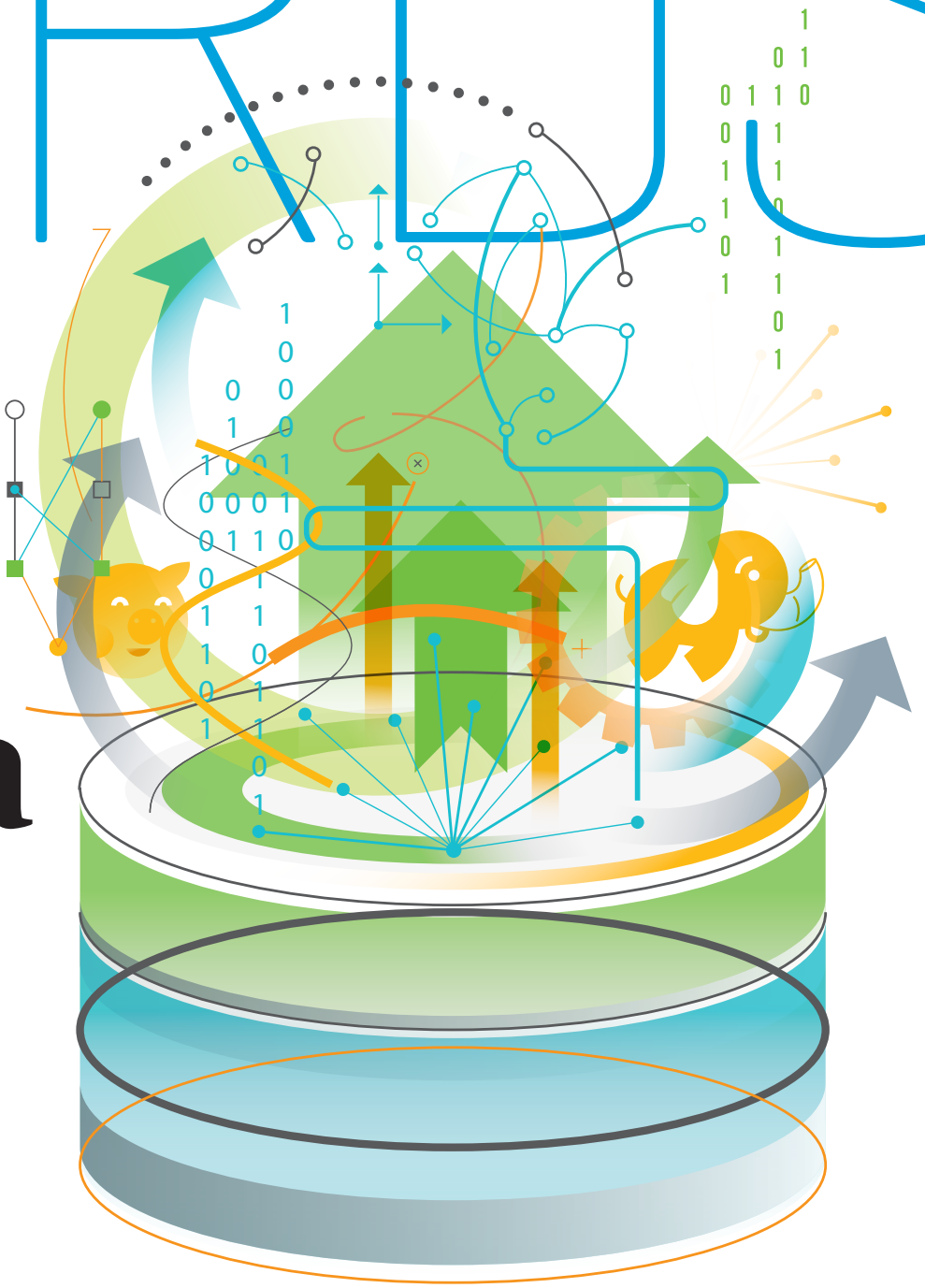
Crossroads The ACM Magazine for Students

FALL 2012 VOL.19 · NO.1

XRDS

XRDS.ACM.ORG



Big Data

The Center for Advanced Spatial Analysis

Creativity and Computing

Big Data at Google

Association for Computing Machinery

The Ultimate Online Resource for Computing Professionals & Students

ACM DL DIGITAL LIBRARY

http://www.acm.org/dl

Association for Computing Machinery

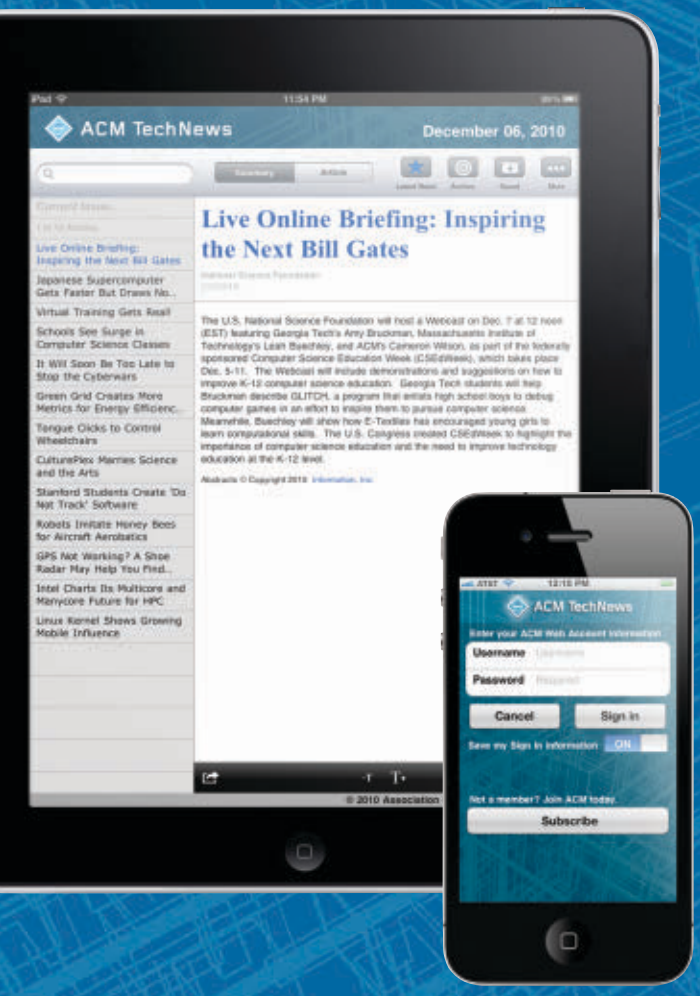
Advancing Computing as a Science & Profession

ACM TechNews Goes Mobile

iPhone & iPad Apps Now Available in the iTunes Store

ACM TechNews—ACM's popular thrice-weekly news briefing service—is now available as an easy to use mobile apps downloadable from the Apple iTunes Store.

