also provides integrity guarantees (and a robust audit log) that the intended training algorithm was run on the specified records with a specified software stack. As a special case of these guarantees, the resulting model may be encrypted, signed, and equipped with key release policies to ensure it will be unlocked only within another enclave that will enforce specific access control and usage restriction. The enclave may, for example, enforce differential privacy by limiting the number of times the model is queried and adding noise to their results.

As an example, a tenant can leverage confidential cloud computing to offer a medical diagnostics service to its own customers, with technical assurances that it cannot steal or misuse a model supplied and maintained by a specialist third-party for this purpose; and it cannot access any personal medical information to query, store, or use the model for any other purpose.

Confidential databases and analytics. Database systems store and process sensitive and business-critical data such as personal records, financial information, and government data. Unauthorized access to such data can have serious consequences, including physical harm and loss of customer trust and competitive advantage. Current database systems provide sophisticated access-control mechanisms such as role-based access control. These mechanisms are limited in effectiveness against stronger attackers such as those with administrative or physical access to the servers. In a common example, the individual or outsourced team responsible for managing the database represents a large insider threat to a company. Because this individual or team is managing the database system, they are able to see all confidential data, even if they have no need to access it.

Over the past few years, the notion of encrypted databases has been proposed as a way of enforcing stronger, cryptographic access control. In an encrypted database, data remains encrypted both at rest and during computation, using keys that are not available even to the database or server administrator. Data appears in cleartext only within trust boundaries defined by data owners.

Encrypted databases can be realized in multiple ways. One approach (proposed by CryptDB and other related systems) is to encrypt data on a trusted client using partially homomorphic encryption schemes. An alternative approach is to decrypt, process, and re-encrypt sensitive data within a trusted execution environment (such as Intel SGX enclaves). This approach was proposed by EnclaveDB,[7] and a related approach has been adopted by Microsoft's SQL Server in the Always Encrypted feature. The designs vary from streaming columnar data into enclaves at the time of processing to placing all sensitive data and queries within enclaves. Approaches based on TEEs have the potential to provide stronger security guarantees such as integrity and freshness of query processing, confidentiality for queries, tamper-proof auditing, and so on. They are also more flexible (that is, they permit any kind of computation and can support complex access-control policies such as attribute-based access control). In addition to data processing, a TEE is a natural choice for enforcing and auditing fine-grained key-release policies.

Confidential multiparty collaboration. Secure and confidential multiparty computation enabled by enclaves[5] also facilitate new types of collaboration between dataset owners, leading to a multiplicative increase in the amount of training data. Hence, instead of being limited to data from a single hospital, multiparty models can be trained on a joint dataset from multiple hospitals, without revealing the constituent datasets (typically subject to complex regulations and commercial considerations). Although it may be possible to train similar models without pooling their datasets, using, for example, federated learning or local differential privacy, these alternatives would involve algorithmic changes, additional resources, and utility losses. Scenarios for confidential multiparty collaboration exist in many other domains, including finance, energy, climate study, and government.

Confidential ledgers. General-purpose applications that run in a cloud datacenter need a way to convince their users that they are running correctly. New frameworks have emerged to help developers build this new class of trusted applications. These frameworks provide a simple way of building trusted applications that run inside of TEEs and produce verifiable ledgers of their execution. For example, the CCF Confidential Consortium Framework (CCF)[4] provides a tamperproof ledger and transactional updates on a key-value store. This is used to build a number of cloud services such as Azure Confidential Ledger, which provides a tamper-evident, high-performance, confidential ledger that applications can use to store general-purpose log records for auditability and verifiability. These properties allow for a new class of applications that require coordination between mutually distrusting parties, as well as applications that require verifiable execution for legal reasons.

Trusted applications commonly use TEE attestation along with secure messaging channels. This gives users confidence that they are connecting to the correct application in a secure and confidential manner. CCF, and other comparable frameworks, additionally distribute and replicate the execution of application logic to ensure liveness and integrity of execution, even when a fraction of the nodes in the system are malicious or compromised. In particular, CCF supports Byzantine fault tolerance, where arbitrary malicious behavior is tolerated, as well as Crash fault-tolerance configurations.

Foundational Services

Usability is critical to any security technology that aims for widespread adoption. It is easy to imagine a simple configuration switch that lets tenants turn on "confidential computing" for their existing workloads, but what does it mean? Encrypting a VM, for example, adds some defense in depth but is largely security theater unless coupled with a trustworthy mechanism for attesting its contents: if the provider controls the initial VM, and the tenant has no mechanism to review it, then the provider is still fully trusted.

To get meaningful end-to-end protection, the workload inputs and outputs must be encrypted, and the associated data keys must be released only to the TEE allocated for the workload. This involves keeping track of the platform and code that are authorized for this workload, while enabling updates for both platform and code. For convenience, most tenants will likely choose to delegate these tasks to cloud services. For security, these services must themselves be deployed in TEEs, relying on one another for core functionalities such as secure identity, keying, and logging.

The next section outlines the services at the core of a confidential computing ecosystem in support of tenants that deploy confidential workloads and application developers who contribute their code.

Key management and attestation services. If a confidential service needs access to any persistent data (for example, a VM disk image) then it needs to retrieve the key from somewhere. For traditional client-side disk encryption, this can be stored in the TPM and unlocked by the tenant entering a passphrase. In a cloud scenario, the storage part is relatively easy to solve either with managed HSMs (hardware security modules) via a service such as Azure Key Vault or a persistent confidential computing service that manages key storage. Asking the tenant to enter a passphrase for every VM, container, or other service that they launch, however, is not a scalable solution.

When tenants grant access to their data, they are making a policy decision based on a review of the information included in the attestation and other metadata at their disposal. This process can be automated by a confidential cloud service, such as the Microsoft Azure Attestation: at the start of the confidential computation, the newly created TEE establishes a secure connection to the attestation service and presents its attestation materials. The service checks them against the authorization policies previously uploaded by the tenant and, if successful, issues the corresponding credentials. The TEE may then use these credentials to access tenant data. It may, for example, present a token issued by the attestation service to obtain the current decryption key from an HSM.

The cloud provider runs the attestation service atop a fleet of authorized TEEs, and then a tenant can do the manual setup step (the equivalent of entering a passphrase) once and provide policies and credentials so that the service can automatically authorize thousands of VMs on its behalf. Within the cloud, for example, the service may automatically authorize migration between TEEs and recovery from their encrypted checkpoints.

To this end, the service maintains an up-to-date, consistent cache of platform certificates for all TEEs provisioned in its datacenters. It performs frequent checks for certificate revocations and can manage, for example, the consistent deployment of firmware or microcode updates for the trusted hardware (typically requiring new collections of certificates). Thus, the service can support precise, stateful policy statements of the form, "This task must run within an SGX enclave, on an Intel SGX v2.1 platform, deployed in the German Azure datacenter, in a VM allocated to the tenant, supported by certificates that are valid as of today," rather than just, "This task must run within an enclave."

From a cloud perspective, it is important both to minimize the number of distinct hardware offerings and to ensure software portability across everything. X86 VMs will happily work on Intel or AMD hardware even though the virtualization extensions on both are different. If confidential VMs needed to be implemented differently between AMD and Intel hardware, this would add a significant barrier to adoption. By factoring out the hardware-specific details, the attestation service enables other services to be platform independent. For example, it enables the integration of legacy HSMs without the need to customize them for different forms of confidential enclaves, VMs, and containers.

Code transparency. Remote attestation enables tenants (or services they trust) to authenticate the platform and software TCB for their computations, but it does not ensure that this TCB is trustworthy. To this end, tenants would ideally gather and review the security of all the source code for their applications, runtime SDKs and libraries, and compilation tool chains. They would then rebuild the software image for their workload and check that its hash matches the one presented for attestation. This task is daunting for several reasons:

- It involves a lot of software from different origins, even for simple applications.
- Parts of this software may be proprietary or confidential, hindering its review.

▸ Modern build systems are sensitive to details in their environment, yielding irreproducible code.

▸ Cloud computing encourages agile development and facilitates seamless code updates, often without requiring a restart of the application.

▸ Emergency patches may be required to mitigate newly disclosed vulnerabilities, and they can hardly wait for a security review of every impacted application.

To facilitate this task, or at least amortize its cost, a code transparency service can securely record the software dependencies, build environments, and resulting binaries used for confidential computing. A cloud provider may operate this service as a confidential ledger that systematically enforces code-update policies, signs the resulting binaries, and maintains a public, immutable log of all its operations. Accordingly, the tenant may configure the cloud attestation and key management services to authorize any such signed binaries for a confidential application.

For example, a tenant may configure the code transparency service to install emergency patches automatically from a reputable software vendor, provided they are correctly signed and published on the release branch of their designated GitHub project. (More conservatively, they may also require signed endorsements from the cloud provider or a designated expert.) Even if the patch is broken, the service will at least provide a strong audit trail to enable its review after the fact. The software vendor might still be able to undermine a tenant's security guarantees but cannot do so without placing its reputation on the line. The code transparency service can also be used to mitigate software supply chain attacks, because it provides auditable provenance and chain-of-custody for a software bill of materials (SBoM).

Confidential Computing Is the Future of The Cloud

Confidential computing provides strong security assurances in the cloud by empowering tenants to remotely control the TCBs for their workloads: It makes explicit the (minimal) hardware, software, and services that they still need to trust; and it provides strong technical protection against any attacks from the rest—preventing potential attacks from other tenants and even from the cloud provider. This enables tenants in turn to develop and deploy their own confidential applications for their most sensitive data.

Confidential cloud computing is in its infancy, but we believe that it will eventually become ubiquitous, the same as other privacy and integrity mechanisms such as TLS and encryption at rest. The hardware for confidential computing (CPUs, FPGAs, and accelerators) should evolve at a rapid pace, as the result of a large, concerted effort across the industry aiming to provide open, robust, standardized, platform-agnostic capabilities. As these become available, they will form the foundation for a cloud fabric that will be compartmentalized with small TCBs, where each component will have access only to the information necessary to perform its function.

This foundation will support running confidential VMs, as well as higher-level platform services. While some services may never run in a fully confidential mode, they will benefit from the new, more secure foundation. As software-engineering tools evolve toward enabling compartmentalized confidential-by-default development, these guarantees will be easy to add.

Imagine a future in which end users have complete and verifiable control over how their data is used by any cloud service. If they want their organization's documents to be indexed, a confidential indexing service could guarantee that no one outside their organization ever sees that data. A confidential videoconferencing service could guarantee end-to-end encryption without sacrificing the ability to record the session or provide transcripts, with the output sent to a confidential file-sharing service, never appearing unencrypted anywhere other than the organization's devices or confidential VMs. A confidential email system could similarly protect privacy without compromising on functionality such as searching or authoring assistance. Ultimately, confidential computing will enable many innovative cloud services, while allowing users to retain full control over their data. C

References

1. AWS Nitro Enclaves. AWS; https://aws.amazon.com/ec2/nitro/nitro-enclaves/.
2. Baumann, A., Peinado, M., Hunt, G. Shielding applications from an untrusted cloud with Haven. In *Proceedings of the 11th Usenix Symp. Operating Systems Design and Implementation*, 2014; https://www.usenix.org/conference/osdi14/technical-sessions/presentation/baumann.
3. Carruth, C. Speculative load hardening. LLVM Compiler Infrastructure, 2018; https://llvm.org/docs/SpeculativeLoadHardening.html.
4. Confidential Consortium Framework. GitHub; https://github.com/microsoft/CCF.
5. Ohrimenko, O. et al. Oblivious multi-party machine learning on trusted processors. In *Proceedings of the 25th Usenix Security Symp.*, 2016; 619–636; https://dl.acm.org/doi/10.5555/3241094.3241143.
6. Open Enclave SDK. GitHub; https://github.com/openenclave/openenclave.
7. Priebe, C., Vaswani, K. and Costa, M. EnclaveDB: a secure database using SGX. In *Proceedings of the 2018 IEEE Symp. Security and Privacy*; https://ieeexplore.ieee.org/document/8418608.
8. Qureshi, M.K. New attacks and defense for encrypted-address cache. In *Proceedings of the 46th Intern. Symp. Computer Architecture*, 2019, 360-371; https://dl.acm.org/doi/10.1145/3307650.3322246.
9. Sakalis, C. et al. Efficient invisible speculative execution through selective delay and value prediction. In *Proceedings of the 2019 Intern. Symp. Computer Architecture*; https://www.researchgate.net/publication/333755760_Efficient_Invisible_Speculative_Execution_through_Selective_Delay_and_Value_Prediction.
10. Schuster, F. et al. VC3: trustworthy data analytics in the cloud using SGX. In *Proceedings of the 2010 IEEE Symp. Security and Privacy*, 38–54; https://dl.acm.org/doi/10.1109/SP.2015.10.
11. SGX-LKL. GitHub; https://github.com/lsds/sgx-lkl.
12. Volos, S. et al. Graviton: Trusted execution environments on GPUs. In *Proceedings of the 13th Usenix Symp. Operating Systems Design and Implementation*, 2018; https://www.usenix.org/system/files/osdi18-volos.pdf.
13. Werner, M. et al. ScatterCache: Thwarting cache attacks via cache set randomization. *Proceedings of the 28th Usenix Security Symposium*, 2019; https://www.usenix.org/system/files/sec19-werner.pdf.
14. Yan, M. et al. InvisiSpec: Making speculative execution invisible in the cache hierarchy. In *Proceedings of the 51st Annual IEEE/ACM Intern. Symp. Microarchitecture*, 2018; https://iacoma.cs.uiuc.edu/iacoma-papers/micro18.pdf.

Mark Russinovich is CTO of Microsoft Azure, where he leads technical strategy and architecture for Microsoft's cloud computing platform.

Manuel Costa is a partner research manager at Microsoft Research Cambridge, where he leads the Confidential Computing research theme.

Cédric Fournet is a senior principal research manager in the Confidential Computing group at Microsoft Research.

David Chisnall is a principal researcher at Microsoft Research Cambridge, where he works on hardware/software co-design for security. He is also an active open source developer, contributing to LLVM since 2008 and having served two terms on the FreeBSD Core Team.

Antoine Delignat-Lavaud is a principal researcher in the Confidential Computing group at Microsoft Research Cambridge and is researching protocols and systems for confidential cloud services built on hardware-security guarantees.

Sylvan Clebsch is a senior principal research engineer at Microsoft Research Cambridge. His work includes the Confidential Consortium Framework, Verona, and the Pony programming language.

Kapil Vaswani is a principal researcher in the Confidential Computing group at Microsoft Research. He has pioneered work on secure databases using trusted hardware and new forms of trusted hardware.

Vikas Bhatia is head of product for Azure confidential computing. Prior to Azure, he led product teams in the Windows Developer Platform group, Cloud Game Streaming, Xbox One, and Visual C++ Compiler.

DOI:10.1145/3418297

A new framework allows intelligence on mainstream end devices without special hardware.

BY HUI GUAN, SHAOSHAN LIU, XIAOLONG MA, WEI NIU, BIN REN, XIPENG SHEN, YANZHI WANG, AND PU ZHAO

CoCoPIE: Enabling Real-Time AI on Off-the-Shelf Mobile Devices via Compression-Compilation Co-Design

MANY BELIEVE THE company that enables real intelligence on end devices (such as mobile and IoT devices) will define the future of computing. Racing toward this goal, many companies, whether tech giants such as Google, Microsoft, Amazon, Apple and Facebook, or startups spent tens of billions of dollars each year on R&D.

Assuming hardware is the major constraint for enabling real-time mobile intelligence, more companies dedicate their main efforts to developing specialized hardware accelerators for machine learning and inference. Billions of dollars have been spent to fuel this intelligent hardware race.

This article challenges the view. By drawing on a recent real-time AI optimization framework CoCoPIE, it maintains that with effective *compression-compiler co-design*, a large potential is yet left untapped in enabling real-time artificial intelligence (AI) on mainstream end devices.

The principle of *compression-compilation co-design* is to design the compression of deep learning models and their compilation to executables in a hand-in-hand manner. This synergistic method can effectively optimize both the size and speed of deep learning models, and also can dramatically shorten the tuning time of the compression process, largely reducing the time to the market of AI products. When applied to models running on mainstream end devices, the method can produce real-time experience across a set of AI applications that had been broadly perceived possible only with special AI accelerators.

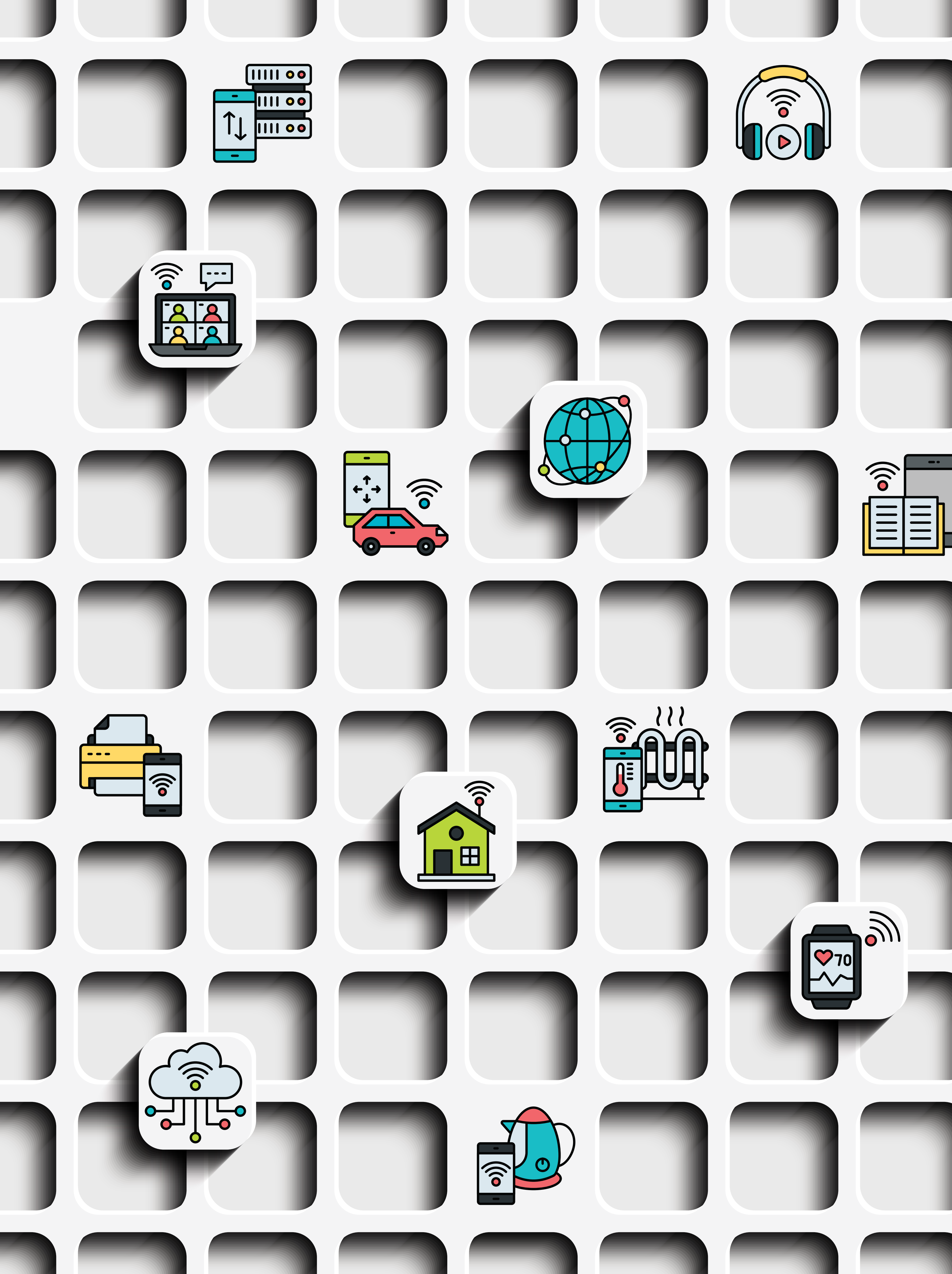
Foregoing the need for special hardware for real-time AI has some profound implications, thanks to the multifold advantages of mainstream processors over special hardware:

▸ *Time to market:* Special hardware often takes multiple years before it reaches the market. The creation of the associated compiler and system software further lengthens the process. Applications using such hardware often need to use the special APIs and meet many special constraints (for example, tiling computations to a certain size), which lengthens the time to market of AI product.

▸ *Cost:* Developing a special ASIC processor is costly and adding them into existing systems incurs extra expenses.

» key insights

- Before eagerly pursuing specialized hardware for real-time AI on end devices, it is worth giving a closer look at whether mainstream end devices are sufficient.
- For real-time AI, there is still a large potential to tap into on mainstream end devices.
- Through innovative compression-compiler co-design, the CoCoPIE framework has demonstrated a promising approach to unlocking the potential.


70

▸ *Technology maturity:* Unlike general-purpose processors, special hardware has a much smaller production volume; the technology available for their production is hence usually several generations behind general-purpose processors. Most AI accelerators, for instance, are based on 28nm to 65nm CMOS technology, with a transistor density over 10× lower than state-of-art mobile CPU or GPU.

▸ *Speed:* As a consequence of the old technology, special processors run much slower than general-purpose processors do.

▸ *Eco-system:* General-purpose processors have a well-developed ecosystem (debugging tools, optimization tools, security measures), which makes the development of high-quality applications much easier than on special processors.

▸ *Adoption:* For all these reasons, the adoption of a special processor is usually limited to the company that creates it and its few close customers. As a result, an AI application developed for the processor can be adopted by a limited number of devices.

These reasons suggest that whenever mainstream processors can meet the speed and efficiency requirements of an AI application, they should be the preferred device to consider. The current industry however puts much emphasis on special AI hardware development, based on assumed insufficiency of mainstream processors in meeting the real-time requirements. In the rest of this article, we explain why the assumption is largely biased when compression-compilation co-design is used, how the principle can be materialized effectively into a practical framework CoCoPIE, and the implications to the AI industry.

Figure 1. (a) Non-structured weight pruning and (b) two types of structured weight pruning.

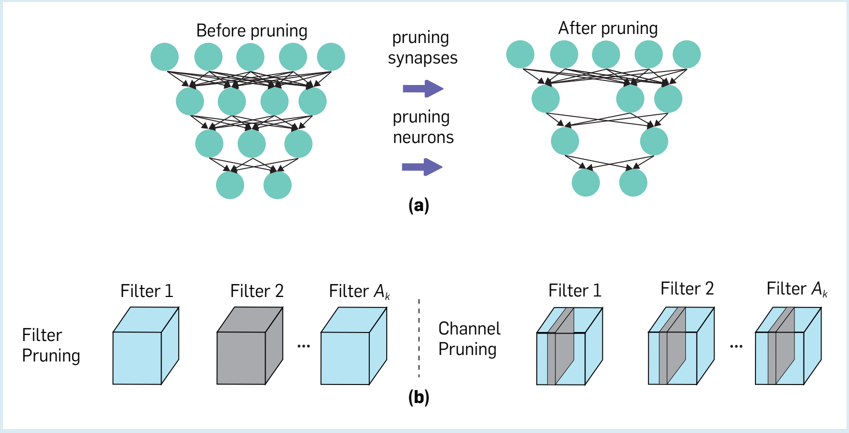
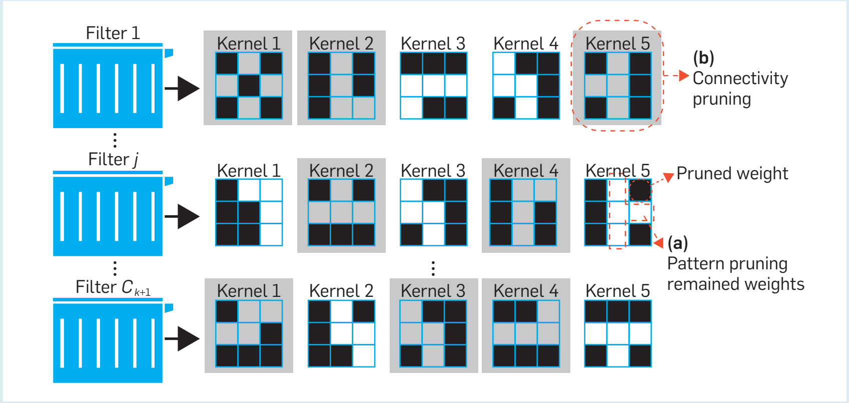


Figure 2. Illustration of (a) kernel pattern pruning on CONV kernels, and (b) connectivity pruning by removing kernels.



Compression-Compilation Co-Design: The Concept

Compression and compilation are the two key steps in fitting a deep learning model on a hardware for efficient executions. Model compression is a common technique for reducing the size and improving the speed of deep learning models. Compression techniques fall into two categories, *pruning* and *quantization*. Pruning removes layers or convolution filters or channels, while quantization reduces the precision of parameters (for example, from floating-point to short integer). Compilation refers to the process of generating executable codes from a given deep learning model. It, in essence, is a process of mapping the high-level operations in deep learning to the low-level instructions that the underlying hardware supports. The process plays a critical role in optimizing the code for efficient executions.

The principle of compression-compilation co-design is to design the two components for AI in a hand-in-hand manner. The synergy may exhibit at three levels:

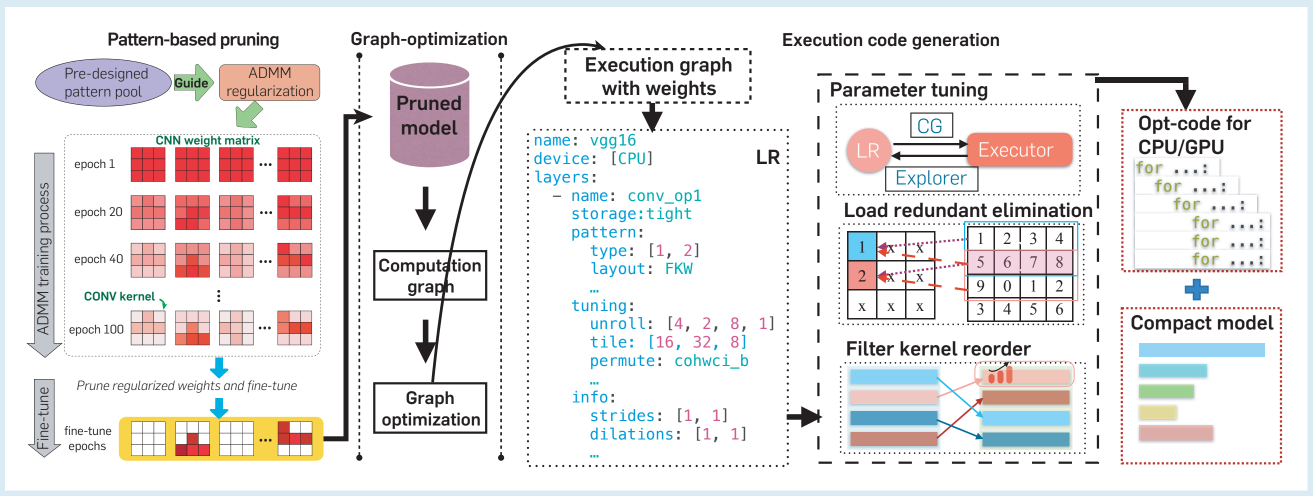
▸ *Demands/Preferences Level:* At this level, the synergy is on taking the preferences or demands of one component into consideration when designing the other component. An example is the *pattern-based pruning*, which creates a regular computation pattern amenable for the compilation step to effectively map to vector units or GPU.

▸ *Perspective/Insight Level:* At this level, the synergy is on taking the perspective or insights in the domain of one component when treating the problems in the domain of the other component. An example is the *composability-based pruning*, which generalizes the principle of composability or modularity in programming systems into a novel DNN model pruning method to dramatically reduce the needed computations.

▸ *Methodology Level:* At this level, the synergy is on closely integrating the methodology of the two components together. We will illustrate this synergy through a compiler framework that automatically generates code to enable a new way of deep learning pruning, which speeds the process by up to 180X.

All the examples mentioned are part of a software framework for mobile AI

Figure 3. Overview of CoCo-Gen acceleration framework.



```
name: vgg16
device: [CPU]
layers:
  - name: conv_op1
    storage:tight
    pattern:
      type: [1, 2]
      layout: FKW
      …
    tuning:
      unroll: [4, 2, 8, 1]
      tile: [16, 32, 8]
      permute: cohwci_b
      …
    info:
      strides: [1, 1]
      dilations: [1, 1]
      …
```

```
for ...:
  for ...:
    for ...:
      for ...:
      for ...:
      for ...:
```

named CoCoPIE. We will next give an overview of CoCoPIE based on previous publications,[8,18] and then use each of its main components to explain the compression-compilation co-design principle and the significant benefits.

CoCoPIE

CoCoPIE stands for Compression-Compilation co-design for Performance, Intelligence, and Efficiency. It is a software framework we have recently put together that holds numerous records on real-time AI on mainstream end devices in both performance and energy efficiency

CoCoPIE consists of two main components, which both reflect the Compression-Compilation co-design principle: *CoCo-Gen* generates efficient DNN execution codes via a synergy of pattern-based DNN pruning and pattern-aware code generation; *CoCo-Tune* dramatically shortens the process in identifying the appropriate set of DNN parameters to prune by a composability-based compiler framework. Here, we explain each of the two components and how compression-compilation co-design makes them possible.

CoCo-Gen: Pattern-based pruning and code generation. *Weight pruning* reduces the number of weights in DNN. As shown in Figure 1, prior weight pruning methods fall into two categories: general and non-structured pruning where arbitrary weights can be pruned; and structured pruning which prunes filters or channels in a way that produces regular and smaller weight matrices. For the better fit of regular structures for GPU/CPU executions, DNNs from regular compression have shown better speeds over those from irregular compression,[10,16,20] but are subject to more notable accuracy loss.[10,20]

We introduce a new method—*pattern-based pruning*—features fine-grained pruning patterns inside coarse-grained structures.

Figure 2 illustrates the basic idea of pattern-based pruning. For each kernel (in a CONV filter), a fixed number of weights are pruned, and the remaining weights (white cells) form specific "patterns." We define the example in Figure 2 as 4-entry pattern pruning, since every kernel reserve four non-zero weights out of the original 3×3 kernel (the most commonly used kernel). It can generalize to other kernel sizes and fully connected layers. Each kernel has the *flexibility* in choosing among a number of predefined patterns.

At theory and algorithm levels, such patterns exhibit similarities to the connection structure in human visual systems.[12,13,15] At compiler level, the known patterns allow a compiler to *re-order and generate codes* at filter and kernel level such that kernels with the same pattern can be grouped together for consecutive executions, thereby maximizing instruction-level parallelism. At hardware level, 4-entry patterns perfectly fit the SIMD architecture in embedded processors, for both CPUs and GPUs.

The selection of appropriate patterns for a kernel can be achieved via search through an extended ADMM-based framework,[15] which can be sped up via a composability-based method as we will discuss.

The method can be used together with *connectivity pruning*, which cuts the connections between certain input and output channels, to achieve even higher weight pruning/acceleration rates.

Figure 3 shows the overview of the internal workflow of CoCo-Gen.[18] After *pattern-based training* performs kernel pattern and connectivity pruning, *execution code generation* performs multiple pattern-based optimizations. Similar to other DNN compilers (for example, TVM[2]), CoCo-Gen converts DNN models into computational graphs and applies multiple graph-based optimizations. It works on a fine-grained DNN layerwise representation (LR), which captures the kernel patterns and tuning-related information. It employs an optimization, filter kernel reorder, to address two challenges of pattern-based pruning—heavy control-flow instructions, and thread divergence and load imbalance—by grouping the filters with similar lengths and patterns together. It stores DNN models in a novel form compressed weight storage, specifically designed for the kernel patterns and connectivity pruning, yielding a much better compression rate than the conventional compressed sparse row (CSR) format does. It further uses a register-level optimization, load redundancy elimination, to maximize memory performance. In sum, allowing compilers to treat pruned kernels as special patterns, the compression-compilation codesign unleashes the power of compiler optimizations for best matching DNN models with underlying hardware (see Niu et al.[18] for more details).

CoCo-Tune: A compiler framework for fast pruning. Finding out what is the best set of filters or connectivities to prune—such as the pattern-based pruning in the previous section—can be very time consuming. For a DNN with *W* filters, the entire configuration space of pruned network can be as large as $2^{|W|}$, even if only filter pruning is considered (adding pattern variations would worsen the complexity further). It often takes hours to evaluate just one configuration (that is, training the pruned network and then testing it).

CoCo-Tune is a compiler-based framework that shortens the time by up to 180X. Its inspiration comes from software engineering. More specifically, it explores *composability*, a property (fundamental in software engineering) that we discovered in the training of a collection of pruned CNN models. The basic observation is that two CNN networks in the promising subspace often differ in only some layers, and the training results of the common layers can be reused across networks to save some training time. More generally, CoCo-Tune views the networks to search as compositions of a set of building blocks (a *block* is a sequence of CNN layers). It pretrains (some of) these building blocks and assembles them into the to-be-explored networks.

To identify the best set of building blocks to pretrain to maximize the benefits, it uses a novel algorithm, which represents all layers of all to-be-explored networks as a sequence of symbols and uses a hierarchical compression algorithm Sequitur[17] to produce a context free grammar (CFG) and uses it to quickly find out the most reusable building blocks.

We integrate the techniques into a compiler-based framework, CoCo-Tune, which, for an arbitrary CNN (in Caffe Prototxt format) and other inputs, automatically generates TensorFlow code to build teacher-student learning structures to materialize composability-based CNN pruning (see Guan et al.[8] for more details).

Evaluation and demos. *Results on DNNs:* We evaluate CoCo-Gen on a Samsung Galaxy S10 cell phone with the latest Qualcomm Snapdragon 855 mobile platform that consists of a Qualcomm Kryo 485 Octa-core CPU and a Qualcomm Adreno 640GPU. Six representative DNNs are used in this evaluation, VGG-16 (VGG), ResNet-50 (RNT), and MobileNet-V2 (MBNT) trained on two datasets, ImageNet and CIFAR-10, respectively. The accompanying table characterizes these DNNs and lists the number of pruning patterns and the loss of prediction accuracy caused by our pruning. Figure 4 shows the CPU and GPU performance of CoCo-Gen compared to TFLite,[6] TVM,[2] and MNN.[1] CoCoGen outperforms all other frameworks for all cases. On CPU, CoCo-Gen achieves 12× to 44.5× speedup over TFLite, 2.3× to 8.1× over TVM, and 1.9× to 15.5× over MNN, respectively. On GPU, CoCo-Gen achieves 2.5× to 20×, 4.1× to 11.4×, and 2.5× to 6.2× speedup over TFLite, TVM, and MNN, respectively.[a] For the largest DNN (VGG) and largest dataset (ImageNet), CoCo-Gen completes CONV layers on a single input within 18.9ms on GPU, meeting the real-time requirement (usually 30 frames/sec, that is, 33ms/frame).

In terms of energy consumption, CoCo-Gen is 8.6× less than TVM. The power consumption rate of the entire mobile device is 4.1W, slightly higher than that of TVM executions, 3.8W (tested by Qualcomm Trepn power profiler). But its 9.3× less execution time leads to the large savings in energy.

a TFLite does not support executing VGG on ImageNet dataset on GPU due to its too large memory footprint.

The results also consistently outperform a number of ASIC and FPGA solutions in both performance and energy efficiency. Figure 5 demonstrates the comparison results on performance and energy efficiency with special ASIC hardware including Google's cloud TPU-V2 and edge TPU,[7] NVIDIA Jetson AGX Xavier, Cambricon MLU-100, Eyeriss,[3] and others and comparison results on accuracy and energy efficiency with the FPGA solution ESE[9] (FPGA 2017 Best Paper Award) from DeePhi. The comparisons are on the same network models, and weight quantization is not used in CoCo-Gen solution (Eyeriss and ESE use 12-bit fixed-point quantizations).

The better results of CoCo-Gen come from three reasons: the compression-compilation codesign more effectively matches models with hardware; smartphone chips are built with the most advanced technology (for example, 7nm, 11nm technology), while FPGA/ASIC solutions are based on older and less energy-efficient 28nm or 40nm technologies. Current ASIC/FPGA solutions are often optimized for a specific DNN type/size (for example, edge TPU for small-scale DNNs, Cambricon MLU-100 for large-scale DNNs), while CoCo-Gen, as a software method, can better adapt to the networks.

Real application demos: We also demonstrate the efficacy of CoCo-Gen through three interesting and key DNN applications, style transfer,[5] DNN coloring,[11] and super resolution.[4] The style transfer model is based on a generative network[23] trained on Microsoft COCO.[14] DNN coloring uses the Places scene[24] dataset to train a novel architecture that can jointly extract and fuse global and local features to perform the final colorization. The super resolution model mainly utilizes residual blocks with wider activation and linear low-rank convolution[21] trained on the DIV2K[19] dataset. With structured pruning and compiler optimization, we implement the models on a Samsung Galaxy S10 mobile phone. We demonstrate that our implementations are able to achieve real-time inference on off-the-shelf mobile device with video demos.

Figure 6 shows sample inputs and

DNNs characteristics (under kernel pattern and connectivity pruning).

Name	Network	Dataset	Layers	Conv	Patterns	Accu (%)	Accu Loss (%)
VGG	VGG-16	ImageNet	16	13	8	91.60	.1
		CIFAR-10	16	13	8	93.9−	0.4
RNT	ResNet-50	ImageNet	50	49	8	92.50	.2
		CIFAR-10	50	49	8	95.6−	1.0
MBNT	MobileNet-V2	ImageNet	53	52	8	90.30	.0
		CIFAR-10	54	53	8	64.6−	0.1

Figure 4. Performance comparison: *x*-axis: different trained DNN models; *y*-axis: average DNN inference execution time on a single input.

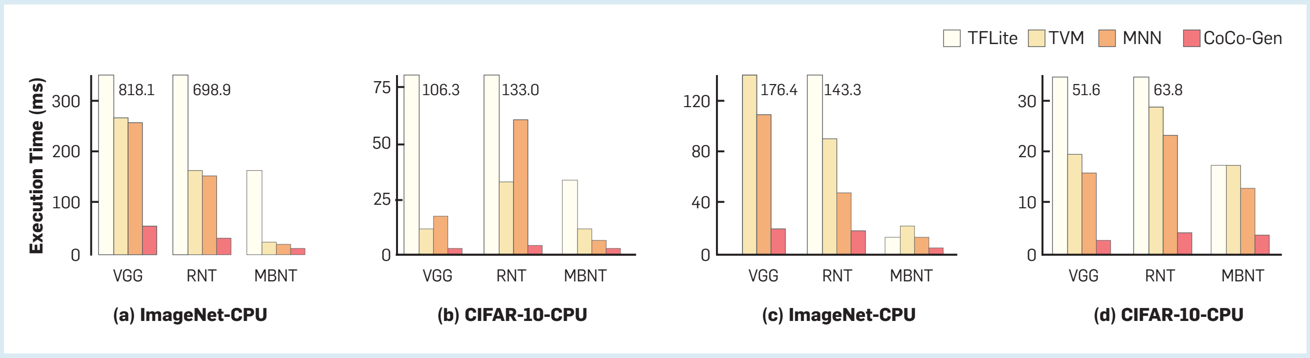
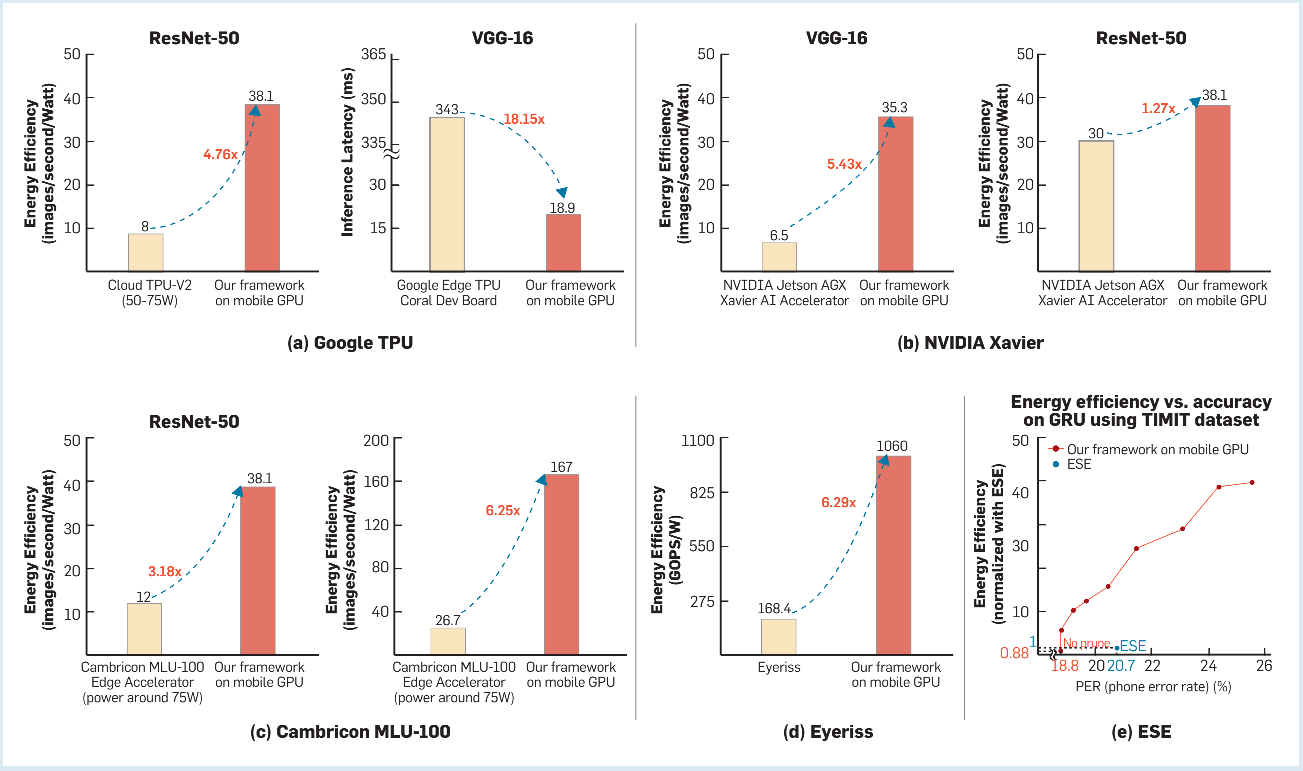


Figure 5. Comparison with existing ASIC and FPGA solutions.

(a) Comparison of energy efficiency and inference latency with Google cloud TPU.
(b) Comparison of energy efficiency with NVIDIA Jetson AGX Xavier.
(c) Comparison of energy efficiency with Cambricon MLU-100.
(d) Comparison of energy efficiency with Eyeriss. (e) Comparison of energy efficiency with FPGA solution ESE.



outputs of the three applications. According to our study,[18] CoCo-Gen on unpruned models can already outperform existing DNN frameworks (for example, TVM) in speed thanks to its advanced optimizations (for example, operator replacement and SIMD optimizations). When combined with pattern-based pruning, CoCo-Gen produces 4.2×, 3.6×, and 3.7× extra speedups on style transfer, coloring and super resolution, respectively. An inference in all the applications can complete within 75ms, demonstrating the promise of real-time performance of complex DNN applications on the off-the-shelf devices. (Please see video demos of the applications at CoCoPIE YouTube channel.[b]

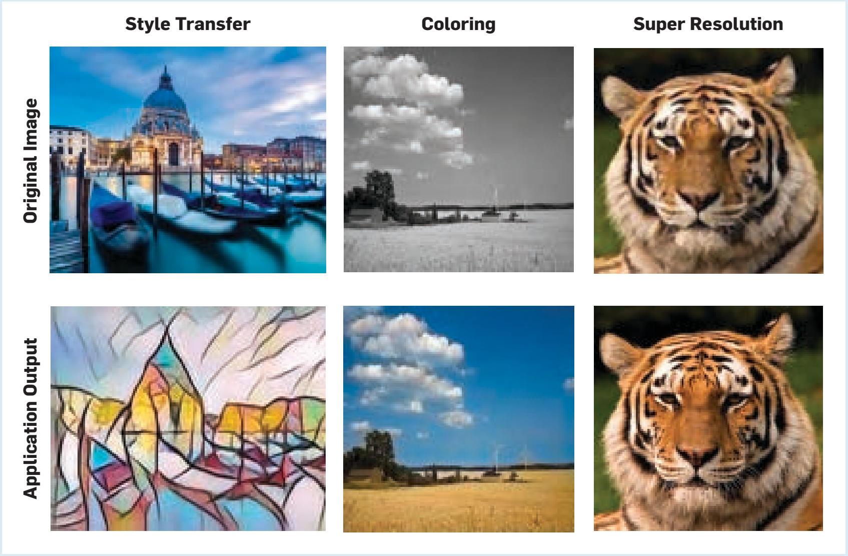
b http://www.youtube.com/channel/UCCKVDt-g2eheRTEuqIJ5cD8A/.)

Conclusion and Future Work

This article has introduced the concept of compression-compilation co-design and how it is materialized into a software framework CoCoPIE for real-time AI on mobile devices. The promising progress opens up many potential directions for future development. We briefly discuss two of them:

The first is to expand the scope of the co-design-based optimizations. So far,

Figure 6. Examples of style transfer, coloring, and super resolution implemented on our mobile device.



the principle of compression-compilation co-design has been focused on DNN models. Besides DNN, a real-world AI application often includes a lot of other parts, such as data collection, data preprocessing, the use of the DNN prediction in follow-up operations, and so on. Even though DNN may play an important role in the overall application, its optimizations may not be sufficient for the entire application to meet users' needs. So, an important direction is on how to generalize the co-design principle into holistic optimizations to the entire AI-based applications.

The second is to increase the applicability of the co-design-based optimizations. This direction relates with privacy and security. As they are two important factors in many AI model constructions and deployments, how to integrate them into the co-design process is worth pursuing. For instance, typically model pruning requires access to both the models and the training datasets, but there are scenarios where datasets may not be accessible to the model optimizer due to either privacy policies or artificial boundaries among corporations. Effective ways to circumvent these roadblocks could expand the applicability of the optimizations.[22] This direction also relates with the way the optimization framework takes to deliver its service (for example, standalone software versus cloud-based service).

In summary, the research reported in this article has provided strong evidence of the promise of the co-design principle, indicating it is possible to instill AI directly on existing commodity computing devices while offering even higher speeds and better energy efficiency than special AI accelerating hardware does. The results open new opportunities for democratizing AI capability on end devices, while cautioning against the broad perception on the indispensability of special AI hardware for real-time AI on end devices. We believe these results will prompt the industry to reexamine the directions and strategies on the pursuit of mobile AI. C

Contact info@cocopie.ai for details on the technology being commercialized through CoCoPIE LLC.

References

1. Alibaba. 2019.
2. Chen, T. et al. TVM: An automated end-toend optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation*, 2018, 578–594.
3. Chen, Y., Krishna, T., Emer, J., and Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *Proceedings of IEEE Intern. Solid-State Circuits Conf. Digest of Technical Papers*, 2016, 262–263.
4. Dong, C., Loy, C., He, K., and Tang, X. Learning a deep convolutional network for image super-resolution. In *European Conf. Computer Vision*. Springer, 2014, 184–199.
5. Gatys, L., Ecker, A., and Bethge, M. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition*, 2016, 2414–2423.
6. Google. Tensorflow lite, 2019.
7. Google Cloud TPU. Google cloud TPU, 2017; https://cloud.google.com/tpu/
8. Guan, H., Shen, X., and Lim, S. Wootz: A compiler-based framework for fast CNN pruning via composability. In *Proceedings of the Programming Language Design and Implementation*, 2019.
9. Han, S. et al. Ese: Efficient speech recognition engine with sparse LSTM on FPGA. *FPGA*, 2017, 75–84.
10. He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the 2017 IEEE Intern. Conf. on Computer Vision*. 2017, 1398–1406.
11. Iizuka, S., Simo-Serra, E., and Ishikawa, H. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Trans. Graphics 3*, 4 (July 2016).
12. Lebedev, V. and Lempitsky, V. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition*, 2016, 2554–2564.
13. Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. Pruning filters for efficient convnets. In *Proceedings of the Intern. Conf. on Learning Representations*, 2017.
14. Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, L. Microsoft coco: Common objects in context. In *Proceedings in European Conf. on Computer Vision*. Springer, 2014, 740–755.
15. Ma, X. et al. PCONV: The missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices. *AAAI*, 2020.
16. Mao, H., Han, S., Pool, J., Li, W., Liu, X., Wang, Y., and Dally, W. Exploring the regularity of sparse structure in convolutional neural networks. 2017; *arXiv:1705.08922*, 2017.
17. Nevill-Manning, C. and Witten, I. Identifying hierarchical structure in sequences: A linear-time algorithm. *J. Artif. Intell. Res. 7* (1997), 67–82.
18. Niu, W., Ma, X., Lin, S., Wang, S., Qian, X., Lin, X., Wang, Y., and Ren, B. PatDNN: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning. *ASPLOS*, 2020.
19. Timofte, R., Agustsson, E., Gool, L., Yang, M., and Zhang, L. Ntire challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition Workshops*, 2017, 114–125.
20. Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, 2016, 2074–2082.
21. Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X., and Huang, T. Wide activation for efficient and accurate image super-resolution. 2018; *arXiv:1808.08718*.
22. Zhan, Z. et al. Priv: A privacy-preserving deep neural network model compression framework. *arXiv preprint*, 2020.
23. Zhang, H. and Dana, K. Multi-style generative network for real-time transfer. 2017; *arXiv:1703.06953*.
24. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, 2014, 487–495.

Hui Guan is an assistant professor in the University of Massachusetts at Amherst, MA, USA.

Shaoshan Liu is co-founder and chair at Perceptin Inc. Mountain View, CA, USA.

Xiaolong Ma is a research assistant at Northeastern University, Boston, MA, USA.

Wei Niu is a student at the College of William & Mary, Williamsburg, VA, USA.

Bin Ren is an assistant professor College of William & Mary, Williamsburg, VA, USA.

Xipeng Shen is a professor of computer science at North Carolina State University, Raleigh, NC, USA.

Yanzhi Wang is an assistant professor at Northeastern University, Boston, MA, USA.

Pu Zhao is a research assistant at Northeastern University, Boston, MA, USA.

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/cocopie

DOI:10.1145/3418292

The requirements engineer role is defined differently within most organizations.

BY XAVIER FRANCH, CRISTINA PALOMARES, AND TONY GORSCHEK

On the Requirements Engineer Role

REQUIREMENTS ENGINEERING (RE) is a critical area in software development, as figuring out what to develop and include in a product is a cornerstone activity which all others depend upon. Countless studies of unsuccessful development projects report that lack in RE is often a core-contributing failure factor.[13] Central in RE is the role that coordinates all its related activities, usually named requirements engineer.

Still, empirical evidence on the way companies implement this role is scarce. In this article, we present the results of an interview-based descriptive study involving 24 IT professionals from 12 companies. As a main outcome, we can affirm that all companies assign IT professionals to the requirements engineer role in their projects, but in many different ways, which might impact efficiency of the function. Furthermore, we uncover that requirements engineers often perform other tasks ranging from project's go vs. no-go decisions to test suite design in addition to handling requirements. Last, the study highlights their need to communicate with many other roles inside the company, from domain experts to system architects.

According to the International Requirements Engineering Board (IREB), a requirements engineer is "a person who—in collaboration with stakeholders—elicits, documents, validates, and

» key insights

- **Organizations nominate requirements engineers in virtually all of their projects, but there is a large diversity in their assignment, from a single person playing the role to several people that collaborate during the project.**
- **Besides RE-specific tasks, requirements engineers may perform other activities, ranging from project management to testing, often as a consequence of requirements engineers playing multiple roles.**
- **In our study, the requirements engineer role is better delimitated in large companies. No other influencing factors were identified, particularly, the development method adopted in the company does not have impact on the requirements engineer role.**

Research Methods

We designed a qualitative study based on semi-structured interviews. We asked the respondents to focus on a single project that they were familiar with. The selected projects were related mainly to embedded systems, websites or mobile applications, and customer business support. Project duration varied from four months to several years and involved from two to up to thousands of people. In order to understand the representativeness of the answers, following Patton's advice,[14] we regularly asked follow-up questions such as "Is this typically how you do this? If not, how do you usually do it?"

The questionnaire used as guideline for the interview had a broad scope, including questions about detailed aspects on elicitation[11] and documentation of requirements. In this paper, we just focused on understanding the requirements engineer. The evidence gathered mainly came from three questions made to all the respondents: "Does the role of "requirements engineer" exist in your company? Is this role played by some designated person or instead it is a hat that a person wears at some moment, and later this very person may become, for example, tester? What are the main responsibilities of this role?" Furthermore, given the nature of semi-structured interviews, some respondents provided related information as part of their responses to other questions. However, we analyzed the responses to the other questions because they also contained some comments related to the objectives of this study.

We coded the responses applying several steps. First, we coded descriptive information, for example the respondent experience. Second, we defined a provisional list of codes coming from a previous survey as baseline.[12] Third, we extracted codes from the responses. Last, we combined similar codes to establish emerging categories and relationships among them. The codes and categories formed a hierarchy (see the figure in this article that summarizes the results of the study).

The full protocol of the study is available at https://tinyurl.com/y3lbpone.

manages requirements."[3] There are several synonyms in place, most of them using the term "analyst," like "requirements analyst," "business analyst," "system analyst," or even simply "analyst."[16]

While the complexity and criticality of RE activities call for such a role,[13] there is not a vision shared in industry about its responsibilities and in fact, its existence as a separate role is not always clear. Ten years ago, Paech started her paper "What is a Requirements Engineer?"[10] stating that "Rarely is there a role called requirements engineer." Afterward, Hermann seconded this view arguing that "in many organizations, the role of the requirements engineer is not defined clearly."[5] Even recently, Wang et al. informed that in spite of practitioners framing requirements engineer as a profession, "there is a significant incongruity regarding the perceptions of the requirements engineering role, tasks, and responsibilities in the IT marketplace."[15] Because of this incongruity, the way in which the requirements engineer is assigned and works in practice may vary largely depending on the organization.

To understand this phenomenon, we have performed a descriptive study investigating the requirements engineer role in the context of industry. Other studies focus on understanding the skills, the competences, and even the educational background of requirements engineers.[1,2,5,15] Instead, we focus on the management of this role in companies. Questions we ask are among others: Who plays the requirements engineer role? How do companies assign IT professionals to this role? How does it relate to other roles? What activities do requirements engineers perform? The answers gathered in the study may help companies to know practices in place that they can eventually adopt to improve their current way of working.

The study. We conducted an interview-based descriptive study with 24 IT professionals in 12 Swedish companies from our local network of industry contacts. Descriptive studies allow investigating a given object, without the commitment of explaining the findings. Along this line, our study serves as an instrument to learn how things work in practice. (See the sidebar "Research Methods" for more details.)

The table here summarizes the most relevant information about the respondents and their companies. Half of the respondents hold a Computer Science or Information Systems degree (BSc or MSc). They occupy different positions, and we knew in advance that all of them are involved in RE activities, although in most cases we didn't know all details of their role.

In the rest of the article, we will refer to respondents using the notation S*x*[*Y*], where S*x* is the respondent's identifier and *Y* is the company's identifier.

Do organizations assign an IT professional to the role of requirements engineer? Most of the respondents explicitly confirmed the existence of the role of requirements engineer in their organizations, dedicated mainly to RE-related tasks. The role was concisely defined by S1[A] as: "A role played by some designated person which in that project is in charge of requirements."

There were a couple of respondents answering "no" to the direct question, but in the explanation that they provided, it became clear that they confused the concepts of "role" ("a function or part performed especially in a particular operation or process"[8]) and "position" ("an employment for which one has been hired"[8]). For example, S5[C] said: "No, the role didn't exist. It was a hat that a person wore when necessary, who did the tasks of a requirements engineer … The main responsibilities when this person was working as a requirements engineer were …" From the later description, it became clear that S5, as the rest of respondents, had a requirements engineer role in his project.

Only respondent S17[I] reported that sometimes nobody played this role in his organization: "The role only exists [in my organization] if the gathering of requirements is necessary." The justification of this response is that company I acts sometimes as subcontractor to perform development activities in external projects, therefore requirements are managed in the contracting company.

How is the requirements engineer role staffed? There exists a large variability in the way that practitioners are assigned to the requirements engineer role. In its simplest form, only one person acts as requirements engineer in a project. This situation was reported by 12 respondents. Three main variations were mentioned:

▸ The organization assigns a person as requirements engineer due to some other role or position s/he currently plays in the project or organization. For instance, S3[B] (and similarly S4[B]) reported that "it is always the same person, the system analyst," while S9[E] informs that "it is a hat worn by the business analyst" and S14[H] and S15[H], "the system engineer." S20[J] mentions two other candidates, namely the system developer and the designer.

▸ The organization assigns a person as requirements engineer in a case-by-case basis, as stated by S1[A], S10[F], S17[I] and S19[J]. S1 informs about a specific name for the role, namely "requirements lead."

▸ It is the client organization that assigns the requirements engineer. Respondents S12[G] and S13[G] reported such situation because "my organization works as a provider of systems or solutions, and the client is the one in charge of providing the requirements." Also as reported here, S17[I] identified this situation for some projects in her organization.

Eight respondents reported that more than one person has assigned the requirements engineer role. Again, we distinguish several situations:

▸ The requirements engineer role is split into different roles that act at different moments. We found two similar situations:

▹ Respondents S6[D] and S7[D] reported one department and one role for managing requirements: the global service department and the system manager. The global service department "manages the business requirements for all the systems in the organization" while the system manager "knows the requirements of a specific product" and acts as a "spoke person for the main requirements from a technical point of view," playing a role similar to that of a project owner in agile development projects. The main reason behind sharing requirements at these two levels is their need to manage all the

Respondents, their projects, and their companies.

Respondent				Project	Company		
R- ID	Highest Educational Background	Years in Industry	Job Position	Method	C-ID	Size*	Main business area
S1	BSc in Computer Science	15	Business Analyst	Waterfall	A	Large	IT Department
S2	MSc in Computer Science	15	Project Manager	Waterfall			
S3	Technical BSc	20	System Analyst	Agile	B	Large	Software Consultancy Company
S4	BSc in Computer Science	13	Requirement Analyst	Agile			
S5	MSc in Computer Science	25	Requirement Analyst	Waterfall	C	Medium	Software House
S6	Technical BSc	20	System Manager	Agile	D	Large	Software House
S7	MSc in Computer Science	19	System Manager	Agile			
S8	BSc in Computer Science	15	Senior Project Manager	Waterfall	E	Very Large	Software Consultancy Company
S9	Technical BSc	20	Senior Business Consultant	Waterfall			
S10	MSc in Computer Science	16	Senior Developer	Agile	F	Small	Software Consultancy Company
S11	Technical MSc	17	Consultant Manager	Agile			
S12	Other MSc	12	Solution Designer	Waterfall	G	Large	Software Consultancy Company
S13	BSc in Computer Science	23	Business Analyst	Waterfall			
S14	Other Ph.D.	10	System Engineer	Waterfall	H	Very Large	IT Department
S15	Other MSc	10	System Engineer	Waterfall			
S16	Technical BSc	25	Product Manager	Agile	I	Very Large	Software House
S17	Technical MSc	8	System Engineer	Waterfall			
S18	Technical MSc	9	Project Leader	Waterfall	J	Very Large	IT Department
S19	Technical MSc	3	Lead Engineer	Waterfall			
S20	Other Ph.D.	23	Software, Manufacturing an Electrical Engineer	Waterfall			
S21	MSc in Computer Science	21	Senior Consultant	Waterfall	K	Large	Software Consultancy Company
S22	Technical BSc	9	Senior Consultant	Agile			
S23	Technical BSc	15	Assignment Manager	Waterfall	L	Large	Public Administration
S24	BSc in Computer Science	26	Requirements Engineer	Waterfall			

* The meaning of the categories is: Small = up to 100 employees; Medium = up to 500 employees; Large = up to 10,000 employees; and Very Large = over 10,000 employees.

main products of the organization in a holistic way.

▹ Respondent S18[J] informed about a system constructor role "who is responsible of translating from function owner requirements (high-level goals) to system requirements … distributed to different modules of the platform" and a requirements manager "who is the person who has a general view of the requirements at the module level." Similarly, S5[C] reported a system analyst closer to the customer, and an interaction designer for elaborating the initial requirements. In both cases, the assignment of particular people to these roles is made case-by-case.

▸ The requirements engineer role is exerted by several people that collaborate during the project. In all the cases, one of the people had a clear lead:

▹ S16[I] reported the product manager did most of the RE work but had the support of a person taking care of a repository of safety requirements, given the importance of this particular type of requirements in company I.

▹ S8[E] reported three roles involved in RE: "The system analyst, the project manager, and the system architect, and these people are also doing other tasks, so it is a hat that a person wears at some moment."

▹ Also, three roles were mentioned by S11[F]: system architect, who is the "person that has more responsibility over requirements;" interaction designer, in charge of requirements related to user interface; and developers, who mainly "add ideas related to technical requirements in the project meetings."

▹ S22[K] informed about a non-complete list of people acting as requirements engineers as needed: product owner, project leader, architects, interaction designers, and so on. However, even if "all the responsibilities are dispersed between these roles," still "the project leader is the main responsible for the requirements."

Last, we observed how context factors may influence in the assignment and performance of the requirements engineering role. Four respondents informed that the concrete way in which the requirements engineer role is covered depends on some context factor. Respondents mentioned specifically project size and current workload:

▸ S23[L] and S24[L] reported that a dedicated person is assigned or not depending on the size of the project: "For large projects, it is a designated person. For small projects, it is a person that later will do further tasks."

▸ S2[A] reported a similar situation with the addition that "for huge projects there could be more than one requirements analyst in the project."

▸ In the same pace, S21[K] informed that usually the product owner or product manager acts as requirements engineer, but "If that person has too much work, they can pass that responsibility to other persons, like consultants or testers." Furthermore, for some big projects or own development projects they appoint a specific business analyst.

What are the activities conducted by people playing the role of requirements engineer? Respondents in our study mentioned requirements engineers to be responsible of the usual RE-related activities: elicitation, negotiation, documentation/specification, validation and change management. However, except in the simplest cases as S1[A], the role "is a hat that a person wears at some moment, but then these people change to do other things" (S10[F]). Consequently, when a system architect or a developer is assigned as requirements engineer, she still needs to design the architecture or develop new software, while participating as requirements engineer when required. This can pose a real problem as it brings a solution focus very early on, especially when high-level requirements should be broken down to concrete needs. In fact, the elaboration of high-level requirements into more detailed ones (that is, "understanding what has to be done from this big first requirement," as said by S7[D]) is mentioned often by the interviewees as a challenge.

Furthermore, some respondents mentioned particular activities at different abstraction and granularity levels that we mention here:

▸ Probably the most critical action was mentioned by S5[C]. In his company, the requirements engineer is

In one respondent's company, the requirements engineer is entitled to "researching the possibility of developing a solution taking into account what the customer wanted," that is, a go vs. no-go decision.

entitled to "researching the possibility of developing a solution taking into account what the customer wanted," that is, a go vs. no-go decision.

▸ Whenever necessary, the requirements engineer may be requested to provide a business view. S21[K] mentions as responsibilities "defining a business model, defining how the return on investment is achieved by the specified requirements."

▸ Some activities arise due to the nature of the organization. For example, respondent S7[D] works in a market-driven company. Therefore, one of the two roles managing requirements, the system manager, needs to do "quick studies or pre-studies and based on that they select what to put in their requirements and how to scope the solution."

▸ Requirements engineers may sometimes be assigned to perform project management tasks. As reported by S3[B], the reason may be that "there is no clear barrier of who should do these tasks."

▸ Interaction inside the organization may be necessary for the requirements engineers. S19[J] reports among the main responsibilities "talking to other groups to get input or data necessary from other systems."

Study Validity

As any other empirical study, the results of this work need to be taken with caution. First of all, the sample of respondents is very small compared to the community of practitioners that work in RE. Therefore, generalization is not possible. However, this is not the intention of this type of studies. The real goal is to gain evidence that can be aggregated with other previous studies to understand better the phenomenon under study. As an indicator about how difficult is trying to generalize, in this study we observed that even subjects from the same company answered differently to some of the questions made. Although surprising in a first glance, this is justified because they can occupy different positions or participate in different types of projects.

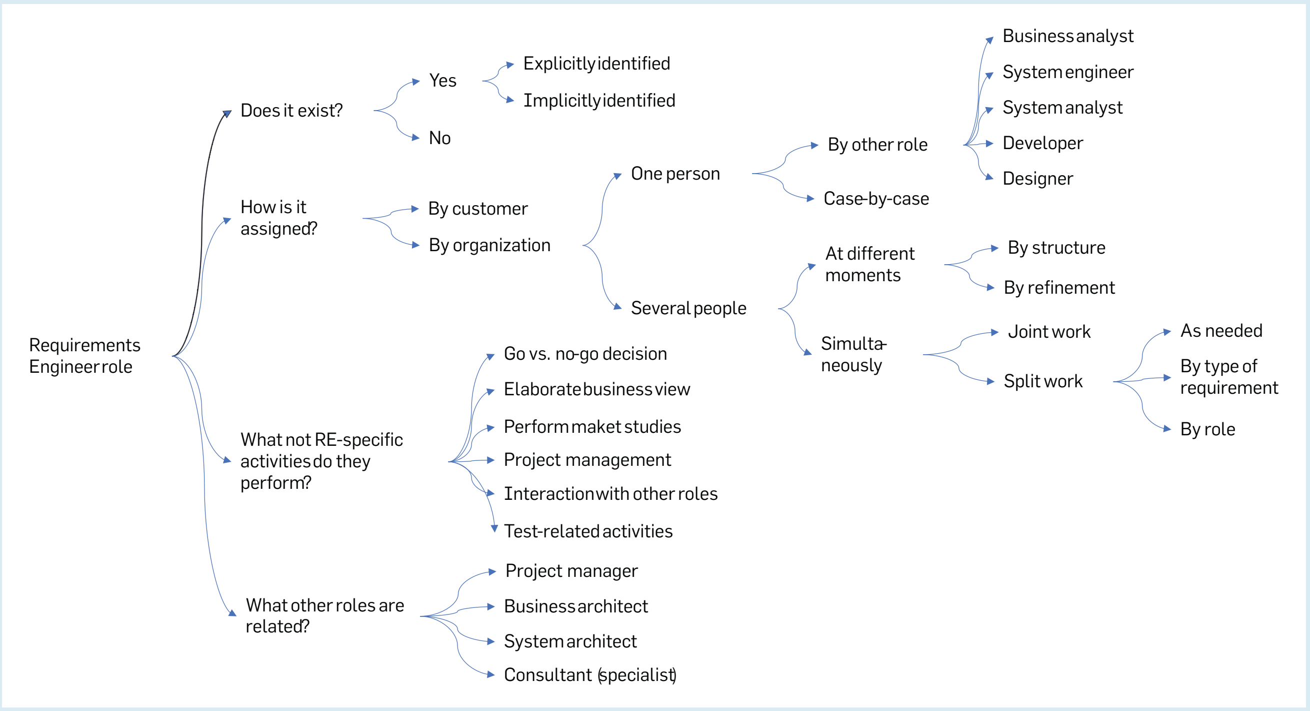
Furthermore, given that the study is based on evidence, there are several threats that may have impacted the results. For instance, the way in which the interviews were made, the fact that the interviews were made in English and not in Swedish, or the tacit information that respondents may have not provided in the interviews. In order to mitigate this and other threats, we carefully designed and piloted a protocol that was used throughout the study (available at https://tinyurl.com/y3lbpone).

▸ Some of the respondents include test-related activities as part of the requirements engineer responsibilities. S2[A] and S3[B] mention the design of tests, while S6[D] reports the specification of tests in general. S5 also run the tests for some requirements.

Which other roles interact with the requirements engineer? When discussing the requirements engineer role, other functions and roles were also lifted by our respondents. A representative example comes from S1[A]. As already mentioned, in the case of this company, the requirements engineer only performs RE-related tasks, so she needs to communicate with other roles: the assignment lead ("it is kind of a project manager but inside the team, working hand to hand with them") and the business architect. The requirements engineer (called system analyst in this organization) is constantly interacting with both of them.

Another typical role reported as important for the requirements engineer is that of system architect. System architects collaborate typically in verification activities, as reported by S5[C]:

Summary of responses in the study.



"[Requirements engineers were in charge of] securing and verifying together with system engineering architects the technical accuracy of the requirements responses provided to the customer." With a similar function, S23[L] and S24[L] report the use of specialists and consultants in their organization, who "know a lot about technical aspects, so they help requirements engineers to understand the technical aspects when specifying requirements."

Some of these roles may be assigned depending on the context. Size is one of them, as reported by S3[B]: "If the project is quite big, the organization even has a project leader for each one of the different stages of the project: requirements, development, etc."

Discussion

The information gathered from the 24 respondents has been very useful to gain insights in the requirements engineer role from the perspective of the organization (See the sidebar "Study Validity" for more details). The figure here summarizes the results as they have emerged from these answers. Here we report some observations:

Large diversity in the staffing of requirements engineers. This is an aspect not sufficiently addressed in the literature that deserves more attention. With so many options, an organization may be hesitant about the best way to proceed. For example, putting several people together to play the role of requirements engineer can be considered beneficial because they provide their own expertise, skills, and judgment and therefore improve the overall quality of the RE process. On the other hand, it may give rise to communication problems, which is one of the typical challenges reported by respondents in this study. For example, S5[C] declared that "there were requirements missing at the end (incompleteness) and some misunderstandings (ambiguity). Especially the problems were with the communication between the two roles related to requirements, as the interaction designers were using the requirements specified by the requirements analysts to create a new more detailed version of the requirements." This observation aligns with the study by Calzanas et al.,[1] which reported good

It may be argued that companies opting to share the role of requirements engineer among several people should be ready to invest resources in training their soft skills.

written and oral communications as two of the most frequently demanded requirements engineers' skills in the Brazilian market. Additional studies have also shown that written/oral comprehension and communication is one of the skills with more difficulties for requirements engineers;[9] therefore, it may be argued that companies opting to share the role of requirements engineer among several people, should be ready to invest resources in training their soft skills.