These new apps allow nearly 100,000 ACM members to keep current with news, trends, and timely information impacting the global IT and Computing communities each day.



TechNews mobile app users will enjoy:

- ***Latest News:*** Concise summaries of the most relevant news impacting the computing world
- ***Original Sources:*** Links to the full-length articles published in over 3,000 news sources
- ***Archive access:*** Access to the complete archive of TechNews issues dating back to the first issue published in December 1999
- ***Article Sharing:*** The ability to share news with friends and colleagues via email, text messaging, and popular social networking sites
- ***Touch Screen Navigation:*** Find news articles quickly and easily with a streamlined, fingertip scroll bar
- ***Search:*** Simple search the entire TechNews archive by keyword, author, or title
- ***Save:*** One-click saving of latest news or archived summaries in a personal binder for easy access
- ***Automatic Updates:*** By entering and saving your ACM Web Account login information, the apps will automatically update with the latest issues of TechNews published every Monday, Wednesday, and Friday

The Apps are freely available to download from the Apple iTunes Store, but users must be registered individual members of ACM with valid Web Accounts to receive regularly updated content.

http://www.apple.com/iphone/apps-for-iphone/ **http://www.apple.com/ipad/apps-for-ipad/**

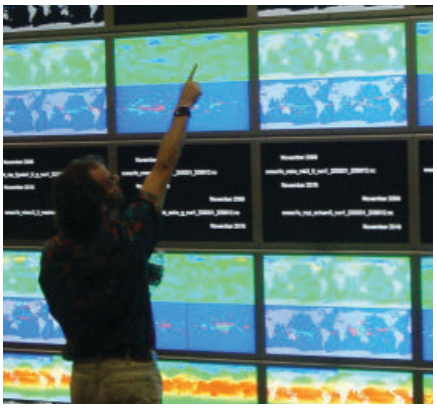
ACM TechNews

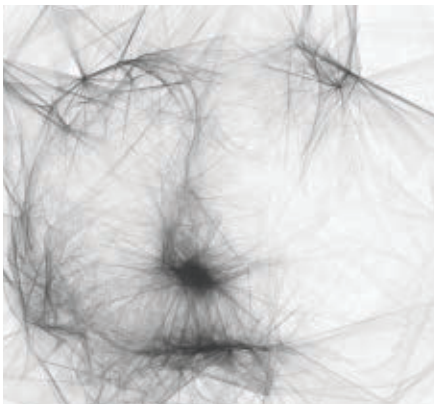
XRDS
Crossroads
The ACM Magazine for Students
FALL 2012 VOL.19 · NO.1

Photo by Éole Wind

begin

Cover Illustration by Leandro Castelao/Dutch Uncle Agency





features

end

Association for Computing Machinery

Photo courtesy of NASA; Generative artwork by Sergio Albiac; Photo courtesy of NASA.

XRDS

Association for Computing Machinery

SUBSCRIBE
Subscriptions ($19 per year includes *XRDS* electronic subscription) are available by becoming an ACM Student Member
www.acm.org/membership/student

Non-member subscriptions:
$80 per year
http://store.acm.org/acmstore

ACM Member Services
To renew your ACM membership or *XRDS* subscription, please send a letter with your name, address, member number and payment to:

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

Postal Information
XRDS (ISSN# 1528-4981) is published quarterly in spring, winter, summer and fall by Association for Computing Machinery, 2 Penn Plaza, Suite 701, New York, NY 10121. Application to mail at Periodical Postage rates is paid at New York, NY and additional mailing offices.

POSTMASTER: Send addresses change to:
XRDS: Crossroads,
Association for Computing Machinery,
2 Penn Plaza, Suite 701,
New York, NY 10121.

Offering# XRDS0171
ISSN# 1528-4972 (print)
ISSN# 1528-4980 (electronic)

Announcing the *XRDS* Blog

Among the goals highlighted in the previous issue was the launching of a new blog to give voice to student concerns, interests, and opinions. To that end we're delighted to announce the *XRDS* blog has been live since May with feature editor Shawn Freeman's coverage of the ICPC World's undergraduate programming competition.

We are currently featuring five bloggers who span continents and fields. Lea Rosen recently graduated from Rutgers School of Law with a J.D. She will be writing about the connections between technology, law, and civil and human rights issues. Lea has worked with the Electronic Frontier Foundation and the National Lawyers Guild among other organizations.

Wolfgang Richter is a Ph.D. student at Carnegie Mellon in computer science focusing on distributed computing Systems. Wolf is developing technologies that will enable new applications in cloud computing. Dimitris Mitropoulos is a Ph.D. candidate at Athens University of Economics and Business in Greece. He has authored numerous open source software libraries and worked on research projects in information security and software engineering.

Lora Oehlberg is a Ph.D. candidate in mechanical engineering at the University of California, Berkeley. Her work extends design theory and methodology for engineers building on, incorporating, and pushing the bounds in human-computer interaction. She has worked at Apple and Autodesk among other places. Finally, Matthew Kay will be writing about Ubiquitous computing (Ubicomp). He is a Ph.D. student working on health sensing and feedback at the University of Washington's Computer Science and Engineering Department.

We'd like to welcome all of our readers to check out the fascinating blog posts that these contributors have already posted and join the conversation online by leaving comments for each post. Going forward, you'll find a few compelling posts in each print edition of the magazine. In that light we've decided to feature the introductory posts from each of the bloggers in this issue. We hope you find them as diverse and informative as we do. By the time this issue is in your hands there should be even more posts from each of the bloggers up on the website and maybe even some new faces there as well.

We're continuing to expand the blog as well, so please write in and let us know what you think. Nominate a blogger or give feedback at eic@xrds.acm.org. In the coming months we'll be reaching out to ACM student chapters at universities worldwide. If you're involved with one of these and would like to partner with *XRDS*, please reach out.

Finally, we're searching for help with promoting *XRDS* content on social networks and external blogs and news sites. If you have any interest in helping out, definitely get in touch! Strong candidates will have a background in computer science or technology and business or marketing. We'd love to hear from you!