Conversely, in the companies that reported a single person playing the requirements engineer role in the project, we found two possibilities reflecting two opposite views on considering RE. Having the same person across several projects is an indicator that RE is recognized as critical in the company. Instead, assigning a person opportunistically in a case-by-case basis may imply at the end that nobody in the company will have the competences required in performing RE activities. Unless it is well implemented and supported, this rotation on the role may turn into an impediment to have continuity in RE competences over projects and thus is an impediment to having a holistic view about the requirements of the company product portfolio. This is especially true considering emerging competences needed to develop complex systems in dynamic environments, such as contextual intelligence and ability to act in complex situations, like learning to learn, sensemaking, mindfulness, and facilitate leadership.[7] These competences are not easy to acquire and require some training that companies may not provide to all their employees when they become assigned as requirements engineer in one project. If the company cannot provide such training to several people, having the same person across several projects seems the best option.

Large variability of non-specific RE activities performed by requirements engineers. Some of these activities are a consequence of the requirement engineers playing multiple roles. For example, making a go vs. no-go decision is aligned with REs assuming some product management responsibilities. This overload may have a negative impact on the RE phase. The fuzzy barrier with project management tasks may be the root cause of a challenge reported by respondents, namely the suboptimal quality of requirements documents: at the end, some requirements may be missing, or their quality may be inadequate (in terms of, say, ambiguity or incompleteness).

Other activities identified in the study connect well with concrete development strategies. Some respondents informed about the requirements engineer being involved in testing activities. This aligns well with the principles of test-driven development, where requirements are quickly turned into test cases. However, this is not always easy to get. S1[A] reported that "The level of specification [of the requirements] was good enough for developers, because they participated in the discussions around requirements, but not for testers. More information or details were missing so testers were not able to completely understand the requirements."

One observation which can be important is that the role assignment, and who gets the assignment in cases of being ad-hoc and/or opportunistic, can be a source of challenges. Due to the demands of the role, working with items from technical depth, to business aspects, the competence of the person needs to be fairly high. In addition, the coordination responsibility calls for demand of personal contacts in the development organization.

Development method not a determinant when it comes to the requirements engineer role. To start with, both companies following an agile approach and other companies more on the "waterfall-ish" side, reported the existence of the requirements engineer role in their projects. The appointment of requirements engineers in agile projects aligns with the observation by Heikkilä et al.[4] who justified the introduction of this role to help with problems with client or customer representatives. We observe there is an influence of the method on the way of staffing the role: the majority of projects in which respondents reported more than one person assigned as requirements engineer were developed Agile, while for the rest of staffing situations, waterfall projects dominated. In the extreme case, the method is a determinant for the contextual appointment of requirements engineer: the four respondents who reported this situation (S2, S21, S23, and S24) used a waterfall method in their projects.

The development method did not appear to be determinant in the other parts of the study. The main challenges reported by our practitioners were instability of requirements (and especially changes in prioritization), the problem of hidden needs, and different issues related to the requirement process, like effort estimation or definition of project scope. All these challenges are reported by some existing studies in agile practices,[4,6] together with others that our respondents do not experience (for example, inappropriate architecture[6]/growing technical debt[4]). However, in our study, these challenges are mentioned by respondents regardless of the development method. For instance, Inayat et al.[6] mention requirements changes as a challenge in agile projects, but in our study, this is mentioned by respondents from companies that work under a waterfall approach, for example, respondents S12, S13, and S17. We may conclude that at the end, requirements engineers must be prepared to face similar challenges regardless the development approach.

Influence of the company size. In order to find determinants in some RE characteristics, we investigated the influence of several factors. We observed the size of the company influences several aspects. Particularly, the nine respondents working in four very large companies (that is, with more than 10,000 employees) reported that REs do not interact with other roles in their projects, the staffing of personnel to the role does not depend on contextual factors, and except in one case, the requirement engineers do not perform tasks unrelated to RE. Considering the three facts together, we can say that the role of requirements engineer is better delimitated in very large companies than in the rest.

Conclusion

This study shows how the role of the requirements engineer in IT companies is elusive (in the sense of "hard to comprehend or define"[8]): every company may understand it and implement it in a different way often dependent on context. However, understanding of the qualifications and level of seniority needed for the role, and to what extent this is a central factor in role assignment is unclear, but possibly critical. With this paper, we hope to shed some light on this critical problem.

Acknowledgments. This work gas been partially supported by the GENESIS project, TIN2016-79269-R. Tony Gorschek's work has been supported by KKS Funded Software Engineering ReThought (SERT) Research Profile @ SERL-SWEDEN, BTH.

References

1. Calzanas, A. et al. Software requirements analyst profile: A descriptive study of Brazil and Mexico. *RE*, 2017.
2. Chang, C.K., Christiansen, M. Blueprint for the ideal requirements engineer. *IEEE Software 13*, 2 (1998).
3. Glinz, M. A glossary of requirements engineering terminology, Ver. 1.7. *IREB* (May 2017).
4. Heikkilä, V.T., Damian, D., Lassenius, C., Paasivaara, M. A mapping study on requirements engineering in Agile software development. SEAA, 2015.
5. Herrmann, A. Requirements engineering in practice: There is no requirements engineer position. REFSQ, 2013.
6. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., S. Shamshirband, S. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior 51*, B (2015).
7. Jantunen, S., Dumdum, R., Gause, D.C. Towards new requirements engineering competencies. CHASE, 2019.
8. *The Merriam-Webster English Dictionary*; https://www.merriam-webster.com/
9. Morales-Ramírez, I., Alva-Martínez, L.H. Requirements analysis skills: How to train practitioners? *REET@ RE*, 2018.
10. Paech, B. What is a requirements engineer? *IEEE Software 25*, 4 (2008).
11. Palomares, C., Franch, X., Quer, C., Chatzipetrou, P., Lopez, L. and Gorschek, T. The state-of-practice in requirements elicitation: An extended interview study of 12 companies. *Requirements Engineering J.*; DOI: 10.1007/s00766-020-00345
12. Palomares, C., Quer, C. and Franch, X. Requirements reuse and requirement patterns: A state of the practice survey. *Empirical Software Engineering 22*, 6 (2017).
13. The Project Management Institute. Pulse of the Profession® In-Depth Report: Requirements Management—A Core Competency for Project and Program Success. 2014.
14. Patton, M.Q. *Qualitative Research & Evaluation Methods.* Sage Publications, 2002.
15. Wang, C., Cui, P., Daneva, M., Kassab, M. Understanding what industry wants from requirements engineers: An exploration of RE jobs in Canada. *ESEM* 2018.
16. Wiegers, K.E., Beatty, J. *Software Requirements, 3rd Ed.* Microsoft Press, 2013.

Xavier Franch (franch@essi.upc.edu) is a professor in the Service and Information System Engineering department at the Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain.

Christina Palomares (cpalomares@essi.upc.edu) is a researcher in the Service and Information System Engineering department at the Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain.

Tony Gorschek (tony.gorschek@bth.se) is a professor in the Software Research Engineering Lab (SERL) at Blekinge Institute of Technology, Karlskrona, Sweden.

DOI:10.1145/3417517

A new dataset significantly revises both scholarly assessment and popular understanding about gender bias in computing.

BY THOMAS J. MISA

Dynamics of Gender Bias in Computing

IN MAY 1948, women were strikingly prominent in ACM. Founded just months earlier as the "Eastern Association for Computing Machinery," the new professional society boldly aimed to "advance the science, development, construction, and application of the new machinery for computing, reasoning, and other handling of information."[36] No fewer than 27 women were ACM members, and many were leaders in the emerging field.[a] Among them were the pioneer programmers Jean Bartik, Ruth Lichterman, and Frances Snyder of ENIAC fame; the incomparable Grace Murray Hopper who soon energized programming languages; Florence Koons from the National Bureau of Standards and U.S. Census Bureau; and noted mathematician-programmer Ida Rhodes.[26] During the war, Gertrude Blanch had organized a massive human computing effort (a mode of computation made visible in the 2016 film *Hidden Figures*[47]) and, for her later service to the US Air Force, became "one of the most well-known computer scientists and certainly the most visible woman in the field."[24,25] Mina Rees, a mathematics Ph.D. like Hopper and Blanch, notably funded mathematics and computing through the Office of Naval Research (1946–1953), later serving as the first female president of the American Association for the Advancement of Science. In 1949, Rees was among the 33 women (including at least seven ACM women) who participated in an international conference at Harvard University, chairing a heavyweight session on "Recent Developments in Computing Machinery."[29]

Their prominence has led to the widespread but inaccurate impression that women were numerically dominant in early computer programming. As one account puts it, "at its origins, computer programming was a largely feminized occupation."[18,19] This view, resting on suggestive but fragmentary data, has become prominent in popular culture, scholarship, and mass media, including the *Wall Street Journal* and National Public Radio and the widely acclaimed 2015 documentary "Code: Debugging the

[a] The May 1948 ACM membership roster is in Margaret R. Fox papers (Charles Babbage Institute 45; purl.umn.edu/41420) box 2, folder 9; other ACM rosters in Frances E. Holberton papers (CBI 94; purl.umn.edu/40810) box 23.

» key insights

- Underrepresentation of women in computing persists, despite energetic reform efforts. To guide strategies for change, we need deeper insight into the changing dynamics of gender bias.
- Analyzing archival data from ACM membership rosters and conference attendees as well as six prominent computer user groups (1950s–1990s) created a new longitudinal dataset of N=50,000.
- This data fills in 'white space' prior to 1970 U.S. government statistics—demonstrating three distinct periods of gender bias since the 1950s while correcting public understanding and scholarly assessment.
- Cultural changes in the 1980s led to today's gender bias in computing—a contingent (not inherent or permanent) result of professionalization.

Gender Gap" by Robin Hauser Reynolds.[14,41] The film popularized the conjecture by some scholars that "women made up 30% to 50% of all programmers" in the 1950s or 1960s and that male programmers subsequently pushed them out. Porter writes, "By the 1960s, women made up 30% to 50% of all programmers, according to [historian] Ensmenger" (specifically citing the Robin Hauser film).[46]

A recent article[45] in *Communications of the ACM* approvingly cites one such source positing a binary switch from a female-dominated field to a male-dominanted one. There, Mundy clearly states the linear view: "after World War II, software programming was considered rote and unglamorous, somewhat secretarial—and therefore suitable for women. The glittering future, it was thought, lay in hardware. But once software revealed its potential—and profitability—the guys flooded in and coding became a male realm."[43] It seems widely accepted that men actively remade computer programming from a female- into a male-dominated field during the 1960s or 1970s just as computer science was professionalizing itself through expansion of research, professional societies, and higher education. This accepted view posits that computing was *born female* and then *made masculine*, with a simple linear dynamic leading straight to today's male-dominated profession. One implication of this conjecture is that gender bias was an inherent part of (male-driven) professionalization in computing. In varied forms, "many computer programmers embraced masculinity as a powerful resource for establishing their professional identity and authority," in Ensmenger's formulation.[19]

The 'Linear Model' Is Too Simple

In the absence of systematic data on gender in the computing workforce, prior to the 1970 U.S. Census, such a linear model once seemed plausible.[38] It was furthermore supported by fragmentary and sometimes cherry-picked evidence and buttressed by theoretical claims about the nature of professionalization.[b] But it is too simple. To start, we need systematic, longitudinal data. For deeper insight on women in computing during these years, this article presents a new dataset with more than 50,000 individuals tabulated by their first (given) names, an indicator of ascribed gender (if not gender identity). The results may be surprising. In 1948, the 27 named ACM women, alongside 330 named ACM men, constituted 7.6% of its membership. Similarly, women were 8.6% and 7.6% of ACM members in 1949 and 1952; and women constituted 7.6% and 5.3% of ACM conference attendees in 1950 and 1952. Women were 5.7% of the 1949 Harvard conference. A retrospective celebration[50] suggests women were 12.7% of the Univac pioneers from 1951 (see Figure 1).[c] This data does not support the common conjecture that women numerically dominated early computing.

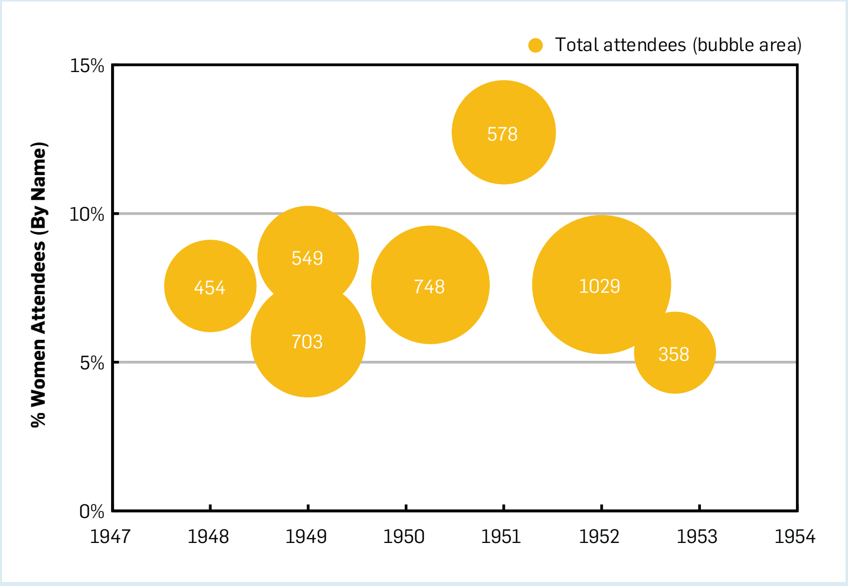
The "pipeline" model is a related linear view, now widely criticized. In Berryman's influential 1983 Rockefeller Foundation report,[5] the pipeline metaphor helped identify the *different* reasons for underrepresentation in the quantitative sciences of African Americans, Hispanics, and American Indians, with structural "losses from the educational pipeline" beginning in high school as well as personal "field choices" (for example, college major) shaping patterns of underrepresentation. For computer science, Camp expanded on Berryman's findings for women that losses were concentrated in a latter stage (from bachelor's to doctoral degrees).[12] In computing, the pipeline model posited a one-way decline of women, from the 1980s, noting that the proportion of women "fell" at each career "stage" from undergraduate student through graduate school and on to full professor. Moshe Vardi recently voiced concern about "puncturing the recruiting pipeline."[51]

A recent critique asks: "What's wrong with the pipeline? Everything. The pipeline assumes a passive flow of women (and men) from one stage to the next culminating in a scientific career. Women's underrepresentation in science results then from their leakage from the pipeline."[9] Such a linear model inadequately acknowledges women's diverse career paths and non-academic career stages, better conceptualized as non-linear "pathways." Fox and Kline caution that "women may linger as tenured associate professors without attaining full rank" and so not fully participate in academic decision-making

b Ensmenger's 1974 source for "reliable contemporary observers"[18,19] claiming 30-plus percent women programmers in fact mentions women on just two pages: a certain single IBM programming group; and a conjecture on women in the "moderating role of 'mother'."[13]

c See UNIVAC Conference 1990, CBI OH 200; purl.umn.edu/104288; and "NCC 1981 Pioneer Day;" http://bit.ly/3sim3UT.

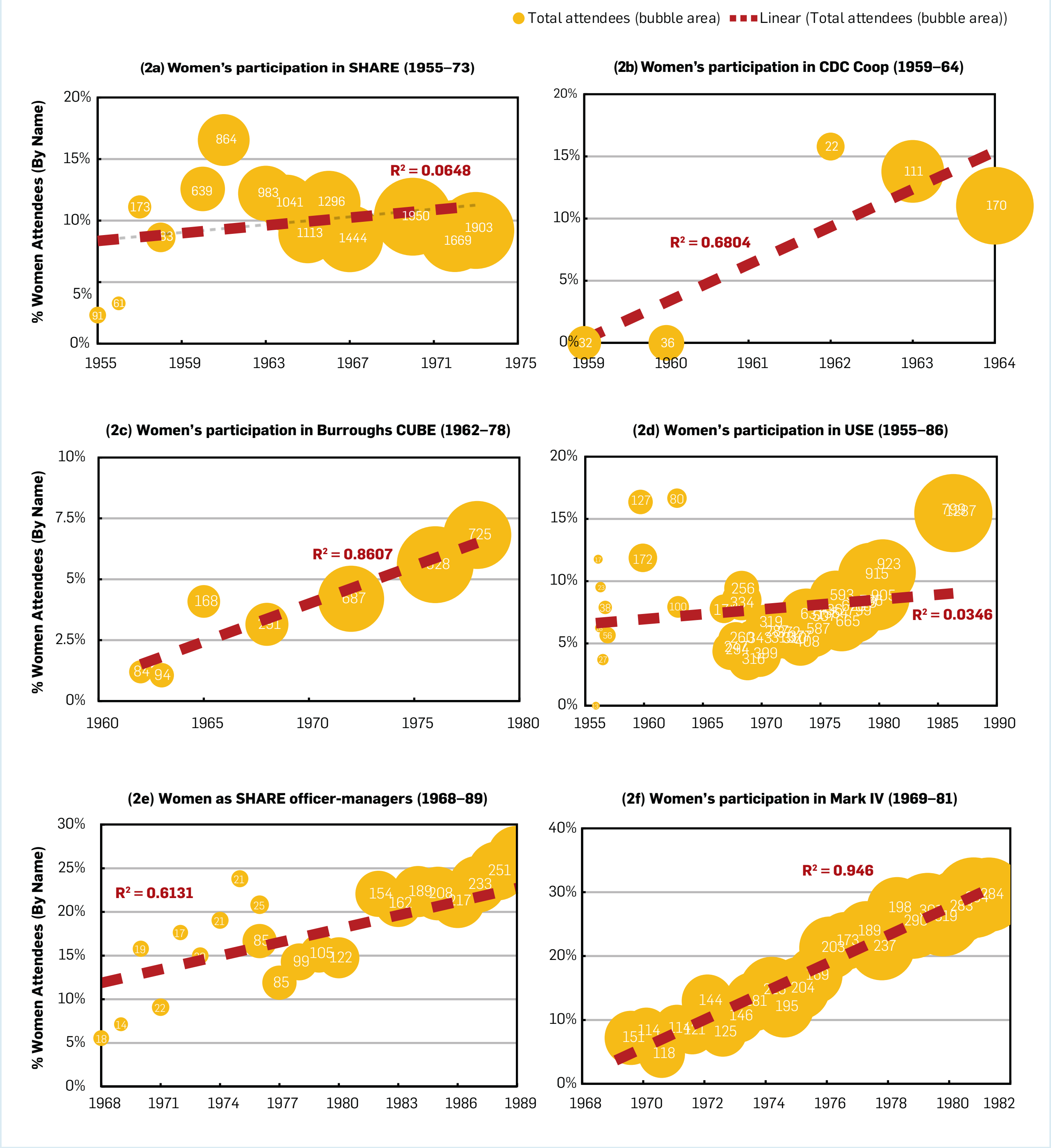
Figure 1. Women's participation in conferences/ACM members (1948–1953).



and professional leadership, even while nominally still within the pipeline; in their view the "pathways" model is a better guide to the "dynamic ... features and forces" of institutional settings, procedures, policies, and cultures in which women faculty members do not always experience orderly, expected, sequential or unidirectional progression through career ranks.[20] Clearly, much more needs to happen than merely "keeping women in the pipeline."[9,52]

To evaluate the 'making programming masculine' thesis and scrutinize the linear-pipeline view, the Charles Babbage Institute analyzed membership and attendee lists of six computer-user groups with available archival records.[41,53] Two of the largest user groups were formed in 1955. SHARE (for IBM computers) and USE (Sperry-Rand Univacs) provided a means for diverse companies, financial institutions, federal agencies and laboratories, and international entities to share algorithms and program

Figure 2. Women's participation in user groups (1955–1989).



code, to identify and address practical problems, to develop novel technical and organizational solutions—and, not least, to give sharp feedback to manufacturers. Both groups compiled attendee lists for their twice-yearly meetings, and many of these list *first names*.

First names, suitably analyzed and methodically tallied, indicate gender; in addition, committee reports identify hundreds of attendees as "Mr." or "Mrs." or "Miss"; oral histories identify others; and the Social Security Administration tabulates all given U.S. birth names by ascribed gender since 1880.[32] Between 80% and 100% of user-group attendees can be gender-identified.[41] Available records also give insight into Control Data's Coop, Burroughs' CUBE, Digital's DECUS, and the best-selling Mark IV software package for IBM computers. For each user-group, a time-series shows the participation of women in professional computing and indicates the rate of growth. The user-group attendees are taken to be samples of the computing workforce. No single user-group, with the possible exception of SHARE, is anything like a representative sample.

All such historical statistics, including government-compiled ones, are formed from sources of data that vary in uniformity (for example, direct personal surveys, company personnel reports, trade literature assessments, and industry or trade-group statistics); "uniform data" for historical statistics are always created by researchers, compilers, and analysts.[2,3] This present longitudinal dataset is the largest available for assessing changes in women's participation in the computing workforce (trade journals occasionally conducted one-time salary surveys[27])—until data from the U.S. Census and Bureau of Labor Statistics in the 1970s. The research method introduced here might be used to create longitudinal data, now lacking or fragmentary, on women in the STEM workforce. This systematic approach convincingly supplants earlier studies' reliance on fragmentary data or anecdotal evidence drawn from scattered or non-representative observations.

Figures 2a–f present new time-series data on women's participation in the U.S. computing workforce from 1955 to 1989. Each graph's *x*-axis gives the years from available archival records;[d] the *y*-axis, the percentage of women identified by first names; and the bubble area, the total analyzed population for each year. Individuals with gender-ambiguous or initials-only names are included in the bubble area (N) but are set aside for tabulation of women's participation. The data establishes varied growth across the 1960s and into the 1980s. Women's participation in SHARE grew slowly but steadily from 1955 to 1973, when, with thousands of attendees, it shifted to initials-only names. Women's participation as SHARE officer-managers similarly grew from 1968 to 1989, with a higher R^2 value supporting the upward linear-trend line. (R^2 is a standard linear regression measure of the 'goodness of fit' of computed trendlines with the underlying data: technically, R^2 is the percent of variation in dependent variable [%-women] that can be attributed to variation in independent variable [years]. All trendlines and statistics computed by Mac Numbers 4.3.1.) Notably, after women officer-managers reached 26.5% in 1989, a wider measure of women as SHARE meeting speakers was lower at 16.8% (N=491) and 19.4% (N=443) during 1991–1992. Women's participation grew steadily in USE during 1955–89, in CDC's Coop during 1959–1964, and in Burroughs' CUBE during 1962–1976, all with moderate R^2 values. Data from the Mark IV software user group 1969–1981, shows strong growth (R^2 =0.94) with women's participation reaching 30%.

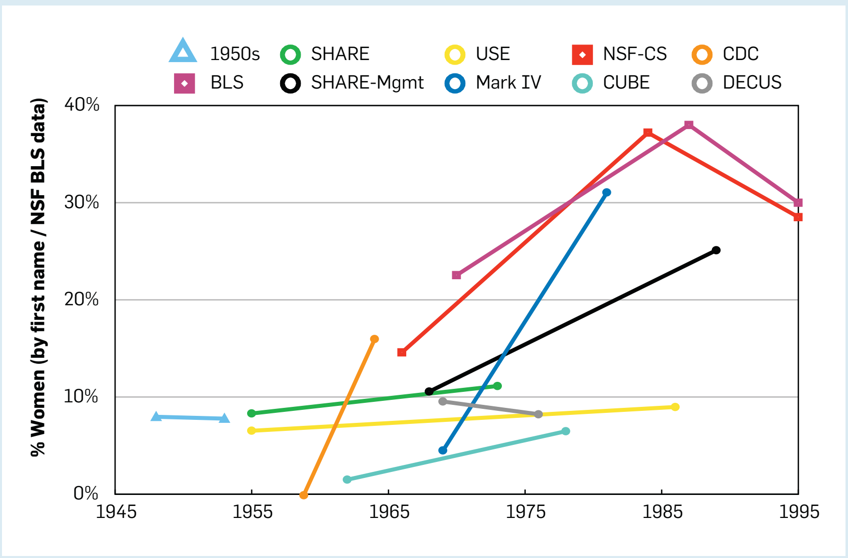
d Archival collections include the Hagley Museum and Library's USE/UNITE records Accession 1881 at findingaids.hagley.org/repositories/3/resources/915 as well as the Charles Babbage Institute's SHARE, USE, Control Data, Burroughs, DECUS, Margaret Fox, and Evan Linick (Mark IV) records at https://bit.ly/3nxsEHG

Gender Bias Is Non-Linear

Figure 3 combines the membership and user-group data across 1948–1995, adding the available federal workforce statistics and US computer-science bachelor degrees from the Bureau of Labor Statistics and NSF, respectively. For clarity, this graph simplifies the time-series data through plotting the underlying trend lines. Figure 3 shows decidedly non-linear dynamics, with varied growth rates and significant declines. The trendlines indicate unmistakable growth 1960s–1980s in women's participation in the computing workforce, refuting the commonly held "linear model" and any supposed masculine takeover. This user-group data tallies with salary surveys,[e] company-wide group

e *Business Automation* in 1960 found women were less than 15% of programmers; in its 1971 survey (N=600,000), women were "14% of systems analysts and 21% of computer programmers."[27]

Figure 3. Women in computing 1948–1995.



photographs,[f] and the NSF and BLS/US Census data.

At least three distinct periods may be discerned. *First:* From 1948 through around 1960, women were a numerically small proportion of the computing community (ranging from 0 to around 10%). There is no systematic data—here or elsewhere—that women were anything like 30% to 50% of the skilled white-collar computing workforce until the 1980s. Growth was modest (see 'USE' [slope = 0.0008]). The apparent sharp growth in CDC [N=371] reflects two years 1959–1960 with *zero* women; data for '1950s' is not a proper time-series. *Second:* From the 1960s through the 1970s, women in computer-user groups grew steadily if slowly to reach roughly 12% to 20% (see 'CUBE' and 'SHARE' [slopes = 0.0031, 0.0016]). Women were entering computing during these years—despite the linear model's speculation about them leaving. (The only time-series showing any downward drift is DECUS in 1968, 1972, 1976 [N=2,116] easing from 9.5% to 8.2% women.) Subsequent data through the mid-1980s suggest accelerating growth in women's participation in computing (see 'SHARE-Mgmt' and 'Mark IV' [slopes = 0.007 and 0.022]). Women attending USE grew to reach 15% in the mid-1980s; women officer-managers of SHARE grew to 26% in 1989; and women attending Mark IV conferences grew to 30% in 1980. These data are consistent with the U.S. Census reporting 22.5% women in the computing workforce (1970) and with the peak years for women's participation in the mid-1980s.[22] *Third:* women indeed left computing—after the peak in the mid-1980s—and this is what has persisted to the present. According to the CRA Taulbee survey, women's share of computer-science bachelor's degrees fell to 11.2% (2009). The U.S. Census American Community Survey reported women constituted 27% of the computing workforce (in 2011), a precipitous drop from the mid-1980s peak of 38%.[38]

f See photos of attendees at NMAA (1951), ACM (various), and *company-wide* photographs from Control Data (1962, 1966, 1982); https://bit.ly/3nzMlhR

There is no systematic data—here or elsewhere—that women were anything like 30%–50% of the skilled white-collar computing workforce until the 1980s.

These three periods demonstrate a non-linear dynamic for gender bias in computing. Instead of one question based on conjecture—"when" did women leave computing?—we now face distinct data-driven research questions. How did women establish a significant presence in the nascent high-skilled computer field in the 1950s? Men solidly dominated the fields that early computing drew on most heavily, such as engineering,[g] physics, and mathematics;[h] and yet computing women took up positions of responsibility and leadership such as Frances Holberton (née Snyder), Grace Hopper, Mina Rees, and many others. Why was women's growth in the computing workforce steady although slow through the mid-1960s? What attracted so many women into computing just as it professionalized during roughly 1965–1985? Computing among scientific and technical fields stood out for its expanding hospitality to women during these two decades, and we should be alert for useful lessons. And, finally, how to understand the exodus of women beginning in the late-1980s that afflicts computing through today?

Research for the Future

Further research is necessary to address these new questions, but it's clear the worrisome sea-change in computing during the late 1980s and 1990s accompanied dramatic cultural shifts. These include the rise of personal computing, gendered avatars in computer gaming, and the media's lionization of male "nerds." The nerd image, which had been previously ambiguous, flexible, and rhetorically situated distant from power, "gets rehabilitated and partially incorporated into hegemonic masculinity" beginning in the 1980s.[34] (Hegemonic masculinity

g Bix writes, "As late as the 1960s, women still made up less than 1 percent of students studying engineering in the United States."[7] Available data are thin or non-existent for women in specific engineering or science *workforces*; many studies make estimates from *educational* data.

h Mathematics prior to 1940 was distinctly open to women, who gained 14% of the field's Ph.D.'s.[23] But during 1945–1960 the number of men gaining math Ph.D.'s roughly tripled; while women experienced stasis in numbers and decline in participation (falling to 4.6–9.3% of total math Ph.D.'s).[30,44]

can be defined as the "configuration of gender practice [that] guarantees [or is taken to guarantee] the dominant position of men and the subordination of women."[17]) Popular media such as "Revenge of the Nerds" (1984) and "Triumph of the Nerds" (1996) sharpened the nerd image as a computing male. And nerds became allied with power. *Wired* magazine offered up Nicholas Negroponte, Stewart Brand, George Gilder, and John Perry Barlow in the 1990s. "Wired is about the most powerful people on the planet today—the Digital Generation," stated its cofounder. Bill Gates graced its cover five times in 15 years (and later gained a sixth with Mark Zuckerberg).[37,55,56] Today, many researchers target computing's gender-slanted culture, ingrained stereotypes, and associated public images as promising sites for positive intervention.[15,16,21,31,33]

The labor-intensive research method reported here might be automated by linking meeting and membership records with the SSA dataset.[32] As a pilot, I analyzed the 1949 ACM roster (N=435) in two ways. First, I did manual spreadsheet tallies of listed individuals as woman's, man's, initials-only, or gender-ambiguous name; as usual, I resolved gender-unclear names though contextual-archival linking or the SSA dataset. Second, I drew on the SSA dataset (year-of-birth = 1925) to directly compute the gender probabilities of each name. All but three "male" names (n=160) had 95% or greater probability of being male. Noel (91%), Francis (90%), and Jan (45%) were the exceptions; in this instance, it was Jan Rajchman, the noted RCA Laboratory engineer and IAS computer designer. Only one "female" name (n=27) had less than a 99% probability of being female. Jean is a woman's name in the U.S. (97.5%) but a Francophone male name; the 'Jeans' from, for example, Hydro-Québec attending these meetings indicate the need for contextual knowledge to correctly infer gender. In addition, 15 first names did not appear in the SSA dataset and were set aside. A weighed sum of the "male" and "female" name probabilities directly computed with one minor adjustment (resolving eight "initials-only" ACM members who were well-known men, namely JH Boekhoff, JG Brainerd, H Campaign, JJ Eachus, RW Hamming, CC Hurd, CC Gotlieb, and MV Wilkes) predicted that 8.52% of that year's ACM members were women, close to the manually tabulated 8.55% women. APIs exist for inferring gender from first names,[42,49] and some may deal with temporal changes in ascribed gender for such names as "Robin" or "Leslie" or even international names beyond the U.S.-based SSA dataset.[48]

Women's advances in the computing profession from the 1960s through the 1980s deserve special scrutiny today; in these years, computing was attractive to literally thousands of women programmers, systems analysts, database specialists, and middle managers.

Women's advances in the computing profession from the 1960s through the 1980s deserve special scrutiny today; in these years, computing was attractive to literally thousands of women programmers, systems analysts, database specialists, and middle managers. It is a mistaken notion that computing was somehow "made masculine" during these years when, in fact, women were flooding into the profession—attending professional meetings, participating in computer-user groups, and earning an increasing share of computer-science bachelor's degrees. The "making programming masculine" thesis has unwittingly obscured the very years when women found computing to be an exciting field where their technical talents could be actively exercised and professionally rewarded.[1,10,28,40,57] Recent retirements of top women executives at IBM, HP, and Xerox underscore the peak years of the 1980s when these women launched computing careers and when the field was nearly 40% women.[i]

More detailed gender-analysis of membership lists and conference attendees of ACM's numerous SIGs could shed light on which branches of computer science evinced greater or lesser openness to women's participation. Some branches of computer security had especially noteworthy women's leadership. For example, pioneering intrusion-detection research was led by Dorothy Denning, Teresa Lunt, Debra Anderson, Rebecca Bace, and others.[40,58] HCI has focused research attention on gender.[6,11,54] Recent findings suggest gender bias may be endemic in the *content* of machine learning, as expressed

i See geekfeminism.wikia.org/wiki/List_of_women_executives_at_tech_companies and www.fastcompany.com/1139328/women-tech-executives in the meme "Man is to Computer Programmer as Woman is to Homemaker."[4,8,35] Data beyond user groups is desirable. ACM members likely possess SIG records that could advance our understanding of the dynamics of gender bias in computing. ACM's History Committee recently launched a SIG-focused archiving initiative.[39] A large-scale data-gathering effort could empirically analyze what computing did right during the 1960s-1980s—focusing on specific SIGs and subfields—as well as what went wrong during the 1990s and beyond. If the preliminary research reported here is extended, perhaps the hard problem of gender bias in computing can be made tractable.

Acknowledgments. This research was supported by Alfred P. Sloan Foundation grant G-B2014-07.

Full data for all the figures in this article is available at https://tjmisa.com/papers/2021-06_CACM-data.zip.

References

1. Abbate, J. The pleasure paradox: Bridging the gap between popular images of computing and women's historical experiences. *Gender Codes.* T. Misa, (Ed), 211–227.
2. Anderson, M. The history of women and the history of statistics. *J. Women's History 4*, 1 (1992), 14–36; https://muse.jhu.edu/article/363006.
3. Anderson, M. The Census, audiences, and publics. *Social Science History 32*, 1 (2008), 1–18; https://muse.jhu.edu/article/232094.
4. Babaeianjelodar, M., Lorenz, S., Gordon, J., Matthews, J., and Freitag, E. Quantifying gender bias in different corpora. In *WWW '20: Companion Proceedings of the Web Conf.* (Apr. 2020), 752–759; https://dl.acm.org/doi/10.1145/3366424.3383559
5. Berryman, S.E. Who will do science? Minority and female attainment of science and mathematics degrees: Trends and causes. Rockefeller Foundation, New York, NY, 1983.
6. Beckwith, L., Burnett, M., Grigoreanu, V., Wiedenbeck, S. Gender HCI: What about the software? *Computer 39*, 11 (2006), 97–101; https://ieeexplore.ieee.org/document/4014778.
7. Bix, A.S. From 'engineeresses' to 'girl engineers' to 'good engineers:' A history of women's U.S. engineering education. *NWSA J. 16*, 1, (2004), 27–49; https://muse.jhu.edu/article/168384.
8. Bolukbasi, T., Chang, K. W., Zou, J., Saligrama, V., Kalai, A. Man is to computer programmer as woman is to homemaker? Debiasing Word embeddings. July 21, 2016; arXiv:1607.06520; https://arxiv.org/abs/1607.06520
9. Branch, E.H. *Pathways, Potholes, and the Persistence of Women in Science: Reconsidering the Pipeline.* Lexington Books, Lanham MD, 2016.
10. Buckholtz, E. Queens of code. *IEEE Annals of the History of Computing 42*, 2 (2020; 55–62.
11. Burnett, M., Peters, A., Hill, C., and Elarief, N. Finding gender-inclusiveness software issues with GenderMag: A field investigation. In *Proceedings of the 2016 CHI Conf. on Human Factors in Computing Systems.* ACM, New York, NY, 2586–2598
12. Camp, T. The incredible shrinking pipeline. *Commun. ACM 40*, 10 (Oct.1997), 103–110; https://dl.acm.org/doi/10.1145/262793.262813.
13. Canning, R. Issues in programming management. *EDP Analyzer 12*, 4 (1974), 1–14.
14. Cass, S. A Review of Code: Debugging the Gender Gap. *IEEE Spectrum* (June 19, 2015); http://bit.ly/3q6gUgI
15. Cheryan, S., Plaut, V.C., Handron, C., and Hudson, L. The stereotypical computer scientist: Gendered media representations as a barrier to inclusion for women. *Sex Roles: A Journal of Research 69*, 1–2 (2013), 58–71.
16. Clayton, K.L., Von Hellens, L.A. and Nielsen, S.H. Gender stereotypes prevail in ICT: A research review. In *Proceedings of the SIGMIS 47th Annual Conf. Computer Personnel Research.* ACM, New York, NY, 2009, 153–158; https://dl.acm.org/doi/10.1145/1542130.1542160.
17. Connell, R.W. *Masculinities.* University of California Press, Berkeley, CA, 1995, 77.
18. Ensmenger, N. Making programming masculine. *Gender Codes.* T. Misa, (Ed.), 115–141.
19. Ensmenger, N. 'Beards, sandals, and other signs of rugged individualism:' Masculine culture within the computing professions. *Osiris* (2015), 38–65; https://www.journals.uchicago.edu/doi/10.1086/682955.
20. Fox, M.F. and Kline, K. Women faculty in computing: A key case of women in science. *Pathways, Potholes, and the Persistence of Women in Science.* E.H. Branch, (ed), 41–55.
21. Frieze, C. Diversifying the images of computer science: Undergraduate women take on the challenge. *SIGCSE Bulletin 37*, 1 (Feb. 2005), 397–400; https://dl.acm.org/doi/10.1145/1047124.1047476.
22. Gilchrist, B. and Weber, R.E. Enumerating full-time programmers. *Commun. ACM 17*, 10 (Oct. 1974), 592–593; https://dl.acm.org/doi/10.1145/355620.361177.
23. Green, J. and LaDuke, J. *Pioneering Women in American Mathematics: The Pre-1940 Ph.D.s.* American Mathematical Society, Providence, R.I, 2008.
24. Grier, D.A. Gertrude Blanch of the Mathematical Tables Project. *IEEE Annals of the History of Computing 19*, 4 (1997), 18–27; https://ieeexplore.ieee.org/document/627896.
25. Grier, D.A. Ida Rhodes and the dreams of a human computer. *IEEE Annals of the History of Computing 22*, 1 (2000), 82–85; https://ieeexplore.ieee.org/document/815468
26. Gurer, D.W. Women's contributions to early computing at the National Bureau of Standards. *IEEE Annals of the History of Computing 18*, 3 (1996), 29–35; https://ieeexplore.ieee.org/document/511941.
27. Haigh, T. Masculinity and the machine man: Gender in the history of data processing. *Gender Codes.* T. Misa, (Ed). John Wiley, 2010, 51–71.
28. Halvorson, M.J. *Code Nation: Personal computing and the learn to program movement in America.* ACM, New York, NY, 2020; https://dl.acm.org/doi/book/10.1145/3368274.
29. Harvard University. *Second Symposium on Large-Scale Digital Calculating Machinery* (Sept. 11–16, 1949); https://bit.ly/2XAec6U
30. Herzig, A.H. Becoming mathematicians: Women and students of color choosing and leaving doctoral mathematics. *Rev. Educational Research 74*, 2 (2004), 171–214; https://www.jstor.org/stable/3516055.
31. Jia, S., Lansdall-Welfare, T., and Cristianini, N. Measuring gender bias in news images. In *Proceedings of the 24th Intern. Conf. World Wide Web.* ACM, New York, NY, 2015, 893–898; https://dl.acm.org/doi/10.1145/2740908.2742007
32. Karimi, F., Wagner, C., Lemmerich, F., Jadidi, M., and Strohmaier, M. Inferring gender from names on the Web: A comparative evaluation of gender detection methods. In *Proceedings of the 25th Intern. Conf. Companion on World Wide Web.* Intern. World Wide Web Conf. Steering Committee, Republic and Canton of Geneva, Switzerland, 2016, 53–54; https://dl.acm.org/doi/10.1145/2872518.2889385.
33. Kay, M., Matuszek, C., and Munson, S.A. Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conf. on Human Factors in Computing Systems.* ACM, New York, NY, 2015, 3819–3828; https://dl.acm.org/doi/10.1145/2702123.2702520.
34. Kendall, L. Nerd nation: Images of nerds in US popular culture. *Intern. J. Cultural Studies 2*, 2 (1999), 260–283; https://journals.sagepub.com/doi/10.1177/136787799900200206.
35. Leavy, S. Gender bias in artificial intelligence: The need for diversity and gender theory in machine learning. In *Proceedings of the 1st ACM/IEEE Intern. Workshop on Gender Equality in Software Engineering* (May 28, 2018), 14–16; https://dl.acm.org/doi/10.1145/3195570.3195580.
36. Longo, B. *Edmund Berkeley and the Social Responsibility of Computer Professionals.* ACM Morgan & Claypool, New York, NY, 2015; https://dl.acm.org/doi/book/10.1145/2787754.
37. Millar, M.S. *Cracking the Gender Code.* Second Story Press, Toronto, Canada, 1998, 96–107.
38. Misa, T.J. *Gender Codes: Why Women Are Leaving Computing.* John Wiley, Hoboken, NJ, 2010.
39. Misa, T.J. Computing is history. *Commun. ACM 58*, 10 (Oct. 2015), 35–37; https://dl.acm.org/doi/10.1145/2814845.
40. Misa, T.J. 2016. *Communities of Computing: Computer Science and Society in the ACM.* ACM Morgan & Claypool, 2016; https://dl.acm.org/doi/book/10.1145/2973856.
41. Misa, T.J. Gender bias in computing. *Historical Studies in Computing, Information, and Society.* W. Aspray, (ed). Springer Nature, Switzerland, 2019, 113–133; https://link.springer.com/chapter/10.1007%2F978-3-030-18955-6_6.
42. Mueller, J. and Stumme, G. Gender Inference using Statistical Name Characteristics in Twitter. In *Proceedings of the 3rd Multidisciplinary Intern. Social Networks Conference on Social Informatics.* Data Science 2016. ACM, New York, NY, https://dl.acm.org/doi/10.1145/2955129.2955182.
43. Mundy, L. Why is Silicon Valley so awful to women? *The Atlantic* (Apr. 2017). 60-73; http://bit.ly/3sfs6cU.
44. Murray, M.A.M. *Women Becoming Mathematicians: Creating a Professional Identity in Post-World War II America.* MIT Press, Cambridge, MA, 2000.
45. Payton, F.C. and Berki, E. Countering the negative image of women in computing. *Commun. ACM 2*, 5 (May 2019), 56–63.
46. Porter, J. The fascinating evolution of brogramming and the fight to get women back. *Fast Company* (Oct. 20, 2014); http://bit.ly/3qcyuQl.
47. Shetterly, M.L. *Hidden Figures: The American dream and the untold story of the Black women mathematicians who helped win the space race.* William Morrow, New York, NY, 2016.
48. Smith, B.N., Singh, M., and Torvik, V.I. A search engine approach to estimating temporal changes in gender orientation of first names. In *Proceedings of the 13th ACM/IEEE-CS Joint Conf. Digital Libraries.* ACM, New York, NY, 2013, 199–208; https://dl.acm.org/doi/10.1145/2467696.2467720.
49. Tran, A. Inferring gender from column of first names in R. Rev. Aug. 21, 2015; gist.github.com/andrewbtran/d3d8e04f5c86dcfa2bb0
50. Univac Pioneers Day. Pioneer Day 1981: UNIVAC I. Annals of the History of Computing 3, 4 (1981), 400–407.
51. Vardi, M.Y. How We Lost the Women in Computing. *Commun. ACM 61*, 5 (May 2018), 9; https://dl.acm.org/doi/10.1145/3201113
52. Vitores, A. and Gil-Juárez, A. The trouble with 'women in computing': A critical examination of the deployment of research on the gender gap in computer science," *J. Gender Studies 25*, 6 (2016), 666–680; https://www.tandfonline.com/doi/full/10.1080/09589236.2015.1087309
53. Vogel, W.F. 'The spitting image of a woman programmer': Changing portrayals of women in the American computing industry, 1958–1985. *IEEE Annals of the History of Computing 39*, 2 (2017), 49–64.
54. Vorvoreanu, M., Zhang, L., Huang, Y. H., Hilderbrand, C., Steine-Hanson, Z., and Burnett, M. From gender biases to gender-inclusive design: An empirical investigation. In *Proceedings of the 2019 Conf. Human Factors in Computing Systems Paper 53* (May 2019), 1–14; https://dl.acm.org/doi/10.1145/3290605.3300283
55. Waern, A., Larsson, A., and Nerén, C. Hypersexual avatars: Who wants them? In *Proceedings of the 2005 ACM SIGCHI Intern. Conf. Advances in Computer Entertainment Technology.* ACM, New York, NY, 2005, 238–241; DOI.
56. *Wired.* Gates, Zuckerberg meet for Wired cover shoot. (Apr. 19, 2010); http://bit.ly/3oAjuLS /
57. Yost, J. Programming enterprise: Women entrepreneurs in software and computer services. *Gender Codes.* T. Misa, (ed), 229–250.
58. Yost, J.R. The march of IDES: Early history of intrusion-detection expert systems. *IEEE Annals of the History of Computing 38*, 4 (2016), 42–54; https://ieeexplore.ieee.org/document/7155454.

Thomas J. Misa is Past President of the Society for the History of Technology (2021–2022) and editorial board member for ACM Books (2013–present). He directed the Charles Babbage Institute (2006–2017) at the University of Minnesota and chaired the ACM History Committee (2014–2016).

Attention: Undergraduate *and* Graduate Computing Students

STUDENT RESEARCH COMPETITION

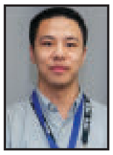
Association for Computing Machinery
Advancing Computing as a Science & Profession

SPONSORED BY Microsoft

The ACM Student Research Competition (SRC), sponsored by Microsoft, offers a unique forum for undergraduate and graduate students to present their original research before a panel of judges and attendees at well-known ACM-sponsored and co-sponsored conferences. The SRC is an internationally recognized venue enabling undergraduate and graduate students to earn many tangible and intangible rewards from participating:

- **Awards:** cash prizes, medals, and ACM student memberships
- **Prestige:** Grand Finalists and their advisors are invited to the Annual ACM Awards Banquet, where they are recognized for their accomplishments
- **Visibility:** opportunities to meet with researchers in their field of interest and make important connections
- **Experience:** opportunities to sharpen communication, visual, organizational, and presentation skills in preparation for the SRC experience

It's hard to put the ACM Student Research Competition experience into words, but we'll try…



"It is an excellent chance for me to participate in the ACM SRC to present my work, and I got much valuable feedback from experts and peers. Moreover, I practiced my communication and presenting skills, laying a solid foundation for my future research life. I also made friends with outstanding students from all over the world who are interested in scientific research. I strongly recommend ACM SRC to all undergraduate and graduate students because it is a perfect platform for all of us to exchange ideas and make improvements."

Shengcheng Yu
Nanjing University | ASE 2019

"ACM SRC provided an excellent opportunity to showcase my research work and solicit feedback from experts in my field. Winning this competition gave a boost to my confidence and motivated me to make a greater impact to my field. In addition, I enjoyed learning about other research areas from fellow participants and made several connections. Thus, I am extremely grateful for the organizers of this competition who provided me an opportunity to present my work. I would strongly encourage undergraduate and graduate students to participate!"

Peter Zhi Xuan Li
Massachusetts Institute of Technology | SIGMICRO 2019

"The SRC was a great opportunity to present our work-in-progress research results to leading experts in the community. I appreciated the different kinds of presentations – extended abstract, poster, and talk – which gave me the opportunity of getting different kinds of feedback. All in all, the SRC was a great experience to both make new connections and discuss early-stage results with expert researchers."

Ari Rasch
University of Muenster | CGO 2020

ACM Student Research Competition

STUDENT RESEARCH COMPETITION
Association for Computing Machinery
Advancing Computing as a Science & Profession
SPONSORED BY Microsoft

"Participating in the ACM SRC was a phenomenal experience for me. As an undergraduate student, it gave me the opportunity to present my work to experts in the field and seek valuable feedback. Interacting with researchers and fellow participants at the conference was a great learning experience and allowed me to network as well. The SRC is a great way to broaden one's horizons and I would strongly encourage student researchers to participate in it."

Milind Srivastava
Indian Institute of Technology Madras | ICCAD 2019

"Participating in the ACM SRC gave me confidence in my research and technical experiences. Receiving feedback during my presentation helped me improve my work and develop my communication skills. It was a fantastic opportunity to present in English at an international conference for the first time and to connect with other researchers during the competition. I really admire the program because it gives opportunity and support so that anyone worldwide can apply and attend the conference. I strongly recommend other undergraduate students submit their research to the competition."

Ana Solórzano
Universidade Federal de Santa Maria, Brazil | GHC 2019

"ACM SRC was a high-visibility platform for sharing research ideas. It was exciting to meet people who share similar interests and get their perspective and thoughts on ideas we presented. We got exposed to the best collection of research work, arts, and emerging technologies. It would be really difficult for our team to recall an instance where we got bored. Rather, the challenge was to choose between multiple, equally enticing events. My heartfelt thanks to my team and ACM for organizing this platform!"

Sai Ganesh
Texas A&M University, College Station | SIGGRAPH 2019

"The ACM SRC made it possible for me to experience the world of research in my field of interest, far more broadly than I had previously been able to from my home institution. At the conference, I was able not only to hear from and talk to prominent researchers and fellow students, but also to better understand my possible future career paths as I prepared to apply to graduate schools. I am incredibly grateful to my mentors for guiding me to participate in this opportunity. If you're a student interested in research, I highly recommend you apply!"

Samuel Estep
Liberty University | SPLASH 2019

Check the SRC Submission Dates: ***https://src.acm.org/submissions***

Questions? Contact Nanette Hernandez, ACM's SRC Coordinator: ***hernandez@hq.acm.org***

DOI:10.1145/3429775

Application-layer and network-layer defenses are critical for fortifying routing attacks.

BY YIXIN SUN, MARIA APOSTOLAKI, HENRY BIRGE-LEE, LAURENT VANBEVER, JENNIFER REXFORD, MUNG CHIANG, AND PRATEEK MITTAL

Securing Internet Applications from Routing Attacks

THE INTERNET IS a "network of networks" that interconnects tens of thousands of separately administered networks. The Border Gateway Protocol (BGP) is the glue that holds the Internet together by propagating information about how to reach destinations in remote networks. However, BGP is notoriously vulnerable to misconfiguration and attack. The consequences range from making destinations unreachable (for example, Google's routing incident caused widespread Internet outage in Japan[a]), to misdirecting traffic through unexpected intermediaries (for example, European mobile traffic routed through China Telecom due to improper routing announcements from a Swiss datacenter[b]), to impersonating legitimate services (for example, traffic to an Amazon DNS server rerouted to attackers who answered DNS queries with fraudulent IP addresses[c]). Efforts to secure the Internet routing system have been underway for many years,[6–8,11,13,14] but the pace of progress is slow since many parties must agree on solutions and cooperate in their deployment.

In the meantime, more users rely on the Internet to access a wide range of services, including applications with security and privacy concerns of their own. Applications such as Tor (The Onion Routing) allow users to browse anonymously, certificate authorities provide certificates for secure access to Web services, and blockchain supports secure cryptocurrencies. However, the privacy and security properties of these applications depend on the network to deliver traffic; Figure 1 illustrates the *cross-layer* interaction between Tor and the underlying network. Application developers abstract away the details of Internet routing, but BGP does not provide a sufficiently secure scaffolding for these applications. This gap leaves the vulnerabilities due to rout-

a Google leaked prefixes—and knocked Japan off the Internet, 2017; http://bit.ly/3sPjWII

b BGP event sends European mobile traffic through China Telecom for 2 hours, 2019; http://bit.ly/3qJrefc

c AWS DNS network hijack turns MyEtherWallet into ThievesEtherWallet, 2018; https://www.theregister.co.uk/ 2018/04/24/myetherwallet dns hijack

» key insights

- **The risks of routing insecurity have been significantly underestimated. Routing attacks can compromise critical Internet applications and have devastating consequences for users.**
- **Application-specific defenses against Internet routing attacks offer immediate protection to users.**
- **Given the serious risks of strategic routing attacks, the community should redouble its efforts to secure the global routing system**

IMAGE BY ALEXEY EROFEJCHEV

ing insecurity significantly underestimated. While routing attacks are well known, they have been viewed primarily as affecting availability (when misdirected traffic is dropped) and confidentiality (when data is not encrypted). This article provides a new perspective by showing that routing attacks on Internet applications can have even more devastating consequences for users—including uncovering users (such as political dissidents) trying to communicate anonymously, impersonating websites even if the traffic uses HTTPS, and stealing cryptocurrency.

This article argues that the security of Internet applications and the network infrastructure should be considered together, as vulnerabilities in one layer led to broken assumptions (and new vectors for attacks) in the other. We first give an overview of routing security. Then, we discuss how cross-layer interactions enable routing attacks to compromise popular applications like Tor, certificate authorities, and the bitcoin network. Given the slow adoption of secure routing solutions, we discuss how applications can take into account the underlying routing properties and employ application-layer defenses to mitigate routing attacks. We believe that application-layer and network-layer solutions are interconnected, and both are essential to secure Internet applications. While application-layer defenses are more easily deployable, we hope to motivate the community to redouble efforts on secure routing solutions and tackle BGP's many security problems once and for all.

Routing Attacks

Routing attacks occur in the wild and are getting increasingly prevalent and more sophisticated. We dissect routing attacks from the perspective of an attacker and review existing defenses. In particular, the ability to divert targeted traffic via routing attacks is an emerging threat to Internet applications. We further demonstrate how routing attacks compromise three applications.

How BGP works. The Internet consists of around 67,000 Autonomous Systems (ASes),[d] each with an AS number (ASN) and a set of IP prefixes. Neighboring ASes exchange traffic in a variety of bilateral relationships that specify which traffic should be sent and how it is paid for. Such agreements can generally be classified into two types: a *customer-provider* relationship, where the customer pays the provider to send and receive traffic to and from the rest of the Internet, and a *peer-to-peer* relationship, where no money is exchanged but traffic must be destined for the peer or its customers.

Routing among the ASes is governed by the Border Gateway Protocol (BGP), which computes paths to destination prefixes. ASes choose one "best" route to a prefix based on a list of factors, with the top two generally being: Local Preference: a path via a customer is preferred over path via a peer, which is preferred over a provider; Shortest Path: a path with the fewest AS hops is preferred. The AS will then add the route into its local *Routing Information Base*, and further propagate the route to its neighbors based on routing policies after prepending itself in the path.

d CIDR Report, 2020; http://www.cidr-report.org/as2.0

ASes forward packets using the path to the longest matching prefix of the destination IP. In Figure 2, AS1 announces 140.180.0.0/22 via neighbor AS2, and 140.180.0.0/24 via neighbor AS3. AS4 forwards packets to 140.180.0.0/24 via AS3 based on the longest prefix match. Note that, in general, the longest prefix that can be successfully propagated is /24; many ASes filter prefixes that are longer than /24 by default.

Goals of routing attacks. By default, ASes trust routing announcements from other ASes. Routing attacks happen when an AS announces an incorrect path to a prefix, causing packets to traverse through and/or arrive at the attacker AS. We discuss the goals of the attacker from two perspectives: *whom to affect* and *what to achieve*.

Whom to affect. Routing attacks affect two groups of victims: destinations, whose prefixes are announced by the attacker, and senders, who send packets to the attacked prefixes.

Destinations. YouTube was the targeted destination of a hijacking incident in 2008, where Pakistan authorities tried to block access to YouTube.[e] Pakistan Telecom (AS17557) announced the prefix 208.65.153.0/24, which was a subnet of 208.65.152.0/22 announced by YouTube (AS36561).

Senders. The attacker can divert global traffic from all senders on the Internet, or selectively target only traffic from certain senders. In the YouTube incident, the goal was to target only senders within Pakistan; however, the attack unintentionally affected all senders around the globe.

What to Achieve. Historically, the most visible effect of routing attacks is *outage*, where attackers drop packets and make the destinations unreachable. This type of attack that "black-

e Pakistan hijacks YouTube, 2008; https://dyn.com/blog/pakistan-hijacks-youtube-1

Figure 1. BGP routing affects who can observe Tor traffic.

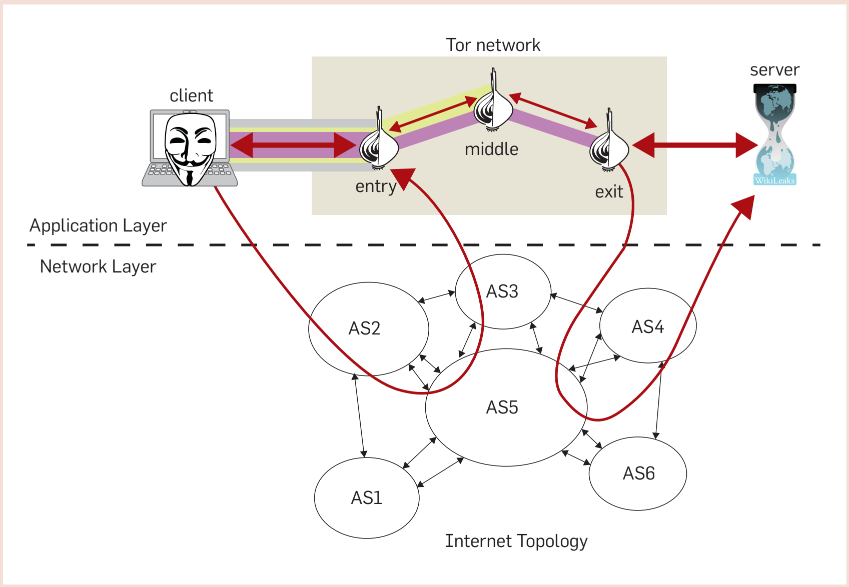
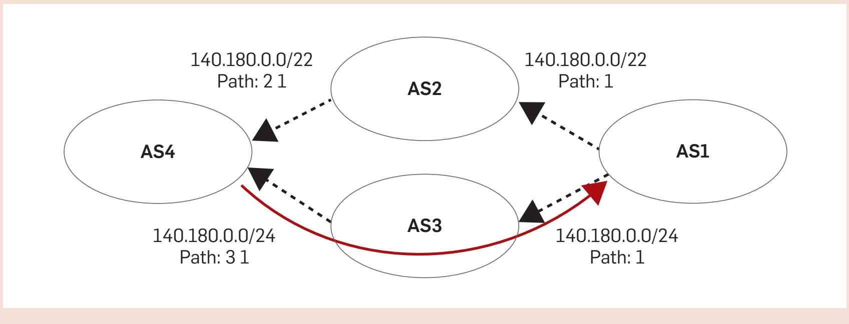


Figure 2. AS4 routes traffic to AS1 via AS3 for destination IPs within 140.180.0.0/24 based on longest matching prefix.



holes" the traffic is also characterized as a *hijack attack*. However, the attacker's goals can be more sophisticated.

Surveillance. Authorities may use routing attacks to perform surveillance and target traffic from senders in certain regions. Intelligence agencies such as NSA could launch routing attacks to make certain traffic easier to intercept for surveillance.[f] Traffic from the targeted region would be rerouted to the authorities, who forward the traffic to the destinations while monitoring the activities. This type of attack is usually characterized as an *interception attack*, where the legitimate destinations still receive the traffic. Interception attacks are much harder to notice than hijack attacks since they do not interrupt the communication, though performance may degrade due to more circuitous paths. Furthermore, authorities could exploit routing attacks to surpass legal restrictions by diverting domestic traffic (for example, emails between Americans) to foreign jurisdictions to conduct surveillance.[9]

Impersonation. Attackers can impersonate destinations to deceive the senders by intercepting packets via either hijack or interception attacks and replying with forged responses. These attacks can have damaging consequences. In 2018, attackers used routing attacks to impersonate Amazon's authoritative DNS service and answered DNS queries for a cryptocurrency website with Russian IP addresses. The users were then directed to a fraudulent site which they believed was their real cryptocurrency service. Consequently, cryptocurrency was stolen. Attackers may also impersonate large number of IP addresses to originate spam or other malicious traffic.[g]

Cross-layer attacks on applications. Attackers may further exploit the diverted traffic to perform more sophisticated attacks on networked systems and applications. The specific goals vary depending on the functionalities of the applications. In this article, we demonstrate routing attacks on three applications: deanonymizing Tor users via traffic analysis on the Tor network, obtaining bogus digital certificates for websites from certificate authorities, and preventing blockchain systems from reaching consensus.

The ability to divert targeted traffic via routing attacks is an emerging threat to Internet applications.

Attack methodology. Attackers must decide which prefix to announce, which path to announce, and which ASes should receive the announcement.

Which prefix to announce. Attackers can announce either a sub-prefix (that is, more-specific prefix) of the target prefix, or an equally specific prefix same as the target prefix. Note that a less-specific prefix would not be used in packet forwarding and hence would not constitute a successful attack.

Affecting global traffic by announcing sub-prefixes. Since forwarding is based on longest prefix match, sub-prefix attacks are highly effective at hijacking traffic from all senders. However, since most ASes filter announcements for prefixes longer than /24, sub-prefix attacks on /24 prefixes would not be effective.

Targeting selective traffic by announcing equally specific prefixes. An AS that receives both the legitimate announcement and the attacker's announcement would pick one based on routing preferences. Note that some ASes may only receive one announcement. In Figure 3, AS2 (attacker) announces the same /24 prefix as the destination AS1, and AS4 prefers the path to AS2 while AS3 still prefers the path to AS1. This attack generally affects only parts of the Internet and does not have global impact. However, it is stealthier due to its local impact and enables targeted attacks on certain senders.

Which path to announce. The attacker may put itself as the origin of the prefix, which naturally constitutes a hijack attack. Yet, a more sophisticated attacker has a range of other options.

Evading detection by forging the victim AS. The attacker can add the legitimate destination AS to the end of its path, so the announcement has the same "last hop" (that is, "origin") AS as a legitimate announcement. This makes the attack stealthier since some defenses (for example, monitoring systems and origin validation) only check the origin AS of the announcement instead of the full path. Note that the path now appears one hop longer, which may reduce the number of ASes

f Network Shaping 101; https://www.documentcloud.org/documents/2919677-Network-Shaping-101.html.

g Shutting Down the BGP Hijack Factory, 2018; https://blogs.oracle.com/internetintelligence/shuttingdown-the-bgp-hijack-factory

Figure 3. AS2 (attacker) announces an equally specific prefix as AS1 (legitimate destination). AS4 prefers the path to AS2, while AS3 prefers the path to AS1. Only AS4 is affected by the attack.

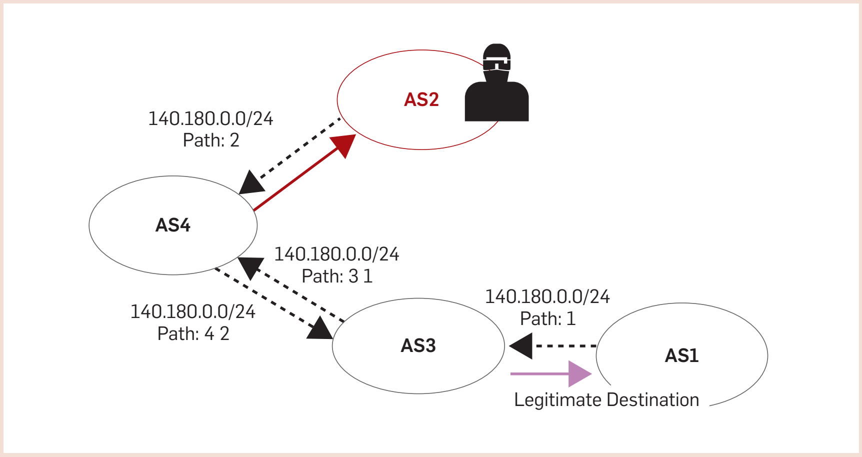
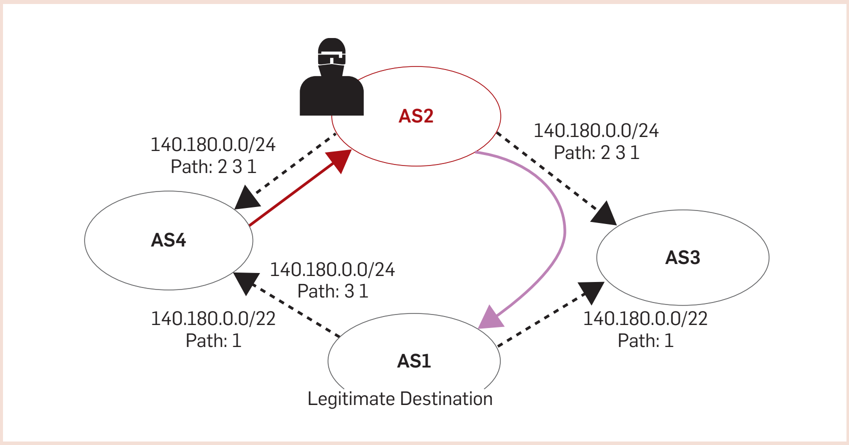


Figure 4. AS2 (attacker) "poisons" the path by appending AS3 and AS1 (legitimate destination) in the path, which preserves a legitimate route from AS2 to AS1.



that pick the attacker's route over the legitimate route.

Interception attack via AS path poisoning. A sophisticated attacker can append a set of carefully selected ASes at the end of the path. These ASes should constitute a legitimate path from the attacker to the destination AS. The appended ASes will ignore the attacker's announcement because of BGP loop prevention, which consequently helps preserve legitimate routes from the attacker AS to the destination AS. This attack is known as the "AS path poisoning attack" (see Figure 4). This attack is very stealthy and effective at performing interception attack while announcing a sub-prefix.

Which ASes should receive the announcement. Instead of sending the announcement to all neighbors, a strategic attacker may attempt to control who can receive the announcement to increase attack stealthiness, perform an interception attack, or target certain senders. We discuss two techniques to limit announcement propagation.

Announcing to certain neighbors. Attackers may exploit routing policies to control attack propagation by only announcing to certain peers and customers. These announcements will only be propagated "down" to the peer's customers, but not to its providers. Consequently, only selected ASes will hear the announcements.

BGP communities. BGP communities are optional attributes that can be added to an announcement to control routing policies in upstream ASes, for purposes such as traffic engineering. Attackers may exploit BGP communities to strategically control attack propagation such that selected ASes will never hear or will not prefer the bogus announcements, and thus increase the effectiveness and viability of interception attacks.[4]

Routing defenses. Defending against routing attacks is challenging due to the lack of "ground truth" to inform whether a path is "correct." Seemingly suspicious announcements could be legitimate paths used by ASes to optimize network performance. Many solutions have been proposed that rely on different sources of information as "ground truth."

Anomaly detection via BGP monitoring. BGP monitoring systems detect anomalous routing announcements by using historical routing data to infer the "expected" origin ASes or paths for prefixes.[10,12,15,19,24] They typically do not require changes to the routing protocol and hence are highly deployable. However, many early efforts on monitoring systems focused on catching "easy" attacks (for example, mismatched origin ASes), but failed to detect more sophisticated attacks such as interception attacks. Furthermore, relying on historical data to infer ground truth is prone to false positives (flagging legitimate routes) and false negatives (missing real attacks).

Defensive filtering via preset knowledge. ASes often perform prefix filtering on announcements received from direct customers. It is effective against attacks launched by customer ASes, but does not prevent ASes from attacking their direct or indirect customers. A more advanced filtering technique is AS path filtering, which uses a whitelist of paths for announcements received from peering ASes based on prior information exchange.[20] It extends the knowledge base further from the sole knowledge of an individual provider on its customers (as in prefix filtering), to a collective knowledge base exchanged and built among a network of trusted peers. The MANRS project[h] has outlined best practices for using filtering techniques to protect the routing infrastructure.

Origin validation. The Resource Public Key Infrastructure (RPKI) is a public key infrastructure that stores crypto-

h MANRS Project, 2020; https://www.manrs.org/

graphic attestations, known as Route Origin Authorizations (ROAs), indicating which ASes are authorized to originate which prefixes.[6] Upon receiving an announcement, ASes perform Route Origin Validations (ROV) to filter routes originated from invalid ASes. RPKI utilizes cryptographic primitives to make the knowledge base available to *all ASes* as opposed to only direct neighbors in defensive filtering. Even though ROV only validates the origin AS instead of the full path, it can already be effective at preventing many attacks. However, currently less than 20% of the prefixes have valid ROAs[i] and even fewer ASes are correctly performing ROV.[16]

Path validation. BGPsec uses cryptographic primitives to validate the *whole AS path*.[13] It is an online protocol, as opposed to a separate offline lookup (like ROV). Each AS in the path generates a cryptographic signature which is added to the path as the announcement propagates through the network. While BGPsec provides validation of the full path, it places a heavy burden on BGP routers. It also requires all ASes along a path to participate, making incremental deployment challenging. We have yet to see real-world deployment of BGPsec.

In this article, we provide a new angle into building defenses—in addition to network-layer defenses, applications can build their own application-layer defenses by taking into account the underlying routing properties. We also highlight the importance of deploying defenses against sophisticated attacks, which are stealthier and effective at compromising Internet applications.

i RPKI Deployment Monitor; https://rpki-monitor.antd.nist.gov/.

The Tor Network

Tor is the most widely used anonymity system.[7] It carries terabytes of traffic every day and serves millions of users.[j] However, network-level adversaries can deanonymize Tor users by launching routing attacks to observe user traffic and subsequently performing correlation analysis. Furthermore, the attacks have broad applicability to low-latency anonymous communication systems beyond Tor (for example, I2P anonymous network or even VPNs).

How Tor works. To prevent an adversary from associating a client with a destination server, Tor encrypts the network traffic and sends it through a sequence of *relays* (proxies) before going to the destination. The client selects three relays (entry, middle, exit), and constructs a *circuit* through them with *layered encryption* by repeatedly encrypting the next hop with the keys of the current hops (see Figure 5). Each relay only learns the previous and next hops, and no relay or local network observer can identify both the source and destination.

However, Tor is known to be vulnerable to network-level adversaries who can observe traffic at both ends of the communication, that is, between client and entry, and between exit and server. By default, Tor does not obfuscate packet timings, so the traffic entering and leaving Tor are highly correlated. An adversary on the path at both ends can then perform traffic correlation analysis on the packet traces to deanonymize the clients.

j Tor metrics; https://metrics.torproject.org/.

Figure 5. The Tor network.