—Peter Kinnaird and Inbal Talgam-Cohen

UPCOMING ISSUES

Winter 2012
[December issue]
ICT for Development
Article deadline: August 31, 2012

Spring 2013
[March issue]
Scientific Computing
Article deadline: November 23, 2012

Summer 2013
[June issue]
CS and Creativity
Article deadline: March 1, 2013

begin

INBOX

THE STARTUP ISSUE

This ["How to be an 'Entrepredemic,'" by Jonathan Friedman, June 2012] was a long overdue introspect in how to navigate the decision of choosing academia, entrepreneurship, and corporate America. The quote—"Your goal should be: 'To help people do X,' where X is something you feel extremely passionate about. The business is just a vehicle to help you get there."—really says it all about entrepreneurship and research. Thanks this really provoked my thoughts.
—**Andrea Johnson**, *Comment on XRDS.acm.org*

Editor's reply:
Thanks, Andrea. Happy to hear the article was helpful!

Great article ["Want a Tenure?" by Eldar Sadikov and Montse Medina, June 2012], it contains a lot of useful information.
—**Marvin Andujar**, *Comment on XRDS.acm.org*

Editor's reply:
Glad you enjoyed it!

Hi guys,
The role of academia in the startup world was a good read. Recently, professor Alan M. Davis (http://reqbib.com/adavis/) visited our campus and gave an insightful lecture on what every computer science students needs to know about start-ups. He referred to his colleague's (Jeff C.) idea that, "you and I could sit down at a bar and think of 100 new business ideas with no problem at all." It is relatively very easy to think about simple ideas (that will eventually not work); the difficulty lies in making simple ideas work.

The "entrepredemic" should not only have academic qualification, skill, and expertise but also should be driven by passion to think different, make difference, and solve real world challenging problems. On one hand there is great risk, in terms of income, investments and mental stress. However, on the other hand, there is far greater likelihood of success if someone opts in for this type of work.
—**Santosh Kalwar**, *Email, Lappeenranta University of Technology, Finland*

AROUND THE WEB

I encourage you to check out ACM *XRDS* Magazine, the ACM's magazine for students xrds.acm.org #acm_xrds
—**Jessica Jones**, *Ph. D. candidate, Clemson University, Twitter (@JessicaNJones_)*

Use ur business as a vehicle to help people do whatever it is you're passionate about. ˜Jonathan Friedman #acm_xrds. Powerful words!
—**A.E. Johnson**, *Ph. D. candidate, Clemson University, Twitter (@hccandrea)*

Thx for the shoutout! RT @XRDS_ACM: Interested in startups check out @PlugandPlayTC launching Plug and Play Moscow plugandplaytechcenter.com/blog/news/plug...
—**Plug and Play**, *Twitter, (@PlugandPlayTC)*

This has been a very quiet group.
—**Shirley Hicks**, *Facebook*

Editor's reply: *We're looking for help on that front! Check out the "Letter from the Editors" in this issue for more information.*

How to contact *XRDS*: Send a letter to the editors or other feedback by email (xrds@acm.org), on Facebook by posting on our group page (http://tinyurl.com/XRDS-Facebook), via Twitter by using #xrds in any message, or by post to ACM Attn: XRDS, 2 Penn Plaza, Suite 701, New York, New York 10121, U.S.

Photo by Éole Wind

INIT

Big Data

Big data is everywhere. In just about every part of the modern world, scientists and engineers are developing new ways to measure events. Whether it's sensors, traffic cameras, sales data, Web usage, gene expression, or just about anything else, we have entered an age of truly massive data. Why do we collect this data? It's simple—to learn. We want to make predictions, quantify reality, or understand the past to optimize the decisions we make.

Massive data leads to many challenges for computer scientists. We're recording petabytes of data every day. Before we even think about learning from it, how and where do we store it? What kinds of systems do we build to retrieve and analyze the data? Can we develop theoretical guarantees about the optimality of these systems and strategies? Even if we can store the data, how do we learn from data sets that we cannot hold on a single computer or even in many computers? Can we learn from data on the fly? Moreover, our data is heterogeneous: We are observing social networks, ad clickthroughs, gene sequences, protein concentrations from cells, as well as confidential personal data that must be kept secret. How do we adapt our systems and algorithms for all kinds of data? These are just some of the exciting challenges facing the big data community.

For such a diverse topic like big data, it is nearly impossible to provide a comprehensive picture. Instead, in this issue we try to highlight some recent developments organized into three main themes: the theoretical foundation providing models and algorithms for reasoning about various data processing tasks, the large-scale computer systems for handling big data, and the range of applications and analyses enabled by big data from a variety of scientific domains. It has been an interesting time for big data with innovations coming simultaneously from theorists, system builders, and scientists or application designers. We hope to provide readers with an idea of the interplay between developments in these three different communities, how ideas and priorities in different communities interact, and together drive forward the development of big data analysis.

Interest in big data has given rise to a lot of recent interest in building systems to support queries and transactions over massive quantities of data.

THEORY

Opening the issue is an introduction to the theo-



The U.N. Global Pulse project is researching the use big data can have in understanding global development.

One of the datasets they are currently providing to researchers is anonymized cell phone records from Cote d'Ivoire, to explore what these might reveal about society and development there.

INIT

retical work for modeling and studying challenges in big data. Jelani Nelson introduces us to the world of streaming algorithms where there is a voluminous stream of data passing by. One can only examine each piece rudimentarily and yet is still able to report meaningful statistics about the whole stream at the end.

Next, Ashwin Machanavajjhala and Jerome P. Reiter describe a principled approach to privacy when dealing with big data. They provide examples of common pitfalls and general methods in both statistics and computer science for protecting privacy while still providing the enormous utility of big data.

Ronitt Rubinfeld follows with a task seeming even more incredible: Computing the answer without even looking at the whole input. She focuses on the problem of understanding distributions just from a few samples, in fact, much fewer than the domain size.

To wrap up the theory foundation, Jeff Ullman provides a gentle introduction to designing algorithms for the map-reduce framework for parallel processing of big data, a hugely successful approach for distributed computing in computer clusters with many practical applications.

SYSTEMS

Interest in big data has given rise to a lot of recent interest in building systems to support queries and transactions over massive quantities of data. A number of important technical developments in this arena have happened outside of academia. We have chosen to present three different perspectives from industry: one from a mature company, one from a small startup, and one from a company that is somewhere in between.

From Cloudera, Yanpei Chen and his coauthors—Andrew Ferguson, Brian Martin, Andrew Wang, Patrick Wendell—provide lessons that can be learned from a small startup on big data and why it makes sense for students to intern in a big data startup.