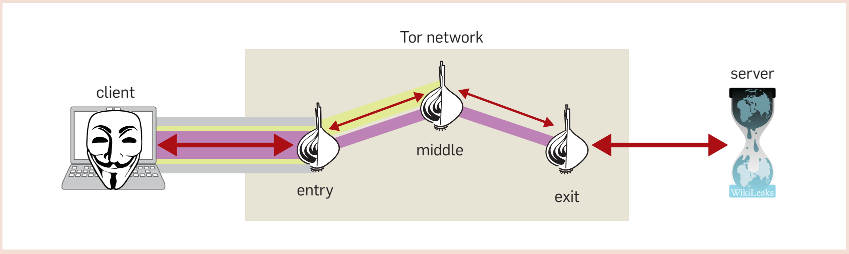
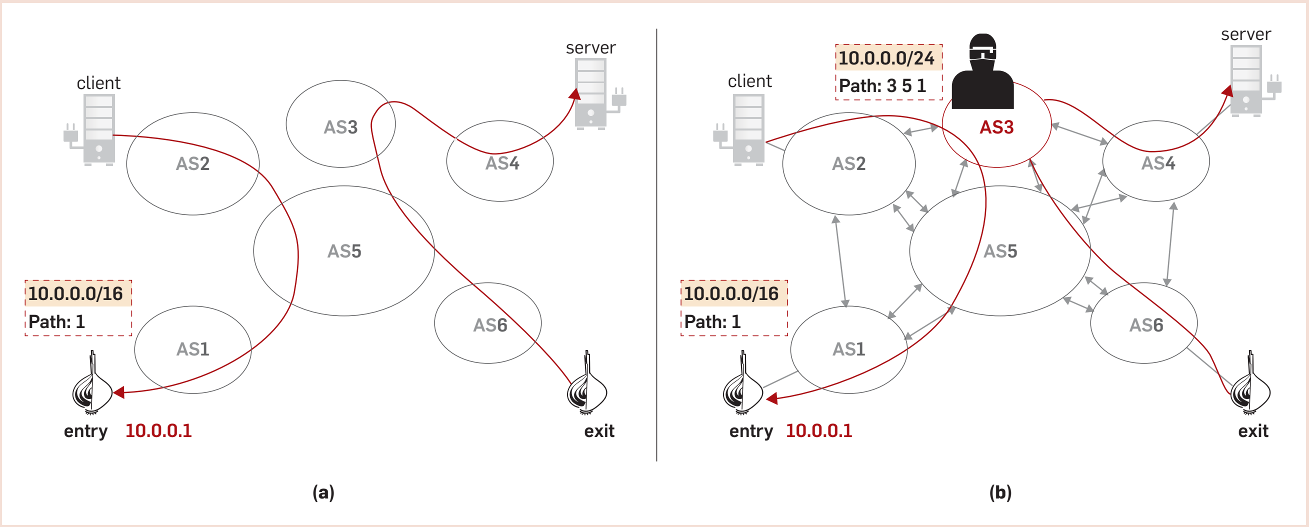


Figure 6. An adversary (AS3) launches an interception attack on the entry relay in AS1 and consequently observes the client-entry traffic in addition to exit-server traffic.



Routing attacks on anonymity systems. Traditional attacks from network-level adversaries focus on passive adversaries who are already on the paths to observe Tor traffic. However, adversaries can exploit active routing attacks to strategically intercept Tor traffic, enabling on-demand and targeted attacks.[22]

Figure 6 illustrates the attack. AS3 (adversary) only sees traffic between the exit and the Web server and needs to intercept the traffic between the client and the entry relay. It also needs to keep the connection alive in order to capture sufficient traffic for the correlation analysis, that is, perform an *interception attack*. AS3 announces an equally specific prefix of the target prefix which covers the entry relay, while maintaining a valid path (via AS5) to the victim AS1. Consequently, traffic from the client gets routed to the adversary AS3, which forwards the traffic to AS1 to keep the connection alive. Similar attacks can be performed to intercept the exit-server connection as well, if the adversary is not already on the path.

The attacks become more threatening given that seeing *either direction of the traffic* is sufficient, which opens the door to more adversaries. Figure 7 illustrates the scenario where the user downloads a f ile from the Web server. The adversary performs an interception attack on the entry relay and only sees one direction of the traffic (client to entry relay), which are mostly TCP ACK packets. The adversaries then use the sequence and acknowledgment numbers from the TCP header (unencrypted) to determine the sizes of the data packets traveling in the other direction.

The attack was successfully demonstrated on the live Tor network (ethically), by having 50 Tor clients download files from 50 Web servers via an entry relay under a prefix controlled by the researchers.[22] Routing announcements were propagated through the PEERING testbed,[18] and an interception attack was launched on the prefix covering the entry relay. No real user was affected during the attack. The attack deanonymized 90% of the clients in less than five minutes.

Defenses to protect anonymity. Many existing defenses cannot sufficiently detect or prevent such interception attacks. Recent works have proposed application-layer defenses for Tor.[21,23]

Proactive defense via relay selection. Sun et al.[21] proposed a new relay selection algorithm to protect the connection between a Tor client and the entry relay. This algorithm defends against equally specific prefix attacks on entry relays, where the effect is localized and only clients in certain locations will get affected. The localized effect opens up the possibility for clients to stay unaffected by choosing the relay wisely and proactively before any attack happens. The algorithm maximizes the probability of clients being unaffected by attacks based on the topological locations of the clients and the relays. It successfully improves the probability by 36% on average (up to 166% for certain Tor client locations).

Reactive defense via monitoring. To complement the proactive defense, Sun et al. proposed a monitoring system on routing activities for Tor relays. The system uses new detection techniques such as time-based and frequency-based heuristics, specifically tuned for Tor. The authors showed that most BGP updates involving a Tor relay are only announced by a single AS (across all updates), effectively differentiating the announcements made by adversary ASes who never announced the prefix in the past. Tan et al.[23] also proposed a data-plane detection approach that periodically runs traceroute to detect longest-prefix attacks and update Tor relay descriptors upon anomaly detection, so that Tor clients can pick entry relays correspondingly.

Certificate Authorities

The Public Key Infrastructure is the foundation for securing online communications. Digital certificates are issued by trusted certificate authorities (CAs) to domain owners, verifying the ownership of a domain. Internet users trust a domain with encrypted commu-

Figure 7. The adversary may only see one direction of the traffic but can still perform asymmetric traffic analysis to deanonymize users.

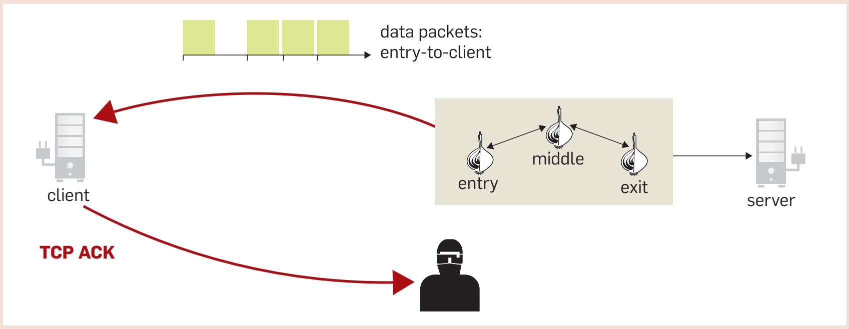
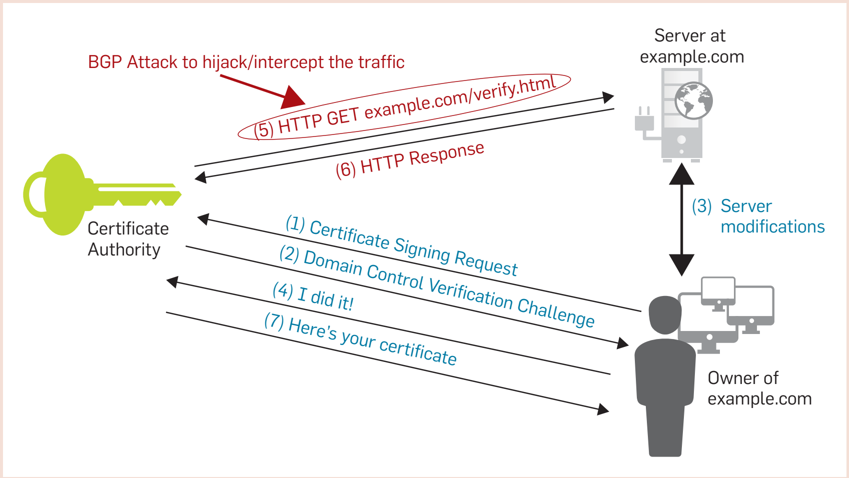


Figure 8. BGP attack on domain control verification.


BGP Attack to hijack/intercept the traffic
Server at example.com
(5) HTTP GET example.com/verify.html
(6) HTTP Response
Certificate Authority
(3) Server modifications
(1) Certificate Signing Request
(2) Domain Control Verification Challenge
(4) I did it!
(7) Here's your certificate
Owner of example.com

nications, such as bank websites, only if a valid certificate signed by a CA is presented. This mechanism effectively prevents Man-In-The-Middle (MITM) attacks that can have disastrous consequences, such as stealing users' financial information.

However, the certificate issuance process is itself vulnerable to routing attacks, allowing network-level adversaries to obtain trusted digital certificates for any victim domain.[3] These attacks have significant consequences for the integrity and privacy of online communications, as adversaries can use fraudulently obtained digital certificates to bypass the protection offered by encryption and launch man-in-the-middle attacks against critical communications.

How certificate authorities work. Domain control verification is a crucial process for domain owners to obtain digital certificates from CAs. Domain owners approach a CA to request a digital certificate, and the CA responds with a challenge that requires the owners to demonstrate control of an important network resource (for example, a website or email address) associated with the domain. Figure 8 illustrates HTTP verification where the CA requires the domain owner to upload a document to a well-known directory on its Web server and verify the upload over HTTP. Upon completion of the challenge, the CA issues the digital certificate to the domain owner.

Routing attacks on digital certificates. The domain control verification process creates a vulnerability to adversaries who can fake control of the network resources. Network-level adversaries can use routing attacks to hijack or intercept the traffic to the victim's domain such that the CA's request is routed to the adversaries instead[3] (step (5) in Figure 8). Adversaries can then answer the CA's HTTP request in step (6) and subsequently obtain a signed digital certificate from the CA for the victim domain. The attacks were successfully demonstrated in the real world, ethically.[3] The attacked domains were run on IP prefixes controlled by the researchers and had no real users or services. The adversary successfully obtained certificates for the victim domain from five top CAs in as little as 35 seconds (see the table here).

Five CAs were attacked and obtained certificates from. All were automated and none had any defenses against BGP attacks.[3]

	Let's Encrypt	GoDaddy	Comodo	Symantec	GlobalSign
Time to issue certificate	35s	<10min	51s	6min	4min
Human Interaction	No	No	No	No	No
Multiple Vantage Points	No*	No	No	No	No
Validation Method Attacked	HTTP	HTTP	Email	Email	Email

* No vantage points were deployed at time of attack. Let's Encrypt has since deployed multiple vantage point verification.

This work highlights the significant damage of routing attacks that can compromise the foundation of secure online communications and shows the urgent need for practical defenses. Furthermore, the attacks also apply to other systems that require demonstration of control on certain resources via verification requests, such as email verifications. The communication with the mail server can be hijacked or intercepted, and there is still a non-negligible amount of email messages that are unencrypted (for example, less than 20% of the emails from "icicibank.com," a bank website, are encrypted[k]).

Defenses to protect digital certificates. Many currently deployed defenses do not sufficiently protect digital certificates. Given the relatively short time required to obtain a fraudulent certificate, adversaries can get a certificate before the attack is mitigated, even if it is detected by monitoring systems. In addition, adversaries can potentially obtain a malicious certificate using only localized routing attacks that do not affect a large portion of the Internet. If a domain does not have a CAA DNS record (which is currently true of the vast majority of domains[17]), any CA is authorized to sign a certificate for that domain. Thus, adversaries only need to affect the route between one (of several hundred) CAs and the target domain to obtain a fraudulent certificate.

Birge-Lee et al.[3] recently proposed two practical application layer defenses. (1) Multiple Vantage Point Verification: building on the key insight that routing attacks may be localized, CAs can significantly decrease their vulnerability to attacks by performing domain verification from multiple vantage points and suspend certificate issuance in the case of inconsistent validation results. By adding only one additional vantage point, the probability of catching a localized routing attack on a domain increases from 61% to 84%. By having two additional vantage points, the probability of catching the attack reaches over 90% for 74% of the 1.8 million domains in the study. (2) BGP monitoring with route age heuristics: building on the key insight that anomalous and suspicious routing announcements are usually short-lived, CAs can require the routes to the domains to be active for a minimum time threshold before signing a certificate. This defense would force attacks to be active for over a day before the routes can be used to obtain a bogus certificate. Both defenses only require minimal deployment effort by the CAs with no change needed from domain owners or the routing infrastructure.

Multiple vantage point verification has gained significant traction. Let's Encrypt, the world's largest CA, has deployed multiple vantage point verification.[l,26] Furthermore, the prominent CDN CloudFlare has developed an API for CAs to perform multiple vantage point verification using its network.[m]

The Bitcoin Network

Bitcoin is the most widely used cryptocurrency to date with over 42 mil-

k Google Transparency Report; https://transparencyreport.google.com/safer-email/.

l Multi-Perspective Validation Improves Domain Validation Security, 2020; https://letsencrypt.org/2020/02/19/multiperspective-validation.html

m Securing Certificate Issuance using Multipath Domain Control Validation, 2019; https: //blog.cloudflare.com/secure-certificate-issuance

Figure 9. (a) New blocks mined by bitcoin nodes in different ASes are propagated to the whole network. (b) The attacker hijacks all prefixes pertaining to bitcoin nodes in the gray zone. Consequently, blocks mined by nodes in the gray zone won't be propagated further, which effectively isolates the gray zone.

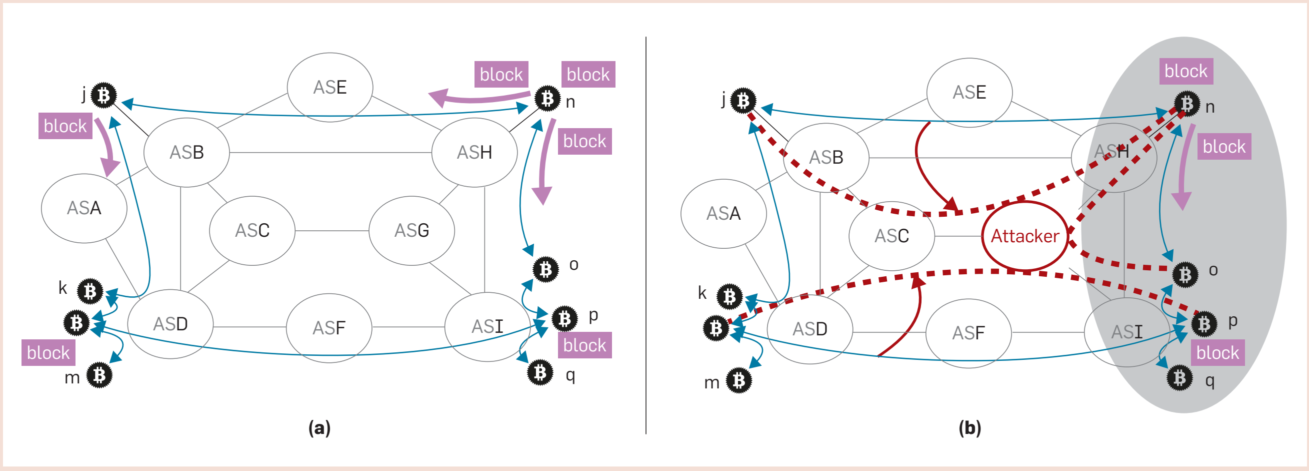
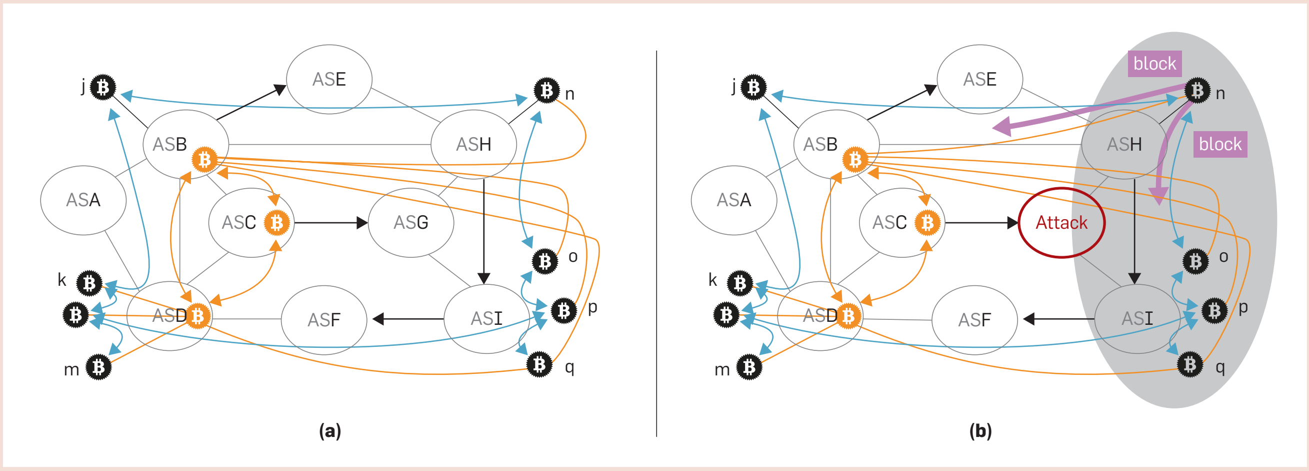


Figure 10. (a) Relay nodes are hosted in ASes that have no customer ASes and compose connected graph of direct peering links. Bitcoin clients connect to at least one relay node. (b) While routing attacks isolate the gray zone from the rest of the bitcoin network, blocks mined in the gray zone are propagated via relay nodes in the overlay SABRE network.



lion users.[n] However, network-level adversaries can launch routing attacks to partition the bitcoin network, effectively preventing the system from reaching consensus.[2] Besides Bitcoin, this attack is generally applicable to many peer-to-peer networks and is particularly dangerous against blockchain systems.

How Bitcoin works. Bitcoin is a peer-to-peer network in which nodes use consensus mechanisms to jointly agree on a (distributed) log of all the transactions that ever happened. This log is called the *blockchain* because it is composed of an ordered list (chain) of grouped transactions (blocks).

Special nodes, known as wallets, are responsible for originating transactions and propagating them in the network using a gossip protocol. A different set of nodes, known as miners, are responsible for verifying the most recent transactions, grouping them in a block, and appending this block to the blockchain. To do so, the miners need to solve a periodic puzzle whose complexity is automatically adapted to the computational power of the miners in the network.

Every time a miner creates a block, it broadcasts it to all the nodes in the network and receives freshly mined bitcoins. Besides the most recent transactions, the block contains a proof-of-work (a solution to the puzzle) that each node can independently verify before propagating the block further. In Figure 9a, node *n* "mines" a block which is then broadcasted hop-by-hop in the network.

As miners work concurrently, several of them may find a block at nearly the same time. These blocks effectively create "forks" in the blockchain, that is, different versions of the blockchain. The conflicts are eventually resolved as subsequent blocks are appended to each chain and one of them becomes longer. In this case, the net-

n Number of Blockchain wallet users worldwide, 2020; https://www.statista.com/statistics/647374/worldwide-blockchain-wallet-users/

work automatically discards the shorter chains, effectively discarding the corresponding blocks together with the miner's revenues.

Routing attacks on consensus. Network-level adversaries can perform routing attacks on bitcoin to partition the set of nodes into two (or more) disjoint components.[2] Consequently, the attacks disrupt the ability of the entire network to reach consensus. The adversary must divert and cut *all* the connections connecting the various components together. To do so, the adversary can perform an interception attack by hijacking the IP prefixes of each component and selectively dropping the connections crossing the components, while leaving the internal connections (within a component) untouched.

In Figure 9b, the adversary hijacks all prefixes pertaining to bitcoin nodes in the gray zone. Having gained control over the traffic toward these nodes (red lines), the adversary drops the connections between the clients that are within the gray zone and outside it, effectively creating a partition.

The impact of partition attacks is worrying. First, a partition attack can act as a denial-of-service attack: clients can neither properly propagate the corresponding transactions, nor verify the ownership of funds. Second, a partition attack can lead to high revenue loss for the miners: once the network reconnects, the shortest chain(s) will be discarded, permanently depriving miners of their rewards.

Defenses to protect the Bitcoin consensus. Apostolaki et al.[1] recently proposed SABRE to protect bitcoin from partition attacks. SABRE is an overlay network, composed of a small set of special bitcoin clients (relays) that receive, verify, and propagate blocks. Regular bitcoin clients can connect to one or more relays in addition to their regular connections. During a partition attack, SABRE relays stay connected to each other and to many bitcoin clients, allowing block propagation among the otherwise disconnected components. In Figure 10b, while clients in the gray zone are isolated from the rest of the network, a block mined by node n is propagated via the relay nodes (colored in orange) to the rest of the network.

SABRE achieves this by strategically choosing the ASes in which to host relay nodes. The key insight is that some ASes, such as those without customers, are naturally protected against routing attacks. By hosting relays in these ASes, SABRE can therefore maintain its connectivity and its ability to propagate blocks on behalf of bitcoin clients, even in the presence of routing attacks. Note that a bitcoin client only requires one unhindered connection to a SABRE relay to be protected.

In the SABRE network shown in Figure 10a, three ASes (*ASB*, *ASC*, *ASD*) are selected to host the relay nodes, which directly peer with each other and have no customer ASes. During routing attacks, the relay nodes stay connected to each other. For instance, if *ASG* (provider of *ASC*) announces the prefix of *ASB*, *ASC* would still prefer the route to *ASB* since it's via a peer. Additionally, all bitcoin clients keep at least one connection to the relay network during the attack. Even nodes such as node q which loses one of the connections to the relay network due to the attack, stays connected via another relay node.

Cross-Layer Solutions

We demonstrated the emerging threats of routing attacks to critical applications. Next, we outline lessons learned from the three applications, and discuss the importance of developing solutions at both the application and network layers.

For application developers. The most important takeaway is the significant impact of routing (in)security on Internet applications. When securing the application layer in isolation becomes difficult to achieve, we should think about cross-layer solutions that take into account routing properties at the network layer.

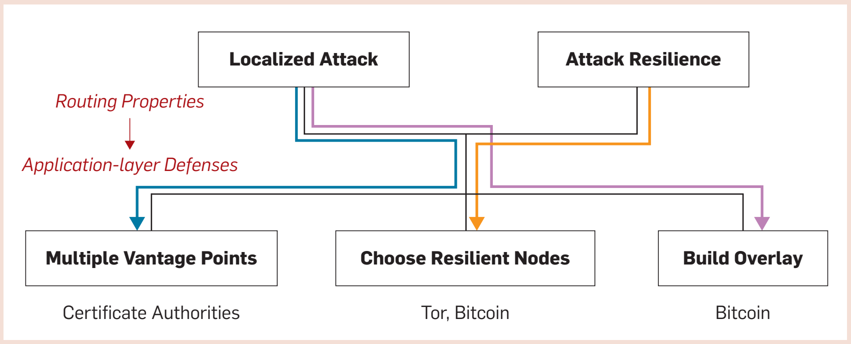
We outline two routing properties that are the key insights in building application-layer defenses: *localized attack*: attack announcements may not be propagated and visible to the whole Internet, and stealthy adversaries can carefully craft announcements to control propagation and only target certain regions; *attack resilience*: some ASes that receive the attack announcement may not be affected, that is, not favoring the malicious path and hence being "resilient" to the attack. This depends on the routing preferences, for example, if an AS receives the attack announcement from a provider while the legitimate path is through a peer, the AS will still prefer the legitimate path.

These two simple routing properties lead to three generalizable application-layer defenses shown in Figure 11, where the key property for each defense is highlighted.

Deploy multiple vantage points. Initiating connections from multiple vantage points increases the likelihood of detecting and circumventing a localized attack. Certificate Authorities can perform domain control verification from multiple vantage points to ensure that routes to the destination are consistent. This approach generalizes to a broad set of verification processes, where verifications from multiple sources would help lower the success of an attack and significantly increase the cost to an adversary. BGP monitoring systems also benefit from having more comprehensive data through multiple vantage points to detect stealthy attack.

Choose resilient nodes. Applications

Figure 11. Two routing properties serve as the key insights in developing application-layer defenses.



can strategically choose nodes/servers that are the most resilient to attacks. Tor clients may choose an entry relay that maximizes the probability of being resilient given the AS locations of the client and the relay. Bitcoin may choose relay nodes in certain ASes (for example, peer AS without customers) to avoid being affected by attacks. The specific implementation can vary based on the need of the applications and may even bring in RPKI as a criteria in choosing resilient nodes.

Build an overlay network. This approach can help mitigate some effects of routing attacks, for example, partitioning Bitcoin nodes, by providing alternative routes. It can be more effective when combined with "choosing resilient nodes," where the nodes in the overlay are carefully chosen to maximize the resilience to attacks. Bitcoin is an example application that benefits from an overlay to mitigate partitioning attacks, but the approach is generally applicable to many peer-to-peer networks.

For network operators. While application-layer defenses can provide immediate protections, we should also push for large-scale deployment of general defenses against sophisticated routing attacks. We recommend that ASes: adopt best practices outlined in the MANRS project, accelerate the adoption of RPKI by publishing ROAs and performing Route Origin Validation (ROV), and build consensus on a pathway to solving routing security issues (including full path security) once and for all. Furthermore, we outline two ways that synergize network operators with application developers.

Applications as starting points. Securing all 800K prefixes and 67K ASes seems like an impossible task. However, only a small portion of the prefixes play a heavy role in each application. For instance, only around 1100 ASes have Tor relays hosted on their prefixes, and one AS alone carries 23% of all Tor traffic.[21] Furthermore, in digital certificate issuance, a handful of certificate authorities issue the vast majority of certificates, and the domains are largely hosted on a few cloud and CDN providers (for example, five ASes including SquareSpace and Amazon host nearly half of the domains[3]). Finally, only five ASes host one third of all Bitcoin clients,[o] while 50% of all mining power is hosted in less that 100 prefixes.[2] If a few thousand ASes can take major steps to deploy routing security, the applications will receive tremendous benefits.

Applications as incentives. Popular applications—and their users—can help incentivize the deployment of routing security solutions by the actions they take, while ensuring the applications' security/privacy goals. For instance, Tor could favor certain relays that are hosted on authenticated prefixes, and domain owners could favor cloud hosting services that provide origin validations and favor certificate authorities hosted on authenticated prefixes. Similarly, miners could prefer hosting their infrastructure in ASes that provide origin validation, while regular client could prefer to connect to peers hosted on authenticated prefixes. These steps may help motivate network operators to validate their prefixes to offer better service to their customers, and eventually lead to a more secure routing infrastructure.

Conclusion

Often times, we focus on individual system layers in isolation. In neglecting routing (in)security, application developers underestimate the risks for their users. In focusing on availability threats, network operators underestimate the risks to Internet applications. By demonstrating the dire consequences of routing attacks on Internet applications, we stress the importance of cross-layer awareness and the need to deploy both application-layer and network-layer solutions. C

o Bitnodes. https://bitnodes.io/dashboard/.

References

1. Apostolaki, M., Marti, G., Muller, J., and Vanbever, L. SABRE: Protecting Bitcoin against routing attacks. In *Proceedings of Network and Distributed System Security Symp.*, 2019.
2. Apostolaki, M., Zohar, A., and Vanbever, L. Hijacking Bitcoin: Routing attacks on cryptocurrencies. In *Proceedings of IEEE Symp. on Security and Privacy*, 2017.
3. Birge-Lee, H., Sun, Y., Edmundson, A., Rexford, J., and Mittal, P. Bamboozling certificate authorities with BGP. In *Proceedings of USENIX Security Symp.*, 2018.
4. Birge-Lee, H., Wang, L., Rexford, J., and Mittal, P. SICO: Surgical interception attacks by manipulating BGP communities. In *Proceedings of ACM Con. Computer and Communications Security*, 2019.
5. Boldyreva, A. and Lychev, R. Provable security of S-BGP and other path vector protocols: Model, analysis and extensions. In *Proceedings of ACM Conf. Computer and Communications Security*, 2012.
6. Bush, R. and Austein, R. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810, RFC Editor, Jan. 2013.
7. Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *Proceedings of USENIX Security Symp.*, 2004.
8. Gill, P., Schapira, M., and Goldberg, S. Let the market drive deployment: A strategy for transitioning to BGP security. *ACM SIGCOMM*, 2011.
9. Goldberg, S. Surveillance without borders: The "traffic shaping" loophole and why it matters. *The Century Foundation*, 2017.
10. Hu, X. and Mao, Z.M. Accurate real-time identification of IP prefix hijacking. In *Proceedings of IEEE Symp. on Security and Privacy*, 2007.
11. Kent, S., Lynn, C., and Seo, K. Secure border gateway protocol (S-BGP). *IEEE J. Selected Areas in Commun. 18*, 4 (2000), 582–592.
12. Lad, M., Massey, D., Pei, D., Wu, Y., Zhang, B., and Zhang, L. PHAS: A prefix hijack alert system. In *Proceedings of USENIX Security Symp.*, 2006.
13. Lepinski, M. and Sriram, K. BGPsec Protocol Specification. RFC 8205, RFC Editor, Sept. 2017.
14. Lychev, R., Goldberg, S., and Schapira, M. BGP security in partial deployment: Is the juice worth the squeeze? *ACM SIGCOMM*, 2013.
15. Qiu, J., Gao, L., Ranjan, S., and Nucci, A. Detecting bogus BGP route information: Going beyond prefix hijacking. *SecureComm*, 2007.
16. Reuter, A., Bush, R., Cunha, I., Katz-Bassett, E., Schmidt, T.C., and Wahlisch, M. Towards a rigorous methodology for measuring adoption of RPKI route validation and filtering. *ACM SIGCOMM Computer Commun. Rev. 48*, 1 (2018), 19–27.
17. Scheitle, Q. et al. A first look at certification authority authorization (CAA). *SIGCOMM Comput. Commun. Rev.*, 48(2):10–23, May 2018.
18. Schlinker, B., Arnold, T., Cunha, I., and Katz-Bassett, E. PEERING: Virtualizing BGP at the edge for research. In *Proceedings of ACM SIGCOMM CoNEXT Conf.* Dec. 2019.
19. Shi, X., Xiang, Y., Wang, Z., Yin, X., and Wu, J. Detecting prefix hijackings in the Internet with Argus. In *Proceedings of Internet Measurement Conf.*, 2012.
20. Snijders, J. Practical everyday BGP filtering with AS PATH filters: PeerLocking. *NANOG-67*, Chicago, June, 2016.
21. Sun, Y., Edmundson, A., Feamster, N., Chiang, M., and Mittal, P. Counter-RAPTOR: Safeguarding Tor against active routing attacks. In *Proceedings of IEEE Symp. Security and Privacy*, 2017.
22. Sun, Y., Edmundson, A., Vanbever, L., Li, O., Rexford, J., Chiang, M., and Mittal, P. RAPTOR: Routing attacks on privacy in Tor. In *Proceedings of USENIX Security Symp.*, 2015.
23. Tan, H., Sherr, M., and Zhou, W. Data-plane defenses against routing attacks on Tor. In *Privacy Enhancing Technologies Symp.*, 2016.
24. Zhang, Z., Zhang, Y., Hu, Y.C., Mao, Z.M. and Bush, R. iSpy: Detecting IP prefix hijacking on my own. *ACM SIGCOMM*, 2008.
25. Zheng, C., Ji, L., Pei, D., Wang, J., and Francis, P. A lightweight distributed scheme for detecting IP prefix hijacks in real-time. *ACM SIGCOMM*, 2007.
26. Birge-Lee, H., Wang, L., McCarney, D., Shoemaker, R., Rexford, J., and Mittal, P. Experiences deploying multi-vantage-point domain validation at Let's Encrypt. In *Proceedings of USENIX Security Symp.*, 2021

Yixin Sun is an assistant professor at University of Virginia, Charlottesville, VA, USA.

Maria Apostolaki is a Ph.D. student at ETH Zurich.

Henry Birge-Lee is a student at Princeton University, Princeton, NJ, USA.

Laurent Vanbever is an associate professor at ETH Zurich.

Jennifer Rexford is a professor at Princeton University, Princeton, NJ, USA.

Mung Chiang is a Dean at Purdue University, West Lafayette, IN, USA.

Prateek Mittal is an associate professor at Princeton University, Princeton, NJ, USA.

research highlights

DOI:10.1145/3460221

To view the accompanying paper, visit doi.acm.org/10.1145/3460223

Technical Perspective
Race Logic Presents a Novel Form of Encoding

By Abhishek Bhattacharjee

MOORE'S LAW AND DENNARD scaling are waning. Yet the demand for computer systems with ever-increasing computational capabilities and power/energy-efficiency continues unabated, fueled by advances in big data and machine learning. The future of fields as disparate as data analytics, robotics, vision, natural language processing, and more, rests on the continued scaling of system performance per watt, even as traditional CMOS scaling ends.

The following paper proposes a surprising, novel, and creative approach to post-Moore's Law computing by rethinking the digital/analog boundary. The central idea is to revisit the idea of data representation and show how it is a critical design choice that cuts across hardware and software layers.

In particular, the authors develop the concept of *race logic*, where the key idea is to encode values as delays from some reference. Unlike pure analog approaches, race logic continues to encode data in binary form. However, unlike traditional digital logic, the *time* at which signals transition from zero to one encodes the value. In other words, relative propagation times of signals, usually considered a design artifact that modern digital technologies must work around, becomes a design feature and is leveraged to perform computation. Because of its reliance on data races, this computation enjoys a low number of "bit flips" and fewer wires versus conventional digital logic. The benefit is significantly better energy efficiency versus conventional digital design.

A key question is the suitability of race logic for different classes of computation. Naturally, not all computations are amenable to these encodings, but those that are stand to benefit significantly. The paper shows that machine learning classification may be one such target. In particular, the authors show how race logic can be used to "reverse" and "flatten" decision trees, widely used and a promising candidate for explainable AI, and architect a programmable race tree hardware accelerator for ensemble tree learning. Via a tour de force of engineering, the authors validate their research hypotheses via energy, throughput, and area utilization studies for an ASIC design of their accelerator, functional RTL implementations on an FPGA, SPICE model synthesis of the underlying primitives of race logic, and a fully automated toolchain for scikit-learn. The upshot is a full-stack and unusually detailed study from software structures down to device configurations.

This paper will be of wide interest to the computing community as it hints at many tantalizing research questions worthy of scientific inquiry. Perhaps the most natural one is race logic's promise for machine learning. The need for ultra-energy-efficient machine learning in edge and IoT devices is already exigent. Dynamic vision sensors, time-based image sensors, time to first spike and time of flight cameras, and address event representation-based sound sensors are just a few systems expected to drive sophisticated learning algorithms and race logic is particularly well-suited to reducing their energy needs. To fully realize these benefits, further research will be needed on automated design tools and flows that enable at-scale race logic, as well as software development environments, domain-specific languages, compilers, and more.

The following paper will be of wide interest to the computing community as it hints of many tantalizing research questions worthy of inquiry.

Perhaps even closer to my heart is the more abstract principles on which race logic rests. At its core, race logic is inspired by several aspects of how neuroscientists believe that the brain computes. These include, for example, the notion that time encodes computation, the concept of radial basis functions where larger signals trigger neurons more rapidly, and the inclusion of race logic primitives that are inspired by inhibitory post-synaptic potentials in the neo-cortex. Computer scientists have long been fascinated by the idea of drawing lessons from biology and nature to build better abstractions and methods for computing, spurring research on neuromorphic systems, natural algorithms, the emergence of intelligence, and more. These endeavors are often faced with the following question: To what degree is it useful for concepts from biology/nature to be replicated in systems/algorithms? Does, for example, the fact that computer systems rely on silicon and digital technologies, which differ from the elements and proteins used to realize life, mean that more abstract principles from natural computing need to be considered instead? And if so, what are the abstractions from nature appropriate for mimicry in computer systems?

Race logic offers perspective on this debate by lifting underlying principles of computation in the brain and abstracting them so that they may be suitable for deployment using silicon technologies. I believe this is what enables race logic to achieve efficiency across all three of sensor layer, learning algorithm, and architecture layer. As the authors point out, achieving all three is a rarity and, I believe, a testament to the educational value of this paper.

I hope you enjoy reading about race logic as much as I have. C

Abhishek Bhattacharjee is an associate professor of computer science at Yale University, New Haven, CT, USA.

DOI:10.1145/3460223

In-Sensor Classification With Boosted Race Trees

By Georgios Tzimpragos, Advait Madhavan, Dilip Vasudevan, Dmitri Strukov, and Timothy Sherwood

Abstract

When extremely low-energy processing is required, the choice of data representation makes a tremendous difference. Each representation (e.g., frequency domain, residue coded, and log-scale) embodies a different set of trade-offs based on the algebraic operations that are either easy or hard to perform in that domain. We demonstrate the potential of a novel form of encoding, race logic, in which information is represented as the delay in the arrival of a signal. Under this encoding, the ways in which signal delays interact and interfere with one another define the operation of the system. Observations of the relative delays (for example, the outcome of races between signals) define the output of the computation. Interestingly, completely standard hardware logic elements can be repurposed to this end and the resulting embedded systems have the potential to be extremely energy efficient. To realize this potential in a practical design, we demonstrate two different approaches to the creation of programmable tree-based ensemble classifiers in an extended set of race logic primitives; we explore the trade-offs inherent to their operation across sensor, hardware architecture, and algorithm; and we compare the resulting designs against traditional state-of-the-art hardware techniques.

1. INTRODUCTION

In embedded applications, where the computation and sensing are close in both time and space, the exact type of data is something that needs to be carefully considered. Typically, a sensor gathers analog information from the physical world and then converts it into a conventional digital signal. For example, a camera captures incident photons and, through the photoelectric effect, uses their energy to guide the charging of a cell. The voltage on the cell is read out to an analog-to-digital converter (ADC) that converts the measured voltage into a stream of zeros and ones. Although this binary-represented integer is perfectly efficient for storage as bits in a memory and for general-purpose computing operations, it is unclear whether this is the most *energy efficient* solution. We posit that there are other encodings that, although still capturing the relative values of the data to be encoded, are more efficient for in-sensor processing.

One such possible representation is pure analog signaling. There is a long history of machine-learning-like computing with analog devices. Although pure analog design is always an option, it comes with a number of challenges of its own. First, well-understood analog design rules always lag far behind digital rules in available technology node. High-density, high-performance, and low-energy CMOS analog parts can be hard to achieve because of this gap. Second, although analog design in these aggressive technology nodes is certainly possible, tighter margins for process variations and noise often drive analog designs to use larger gates than their digital counterparts. Ideally, we could keep the good parts of analog behavior, where the computation closely matches the capabilities of the underlying devices, without sacrificing the noise tolerance and layout simplicity of digital designs.

One class of logic that attempts to strike this balance is race logic.[10] The key idea behind race logic is to encode values as a *delay* from some reference. All signals, unlike pure analog approaches, are supposed to be 0 or 1 at all times. However, the *time* at which $0 \rightarrow 1$ transition happens encodes the value. Computations are then based on the relative propagation times of signals injected into a configurable circuit. In prior work, it was shown that the basic temporal operators MAX, MIN, and ADD-CONSTANT could efficiently solve a class of dynamic programming algorithms, and both synchronous and asynchronous versions have been evaluated.[10, 11] The inclusion of INHIBIT[20] opens the door to new computations, but the question of the efficiency of this approach to computing on larger and more general problems remains open.

To establish the interesting new capabilities that this more general race logic provides, we propose its application to a sensor-friendly yet machine-learning-ready encoding. For the experimental validation of our hypothesis, we complete an end-to-end evaluation that includes energy, throughput, and area utilization estimates for an ASIC design, a fully functional RTL implementation working in both simulation and on FPGA, SPICE models of the underlying primitives on which our system is built, a fully automated toolchain linking scikit-learn[15] software structures down to device configurations, and accuracy versus energy analysis across a set of decision tree ensembles and design parameters. Even without accounting for the extra energy savings of using an encoding more natural to the sensor, the presented system dramatically reduces the total energy usage required for classification with very low latency.

The original version of this paper is entitled "Boosted Race Trees for Low Energy Classification" and was published in *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, New York, NY, USA, 215–228; https://doi.org/10.1145/3297858.3304036

2. GENERALIZED RACE LOGIC

Race logic encodes information as timing delays. Computation then may happen through the purposeful manipulation of those delays rather than final logic levels, and the functions forming this logic's foundation are MAX, MIN, ADD-CONSTANT, and INHIBIT instead of AND, OR, and NOT.

Under the assumption that smaller delays in rise time encode smaller magnitudes and longer delays encode larger magnitudes, a MAX function should output a logical high only when all of its inputs have arrived (e.g., "gone high"). Therefore, only a single AND gate between its input wires is needed for its implementation. Figure 1(a) displays the symmetric nature of this function; the input that arrives first has to wait for the second one to arrive before the output responds. In the case of MIN, the function outputs a logical high when the first input arrives, and thus a single OR gate is all that is needed—Figure 1(b).

Furthermore, as the arrival time of the rising edge is what encodes information, delaying the $0 \rightarrow 1$ transition by a fixed amount of time is equivalent to constant addition (ADD-CONSTANT). Delaying a signal can be performed in multiple ways depending upon the implementation. In conventional synchronous digital logic, a sequence of flip-flops can be used, as shown in Figure 2. Asynchronous delay elements constructed out of current-starved inverters can provide an alternative, more energy-efficient method for performing the desired delay operation.[11]

Finally, the INHIBIT function, inspired by the behavior of inhibitory postsynaptic potentials in the neurons of the neocortex,[20] works as a nonlinear filter that has two inputs: an inhibiting signal and a data signal (that gets inhibited). If the inhibiting signal arrives first, the output is prevented from ever going high (no state transition), which corresponds to ∞ in the race logic world. On the other hand, if the data signal arrives before or at the same time as the inhibiting signal, the former is allowed to pass through unchanged.

Figure 1. Panels (a) and (b) show the implementation of MAX and MIN functions in race logic. Panel (c) represents an example waveform for $x = 2$ and $y = 4$.

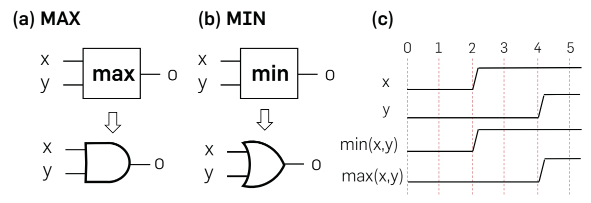


Figure 2. In race logic, adding a constant value k to a variable x is equivalent to delaying the rising edge of x by k clock cycles. Panel (a) shows how this delay can be achieved in conventional synchronous digital logic with the use of a shift-register. Panel (b) shows an example waveform for $x = 2$ and $k = 3$.

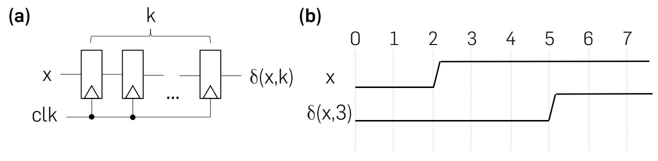


Figure 3 shows (a) the symbol used for i inhibiting j, (b) the function's state diagram as a Mealy machine, (c) and (d) two possible CMOS implementations, and (e) a waveform depicting its functionality through two examples. An even more efficient implementation, consisting of only a single PMOS pass gate, is also possible with a little customization.

Together this set of four operations allows us to deliberately engineer "race conditions" in a circuit to perform useful computation. The energy efficiency of the scheme comes from the very low number of "total bit flips" required. Compared to traditional approaches, race logic implementations require fewer wires because each of them can hold a multivalued signal (the delay). Moreover, these wires flip from 0 to 1 at most once through a logic evaluation as the signal-front washes across the circuit. Although not all computations are amenable to such an encoding, those that are have the potential to operate with very little energy. An open question answered in this work is if such a logic is applicable to any general learning or classification task.

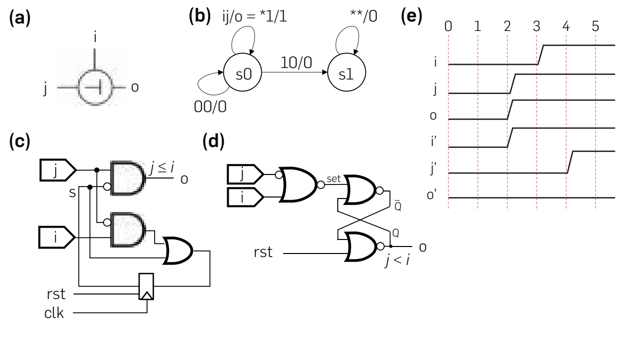
3. RETHINKING DECISION TREES

Although monolithic neural networks receive the lion's share of attention from the architecture community with respect to machine learning, decision trees have proven to be incredibly useful in many contexts and a promising solution towards explainable, high-performing AI systems. A decision tree, as its name denotes, creates a hierarchy of decisions, which consists of a set of leaves (labels) and a set of decisions to be made (branches) that lead one to those labels. One normally starts at the root and branches down the tree to find the relevant answer. Thus, classification is reduced to a sequence of binary decisions.

3.1. Reverse race trees

Existing race logic implementations, such as the DNA sequence alignment engine,[10] perform computation by observing the

Figure 3. Panel (a) introduces the symbol that from now on we will use to represent the INHIBIT operator. Panel (b) presents the state diagram of the corresponding Mealy machine. Each transition edge is labeled with the value of inputs i and j and the value of the output. The machine starts in state $s0$, which denotes that input j has not been inhibited by i, whereas state $s1$ indicates the opposite; j has been inhibited. Panels (c) and (d) show two possible implementations of the operator in a purely digital context. Finally, the waveform in Panel (e) depicts INHIBIT's functionality through two examples: (1) $i = 3$ and $j = 2$, and (2) $i' = 2$ and $j' = 4$. In this and other examples below, we assume that low to high transitions in the input signals are synchronized with the clock.

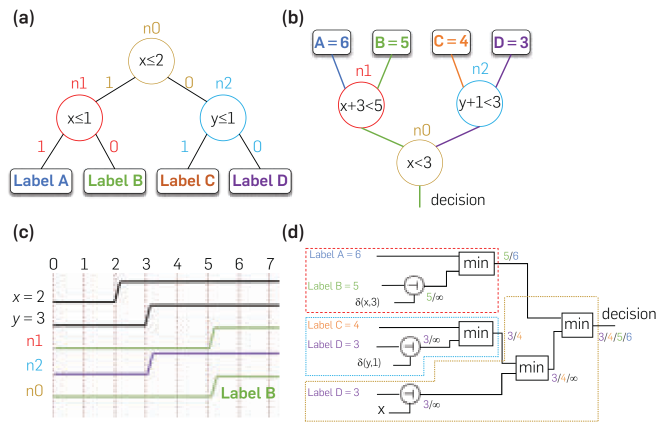


relative propagation times of signals injected into the circuit. Following this example, one approach to implement decision trees is by virtually turning them upside down; we can think of them as reverse tree networks that route possible packets from the leaves to the root. Initially, a unique delay-encoded label is assigned to each leaf. These labels then race against one another, and where two of them "meet," only one is allowed to propagate further. In the end, only the label associated with the "correct" leaf survives—the packet at the output of the network is unchanged, whereas all others get discarded along the way.

The decision tree that we use as a running example is presented in Figure 4(a). Figure 4(b) shows the flow of the four temporally encoded labels in the reverse tree for $x = 2$ and $y = 3$, with the corresponding waveform being illustrated in Figure 4 (c). Its race logic implementation is depicted in Figure 4(d). The upper two blocks, colored in red and blue, correspond to the tree's internal nodes ($x \leq 1$ and $y \leq 1$) and are implemented with the use of one INHIBIT and one MIN operators, whereas the bottom one, colored in yellow, is slightly more complicated as the label coming from its *False* path can take more than one values (either label C or label D).

Note that when reversing a tree, the *if* clauses in its nodes should be revised. For example, for label $D = 3$, $y \leq 1$ in node $n2$ must be rewritten as $y + 1 < 3$. To implement $y + 1 < 3$, feature y must be delayed by one clock cycle. As already discussed, when we rely on off-the-shelf digital circuits, shift-registers must be used to perform constant addition. However, these clocked components are relatively costly, and ideally, their usage should be constrained.

Figure 4. Panel (a) depicts an example decision tree. Panel (b) shows its "reverse" equivalent as well as the flow of four temporally encoded labels for $x = 2$ and $y = 3$. Panel (c) displays the corresponding waveform for the given example. Panel(d) presents its race logic implementation. Note that the leaf label associated with the *False* branch of a node plays the role of *j* in the INHIBIT operator, and the node's attribute (*x* or *y* in this example) serves as the inhibiting input *i*. Given that subtraction and variable addition are not supported by race logic, the attribute routed to an INHIBIT's controlling input must be adjusted accordingly; for example, $y \leq 1$ must be rewritten as $y + 1 < 3$.



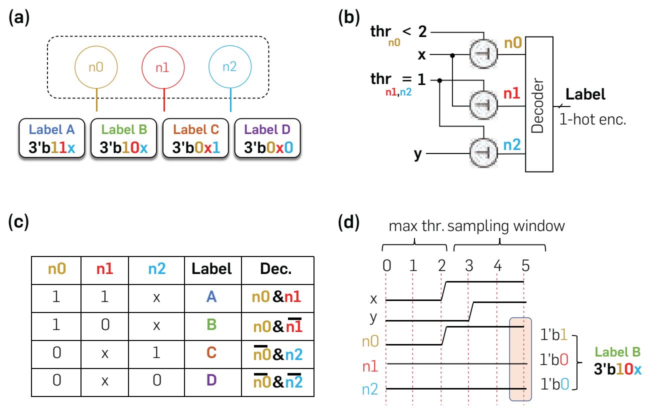
3.2. Flat race trees

An alternative way to look at a decision tree is as a set of independent and parallel rather than sequential decision rules that lead to a final prediction when combined accordingly.[1] Each leaf now can be represented as a logical function of the binary decisions encountered at the nodes on its path to the root. In other words, the tree gets "flattened" and each path from the tree root to a leaf corresponds to a unique combination of attribute test outcomes. The big idea behind the parallel execution of all these independent *if* clauses is shown in Figure 5(a). For example, the leftmost leaf is reached only when both $n0$ and $n1$ return *True*, whereas the output of $n2$ is inconsequential. The order that the outcomes of these conditions reveal to appear does not affect the final decision.

Figure 5(b) presents the implementation of such a flat tree in race logic. In contrast to conventional digital logic approaches, where the size and performance of the circuit realizing the desired threshold functions (each node is a binary decision) are directly related to the resolution of the associated attribute and threshold values, this is not the case here. In race logic, a range of magnitudes can be encoded on a single wire with a single edge. Thus, only one INHIBIT gate is required per tree node.

Moreover, because the decisions related to the various tree paths are mutually exclusive and the maximum threshold value is statically known, the transition from the temporal domain to binary happens seamlessly, without the need for any special circuitry. Figure 5(c) presents the truth table describing the functionality of the decoder that associates nodes' decisions with one of the leaf labels. Figure 5(d) shows the resulting waveform for $x = 2$ and $y = 3$. In the given example, the maximum threshold value is 2; thus, all node decisions can be safely considered final after 2 clock cycles, and the outcome of $n0$, $n1$, and $n2$ conditions can be read at any time after that.

Figure 5. A decision tree can be viewed as a set of independent decision rules that lead to one and only one leaf when combined accordingly. Hence, the threshold functions corresponding to each node (and lead to these decisions) can be executed in parallel, as shown in Panel (a). Panel (b) depicts the race logic implementation of this flattened decision tree with the use of INHIBITs, where thresholds play the role of the gates' controlling inputs. Panel (c) presents the truth table that defines the decoder's functionality. Panel (d) displays the resulting waveform for $x = 2$ and $y = 3$.



n0	n1	n2	Label	Dec.
1	1	x	A	n0&n1
1	0	x	B	n0&$\overline{\text{n1}}$
0	x	1	C	$\overline{\text{n0}}$&n2
0	x	0	D	$\overline{\text{n0}}$&$\overline{\text{n2}}$

4. END-TO-END ARCHITECTURE

4.1. From sensor to delay coded input

Whenever a different encoding is considered, the cost of translation in and out of that encoding should be taken into account. However, race logic is such a natural direct target for sensors that we can instead consider a case where sensing and processing are tightly integrated. Figure 6 presents an end-to-end architecture for temporal processing.

Because sensory input is analog in nature, most sensors begin with a measured voltage or current, which is then converted to a digital output with the use of ADCs. ADCs traditionally return digital binary values; however, given that information in the time domain is now useful for computation, the design of these components can be significantly simplified. For instance, the costly time-to-digital conversion (TDC) in the ADCs is redundant and can be skipped.[6] Examples of sensing systems that can provide directly time-encoded outputs, without TDCs, include Dynamic Vision Sensors (DVS),[9] Asynchronous Time-based Image Sensors (ATIS),[16] Time To First Spike (TTFS)[17] and Time-of-Flight (ToF)[14] cameras, and AER (Address Event Representation) Ear[2] sound sensors.

Given the properties of race logic and the nature of the above-described sensing systems, raw delay-coded data can be directly provided to temporal accelerators to achieve a more efficient sensor-accelerator integration.

4.2. Programmable race trees architecture

Reverse and flat tree encodings provide two ways of implementing decision trees in race logic. The reverse tree idea is of particular interest as it is unlike any other network. Typically, in a network, the packet contents are inert with respect to routing. For example, in the case of sorting networks, the packet values are used for routing, but they also have numerical content external to the network.[12] In reverse trees, the packets are externally assigned values that are symbolic and contain no useful numerical content—in much the same way that numbers in Sudoku are used symbolically, but not numerically. However, internal to the network, as part of the routing architecture, packet values do take part in numerical operations.

The idea behind the flat tree approach is much simpler. This simplicity results in a more compact and efficient hardware design—fewer shift-registers and a smaller interconnection network are needed. Due to these reasons, we consider flat trees as our design of choice for the rest of the paper. Figure 7 presents a programmable architecture for the hardwired design of Figure 5.

To ensure any delay-coded threshold value and any input feature can be routed to the necessary nodes of the tree, we use two configurable crossbars. The decoder, which transforms INHIBITs' outputs into a memory address, is built from AND gates and inverters. Note that although INHIBITs are returning temporal signals, the sampled outputs form a typical binary vector. Prior to the next computation, the circuit must be reset.

Figure 8 presents our system architecture for a tree-based ensemble learner. To keep the overhead of the clocked components low, we organize trees into groups and share the same shift-register and buffer (used for the generation of the delay-coded thresholds and the local buffering of the delay-coded input features). Of course, the cost of the crossbars increases with the size of these groups. This trade-off should be further analyzed for maximum efficiency; crossbars can be replaced by any other more efficient configurable routing network without any effect in the system's functionality. Finally, because the data retrieved from memory are in a regular binary encoding, the implementation of the weighted voting scheme is based on typical binary adders. Once the prediction values of all trees have been summed, a comparison between them takes place to find the class with the highest score and determine the system's final "guess."

5. EVALUATION

5.1. Methodology

To evaluate the proposed designs and identify opportunities for further improvement, we created analytical and empirical

Figure 6. End-to-end temporal architecture for in-sensor processing.

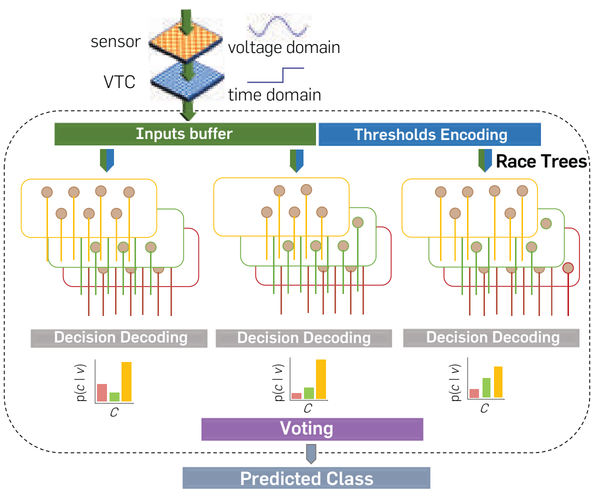
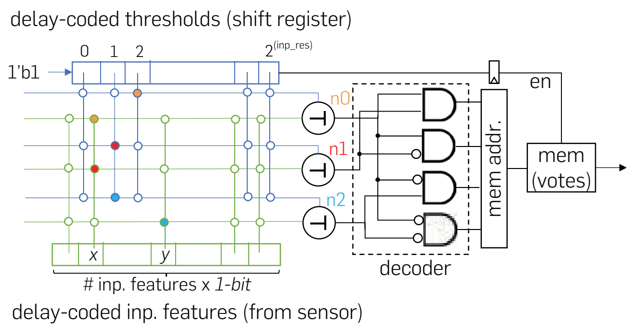
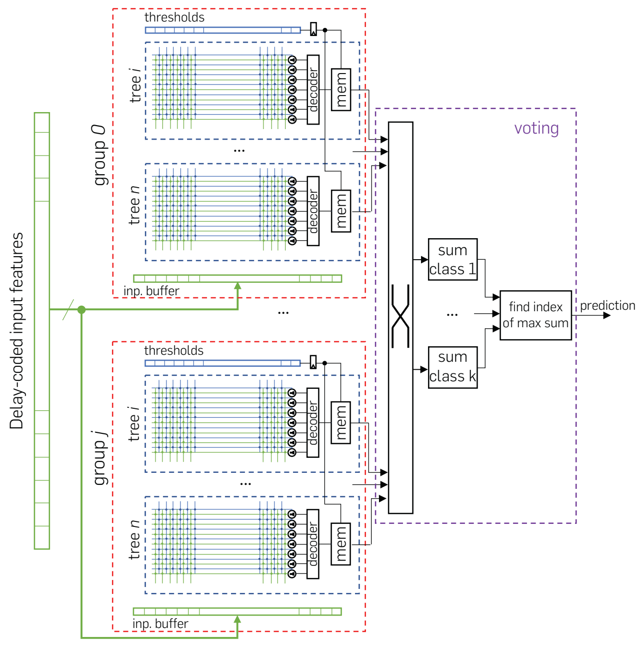


Figure 7. Programmable race logic accelerator for a decision tree of depth 2 (before flattening). The length of the shift-register, used for the temporal encoding of thresholds, is defined by the resolution of input features. The memory block shown on the right side of the decoder is useful in the case of tree ensembles, where a weighted voting scheme follows.



power and area models for the basic components of our architecture. We also build a Python-based development flow, as shown in Figure 9, which attaches to the popular scikit-learn library[15] and leverages the power of hardware templates and PyRTL[4]—a Python embedded hardware design language—to generate synthesizable RTL code. More specifically, once the model is trained, the tool analyzes the importance of input features, explores the learners' performance against lower resolution data, proceeds with votes (content of tree leaves) quantization, generates either a customized hardware design or a configuration file, and performs cross-checking verification. To obtain the desired implementation results, we use open-source tools[22,24] and a publicly available 14 nm standard cell library.[3] The operational voltage and frequency are 0.55 V and 1,000 MHz, respectively. In our energy and throughput calculations, we assume a 500 MHz clock and, to compensate for the lack of a wire load model, we do not scale our power numbers; under this assumption, each operation consumes twice its nominal energy.[19]

Figure 8. Lower-level diagram of the architecture presented in Figure 6. The shown circuit implements a configurable race logic accelerator for tree-based ensemble learners.



5.2. Implementation results

In recent years, an explosion of hardware accelerated machine learning activity has resulted in a wide variety of ASIC architectures, which we can use for comparison. While MNIST is very simple, it is complex enough to demonstrate the principles involved and, because it is the most commonly used dataset in the context of extremely low-power classifiers, facilitates a comparison with state-of-the-art.

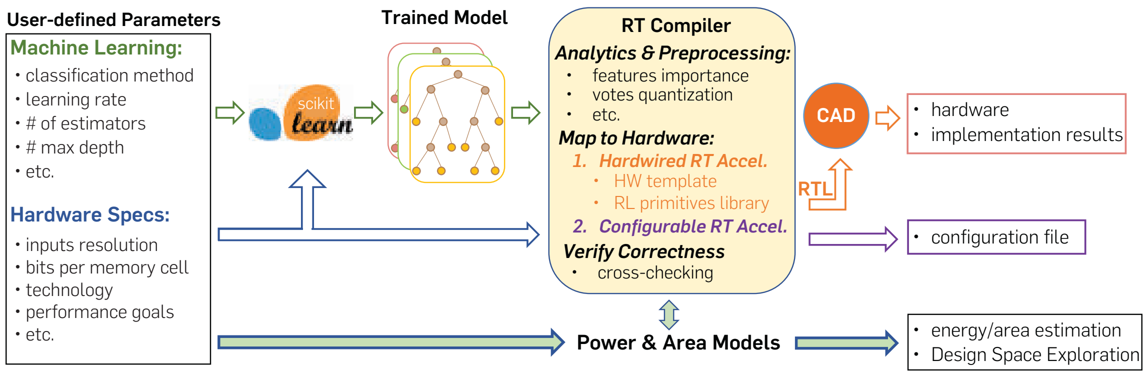
An accuracy versus energy comparison between the proposed race trees – represented by green dots – and state-of-the-art low-power classifiers is shown in Figure 10. Moreover, Figure 11 illustrates an accuracy versus energy-delay product comparison, which better elucidates the efficiency gap between race trees and its counterparts. The technique used for training the race trees is gradient boosting. We note that we do not perform any parameter fine-tuning to improve learners' performance and that race logic does not introduce any sources of inaccuracy.

In more detail, a classifier consisting of 1,000 race trees of depth 6 (before flattening) gets 97.45% accuracy and dissipates 31.35 nJ of energy per prediction. A more efficient solution, consisting of 200 trees of depth 6, achieves a performance of 95.7% with energy numbers as low as 7.8 nJ per prediction. By increasing the trees' depth to 8, the accuracy increments by 0.5%. This improvement comes at the expense of 16.1 nJ of additional energy per prediction. More results can be found in Table 1.

6. CONCLUSION

If machine learning is the engine, then raw data is the fuel, and most approaches consume a great deal of it. As machine learning techniques continue to find new and compelling applications across a wide range of computing

Figure 9. Overview of the developed toolchain. For the training part, the open-source scikit-learn framework[15] is used. The tool (a) provides the user with the options to quantize input features and/or trees' votes, (b) supports the automatic generation of race trees circuitry or the configuration of a programmable architecture directly from scikit-learn structures, (c) assists trade-off analysis with the use of analytical architectural models, and (d) facilitates hardware design evaluation through cross-checking with software models.



tasks, the desire to bring that computational power into even our lowest power devices will only continue to grow. Applying these complex algorithms without resorting to the use of significant amounts of energy remains an important challenge and the choice of data representation is an important cross-layer factor that impacts everything from the sensor to the end product of learning. In this paper, we show that the natural relationship between modern decision tree algorithms, new advances in race logic, and the underlying sensors themselves provide new opportunities for extremely efficient classification. Although it is rare to come across a change that appears simultaneously beneficial across all three of the sensor, learning algorithm, and architecture layers, a delay code seems to be one such rarity. Others have already shown that it is advantageous from an analog perspective to leave signals as they are, and trivially convert them to a race encoding, than to convert them to a pure digital representation. At the algorithm level, little is needed in the way of changes—one must just be mindful of the depth and configuration of the existing decision tree algorithms. At the architecture level, the improvements for these considerations are dramatic both in hardwired and programmable configurations. The resulting system has a shallow critical path and induces exceedingly few bit transitions as the computation propagates through race trees. Example designs that demonstrate this behavior are available at our GitHub repository.[a]

Looking forward, the end of traditional CMOS scaling has reenergized the search for novel models of computation and a requestioning of digital/analog boundaries. As our ability to deliver useful computation becomes fully bounded by energy consumption, the objective shifts from performing operations at the lowest possible latency to the highest possible efficiency. We believe that this work is an important step in that direction and invites a serious reconsideration of the interfaces between the analog and digital worlds. When the goal is to process sensor data locally, the choice of data representation does not only affect the way computing happens but also dictates the structure and efficiency of the required converters, which often impose a nonnegligible overhead to the system's performance. Although we have not explicitly considered the gains in efficiency possible at the circuit level from avoiding the full transition to binary, there is reason to believe it could be significant.[13] In fact, there may be many other advantages to temporal models of computation as a more general proposition, perhaps even as a way of encoding more general learning systems inspired by neural-computation[20] or enabling emerging circuit technologies.[21]

Figure 10. Accuracy vs. energy scatter plot for state-of-the-art machine learning accelerators: a,[23] b,[7] c,[18] d,[8] e,[5] f.[5] For easier comparison, all results have been scaled to 28 nm. Green dots represent race trees.

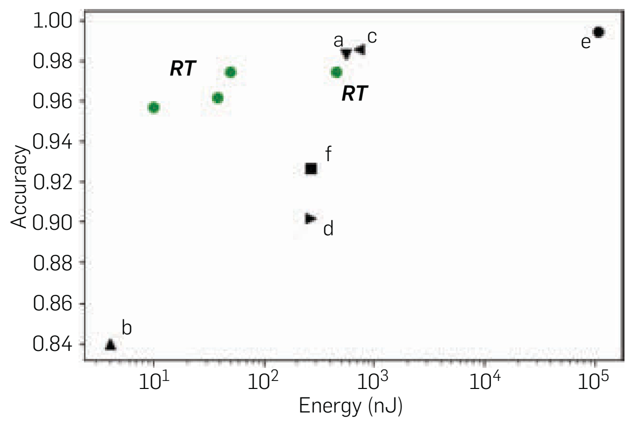
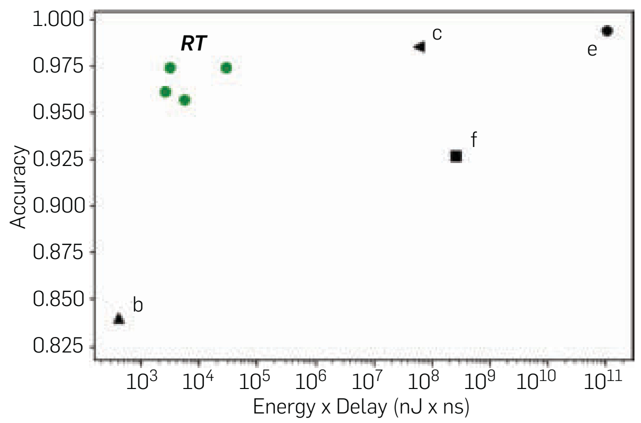


Figure 11. Accuracy vs. energy-delay product scatter plot for state-of-the-art machine learning accelerators: b,[7] c,[18] e,[5] f.[5] Green dots represent race trees.



Acknowledgments

This material is based upon the work supported by the National Science Foundation under Grants No. 1763699, 1740352, 1730309, 1717779, 156393.

Advait Madhavan acknowledges support under the Cooperative Research Agreement between the University of Maryland and the National Institute of Standards and

a https://github.com/UCSBarchlab/RaceLogic

Table 1. Synthesis results for hardwired race trees produced by Yosys[24] using a publicly available 14nm standard cell library[3].

# Trees	Depth	Inp. res. (bits)	Mem. bits (per vote)	Accuracy	Latency (CCs)	Power (mW)	Area (mm^2)	Freq. (MHz)
1,000	6	8	8	97.48%	273	521	0.46	1,000
1,000	6	4	4	97.45%	33	475	0.45	1,000
200	8	4	4	96.18%	31	384	0.33	1,000
200	6	4	4	95.72%	31	125	0.13	1,000

Technology, Physical Measurement Laboratory. Award 70NANB 14H209 through the University of Maryland.

Dilip Vasudevan was supported by the Advanced Scientific Computing Research (ASCR) program and funded by the U.S. Department of Energy, Office of Science. Lawrence Berkeley National Laboratory operates under Contract No. DE-AC02-05CH11231.

Last but not least, the authors would like to thank James E. Smith, Jennifer Volk, Georgios Michelogiannakis, David Donofrio, John Shalf, and the anonymous reviewers for their helpful comments.