Our interview with Surajit Chaudhuri from Microsoft Research provides a lens into big data systems design from a company that has been designing database systems for decades.

Raghotham and Rajat from Facebook, which has been in the forefront of the NoSQL big data movement (as it is called), tell us how Facebook designs systems used internally to support queries over the massive quantities of data.

If the industry perspective on building systems wasn't enough, we present an article from Mike Carey, Chen Li, and Vinayak Borkar from UC Irvine, who have been rethinking the design of these big data systems from first principles, and have been making some exciting progress.

It has been an interesting time for big data with innovations coming simultaneously from theorists, system builders, and scientists or application designers.

APPLICATIONS

In such massive data contexts, getting data into a form amenable to analysis and visualization is challenging. Jeff Heer and Sean Kandel write about cutting-edge work that enables data analysts to quickly gain valuable insights from their data.

Social network analysts have been using massive graph data to understand social interactions and behavior. B. Aditya Prakash presents some of the challenges and strategies for studying propagation and immunization in the realm of large social networks.

John Langford from Microsoft Research gives an overview of the challenges in machine learning on big data. He addresses approaches to thinking about learning from data in parallel and some interesting applications.

We also have massive datasets on the cell-by-cell and genome levels. Cliburn Chan reviews the current issues facing the computational biology community and current computational strategies for tackling these problems.

SUMMARY

Overall, through this issue, we are providing a peek into the exciting world of big data—through the lens of theorists, systems designers, scientists and application developers. Indeed, it is undeniable that big data is going to grow in importance in all fields, and will need our expertise. The expertise of an educated bunch of young researchers, scientists, and engineers.

—Andrew Cron, Huy L. Nguyen, and Aditya Parameswaran, Issue Editors

1.8 zettabytes (1.8 trillion gigabytes)

of information created or copied in 2011.

BENEFIT

ACM Author-Izer

Anyone who has had to perform research knows the value of the ACM Digital Library (DL). What they may not realize is this access is paid for either by their institution or by their own membership. The pay wall restricts access to the population at large—a controversial decision given how much research is funded from public sources. This has meant that for many years individuals have chosen to self-archive and promote their publications on their websites, running the risk of copyright infringement and resulting in inaccurate download counts in the DL.

ACM has come up with an awesome way of dealing with this. Author-Izer enables authors to generate and post links on their websites, allowing visitors to download the definitive version of their paper from the DL.

So why should students be interested? In developing your career, it's important to share the work you've produced. Not only does Author-Izer allow you to do this, it does it in such a way that any downloads off your personal webpage are counted into the download statistics.

For more info, visit http://www.acm.org/publications/acm-author-izer-service.

—Daniel Gooch

ADVICE

Six Tips for Students Interested in Big Data Analytics

This department is quite different than previous ones. Instead of having students advising you, we've asked a big data expert at IBM what it takes to get a job in big data. Here is what she had to say.

—Vaggelis Giannikas

Forrester, a global research and advisory firm, estimates firms effectively utilize less than five percent of available data, mainly due to the lack of training and skills necessary for the type of information gathering and analysis needed to transform big data. If you are considering a career in information technology, business analytics, and/or computer science, you may want to consider the data scientist role. As I'm often asked where students can begin, I've pulled together six tips for those interested in a career in the growing and exciting field of big data.

1. Look for programs offering hands-on training. Large organizations like IBM are working with academic organizations around the world specifically in the area of expanding and strengthening data and analytics curricula to meet the growing demand of highly skilled business professionals of the future. As an example, Yale's School of Management partnered with IBM to examine existing case studies and apply big data analytics software to solve problems.

2. Gain domain expertise, or industry knowledge, across several fields before applying data analysis to gain the best insights. As Manish Parashar, director of Rutgers Discovery Informatics Institute says, "Students not only need to learn tools to handle big data—they must learn how to integrate big data into their reasoning." Domain expertise will be important across multiple industries as the applications of big data analytics expand.

3. Develop an understanding of business practices. In a recent conversation on big data skills, educator and author Terri Griffith believes you can't manage people, technology, or organizational processes in silos. Instead, an effective data professional in the business world knows how to use all of their resources as well as mix and match solutions of people, technology, and other assets across the organization.

4. Gain background in computer science and software development, but do not ignore basic analytical skills such as statistics. During an online panel discussion, Anders Rhod Gregersen of Vestas Wind Systems stated the "power-users" of analytics are statisticians. By combining math with computer science, data professionals can develop unique algorithms that analyze data in organization-specific ways. This unique overlap is known as "machine learning" and is one of the most in-demand skills.

5. Learn to think like a scientist. Scientists see the world very differently. They do research, ask questions, and set up experiments to get answers. They question everything and don't just look at data, they dissect it.

6. Never stop learning. Ask people to explain what they do to expand your skills and explore supplemental learning opportunities.

Biography

Anjul Bhambhri has 23 years of experience in the database industry and is currently IBM's Vice President of Big Data Products, overseeing product strategy and business partnerships. In 2009, she received the YWCA of Silicon Valley's "Tribute to Women in Technology" Award.

$40,000

The cost of the first 1GB hard drive in 1980.

66¢

The cost of 1GB storage for one year on Amazon S3.

300,000

servers in just one of Microsoft's data centres.

UPDATES

Creativity and Computing
An ACM Student Chapter Initiative

Who says computer science and creativity can't mingle? In a creative and novel twist on traditional ACM student chapter initiatives, the PUCIT-ACM chapter in Pakistan fused the worlds of computer science and cinema with their first ever Short Movie Competition. The activity gave students the opportunity to demonstrate their acting, directing, and editing capabilities—all in a computer science environment.

"Entertainment and enjoyment is everybody's right and without this life seems dull and boring. If we put some technical and soft skills in it, then it produces the best results in the form of learning and inspiration of students," explained PUCIT-ACM chairman Jawad Javed. The goals of the program were numerous: to encourage a good combination of technical and creative skill, to promote awareness of various social issues, to attract and inspire students to engage with ACM, to provide a platform through which students could express the problems of society, and to induce intellectual discussion among students.

Entries were divided into three categories: serious, documentary, and humorous. The movies were limited to five minutes in length. Students were encouraged to apply their knowledge of image processing and multimedia editing software to glamorize their films. Submission titles included a wide range of topics such as "History Repeats Itself" (serious), "Brain Drain" (documentary), and "Recursion" (humorous). The activity recruited computer science faculty to sit on a panel of judges. The short films were critiqued on the basis of story, acting, effectiveness, audience response, and most importantly, visual effects. The winners in each category represented three different institutions in Pakistan: PUCIT, UET, and GCU. To boost entertainment, the animated film "Despicable Me" was screened following the competition.

With their eyes glued to the screen, attendees of the PUCIT-ACM Short Movie Competition enjoy a submitted film.

But what is an ACM event without a bit of computing innovation? In an idea unheard of at PUCIT-ACM, the organizers decided to employ an automated judging panel and scoreboard. The head of the chapter's IT branch teamed up with the creative art director to develop a Web-based scoring system in PHP. The software, MC^2 (Movie Competition Squared), was deployed to the student chapter's website and every judge was given a login to use during the competition. The software automated the calculation of the competition results.

The Short Movie Competition is one example of a fantastic and creative way to broaden the range of activities and services provided by an ACM student chapter. PUCIT-ACM did a wonderful job of engaging their students and faculty in a fun, novel event that fused creativity and computer science. This event helps remind us of the value of interdisciplinary activity in computing—not even movie making is too far a stretch for ACM student chapters!

—Ben Deverett

The newly launched *XRDS* blog highlights a range of topics from conference overviews to privacy and security, from HCI to cryptography. Selected blog posts, edited for print, will be featured in every issue. Please visit xrds.acm.org/blog to read each post in its entirety.

BLOGS

The Changing Nature of (Ubiquitous) Computing

By Matthew Kay

I seem lately to be having recurring conversations on the same theme: The changing nature of computing and the movement from desktop to mobile/ubiquitous computing. I think, like social change, much of technological change comes through new generations that grow up with realities their parents had to adopt—computers, the Internet, social media. Wonderful clichés like, "Back in my day, we had to know how to read a map!" betray fundamentally different views of the world that are symptomatic of technological shift. When kids are so used to a technology being there that they can't conceive of its absence—the 2-year-old pinch-zooming a magazine in vain—that is when a new generation of people, whose underlying worldview is not shaped by old ideas but built on a foundation of new technology, develop solutions that are truly native to that technological landscape.

So, what does this have to do with ubicomp? Ubiquitous computing is a thing—separate from other instantiations of interactive computing—only insofar as it isn't ubiquitous. Once it underlies, as it increasingly does, so much of how we interact with technology on a day-to-day basis, it becomes less meaningful to say one does work in ubiquitous computing apart from other areas of human–computer interaction (HCI).

For example, my own interest in pervasive health sensing and feedback (i.e. mobile, or in-home, or ubiquitous health tech) did not arise from my interests in ubiquitous computing as an area—I had none. It arose, broadly, from my interest in human–computer interaction and a particular application area. It happens that many of the problems and questions I am interested in draw on ubicomp solutions, and are appropriate for a ubicomp audience, but if (for example) my research takes me into Web-based or desktop-based solutions I will follow my way there. I suspect many other people ostensibly in ubicomp today feel similarly. Ten or 20 years out from now, when the kids who today are frustratedly pinch-zooming magazines have become researchers and app developers, it won't occur to them that building interactive systems doesn't involve ubicomp, since in their technological landscape the two will be the same.

As the computing everyone uses moves off the desktop, more and more questions in human–computer interaction involve ubiquitous computing technology, such as smartphones, even if only as a platform. Does that research then become ubicomp work? Or will the notion of ubicomp become so embedded in much of the rest of HCI that this distinction is meaningless? Like most things there is a grey area here, but as ubicomp becomes integral to much of HCI it might be useful to ask if we need to rethink the boundaries of these concepts. I suspect the coming generation of pinch-zoomers will have difficulty seeing the difference.

Matthew Kay is a Ph.D. student in computer science and engineering at the University of Washington. He studies health sensing and feedback.

How Secure is Your Software?

By Dimitris Mitropoulos

When you are implementing an application, your first goal is to achieve a specific functionality. You will follow some code conventions during implementation while simultaneously checking your code quality. But how secure is your code? Is there a way for a malicious user to harm you or your application by taking advantage of potential bugs that exist in your code? Unfortunately, most programmers have been trained in writing code that implements the required functionality without considering its many security aspects. Most software vulnerabilities derive from a relatively small number of common programming errors that lead to security holes.

In 2001 when software security was first introduced as a field, information security was mainly associated with network security, operating systems security, and viral software. Until then, there were hundreds of millions of applications implemented but not with security in mind. As a result, the vulnerabilities "hidden" in these (now legacy) applications can still be used as backdoors that lead to security breaches.

Although, nowadays computer security is standard fare in academic curricula around the globe, few courses emphasize secure coding practices. For instance, during a standard introductory C course, students may not learn that using the gets function could make their code vulnerable to an exploit. Even if someone includes it in a program, while compiling he or she will get the following obscure warning: *"the 'gets' function is dangerous and should not be used."* Well, gets is dangerous because it is possible for the user to crash the program by typing too much into the command prompt. In addition, it cannot detect the end of available memory, so if you allocate an amount of memory too small for the purpose, it can cause a segmentation fault and crash.

27% the reduction in burglaries from July 2010 to July 2011

the city of Santa Cruz, CA was able to achieve by using crime data to predict where future crimes would happen.

BLOGS

The situation is similar in Web programming. Programmers are not aware of security loopholes inherent to the code they write; in fact, knowing they program using higher level languages than those prone to security exploits, they may assume these render their application immune from exploits stemming from coding errors. Common traps into which programmers fall concerns user input validation, the sanitization of data that is sent to other systems, the lack of definition of security requirements, the encoding of data that comes from an untrusted source, and others.

Do not panic, you are not obliged to become an expert in secure coding. There are numerous tools that can help you either build secure applications or protect existing ones.

Dimitris Mitropoulos is a Ph.D candidate at the Athens University of Economics and Business. His research interests include information security and software engineering.

Eyes Clouded by Distributed Systems

By Wolfgang Richter

You are probably reading this article with a dual- or quad-core processor, and perhaps with even more cores. Your computer is already a distributed system, with multiple computing components—cores—communicating with each other via main memory and other channels such as physical buses—or wires—between them. As you browse multiple Web pages you are interacting with the largest distributed system ever created—the Internet. Every Internet company depends on distributed systems, and, by extension, the economies of the world are now tied to them. Companies such as Google, Facebook, and Amazon are all interested in building highly efficient large-scale distributed systems, which power their businesses.

With world economies tied to distributed systems, it is no mistake that the study of distributed systems is paramount to the future of computing and research reflects this with efforts such as the Exascale project. The Exascale project explores what future distributed systems might look like beyond the largest scale imaginable today. No problem moving forward will be able to avoid the often messy, although ultimately satisfying when overcome, challenges of distributed computing.

We must come to agreement on the definition for distributed system. Of course, my view is clouded by a lens through which I see everything as a distributed system. You may not agree with me, and we encourage discourse, so please feel free to comment in with your criticism. I hope that the definition below crisply defines what a distributed system is in your mind, as I hope to dissect many of the most interesting developments in distributed systems research in future articles:

In computer science, a distributed system is any set of entities capable of computation, which also have the capability of communicating via a set of mechanisms such that computation may be organized among them.

Examples of distributed systems:

1. Car; multiple embedded microprocessors
2. Single core computer with graphics card; two discrete computation entities communicating via shared buses
3. Multicore computer; clearly a distributed system with multiple cores
4. Networked computers; at a minimum they cooperate via network protocols, in the limit they could be architected together for high performance or scientific computing.

Wolfgang Richter is a fourth year Ph.D. student in the CS Departmentat at Carnegie Mellon University. His research focus is distributed systems, specifically both retrospecting and introspecting clouds and other collections of virtual machines.

Dear HCI, Thank you. Love, Mechanical Engineering

By Lora Oehlberg

My entire academic background—B.S., M.S., Ph.D.—is in mechanical engineering, specializing in design. However, in addition to conferences hosted by the American Society of Mechanical Engineering, I also attend the suite of ACM's Human-Computer Interaction (HCI) Conferences. So, why should mechanical engineering care about HCI?

Product design refers to the blend of mechanical engineering and industrial design. Design is the "outward facing" side of mechanical engineering; product designers conceptualize, design, and implement many of the physical products you interact with on a daily basis. In the cafe that I'm currently writing from, a design engineer was involved in everything from the teacup, the teapot, the table, the chair, and the laptop I'm writing on... and all the packaging that each of those products arrived in. These traditional products still have interfaces—examples from Don Norman's infamous *Design of Everyday Things* address how people physically interact with "non-smart" products and devices such as teapots, doorknobs, or rotary telephones. Today's product designers are asked to not only design the physical product, but also weigh in on how the user should interact with smart products.

Early-stage phases of new product development—

6 hrs., 2 mins.

for Google to sort 1PB 100-byte records in 2008.

particularly user research and concept generation—are agnostic to whether or not the final "product" is a physical product, software, a physical or digital service, or an architectural space. As a result, many of the same design theory principles coming out of the interaction design community are broadly applicable to other design domains, including product design or new product development, within some level of translation.

While computer scientists frequently design new programming environments for themselves, mechanical engineers and new product developers are not always the subject of thoughtful, human-centered technology design. Taking an HCI perspective to understand how engineers and designers are users of software opens up the possibility for better-designed tools in the future (I'm looking at you, CAD!).

It's sometimes easy to get lost in cognition, perception, algorithms, and pixels. However, when mechanical engineers check their gut, they see the physical interface between humans and computers. You'll see plenty of relevant contributions from mechanical engineering in the areas of ergonomics, haptic feedback, or tangible interfaces. But more broadly, mechanical engineers offer the reminder that humans (and computers) still primarily exist in a physical world.

Lora Oehlberg is a Ph.D. candidate in mechanical engineering at the University of California, Berkeley. Her research focuses on design theory and methodology, and frequently extends into HCI.

If You Think Network Security is a Safety Issue, You'll Need to Deal with Cost-Benefit Analysis

By Lea Rosen

I gave a short presentation at Hive76 recently, and after it was over I hung around answering questions. One man asked me, repeatedly, whether the FDA has codified security standards for networked and wireless medical devices like insulin pumps and pacemakers. The answer wasn't satisfactory for either of us; he was sincerely alarmed, and I couldn't reassure him. When I said they have no such standards, he asked me why not. If their mandate is to ensure the safety of medical devices, he said, how could they reasonably neglect such an obvious security risk?

Many of us believe this kind of security bears seriously on human safety and human rights. I think this perspective is common in engineers, designers, and hackers—people for whom network and wireless security are tangible realities, who are accustomed to translating abstract into concrete. And yet, government agencies tasked with ensuring our safety seem oblivious to the danger posed by insecure networks. That's why the man I spoke to was so frustrated—he saw technological security as a natural extension of the FDA's mandate to ensure the safety of medical implants and devices. It seemed to him like they were failing to adequately do their job.

Look, civil servants are as able as anyone to comprehend the danger that insecure pacemakers or insulin pumps might pose. But they're also required to prepare and present cost-benefit analysis reports to the Office of Management and Budget before they can do anything. Cost-benefit analysis (CBA) is probably familiar to most readers of this blog. It's a method of decision-making that aims to maximize welfare along an economic model. After accumulating data on the benefits and costs of each project, the analysis determines whether the benefits o outweigh the costs. Since 1981, all federal agencies have been required by law to use CBA to determine how they will carry out their individual mandates. The question they need to answer is not whether a project would be useful, or a logical extension of an agency's mandate, but whether the project is economically rational.

The risk posed to human life by bad or nonexistent security is difficult to quantify. The FDA would need to do so much outside work just to prepare itself to do these initial evaluations that it's unlikely to ever seem worth it compared to the current status quo. It is possible that investing in security—developing standards, hiring or outsourcing experts, training employees in an area with which they are presumably not familiar—just isn't an economically rational thing for the FDA to do.

There is no federal agency that regulates the network security of consumer devices, and it seems unlikely that existing agencies will find it economically rational in the near future to invest in learning how to evaluate the risk and harm that might come of bad security. If secuarity experts want to persuade the administrative state to take a serious interest in this problem—and many of us believe that it should—the language and values of cost-benefit analysis are important to consider. If CBA can be used to make a case for device security, we should make it. If the numbers don't add up in our favor, it'll be all the more essential to articulate why agencies like the FDA should look at network security in a different way.

Lea Rosen, J.D., writes about technology and law, with a focus on civil and human rights issues inherent in the creation and adoption of new technologies.

Sketching and Streaming Algorithms for Processing Massive Data

The rate at which electronic information is generated in the world is exploding. In this article we explore techniques known as sketching and streaming for processing massive data both quickly and memory-efficiently.

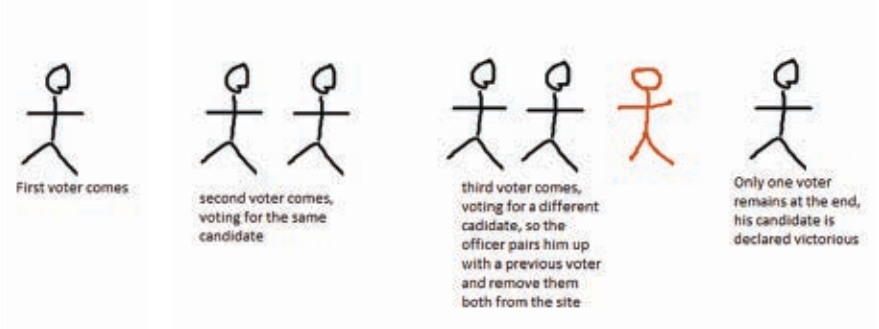
By Jelani Nelson

DOI: 10.1145/2331042.2331049

Several modern applications require handling data so massive that traditional algorithmic models do not provide accurate means to design and evaluate efficient algorithms. Such models typically assume that all data fits in memory, and that running time is accurately modeled as the number of basic instructions the algorithm performs. However in applications such as online social networks, large-scale modern scientific experiments, search engines, online content delivery, and product and consumer tracking for large retailers such as Amazon and Walmart, data too large to fit in memory must be analyzed. This consideration has led to the development of several models for processing such large amounts of data: The external memory model [1, 2] and cache-obliviousness [3, 4], where one aims to minimize the number of blocks fetched from disk; property testing [5], where it is assumed the data is so massive that we do not wish to even look at it all and thus aim to minimize the number of probes made into the data; and massively parallel algorithms operating in such systems as MapReduce and Hadoop [6, 7]. Also in some applications, data arrives in a streaming fashion and must be processed on the fly. Such cases arise, for example, with packet streams in network traffic monitoring, or query streams arriving at a Web-based service such as a search engine.

In this article we focus on this latter streaming model of computation, where a given algorithm must make one pass over a data set to then compute some function. We pursue such streaming algorithms, which use memory that is sublinear in the amount of data, since we assume the data is too large to fit in memory. Sometimes it becomes useful to consider algorithms that are allowed not just one, but a few passes over the data, in cases where the data set lives on disk and the number of passes may dominate the overall running time. We also occasionally discuss

Figure 1: Explaining the MJRTY algorithm.



sketches. A sketch is with respect to some function f, and a sketch of a data set x is a compressed representation of x from which one can compute $f(x)$. Of course under this definition $f(x)$ is itself a valid sketch of x, but we often require more of our sketch than just being able to compute $f(x)$. For example, we typically require that it should be possible for the sketch to be updated as more data arrives, and sometimes we also require sketches of two different data sets that are prepared independently can be compared to compute some function of the aggregate data, or similarity or difference measures across different data sets.

Our goal in this article is not to be comprehensive in our coverage of streaming algorithms. Rather, we discuss in some detail a few surprising results in order to convince the reader that it is possible to obtain some non-trivial algorithms within this model. Those interested in learning more about this area are encouraged to read the surveys [8, 9], or view the notes online for streaming courses taught by Chakrabarti at Dartmouth [10], Indyk at MIT [11], and McGregor at UMass Amherst [12].

PROBABILISTIC COUNTING

How many bits does it take to store an integer between 1 and n? The answer is clearly $\lceil \log_2 n \rceil$ bits, else two integers would map to the same bitstring and be indistinguishable. But what if we only care about recovering the integer up to a constant factor? Then it suffices to only recover $\lceil \log n \rceil$, and storing $\lceil \log n \rceil$ only requires $O(\log \log n)$ bits.

This observation was behind one of the oldest known streaming algorithms, proposed by Morris, former chief scientist of a division of the NSA (and father of the inventor of the first Internet worm) [13]. Consider the streaming problem where we see a stream of n increments. We would like to compute n, though approximately, and with some potential small probability of failure. We could keep an explicit counter in memory and increment it after each stream update, but that would require $\lceil \log_2 n \rceil$ bits. Morris' clever algorithm works as follows: initialize a counter c to 1, and after each update increment c with probability $1/2^c$ and do nothing otherwise. Flajolet showed the expected value of 2^c is $n + 2$ after n updates [14], and thus $2^c - 2$ is an unbiased estimator of n. The same work showed the variance is bounded such that $2^c - 2$ will be within a constant factor of n with constant probability. By a combination of averaging many independent estimators, as well as attempting to store $\log_{1+\gamma} n$ in memory instead of $\log_2 n$ for some small $\gamma > 0$ by incrementing with higher probability, it is possible to obtain more precise approximations of n in small memory with very large probability.

FREQUENT ITEMS

A common desired ability in many software systems is the ability to track "hot" items. For example, Google Trends keeps track of the search queries and topics that have been the most popular over a recent time window. Large ISPs like AT&T want to monitor IP traffic being routed through their network to understand, for example, which servers are receiving the largest amounts of traffic. Such knowledge can help in detecting Denial of Service attacks, as well as designing network infrastructure to minimize costs. For companies serving similar or identical content to large numbers of users, such as Akamai or Dropbox, it may be beneficial to detect whether certain content becomes hot, i.e. frequently downloaded, to know which files to place on servers that are faster or have connections with higher bandwidth.

The formal setup of this problem is as follows. There is some stream of tokens $i_1, i_2, \ldots, i_m$ with each i_j coming from some fixed set of size n (e.g. the set of all 232 IPv4 IP addresses, or the set of all queries in some dictionary). Let us just suppose this fixed set is $[n]$ (denoting the set $\{1, 2, \ldots, n\}$). For some $0 < \varepsilon \leq 1/2$ known to the algorithm at the beginning of the stream, we would like to report all indices $i \in [n]$ such that i appeared in the stream more than εm times. This formalization models the examples above: A query stream coming into Google, a packet stream going through a router, or a stream of downloads over time made from some content delivery service.

One of the oldest streaming algorithms for detecting frequent items is the MJRTY algorithm invented by Boyer and Moore [15]. MJRTY makes the following guarantee: If some $i \in [n]$ appears in the stream a strict majority of the time, it will be found. If this guarantee does not hold, MJRTY may output anything. Note that if given a second pass over the stream, one can verify whether the output index actually is a majority index. Thus, MJRTY solves the frequent items problem for $\varepsilon = 1/2$.

Before describing MJRTY, first consider the following means of carrying out an election. We have m voters in a room, each voting for some candidate $i \in [n]$. We ask the voters to run around the room and find one other voter to pair up with who voted for a different candidate (note that some voters may not be able to find someone to pair with, for example if everyone voted for the same candidate). Then, we kick everyone out of the room who did manage to find a partner (see Figure 1). A claim whose proof we leave to the reader as an exercise is that if there actually was a candidate with a strict majority, then

some non-zero number of voters will be left in the room at the end, and furthermore all these voters will be supporters of the majority candidate.

The MJRTY algorithm is simply the streaming implementation of the election procedure stated in the previous paragraph. We imagine an election official sitting at the exit door, processing the voters one by one. When the next voter is processed, he will either be asked to sit aside amongst a pool of people waiting to be paired off (clearly everyone in this pool supports the same candidate, else the official could pair two people in the pool with each other and kick them out of the room), or he will be paired up with someone in the pool and removed. Now, when a new voter approaches the official one of several things may happen. If the pool is empty, the official adds him to the pool. If the pool is not empty and he is voting for a different candidate than everyone in the pool, the official grabs someone else from the pool, pairs them off, and kicks them both out of the room. Else if his vote agrees with the pool, the official adds him to the pool. If the pool is non-empty at the end, then the candidate the pool supports is labeled the majority candidate. Note that this algorithm can be implemented to discover the majority by keeping track of only two things: The size of the pool, and the name of the candidate everyone in the pool supports. Maintaining these two integers requires at most $\lceil \log_2 n \rceil + \lceil \log_2 m \rceil$ bits of memory.

What about general $\varepsilon < 1/2$? A natural generalization of the MJRTY algorithm was invented by Misra and Gries [16] (and has been rediscovered at least a couple times since then [17, 18]). Rather than pair off voters supporting different candidates, this algorithm tells the voters to form groups of size exactly k such that no two people in the same group support the same candidate. Then everyone who managed to make it into a group of size exactly k is kicked out of the room. It can be shown that any candidate receiving strictly more than m/k votes will be supported by one of the last candidates standing, so we can set $k = \lceil 1/\varepsilon \rceil$. Furthermore, a simple extension of the MJRTY algorithm implementation using k-1 ID/counter pairs (and thus using $O(k \log(n + m))$ bits of space) provides a streaming algorithm. When a new voter enters, if he matches with any candidate in the pool then we increment that counter by one. Else, we decrement all counters by one (corresponding to forming a group of size k and removing them).

Several modern applications require handling data so massive that traditional algorithmic models do not provide accurate means to design and evaluate efficient algorithms.

DISTINCT ELEMENTS

On July 19, 2001 a variant of the Code Red worm began infecting machines vulnerable to a certain exploit in an older version of the Microsoft IIS web server. The worm's activities included changing the website hosted by the infected Web server to display

HELLO!
Welcome to http://www.worm.com!
Hacked By Chinese!

as well as an attempted Denial of Service attack against www1.whitehouse.gov.

In August 2001, while trying to track the rate at which the worm was spreading, Moore and Shannon at The Cooperative Association for Internet Data Analysis (CAIDA) needed to track the number of distinct IP addresses sending traffic on particular links whose packets contained the signature of the Code Red worm. This setup turns out to precisely be an instantiation of the distinct elements problem introduced and studied by Flajolet and Martin [19]. In this problem, one has a stream of elements $i_1, i_2, \ldots, i_m$ each being an integer in the set $\{1, 2, \ldots, n\}$. Then, given one pass over this stream, one must compute F_0, the number of distinct integers amongst the i_j. In the case of tracking the Code Red worm, $n = 232$ is the number of distinct IP addresses in IPv4, and m is the number of packets traversing a monitored link while carrying the signature of the worm. Aside from network traffic monitoring applications, the distinct elements problem naturally arises in several other domains: estimating the number of distinct IP addresses visiting a website, or number of distinct queries made to a search engine, or to estimate query selectivity in the design of database query optimizers.

An obvious solution to the distinct elements problem is to maintain a bitvector x of length n, where we initialize $x = 0$ then set $x_i = 1$ if we ever see i in the stream. This takes n bits of memory. Another option is to remember the entire stream, taking $O(m \log n)$ bits. In fact Alon, Matias and Szegedy [20] showed $\Omega(\min\{n, m\})$ bits are necessary for this problem unless slack is allowed in two ways:

1. **Approximation.** We do not promise to output F_0 exactly, but rather some estimate such that $|\widetilde{F}_0 - F_0| \leq \varepsilon F_0$.
2. **Randomization.** Our algorithm may output a wrong answer with some small probability.

Our goal is now to produce such an estimate $\widetilde{F}_0$, which is within εF_0 of F_0 with probability at least two-thirds. This success probability can be amplified arbitrarily by taking the median estimate of several independent parallel runs of the algorithm. Our goal is to design an algorithm using $f(1/\varepsilon) \cdot \log n$ bits of memory, e.g. $O(\log n)$ memory for a constant factor approximation. Note that just storing an index in $[n]$ requires $\lceil \log_2 n \rceil$ bits, which we presume fits in a single machine word, so we are aiming to use just a constant number of machine words!

In fact, such an algorithm is possible. Suppose for a minute that we had access to a perfectly random hash function h mapping $[n]$ to the continuous interval $[0, 1]$ (although one might work with a sufficiently fine discretization of this interval since computers can only store numbers to finite precision). We maintain a single

number X in memory: The minimum value of $h(i)$ we have ever encountered over all i appearing in the stream. One can show that the expected minimum value satisfies

$$EX = 1/(F_0 + 1).$$

Thus a natural estimator is to output $1/X - 1$. Unfortunately a calculation shows the standard deviation of X is almost equal to its expectation, so that $1/X - 1$ is poorly concentrated around a good approximation of F_0. This can be remedied by letting $\bar{X}$ be the average of many independently such Xs obtained independently at random in parallel, then instead returning $1/\bar{X}-1$. A more efficient remedy was found by Bar Yossef et al. [21], and further developed (and named as the KMV algorithm that stands for "k minimum values") by Beyer et al. [22]. The algorithm maintains the k minimum hash values for $k = O(1/\varepsilon^2)$. Now let X_k denote the k^{th} minimum hash value. Then

$$EX_k = k/(F_0 + 1),$$

and the returned estimate is thus given as $k/X_k - 1$. This algorithm can be shown to return a value satisfying the desired approximation guarantees with large constant probability.

LINEAR SKETCHES

In some situations we do not simply want to compute on data coming into a single data stream, but on multiple datasets coming from multiple data streams. For example, we may want to compare traffic patterns across two different time periods, or collected at two different parts of the network. Another motivating force is parallelization: Split a single data stream into several to farm out computation to several machines, then combine the sketches of the data these machines have computed later to recover results on the entire data set.

One way of