References

1. Bermak, A., Martinez, D. A compact 3d vlsi classifier using bagging threshold network ensembles. *IEEE Trans. Neural Netw. Learn. Syst. 14*, 5 (2003), 1097–1109.
2. Chan, V., Liu, S.C., van Schaik, A. Aer ear: a matched silicon cochlea pair with address event representation interface. *IEEE Trans Circuits Syst I Regul Pap. 54*, 1(2007), 48–59.
3. Chen, S., Wang, Y., Lin, X., Xie, Q., Pedram, M. Performance prediction for multiple-threshold 7nm-FinFET-based circuits operating in multiple voltage regimes using a cross-layer simulation framework. In *2014 SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*(Millbrae, CA, 2014), 1–2. doi: 10.1109/S3S.2014.7028218. https://ieeexplore.ieee.org/document/7028218.
4. Clow, J., Tzimpragos, G., Dangwal, D., Guo, S., McMahan, J., Sherwood, T. A pythonic approach for rapid hardware prototyping and instrumentation. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)* (Ghent, 2017), 1–7, doi: 10.23919/FPL.2017.8056860. https://ieeexplore.ieee.org/document/8056860.
5. Esser, S.K., Appuswamy, R., Merolla, P., Arthur, J.V., Modha, D.S. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems*, 2015, 1117–1125. https://papers.nips.cc/paper/2015/hash/10a5ab2db37feedfdeaab192ead4ac0e-Abstract.html.
6. Guo, X., Qi, X., Harris, J.G. A time-to-first-spike cmos image sensor. *IEEE Sens. J. 7*, 8(2007), 1165–1175.
7. Kim, J.K., Knag, P., Chen, T., Zhang, Z. A 640M pixel/s 3.65mW sparse event-driven neuromorphic object recognition processor with on-chip learning. In *2015 Symposium on VLSI Circuits (VLSI Circuits)* (Kyoto, 2015), C50–C51, doi: 10.1109/VLSIC.2015.7231323. https://ieeexplore.ieee.org/document/7231323.
8. Kung, J., Kim, D., Mukhopadhyay, S. A power-aware digital feedforward neural network platform with backpropagation driven approximate synapses. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2015, 85–90.
9. Lichtsteiner, P., Posch, C., Delbruck, T. A 128×128 120db 15μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-St. Circ. 43*, 2(2008), 566–576.
10. Madhavan, A., Sherwood, T., Strukov, D. Race logic: a hardware acceleration for dynamic programming algorithms. *Comput. Architect. News 42*, 3(2014), 517–528.
11. Madhavan, A., Sherwood, T., Strukov, D. Energy efficient computation with asynchronous races. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)* (Austin, TX, 2016), 1–6. doi: 10.1145/2897937.2898019. https://ieeexplore.ieee.org/document/7544351.
12. Najafi, M.H., Lilja, D.J., Riedel, M., Bazargan, K. Power and area efficient sorting networks using unary processing. In *2017 IEEE International Conference on Computer Design (ICCD)* (Boston, MA, 2017), 125–128. doi: 10.1109/ICCD.2017.27. https://ieeexplore.ieee.org/document/8119200.
13. Naraghi, S. Time-based analog to digital converters, 2009. https://oatd.org/oatd/record?record=handle%5C%3A2027.42%5C%2F64787.
14. Niclass, C., Soga, M., Matsubara, H., Kato, S., Kagami, M. A 100-m range 10-frame/s 340×96-pixel time-of-flight depth sensor in 0.18-μm cmos. *IEEE J. Solid-St. Circ. 48*, 2(2013), 559–572.
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot M., Duchesnay, E. Scikit-learn: machine learning in Python. *J. Mach. Learn. Res. 12* (2011), 2825–2830.
16. Posch, C., Matolin, D., Wohlgenannt, R., Hofstäatter, M., Schäon, P., Litzenberger, M., Bauer, D., Garn, H. Biomimetic frame-free hdr camera with event-driven pwm image/video sensor and full-custom address-event processor. In *2010 IEEE on Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2010, 254–257.
17. Qi, X., Guo, X., Harris, J.G. A time-to-first spike CMOS imager. In *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)* (Vancouver, BC, 2004). IV–824. doi: 10.1109/ISCAS.2004.1329131. https://ieeexplore.ieee.org/document/1329131.
18. Reagen, B., Whatmough, P., Adolf, R., Rama, S., Lee, H., Lee, S.K., Hernández-Lobato, J.M., Wei, G.Y., Brooks, D. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)* (2016). IEEE Press, 267–278. doi: 10.1109/ISCA.2016.32. https://dl.acm.org/doi/10.1145/3007787.3001165.
19. Shalf, J., Dosanjh, S., Morrison, J. Exascale computing technology challenges. In *High Performance Computing for Computational Science – VECPAR 2010*. J. M. L. M. Palma, M. Daydé, O. Marques, and J. C. Lopes, eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, 1–25.
20. Smith, J. Space-time algebra: A model for neocortical computation. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)* (Los Angeles, CA, 2018), 289–300. doi: 10.1109/ISCA.2018.00033. https://ieeexplore.ieee.org/document/8416835.
21. Tzimpragos, G., Vasudevan, D., Tsiskaridze, N., Michelogiannakis, G., Madhavan, A., Volk, J., Shalf, J., Sherwood, T. A computational temporal logic for superconducting accelerators. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20. Association for Computing Machinery, New York, NY, USA, 2020, 435–448.
22. Vasudevan, D., Butko, A., Michelogiannakis, G., Donofrio, D., Shalf, J. Towards an integrated strategy to preserve digital computing performance scaling using emerging technologies. In *High Performance Computing*. J. M. Kunkel, R. Yokota, M. Taufer, and J. Shalf, eds. Springer International Publishing, Cham, 2017, 115–123.
23. Whatmough, P.N., Lee, S.K., Lee, H., Rama, S., Brooks, D., Wei, G. 14.3 A 28nm SoC with a 1.2 GHz 568nJ/prediction sparse deep-neural-network engine with > 0.1 timing error rate tolerance for IoT applications. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. (San Francisco, CA, 2017), 242–243. doi: 10.1109/ISSCC.2017.7870351. https://ieeexplore.ieee.org/document/7870351.
24. Wolf, C., Glaser, J. Yosys–A free verilog synthesis suite. In *Proceedings of Austrochip* (2013). https://www.semanticscholar.org/paper/Yosys-A-Free-Verilog-Synthesis-Suite-Wolf-Glaser/65b4a1136599d74ada27ce5226f02dda06d2ccda.

Georgios Tzimpragos (gtzimpragos@cs.ucsb.edu), Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA, USA.

Advait Madhavan (advait.madhavan@nist.gov), Physical Measurements Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, USA.

Dilip Vasudevan (dilipv@lbl.gov), Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

Dmitri Strukov (strukov@ece.ucsb.edu), Electrical and Computer Engineering Department, University of California at Santa Barbara, Santa Barbara, CA, USA.

Timothy Sherwood (sherwood@cs.ucsb.edu), Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA, USA.

DOI:10.1145/3460225

To view the accompanying paper, visit doi.acm.org/10.1145/3460227 rh

Technical Perspective
A Chiplet Prototype System for Deep Learning Inference

By Natalie Enright Jerger

THE FOLLOWING PAPER, "Simba: Scaling Deep-Learning Inference with Chiplet-Based Architecture," by Shao et al. presents a scalable deep learning accelerator architecture that tackles issues ranging from chip integration technology to workload partitioning and non-uniform latency effects on deep neural network performance. Through a hardware prototype, they present a timely study of cross-layer issues that will inform next-generation deep learning hardware, software, and neural network architectures.

Chip vendors face significant challenges with the continued slowing of Moore's Law causing the time between new technology nodes to increase, skyrocketing manufacturing costs for silicon, and the end of Dennard scaling. In the absence of device scaling, domain specialization provides an opportunity for architects to deliver more performance and greater energy efficiency. However, domain specialization is an expensive proposition for chip manufacturers. The non-recurring engineering costs of producing silicon are exorbitant including design and verification time for chips containing billions of transistors. Without significant market demand, it is difficult to justify this cost.

Fortunately for computer architects, machine learning is a domain where specialized hardware can reap performance and power benefits. Machine learning has seen widespread adoption in recent years and its need for more compute, storage, and energy-efficiency is only growing as models take on more complex tasks. Domain specialization improves performance and energy-efficiency by eschewing all of the hardware in modern processors devoted to providing general-purpose capabilities. Furthermore, architectures targeting machine learning feature regular arrays of simple processing elements (primarily doing multiply accumulate) that can potentially be scaled to large numbers and may offer opportunities to ease verification.

Given the billions of transistors that can fit on a single large die, is scaling up the number of processing elements in a machine learning accelerator trivial? The slowing of Moore's Law makes it increasingly difficult to pack more functionality on a single chip. If transistor sizes stay constant, more functionality could be integrated via larger chips. However, larger chips are undesirable due to significantly higher costs. Verification costs are higher. Manufacturing defects in densely packed logic can dramatically reduce the wafer yield. Lower yield translates into higher manufacturing cost.

A promising solution to combat these yield and verification challenges is to design and fabricate smaller chips (chiplets) and integrate those chiplets into one system via a package-level solution such as a silicon interposer or organic substrate. Small chiplets are cheap to manufacture; a manufacturing defect on a chiplet has a smaller impact on the total wafer yield. The reduced functionality of an individual chiplet is compensated for by integrating a large number of chiplets into the system. This concept of chiplet-based architectures has been explored in CPUs and in GPUs. Simba develops an architecture and hardware prototype to demonstrate how chiplets can be effective employed in machine learning accelerators.

While the focus of the paper is a scalable approach to deliver increasing performance and energy efficiency in datacenter-scale inference accelerators, one exciting feature of the proposed chiplet-based approach as is the ease with which it can be scaled across different market segments. Each chiplet can standalone as a complete system; therefore, a single chiplet could be used as an edge device or a small number of chiplets could be integrated for a consumer-class device. Given the design, verification, and manufacturing costs associated with fabricating silicon, a single chiplet design that delivers for all market segments is a compelling solution.

Another insightful aspect of this paper is the emphasis on hardware/software co-design. Given the myriad challenges facing hardware design and manufacturing, it is imperative that software systems be thoughtfully designed to combat any non-uniformities introduced by the hardware solutions. Non-uniform memory access (NUMA) effects have long been studied for multi-socket, multi-board designs. However, this study provides new insights specifically targeting machine learning applications and hierarchical interconnects with different bandwidth and latency characteristics that will be found in these future chiplet-based architectures. On the software side, they consider the impact of workload partitioning and communication-aware data placement. Through detailed case studies, this paper makes a compelling, evidence-based argument for co-design.

The deep neural network accelerator design space is rich with exciting start-ups and big-name companies producing new silicon. A number of open challenges and questions remain. In addition to hardware/software co-design, can the neural network architectures themselves be adapted to run more efficiently on the given hardware. If a single chiplet can serve a range of market segments, we need software and runtime solutions to adapt the network architecture to run efficiently on each instance of the system. How can we adapt this chiplet-based approach to build custom, heterogeneous hardware solutions at low cost? The hardware prototype in this paper provides a compelling foundation for further research in chiplet-based accelerator architectures. C

Natalie Enright Jerger is a professor in the Department of Electrical and Computer Engineering at the University of Toronto, where she also serves as the Canada Research Chair in Computer Architecture.

DOI:10.1145/3460227

Simba: Scaling Deep-Learning Inference with Chiplet-Based Architecture

By Yakun Sophia Shao, Jason Cemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler

Abstract

Package-level integration using multi-chip-modules (MCMs) is a promising approach for building large-scale systems. Compared to a large monolithic die, an MCM combines many smaller chiplets into a larger system, substantially reducing fabrication and design costs. Current MCMs typically only contain a handful of coarse-grained large chiplets due to the high area, performance, and energy overheads associated with inter-chiplet communication. This work investigates and quantifies the costs and benefits of using MCMs with fine-grained chiplets for deep learning inference, an application domain with large compute and on-chip storage requirements. To evaluate the approach, we architected, implemented, fabricated, and tested Simba, a 36-chiplet prototype MCM system for deep-learning inference. Each chiplet achieves 4 TOPS peak performance, and the 36-chiplet MCM package achieves up to 128 TOPS and up to 6.1 TOPS/W. The MCM is configurable to support a flexible mapping of DNN layers to the distributed compute and storage units. To mitigate inter-chiplet communication overheads, we introduce three tiling optimizations that improve data locality. These optimizations achieve up to 16% speedup compared to the baseline layer mapping. Our evaluation shows that Simba can process 1988 images/s running ResNet-50 with a batch size of one, delivering an inference latency of 0.50 ms.

1. INTRODUCTION

Deep learning (DL) has become critical for addressing complex real-world problems. In particular, deep neural networks (DNNs) have demonstrated their effectiveness across a wide-range of applications. State-of-the-art DNNs[12] require billions of operations and hundreds of megabytes to store activations and weights. Given the trend toward even larger and deeper networks, the ensuing compute and storage requirements motivate the large-scale compute capability in DL hardware, which is currently addressed by a combination of large monolithic chips and homogeneous multi-chip board designs.[9, 15] Previously proposed multi-chip DL accelerators have focused on improving total compute throughput and on-chip storage size but have not addressed the scalability challenges associated with building a large-scale system with multiple discrete components.

Recently, the need for high compute throughput in an era of slowing transistor scaling has motivated advances in multi-chip-module (MCM) integration to build large-scale CPUs[3] and GPUs.[1] MCM packaging approaches can also reduce cost by employing smaller chiplets connected together postfabrication, as yield losses cause fabrication cost to grow super-linearly with die size. Packaging technologies such as organic substrates[13] and silicon interposers[11] can be used to assemble a large-scale MCM system. In addition, the recent advances in package-level signaling offer the necessary high-speed, high-bandwidth signaling needed for a chiplet-based system.[24] As a result, chiplet-based MCM systems can provide improved performance more efficiently than the board-level integration but with lower cost than monolithic chips. Although MCMs have been used for general compute systems, applying MCMs to high-performance DNN inference algorithms has not been previously examined. Specific challenges stem from the natural nonuniformity between on-chip and on-package bandwidth and latency. Although multi-chip systems also exhibit similar forms of nonuniformity, this paper focuses on the specific characteristics of MCM-based systems as they provide a natural progression beyond monolithic single-chip inference accelerators.

This paper presents Simba, a scalable deep-learning inference accelerator employing multi-chip-module-based integration. Each of the Simba chiplets can be used as a standalone, edge-scale inference accelerator, whereas multiple Simba chiplets can be packaged together to deliver a data-center-scale compute throughput. To explore the challenges and evaluate the benefits of MCM-based inference accelerator architectures, we designed, implemented, and fabricated a prototype of Simba, consisting of 36 chiplets connected via a mesh network in an MCM.[25] We specifically examine the implications of the nonuniform network access (NUNA) architecture with nonuniform latency and bandwidth for on-chip and on-package communication that lead to significant latency variability across chiplets. Such latency variability results in a long tail latency during the execution of individual inference layers. As a result, the overall performance for each layer is restricted by the slowest chiplet in the system, limiting scalability. To address these challenges, we propose three tail-latency-aware, nonuniform tiling optimizations targeted at improving locality and minimizing inter-chiplet communication: (1) nonuniform work partitioning to balance

The original version of this paper appeared in the *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (Columbus, OH, USA, Oct. 2019).*

compute latency with communication latency; (2) communication-aware data placement to minimize inter-chiplet traffic; and (3) cross-layer pipelining to improve resource utilization.

2. BACKGROUND

Package-level MCM integration is a promising alternative for assembling large-scale systems out of small building blocks known as *chiplets*. Such systems consist of multiple chiplets connected together via on-package links using a silicon interposer or an organic substrate and employing efficient intra-package signaling circuits.[3, 24] Compared to a large monolithic die, MCMs can reduce (1) design costs, because logic design, verification, and physical design are all easier on a small chip than a large chip; and (2) fabrication costs, as the much lower manufacturing yield of large chips makes them far more expensive than small chips. In addition, different scales of systems can be created merely by adjusting the number of chiplets placed in a package, without requiring a different chip tapeout for each market segment. MCMs have been recently applied to a general-purpose CPU design[3] as an alternative to building multi-core CPUs on reticle-limited large die. They have also been an active research area for scaling of multi-CPU[16] and multi-GPU systems.[1] However, package-level wires do not provide the same communication density or energy/bit as on-chip wires. Consequently, MCM architects and software developers must still consider the nonuniform bandwidth, latency, and energy present in these systems to achieve an efficient application performance.

An MCM-based system has a heterogeneous interconnect architecture, as the available intra-chiplet bandwidth is expected to be significantly higher than available inter-chiplet bandwidth. In addition, sending data to remote chiplets incurs additional latency. This latency may include on-chip wire delays to move data to the edge of the chiplet, synchronizer delays for crossing clock domains, serialization and deserialization latency in high-speed communication links, and the on-package wire delays of inter-chiplet links. As a result, the communication latency between two elements in an MCM depends heavily on their spatial locality on the package.

Mapping DNN layers to a tile-based architecture is a well-studied research problem.[6, 19] The state-of-the-art DNN tiling typically assumes a flat architecture with uniform latency and bandwidth across processing elements (PEs) and focuses on data reuse for reducing global bandwidth demands. This assumption is acceptable for small-scale systems, as the communication latency variability is small and the computation is often tolerant of communication latencies. However, as the DNN inference performance is scaled-up to larger systems, the execution time decreases and latency-related effects become more important. Furthermore, in the large-scale systems with heterogeneous interconnect architectures such as MCMs, the assumptions of uniform latency and bandwidth in selecting DNN tiling can degrade the performance and energy efficiency. Simba is the first work that quantitatively highlights the challenge of mapping DNN layers to nonuniform, MCM-based DNN accelerators and proposes communication-aware tiling strategies to address the challenge.

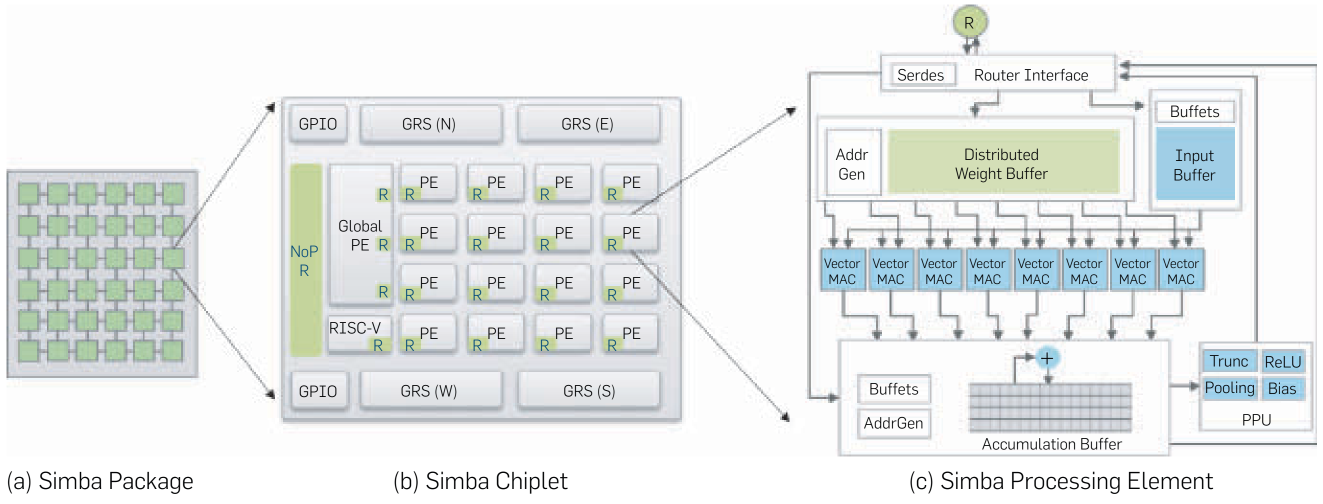
3. SIMBA ARCHITECTURE AND SYSTEM

To understand the challenges and opportunities of using MCMs for building large-scale, deep-learning systems, we designed, implemented, fabricated, and characterized Simba, the first chiplet-based deep-learning system. This section first presents an overview of the Simba architecture and its default uniform tiling strategy. We then describe Simba's silicon prototype and present a detailed characterization of the Simba system in Section 4.

3.1. Simba architecture

Tile-based architectures have been frequently proposed for deep-learning accelerator designs.[6, 5, 20] Our design target is an accelerator scalable to data center inference, where state-of-the-art data center accelerators deliver around 100 tera-operations-per-second (TOPS). For example, the first generation of the tensor processing unit (TPU) delivers 92 TOPS[15] and is designed for inference applications. One simple approach to achieve this design goal is to increase the number of tiles in a monolithic single chip. However,

Figure 1. Simba architecture from package to processing element (PE).



building a flat network with hundreds of tiles would lead to high tile-to-tile communication latency, as examined in both multi-core CPU[7] and accelerator[10] research.

Simba adopts a hierarchical interconnect to efficiently connect different processing elements (PEs). This hierarchical interconnect consists of a network-on-chip (NoC) that connects PEs on the same chiplet and a network-on-package (NoP) that connects chiplets together on the same package. Figure 1 illustrates the three-level hierarchy of the Simba architecture: package, chiplet, and PE. Figure 1(a) shows a Simba package consisting of a 6×6 array of Simba chiplets connected via a mesh interconnect. Each Simba chiplet, as shown in Figure 1(b), contains an array of PEs, a global PE, a NoP router, and a controller, all connected by a chiplet-level interconnect. To enable the design of a large-scale system, all communication between the PEs, Global PEs, and controller is designed to be latency-insensitive[4] and is sent across the interconnection network through the NoC/NoP routers.

Simba PE. Figure 1(c) shows the microarchitecture of the Simba PE, which includes a distributed weight buffer, an input buffer, parallel vector MAC units, an accumulation buffer, and a postprocessing unit. Each Simba PE is similar to a scaled-down version of NVDLA, a state-of-the-art DL accelerator product.[22] The heart of the Simba PE is an array of parallel vector multiply-and-add (MAC) units that are optimized for efficiency and flexibility. The Simba PE uses a weight-stationary dataflow[6]: weights remain in the vector MAC registers and are reused across iterations, although new inputs are read every cycle. Each vector MAC performs an 8:1 dot-product along the input channel dimension C to exploit an efficient spatial reduction. To provide flexible tiling options, the Simba PE also supports cross-PE reduction with configurable producers and consumers. If the current PE is the last PE on the reduction chain, it first sends partial sums to its local postprocessing unit that performs ReLU, truncation and scaling, pooling, and bias addition. The final output activation is sent to the target Global PE for computation of the next layer.

Simba global PE. The Global PE serves as a second-level storage for input/output activation data to be processed by the PEs. To support flexible partitioning of the computation, the Global PE can either unicast data to one PE or multicast to multiple PEs, even across chiplet boundaries. The Global PE has a multicast manager that oversees these producer-consumer relationships. The Global PE also serves as a platform for near-memory computation. Many DNNs feature some computation that has low data reuse, such as element-wise multiply/add or depth-wise convolution. The Global PE can perform such computations locally to reduce communication overhead for these types of operations.

Simba controller. Each Simba chiplet contains a RISC-V processor core[2] that is responsible for configuring and managing the chiplet's PEs and Global PE states via memory-mapped registers using an AXI-based communication protocol. After all states are configured, the RISC-V triggers the execution in the active PEs and Global PEs and waits for these blocks to send *done* notifications via interrupts. Synchronization of chiplet control processors across the package is implemented via memory-mapped interrupts.

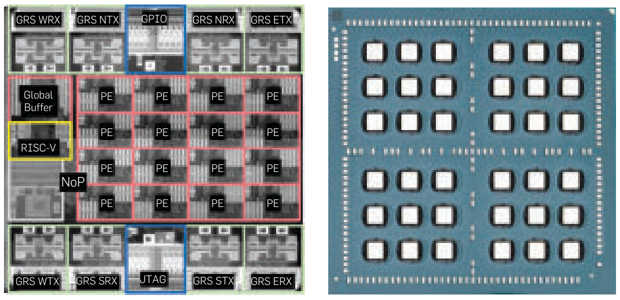
Simba interconnect. To efficiently execute different neural networks with diverse layer dimensions, Simba supports flexible communication patterns across the NoC and NoP. Both NoC and NoP use a mesh topology with a hybrid wormhole/cut-through flow control. Specifically, unicast packets use wormhole flow control for large packet size, whereas multicast packets are cut-through to avoid wormhole deadlocks. Each Simba PE can unicast to any local or remote PE for cross-PE partial-sum reduction, to any local or remote Global PE to transmit output activation values, and to any local or remote chiplet controller to signal execution completion. A PE does not need to send multicast packets as its computation requires only point-to-point communication. In addition to unicast communication, a Global PE can also send multicast packets to local and remote PEs for flexible data tiling.

3.2. Simba silicon prototype

We implemented, fabricated, and tested a silicon prototype of the Simba system, as shown in Figure 2, with the microarchitecture parameters listed in the original Simba paper.[21] We chose parameters so that a Simba chiplet has area and power similar to an efficient edge system, such as DianNao[5] or Eyeriss,[6] whereas a full Simba package is comparable to a data-center-scale system such as TPU.[15] Table 1 shows the synthesis area breakdown of key components in the Simba chiplet architecture.

```
//Package level
for p3 = [0: P3):
  for q3 = [0: Q3):
    parallel_for k3 = [0: K3):
      parallel_for c3 = [0: C3):
```

Figure 2. Simba silicon prototype.



(a) Simba chiplet (b) Simba package

Table 1. Area breakdown of the Simba system.

Partition	Component	Area (λm)²
PE	Vector MACs	12K
	Weight Buffer	41K
	Input Buffer	11K
	Accumulation Buffer	24K
	NoC Router	19K
Global PE	Distributed Buffer	125K
	NoC Routers	27K
RISC-V	Processor	109K
NoP	NoP Router	42K

```
// Chiplet level
for p2 = [0: P2):
  for q2 = [0: Q2):
    parallel_for k2 = [0: K2):
      parallel_for c2 = [0: C2):
// PE level
for r = [0: R):
  for s = [0: S):
    for k1 = [0: K1):
      for c1 = [0: C1):
        for p1 = [0: P1):
          for q1 = [0: Q1):
// Vector-MAC level
parallel_for k0 = [0: K0):
  parallel_for c0 = [0: C0):
   p = (p3 * P2 + p2) * P1 + p1;
   q = (q3 * Q2 + q2) * Q1 + q1;
   k = ((k3 * K2 + k2) * K1 + k1) * K0 + k0;
   c = ((c3 * C2 + c2) * C1 + c1) * C0 + c0;
  OA[p,q,k] += IA[p-1+r,q-1+s,c] * W[r,s,c,k];
```

Listing 1. Simba baseline dataflow.

As shown in Figure 2a, the 2.5 × 2.4 Simba chiplets were implemented in a TSMC 16 nm FinFET process technology.[25] Each Simba package (Figure 2b) contains an array 6×6 of chiplets connected on an organic package substrate using a ground-referenced signaling (GRS) technology for intra-package communication.[24] The top and bottom rows of each chiplet include eight chiplet-to-chiplet GRS transceiver macros. Four macros are configured as receivers and four as transmitters. Each transceiver macro has four data lanes and a clock lane with configurable speed from 11 Gbps/pin to 25 Gbps/pin, consuming 0.82–1.75 pJ/bit, with a total peak chiplet bandwidth of 100 GB/s. We chose GRS as our communication mechanism because it delivers 3.5× higher bandwidth per unit area and lower energy per bit compared to other MCM interconnects.[3]

The prototype chiplets were implemented using a globally asynchronous, locally synchronous (GALS) clocking methodology,[8] allowing independent clock rates for individual PEs, Global PEs, RISC-V processors, and NoP routers. Running in a single-chiplet configuration, Simba prototypes have been measured to operate correctly in the lab at a minimum voltage of 0.42 V with a 161 MHz PE frequency, achieving 0.11 pJ/Op (9.1 TOPS/W) core power efficiency on a peak-utilization convolution micro-benchmark. At 1.2 V, each chiplet operates with a 2 GHz PE frequency for a peak throughput of 4 TOPS. The 36-chiplet Simba system is functional over a slightly narrower voltage range, from 0.52 to 1.1 V, achieving 0.16 pJ/op at 0.52 V and 484 MHz; at 1.1 V, the 36-chiplet system achieves a 1.8 GHz PE frequency and 128 TOPS.

3.3. Simba baseline tiling

To map the DNN layers onto the hierarchical tile-based architecture, we first use a state-of-the-art DNN tiling strategy that uniformly partitions weights spatially, leveraging model parallelism.[22, 15, 5, 20] Listing 1 shows the default tiling in a loop-nest form. Each dimension of a DNN layer can be tiled temporally, spatially, or both at each level of the system hierarchy: package, chiplet, PE, and vector MAC. The loop bounds and orderings in Listing 1 are configurable in Simba so that users can flexibly map computation to the Simba system. In particular, the default dataflow uniformly partitions weights along the input channel (C) and the output channel (K) dimensions, as noted in the `parallel_for` loops. In addition, Simba can also uniformly partition along the height (P) and width (Q) dimensions of an output activation across chiplets and PEs to support flexible tiling. Section 5 highlights the limitations of this approach when mapping networks onto a large-scale, nonuniform network access architecture with an MCM-based integration.

We developed a flow that uses Caffe[14] to map a DNN inference application to the Simba system, which primarily determines an efficient tiling strategy for the dataflow that best exploits data reuse in the memory hierarchy. To facilitate the evaluation of different mapping alternatives, we also developed a fast, analytical energy model for Simba that quantifies the energy cost of a particular mapping, similar to the methodology discussed in prior work.[6, 19] The compilation process starts with a *mapper* that is provided with data regarding available system resources (such as the number of PEs, the number of Global PEs, and the sizes of buffers in the system) and the parameters of a given layer from the Caffe specification. The mapper determines which PE will run each portion of the loop nest and in which buffers the activations and weights are stored. As this mapping is a logical one, the mapper is followed by a *binder* which decides in which physical resource in the Simba topology the loop nests and data structures are placed. We use a random search algorithm to sample the mapping space and use the energy and performance models to select good mappings and placements. Finally, the flow generates the configuration binaries for each chiplet that implement the execution created by the mapper and binder.

4. SIMBA CHARACTERIZATION

This section details the performance characterization of Simba, focusing on achieved scalability using the uniform-tiling baseline. All evaluation results are measured using the prototype system.

4.1. Methodology

Figure 3 shows the experimental setup for measuring the performance and power of the Simba prototype system. The silicon prototype test board is attached to an x86 host through PCI-E using a Xilinx FPGA. To measure the performance of the Simba prototype system, we use software running on the RISC-V to query cycle counters built into the RISC-V microcontrollers. Power and performance measurements begin after the weights have been loaded into each PE's weight buffer and the inputs have been loaded into the Global PE buffers. Unless otherwise noted, the chiplets operate at a core voltage of 0.72 V, a PE frequency of 1.03 GHz, and GRS bandwidth of 11 Gbps. We use sense resistors on the board power supplies and a digital acquisition module to measure energy during experiment execution. As the chiplets support independent clock frequencies for different units (PEs, Global PEs, RISC-V, and NoP routers), we can vary these frequencies to change the compute-to-bandwidth ratios for our experiments. The NoC and NoP routing tables use dimension-ordered X-Y routing for all inter-chiplet communication. Although all 36 Simba chiplets are functional, our evaluation uses 32 chiplets, as it is easier to partition computation by powers of two because the number of input channels (C) and output channels (K) are typically powers of two.

We focus our application measurements on ResNet-50,[12] a state-of-the-art, representative deep-learning network, and evaluate its layers running on the Simba system with a batch size of one, as low-latency inference is a highly critical deployment scenario for data center inferencing.[15] We compile and run each layer independently, except when we map multiple layers to different physical partitions of Simba and execute them in a pipelined manner. Networks are pretrained and quantized to 8-bit using TensorRT without accuracy loss. We believe that the diversity of layers in ResNet-50 provides a sufficient breadth to cover a wide range of behaviors across different convolutional networks.

4.2. Performance/energy overview

Figure 4 summarizes the performance and energy measurements across all ResNet-50 layers. Each point represents a unique mapping for that layer, whereas different colors show

Figure 3. Bench measurement setup for the Simba prototype.

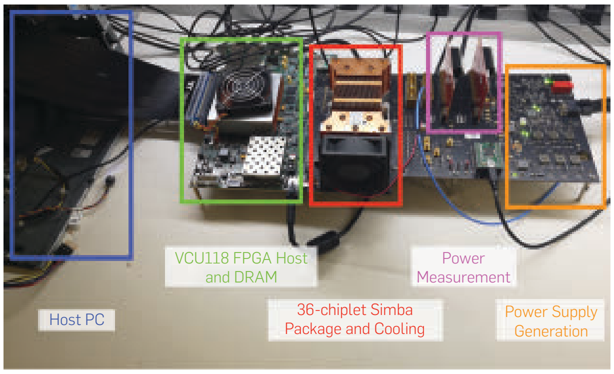
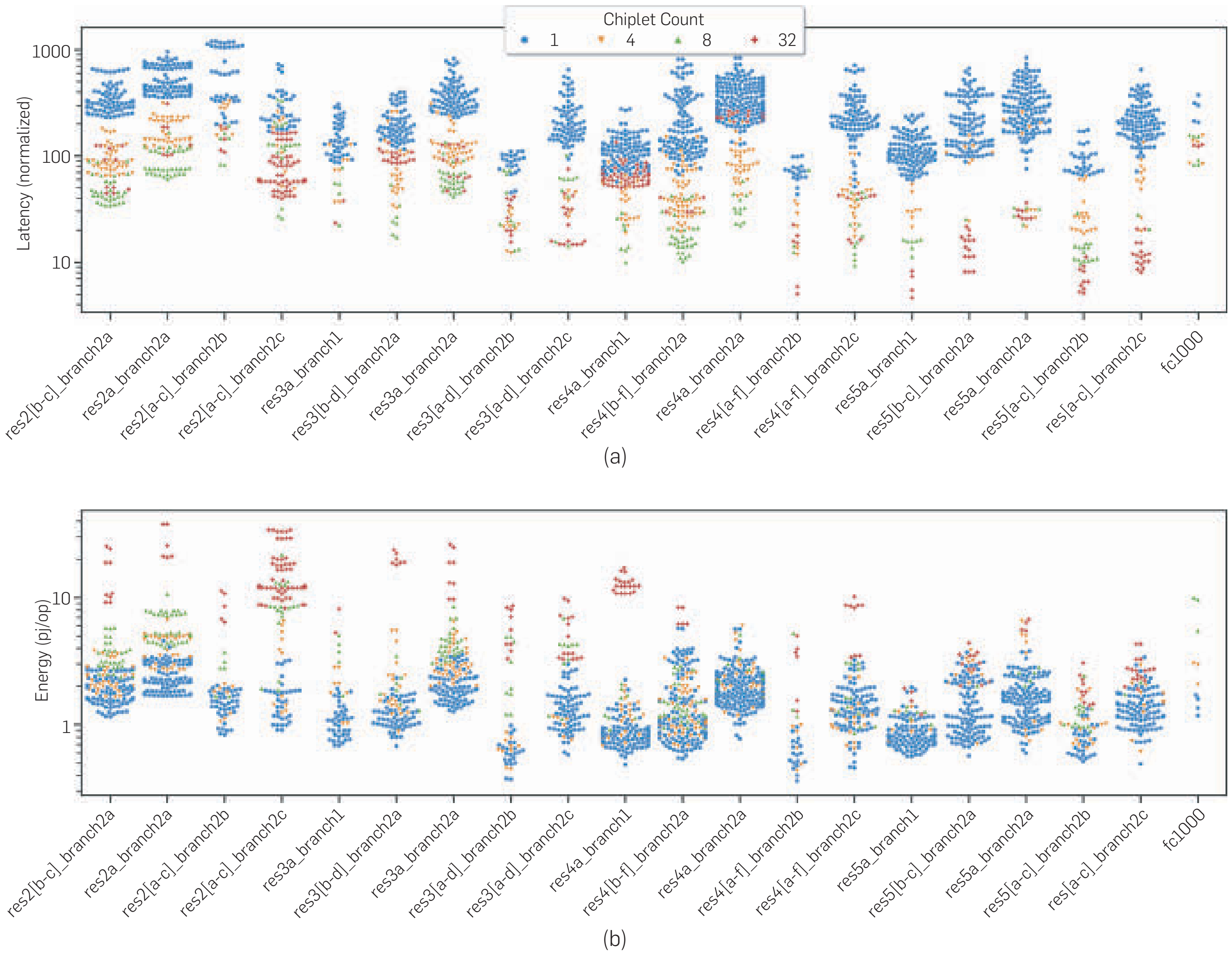

VCU118 FPGA Host and DRAM
Power Measurement
Host PC
36-chiplet Simba Package and Cooling
Power Supply Generation

Figure 4. Measured performance and energy and running ResNet-50 on the fabricated Simba prototype. Each point is a valid workload mapping onto the system; each column cluster shows the different performance and energy achieved by different mappings of the same workload. Each symbol shape represents a different number of active chiplets used for the mapping.



the number of chiplets active for that mapping. Latency is normalized to a hypothetical best-achievable latency that would be realized if each of the 576 PEs of the system operated with 100% utilization and no communication or synchronization overheads. Simba provides a large number of mapping options with drastically different performance and energy profiles, highlighting the importance of strategies for efficiently mapping DNNs to hardware. The figure also demonstrates the highly variable behavior of different layers. For example, the most energy-efficient configurations of layer `res3[b-d]_branch2b` achieve almost an order of magnitude better efficiency than those of `res3a_branch2a`. The degree of data reuse highly influences the efficiency; layers with high reuse factors, for example, 3×3 the convolution in `3[b-d]_branch2b`, tend to perform computation more efficiently than layers that require more data movement. Finally, although increasing the number of chiplets used in the system improves performance, it also leads to increased energy cost for chiplet-to-chiplet communication and synchronization. Efficiency can drop by nearly an order of magnitude for some layers, which further emphasizes the effect of data movement on overall efficiency. To better understand the system-level trade-offs, the remainder of this section characterizes the sensitivity of Simba to mapping alternatives, layer parameters, bandwidth, latency, and weak scaling, and includes a comparison to modern GPUs.

4.3. Layer sensitivity

Figure 5 shows the performance scalability of running three different layers in ResNet-50 across different numbers of chiplets. Although the performance of `res2[a-c]_branch2b` initially improves with increased chiplet count, the performance gains cease beyond eight chiplets. As one of the early layers of the network, the number of weights in this layer is so small that it cannot fully utilize the compute throughput of Simba. The performance degrades with 32 chiplets because the inter-PE communication costs overwhelm the limited parallelism. By contrast, the performance of the `res2a_branch1` layer improves when increasing the number of chiplets from 1 to 8, though it stops improving from 8 to 32 chiplets. This layer has more compute parallelism than `res2[a-c]_branch2b`, but it still does not have enough to fully overcome the overheads of inter-chiplet communication.

The `res5[a-c]_branch2b` layer demonstrates the best performance scaling, with improvements observed up to 32 chiplets. However, the performance scaling slows down significantly past eight chiplets due to communication overheads. Of the 53 layers of ResNet-50, 12 follow the behavior of `res3a_branch1`, 24 follow that of `res5[a-c]_branch2b`, and the remaining 17 have behavior similar to `res2[a-c]_branch2b`. These measurements demonstrate that the amount of compute parallelism that an MCM can leverage varies from layer to layer, and that the cost of communication can hinder the ability to exploit that parallelism, even on a single chiplet.

4.4. NoP bandwidth sensitivity

Figure 6 shows how execution time is affected by NoP bandwidth for two representative ResNet layers when mapped to 32 chiplets. We adjusted the bandwidth of the NoP relative to the intra-chiplet compute performance to measure the network bandwidth sensitivity. For `res3[a-d]_branch2b`, the increased bandwidth between chiplets results in only a 5% decrease in execution time, indicating that this layer is not bound by the NoP bandwidth or inter-chiplet communication latency. However, for `res3a_branch1`, the increased

Figure 5. Simba scalability across different layers from ResNet-50. Latency is normalized to the latency of the best-performing tiling with one chiplet.

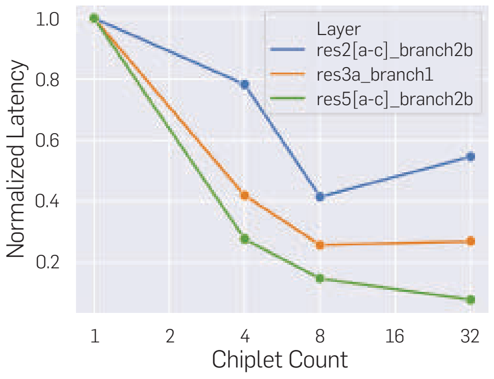


Figure 6. Simba scalability with different chiplet-to-chiplet communication bandwidths.

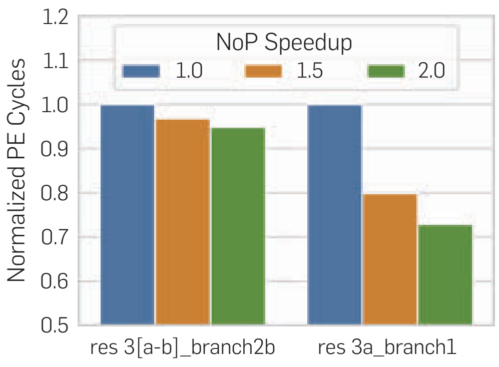
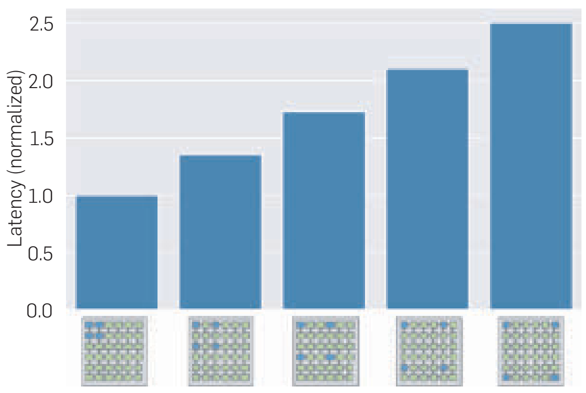


Figure 7. Simba scalability with different chiplet-to-chiplet communication latencies running `re+s4a_branch1` with four chiplets (using the same tiling). Different bars represent different selections of the active four chiplets, as shown under the *X*-axis; the active chiplets are highlighted in blue.



bandwidth decreases execution time by 27%, indicating that this layer is bottlenecked by communication between chiplets. Because an MCM-based system intrinsically has a nonuniform network architecture between intra-chiplet and inter-chiplet PEs, mapping policies must consider the different latency and bandwidth parameters to deliver good performance and efficiency.

4.5. NoP latency sensitivity

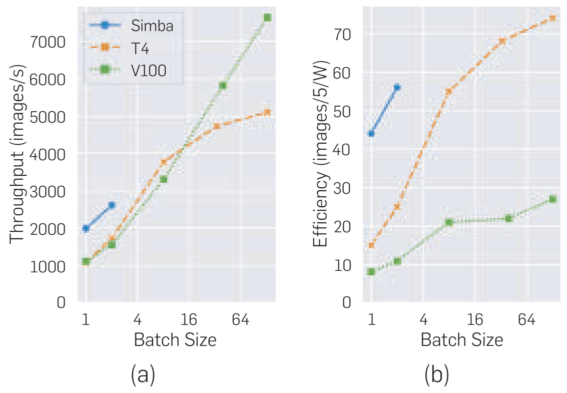
In addition to lower bandwidth, the NoP has higher latency than the NoC due to inter-chiplet signaling overheads. To isolate the effect of NoP latency, we ran experiments mapping layers to four chiplets, but adjusted the locations of the selected chiplets in the package to modulate latency. Figure 7 shows the effect of increasing the longest inter-chiplet latency from 2 hops to 12 hops for `res4a_branch1`. The figure shows the execution time normalized to a configuration of adjacent chiplets, with the chiplet selection shown under each bar. With active chiplets further apart, the overall execution time increases by up to 2.5× compared to execution on adjacent chiplets. Communication latency is typically less pronounced for small-scale systems but plays a significant role in achieving good performance and energy efficiency for a large-scale, MCM-based system like Simba.

4.6. Comparisons with GPUs

Figure 8 compares Simba to NVIDIA's V100 and T4 GPUs. We run ResNet-50 with different batch sizes and compare to the GPU results that are published.[18] Due to Simba's limited on-package storage capacity for input activations, we only run Simba at batch size one and two. Unlike GPUs, Simba is designed for low-latency inference with a small batch size, which motivates the use of distributed and persistent weight storage to reduce data movement. The Simba package, such as the MCM interface, has substantially a smaller total silicon area (216) than T4 (525) or V100 (815), due to differences in math precision, on-chip storage, DRAM interface, and types of computation supported in these architectures.

Figure 8a shows the throughput of Simba, V100, and T4 running ResNet-50. Simba delivers 1.8× and 1.9× better throughput at batch size one compared to V100 and T4, respectively. Figure 8b illustrates the corresponding energy efficiency

Figure 8. Throughput and efficiency of Simba, V100, and T4 running ResNet-50 with different batch sizes.



improvement of Simba compared to V100 (5.4×) and T4 (2.9×). When running ResNet-50 with a larger batch size, instead of exploiting the batch-level parallelism like GPUs, Simba would run each batch sequentially. As a result, we expect that the throughput of Simba is close to that of running with a batch size of one.

5. SIMBA NONUNIFORM TILING

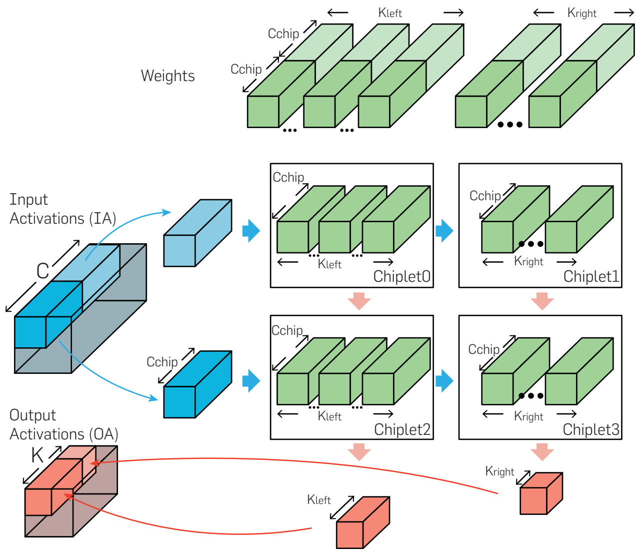
This section presents the details of two novel DNN workload tiling techniques that target the nonuniform latency and bandwidth presented by large-scale MCM-based systems, nonuniform work partitioning and communication-aware data placement. Our results indicate the importance of communication-latency-aware tiling when mapping DNN workloads to large-scale, hierarchical systems.

5.1. Nonuniform work partitioning

Efficient use of parallel systems requires proper load balancing among the components in the system. Failure to properly balance the load on the system leads to higher latency and energy caused by resources waiting for the slowest unit to complete, that is, increased tail latency. The total execution time can be subdivided into two major components: communication latency and compute latency. The state-of-the-art DNN tiling strategies typically assign the same amount of the work to each of the available resources.[23] However, this approach breaks down for large-scale systems, especially when the PEs are spatially distributed with different communication latencies among them.

To address this limitation, we propose a nonuniform work partitioning strategy that considers communication latencies. Instead of uniformly assigning the same amount of work to each PE, we nonuniformly partition the work across the PEs. PEs closer to the data producers will perform more work to

Figure 9. Illustration of communication-aware, nonuniform work partitioning. The top green tensors represent weights (W), the left blue tensors represent input activations (IA), and the bottom red tensors represent the output activation (OA). In this example, IA is stored in Chiplet0 and Chiplet2.



maximize physical data locality, whereas PEs that are further away will do less work to decrease the tail latency effects. Figure 9 illustrates an example of nonuniform work partitioning using a 4-chiplet system. In this example, we assume that the input activation (IA) is physically stored in the Global PEs of `Chiplet0` and `Chiplet2`, whereas the weights (W) and the work are partitioned across all the four chiplets. During execution, the Global PE of `Chiplet0` will multicast a slice of IA to PEs in both `Chiplet0` and `Chiplet1`. Because the communication latencies from the `Chiplet0` Global PE to the PEs in `Chiplet0` and `Chiplet1` are different, `Chiplet1` will fall behind. To prevent the longer communication to `Chiplet1` from increasing the tail latency of the execution, we can adjust the amount of computation that each chiplet is assigned in a manner inversely proportional to its communication distance from the source. In the example as shown in Figure 9, `Chiplet0` and `Chiplet2` are provided with larger chunks of work (K_{left}), whereas `Chiplet1` and `Chiplet3` get the smaller chunks (K_{right}). This work schedule evens out the completion time across the chiplets, thereby improving overall system performance. For simplicity, this example only shows nonuniform partitioning with respect to input activations. A similar technique can be used to mitigate the communication latency for output activations to the destination chiplets by using nonuniform partitioning along the C dimension.

The variation in communication latency is quite pronounced in large-scale systems such as Simba, with hundreds of spatially distributed PEs. To dynamically adjust the amount of work that each PE performs, we use the performance counters within each PE to collect accurate latency and utilization information during the initial execution of a layer. We then adjust the work distribution for the subsequent executions of each layer based on the latency variation across PEs.

Figure 10 illustrates the measured performance improvement using nonuniform work partitioning. For each of the layers, we pick the highest performance uniform tiling from Figure 5 as the baseline. We then measure the execution time of different chiplets and identify layers with a large tail latency. For these layers, we use nonuniform work partitioning to shift the computation from the tail PEs to the PEs that are closer to the data. Depending on layer dimensions, we achieve up to 15% performance improvement compared to the best uniform tiling for a given layer. The achievable performance improvement is highly sensitive to the compute-and-communication ratio in a given mapping. For example, when either compute or communication is significantly dominating the overall execution latency, as in the case of `res5a_branch1`, incrementally modulating the amount of work each PE performs provides little performance improvement. However, when compute and communication latencies are more comparable, which are typically desired to achieve good mapping, the performance improvement is more pronounced, as in the case of `res2[b-c]_branch2c`.

5.2. Communication-aware data placement

The communication latency in a parallel system can have a large effect on the overall system performance, as observed in multi-core and multi-GPU characterizations.[17] Due to the limited scale of today's deep-learning accelerators, most have one unified global buffer that supplies data to all of the PEs.[6, 5, 22] However, in large-scale MCM systems where on-chip buffers are spatially distributed among the chiplets, the communication latency becomes highly sensitive to the physical location of data. Figure 11 illustrates how data placement affects communication distances and latencies. For example, if the `Src` chiplet in Figure 11(a) broadcasts data to the other chiplets, the arrival time of the data will vary greatly depending the distance of the receiving chiplets from the `Src`. Depending on the amount of computation each

Figure 10. Nonuniform work partition for ResNet-50 with speedup normalized to the best-performing tiling.

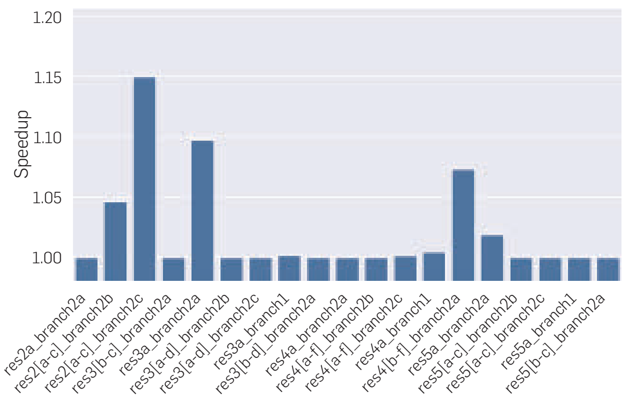
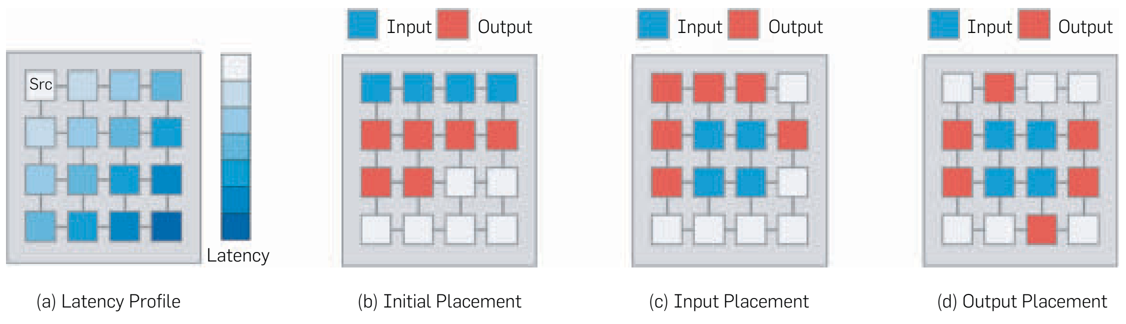


Figure 11. Data placement on the Simba system. (a) Assessment of the relative latency to different chiplets that receive data from Src. (b) Default input activation (IA) and output activation (OA) placement where data is sequentially placed from the Global PE of the first chiplet. (c) An improved IA placement at the center of the package so that data can be multicast to all chiplets. (d) OA placement with even distribution along the periphery of the package to minimize OA communication latency.



chiplet performs, such variations in communication distance could significantly limit the achievable speedup in a distributed, tile-based system like Simba, motivating the need for data placement optimizations.

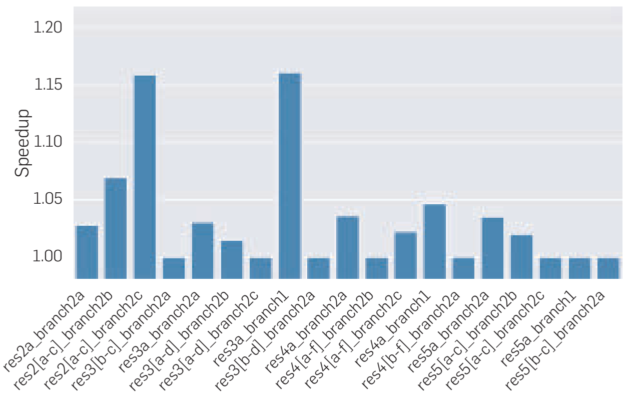
Although optimal data placement is an NP-hard problem, we use a practical greedy algorithm to iteratively determine where input and output activation data should be placed in the Simba system. The algorithm starts by performing a placement of the input activation data blocks. Once the input activations are placed, the same greedy algorithm is executed for tiles of output activations. Because a previous stage of the mapping process has already determined the data tiling, this stage need only focus on data placement and not re-tiling. Figure 11(b) shows a naive data placement with a sequential allocation of input activations to the chiplets on the top row and output activations in the next six chiplets. Figure 11(c) shows a better assignment of IAs to chiplets, selecting the four in the middle that minimize aggregate multicast hop-count to all chiplets. Finally, Figure 11(d) shows a placement of output activations on chiplets in regions where OA accumulation can occur.

Figure 12 shows the performance improvement of ResNet-50 layers with optimized data placement. Although all of the layers use 32 chiplets, many of them have different communication patterns. For example, layers like `res2[a-c]_branch2c` communicate frequently within a group of eight chiplets. In this case, it is better to group those chiplets together to minimize communication cost. In contrast, layer `res4a_branch1` must broadcast from a single chiplet to all 32 chiplets. In this case, instead of placing the source chiplet sequentially at the upper left corner, placing it at the center of the package leads to a 5% performance improvement. Data placement optimization results in up to 15% improved performance compared to the best achieved baseline.

6. CONCLUSION

This work presented Simba, a scalable MCM-based deep-learning inference accelerator architecture. Simba is a heterogeneous tile-based architecture with a hierarchical interconnect. We developed a silicon prototype system consisting of 36 chiplets that achieves up to 128 TOPS at high energy efficiency. We used the prototype to characterize the overheads of the nonuniform network of an MCM-based architecture, observing that load imbalance and communication latencies contribute to noticeable tail-latency effects. We then showed how considering the nonuniform nature of system can help improve performance through techniques such as nonuniform work partitioning, communication-aware data placement, and cross-layer pipelining. Applying these optimizations results in performance increases of up to 16% compared to naive mappings.

Figure 12. Data placement for ResNet-50 layers with speedup normalized to the best performing tiling.



Acknowledgments

The authors would like to thank Frans Sijstermans, Dan Smith, Don Templeton, Guy Peled, Jim Dobbins, Ben Boudaoud, Randall Laperriere, Borhan Moghadam, Sunil Sudhakaran, Zuhair Bokharey, Sankara Rajapandian, James Chen, John Hu, and Vighnesh Iyer for package, PCB, signal integrity, fabrication, and prototyping support. This research was, in part, funded by the U.S. government under the DARPA CRAFT program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

References

1. Arunkumar, A., Bolotin, E., Cho, B., Milic, U., Ebrahimi, E., Villa, O., Jaleel, A., Wu, C.-J., Nellans, D. MCM-GPU: Multi-chip-module GPUs for continued performance scalability. In *Proceedings of the International Symposium on Computer Architecture (ISCA)* (Toronto, ON, Canada, 2017), Association for Computing Machinery, New York, NY, USA.
2. Asanovic, K., Avizienis, R., Bachrach, J., Beamer, S., Biancolin, D., Celio, C., Cook, H., Dabbelt, D., Hauser, J., Izraelevitz, A., Karandikar, S., Keller, B., Kim, D., Koenig, J., Lee, Y., Love, E., Maas, M., Magyar, A., Mao, H., Moreto, M., Ou, A., Patterson, D.A., Richards, B., Schmidt, C., Twigg, S., Vo, H., Waterman, A. *The Rocket Chip Generator*. Technical Report, EECS Department, University of California, Berkeley, 2016.
3. Beck, N., White, S., Paraschou, M., Naffziger, S. Zeppelin: An SoC for multichip architectures. In *Proceedings of the International Solid State Circuits Conference (ISSCC)* (2018), IEEE, San Francisco, CA, USA.
4. Carloni, L.P., McMillan, K.L., Saldanha, A., Sangiovanni-Vincentelli, A.L. A methodology for correct-by-construction latency insensitive design. In *Design Automation Conference* (1999), IEEE, San Jose, CA, USA.
5. Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., Temam, O. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operation Systems (ASPLOS)* (Salt Lake City, Utah, USA, 2014), Association for Computing Machinery, New York, NY, USA.
6. Chen, Y.-H., Emer, J., Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Proceedings of the International Symposium on Computer Architecture (ISCA)* (2016), IEEE, Seoul, South Korea.
7. Das, R., Eachempati, S., Mishra, A.K., Narayanan, V., Das, C.R. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)* (2009), IEEE, Raleigh, NC, USA.
8. Fojtik, M., Keller, B., Klinefelter, A., Pinckney, N., Tell, S.G., Zimmer, B., Raja, T., Zhou, K., Dally, W.J., Khailany, B. A fine-grained GALS SoC with pausible adaptive clocking in 16nm FinFET. In *International Symposium on Asynchronous Circuits and Systems (ASYNC)* (2019), IEEE, Hirosaki, Japan.
9. Fowers, J., Ovtcharov, K., Papamichael, M., Massengill, T., Liu, M., Lo, D., Alkalay, S., Haselman, M., Adams, L., Ghandi, M., Heil, S., Patel, P., Sapek, A., Weisz, G., Woods, L., Lanka, S., Reinhardt, S., Caulfield, A., Chung, E., Burger, D. A configurable cloud-scale DNN processor for real-time AI. In *Proceedings of the International Symposium on Computer Architecture (ISCA)* (2018), IEEE, Los Angeles, CA, USA.
10. Gao, M., Yang, X., Pu, J., Horowitz, M., Kozyrakis, C. Tangram: Optimized coarse-grained dataflow for scalable NN accelerators. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operation Systems (ASPLOS)* (2019), Association for Computing Machinery, New York, NY, USA.
11. Greenhill, D., Ho, R., Lewis, D., Schmit, H., Chan, K.H., Tong, A., Atsatt, S., How, D., McElheny, P., Duwel, K., Schulz, J., Faulkner, D., Iyer, G., Chen, G., Phoon, H.K., Lim, H.W., Koay, W.-Y., Garibay, T. A 14nm 1GHz FPGA with 2.5D transceiver integration. In *Proceedings of the International Solid State Circuits Conference (ISSCC)* (2017), IEEE, San Francisco, CA, USA.
12. He, K., Zhang, X., Ren, S., Sun, J. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), IEEE, Las Vegas, NV, USA.
13. Iyer, S.S. Heterogeneous integration

for performance and scaling. *IEEE Transactions on Components, Packaging and Manufacturing Technology* (2016), IEEE.

14. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R.B., Guadarrama, S., Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia* (2014), Association for Computing Machinery, New York, NY, USA.
15. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., luc Cantin. P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T.V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C.R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., Yoon, D.H. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the International Symposium on Computer Architecture (ISCA)* (Toronto, ON, Canada, 2017), Association for Computing Machinery, New York, NY, USA.
16. Loh, G.H., Jerger, N.E., Kannan, A., Eckert, Y. Interconnect-memory challenges for multi-chip, silicon interposer systems. In *Proceedings of the International Symposium on Memory System (MEMSYS)* (Washington DC, USA, 2015), Association for Computing Machinery, New York, NY, USA.
17. Mirhoseini, A., Pham, H., Le, Q.V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., Norouzi, M., Bengio, S., Dean, J. Device placement optimization with reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)* (2017), JMLR.org, Sydney, NSW, Australia.
18. NVIDIA. NVIDIA Tesla deep learning product performance. https://developer.nvidia.com/deep-learning-performance-training-inference, 2019.
19. Parashar, A., Raina, P., Shao, Y.S., Chen, Y.-H., Ying, V.A., Mukkara, A., Venkatesan, R., Khailany, B., Keckler, S.W., Emer, J. Timeloop: A systematic approach to DNN accelerator evaluation. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)* (2019), IEEE, Madison, WI, USA.
20. Parashar, A., Rhu, M., Mukkara, A., Puglielli, A., Venkatesan, R., Khailany, B., Emer, J., Keckler, S.W., Dally, W.J. SCNN: An accelerator for compressed-sparse convolutional neural networks. In *Proceedings of the International Symposium on Computer Architecture (ISCA)* (Toronto, ON, Canada, 2017), Association for Computing Machinery, New York, NY, USA.
21. Shao, Y.S., Clemons, J., Venkatesan, R., Zimmer, B., Fojtik, M., Jiang, N., Keller, B., Klinefelter, A., Pinckney, N., Raina, P., Tell, S.G., Zhang, Y., Dally, W.J., Emer, J.S., Gray, C.T., Keckler, S.W., Khailany, B. Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In *Proceedings of the International Symposium on Microarchitecture (MICRO)* (Columbus, OH, USA, 2019), Association for Computing Machinery, New York, NY, USA.
22. Sijstermans, F. The NVIDIA deep learning accelerator. In *Hot Chips* (2018).
23. Venkataramani, S., Ranjan, A., Banerjee, S., Das, D., Avancha, S., Jagannathan, A., Durg, A., Nagaraj, D., Kaul, B., Dubey, P., Raghunathan, A. ScaleDeep: A scalable compute architecture for learning and evaluating deep networks. In *Proceedings of the International Symposium on Computer Architecture (ISCA)* (Toronto, ON, Canada, 2017), Association for Computing Machinery, New York, NY, USA.
24. Wilson, J.M., Turner, W.J., Poulton, J.W., Zimmer, B., Chen, X., Kudva, S.S., Song, S., Tell, S.G., Nedovic, N., Zhao, W., Sudhakaran, S.R., Gray, C.T., Dally, W.J. A 1.17pJ/b 25Gb/s/pin ground-referenced single-ended serial link for off- and on-package communication in 16nm CMOS using a process- and temperature-adaptive voltage regulator. In *Proceedings of the International Solid State Circuits Conference (ISSCC)* (2018), IEEE, San Francisco, CA, USA.
25. Zimmer, B., Venkatesan, R., Shao, Y.S., Clemons, J., Fojtik, M., Jiang, N., Keller, B., Klinefelter, A., Pinckney, N., Raina, P., Tell, S.G., Zhang, Y., Dally, W.J., Emer J.S., Gray, C.T., Keckler, S.W., Khailany, B. A 0.11 pJ/Op, 0.32–128 TOPS, scalable multi-chip-module-based deep neural network accelerator with ground-reference signaling in 16nm. In *Proceedings of the International Symposia on VLSI Technology and Circuits (VLSI)* (2019), IEEE, Kyoto, Japan, Japan.

Yakun Sophia Shao* ([ysshao]@berkeley.edu), UC Berkeley, CA, USA.

Jason Cemons, Nathaniel Pinckney, Brucek Khailany, and Stephen W. Keckler ([jclemons, npinckney, bkhailany, skeckler]@nvidia.com), NVIDIA, Austin, TX, USA.

Rangharajan Venkatesan, Brian Zimmer, Ben Keller, and Yanqing Zhang ([rangharajanv, bzimmer, benk, yanqingz]@nvidia.com), NVIDIA, Santa Clara, CA, USA.

Matthew Fojtik, Alicia Klinefelter, Stephen G. Tell, and Tom Gray ([mfojtik, aklinefelter, stell, tgray]@nvidia.com), NVIDIA, Durham, NC, USA.

Nan Jiang ([tedj]@nvidia.com), NVIDIA, Westford, MA, USA.

Priyanka Raina ([praina]@stanford.edu), Stanford University, Stanford, CA, USA.

William J. Dally ([bdally]@nvidia.com), NVIDIA, Incline Village, NV, USA/Stanford University, Stanford, CA, USA.

Joel Emer ([jemer]@nvidia.com), Massachusetts Institute of Technology, Cambridge, MA, USA/NVIDIA, Westford, TX, USA.

*Work done at NVIDIA

[CONTINUED FROM P. 120] a week. And at the end of the week, we didn't see a flaw in one another's arguments. So we proved what was, as Jeff mentioned, an inconsequential theorem, but on the other hand, it was a fun proof and it fit into the kind of language theory that was being done at the time.

Eventually, you drew on your work in language theory to make groundbreaking contributions to the field of compiler design.

AHO: At the beginning, we launched a systematic study of the theory of parsing translation and compiling—what models there were, what algorithms, and what needed to be extended. This resulted in a two-volume book that we wrote in the early 1970s, *The Theory of Parsing Translation and Compiling*. Then, as we learned more about compilers and compiling, and as we worked with some of our colleagues to develop components for creating lexical analysis tools, like Lex, and syntax analysis tools, like yacc, we codified what we knew into another book, called *Principles of Compiler Design*. And Jeff had this brilliant idea of putting a dragon on the front cover with the complexity of compiler design tattooed on its chest.

This is the text that eventually became known as the Dragon Book. I was going to ask whose idea that was.

AHO: The book became widely adopted, and the field grew rapidly. We had to write subsequent books to try to keep up. The color of the dragon changed, and we added a coauthor, Ravi Sethi, and then another, Monica Lam. But in the period of 1977 to 2007, we wrote this collection of books and did research that covered the sweep of programming language translators, and in the process created tools that our colleagues could use to create domain-specific languages and harness theory without having to become theoreticians. Don Knuth once said that the best theory is motivated by practice and the best practice by theory. That fits our work quite nicely.

Jeff, you've also thought a lot about the relationship between theory and practice.

ULLMAN: Different fields have different ways of looking at it. Parsing yields amazingly well to theory. At the time of the first Fortran compiler, it took several person-years to write a parser. By the time yacc came around, you could do it in an afternoon. So the compiler programming languages community has always had a positive view of theory. In the database field, where I spent my subsequent career, it was initially quite a struggle to get researchers to accept the idea that theory has some impact. That's since changed.

What about the relationship between industry and academia?

AHO: I was always interested in teaching. There was a Turing laureate at Bell Labs by the name of Richard Hamming, and he had an opinion on everything. In my first week, he told me that if you want to be a great scientist, you have to do more than great work; you have to teach others how to use your work. I found this combination of doing research, writing books, and teaching a very fruitful combination.

ULLMAN: My original plan was to become an academic, but in those days, assistant professors were the scum of the Earth—all of the hard work and none of the power or opportunities. It's very different today. Back then, I figured I could go work in industrial research for a while first, and it certainly paid off.

Speaking of academia, Jeff, I wonder what you make of the future of traditional "live" education, which you've been thinking about since at least 2001, when you proposed the idea of creating a company that acts as an ASP (application service provider) for computer science education, providing centralized lectures supported by local in-person instruction and a 24/7 helpline.

ULLMAN: I should have been more aggressive about building a company like Coursera around that idea. But the thing that I got hung up on, and that has never really come to fruition, is this idea of a universal TA system. If you're having problems with your PC, you can call tech support. Sometimes it's useful help, sometimes not, but there's a system for fixing problems. I never understood why it wasn't feasible to do the same thing for assistance with homework. But it's hard to have a global system where students from 1,000 different courses, even on the same subject, can ask about their homework, unless instructors are aligned in what questions they ask.

Your online course on automata got great reviews.

ULLMAN: The online courses are helpful. People are still using them. Maybe one day, these course videos will replace textbooks, because there's no question that the demand for texts and courses requiring texts is just gone.

AHO: On the subject of education, I feel that what you teach is perhaps more important than how you teach it. I'm a big fan of finding the appropriate abstractions for talking about a problem area and then finding solutions in terms of those abstractions. It's wonderful to be in an area like computer science because, as we expand its reach, we encounter problems for which we don't even have the appropriate abstractions to be able to think about them.

When you imagine the future of the field, what areas do you think hold the most promise?

AHO: That is a great question. Particularly with fields like AI, we're starting to replace people who do routine cognitive jobs with computer programs. What will the job market of the future be with this increasing capacity and power of computing? And as a consequence, what should we be educating students in so they'll be employable in the future? One of the feelings that I have is that the basics of the field don't go out of style nearly as fast as the details of the technology.

ULLMAN: As technology improves, we are able to deal with larger and larger datasets. So far, that's opened up more and more opportunities for solving problems. There are problems, like driverless cars, that yield to an abundance of data that just don't yield to small amounts. Throughout my life, we've never hit a limit, so when we can deal with 10 times more data than is currently possible, there will probably be other things we can do that I can't even imagine.

Leah Hoffmann is a technology writer based in Piermont, NY, USA.

DOI:10.1145/3460442

Leah Hoffmann

Q&A
Shaping the Foundations of Programming Languages

ACM A.M. Turing Award recipients Alfred Aho and Jeffrey Ullman discuss their early work, the 'Dragon Book,' and the future of 'live' computer science education.

ACM TURING AWARD recipients **ALFRED AHO** and **JEFFREY ULLMAN** met serendipitously, in the registration line for Princeton University's Ph.D. program. After graduate school, both joined the newly established Computing Science Research Center at Bell Laboratories, and their friendship turned into a productive collaboration that shaped the foundations of programming language theory and implementation. Here, they talk to us about languages, compilers, and the future of CS education.

Elements of the 1960s Unix Room have been preserved at Nokia Bell Labs. Ullman and Aho sit on the couch from Dennis Ritchie's office where many discussions on what would become the Unix operating system were rooted. Ritchie, with Ken Thompson, received the 1983 ACM A.M. Turing Award.

PHOTO BY ALEXANDER BERG

You joined Bell Labs in 1967, during what many consider its heyday. What was it like?

AHO: Bell Labs was a researcher's nirvana. When my boss hired me, he said, "Why don't you work on what you think is important?" And we were surrounded by all these brilliant colleagues—people like Ken Thompson and Dennis Ritchie. People had the tradition of keeping their office doors open and welcoming strangers with questions rather than shooing them out.

ULLMAN: My understanding is that, by law, AT&T had to put 1% of its revenue into research. I don't think there's any precedent for anything like that.

AHO: It was definitely the heyday of long-term unfettered research. And the results demonstrate the viability of that model. In my first 10 years at Bell Labs, I was able to write 40 papers and five books, almost all of them co-authored, because the environment was so rich with problems.

What about your collaborations with each other? Jeff, you've called your result on two-way deterministic classes of automata the work you're most proud of from that period.

ULLMAN: We have to understand it's of no consequence whatsoever. A finite automaton—something like a lexical analyzer—is ordinarily designed to chew up its input in one direction; it reads stuff and never goes back. But there was this question of what happens if you allow it to move back and forth on its tape, and we were able to show that that doesn't add to the power. It's a very hard proof, or at least, it was at the time. But I enjoyed the work, and I think Al did, too.

AHO: Jeff would walk into my office and say, "I've got the proof now." And I would listen to it and say, "What about this step?" An hour or two later, I would pop back into his office to say, "We could get around that this way." And he'd expose a flaw in my reasoning. We kept this up for [CONTINUED ON P. 119]

SIGGRAPH 2021

THE PREMIER **CONFERENCE** & **EXHIBITION** IN COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

SAVE THE DATE

9-13 AUGUST

Join us at SIGGRAPH 2021 to celebrate and honor the past, present, and future of computer graphics and interactive techniques. Celebrating 48 years of excellence, SIGGRAPH presents a high-quality experience showcasing the latest research, cutting-edge ideas, and breakthrough discoveries.

S2021.SIGGRAPH.ORG

Sponsored by ACM**SIGGRAPH**

THE 48TH INTERNATIONAL CONFERENCE & EXHIBITION ON COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES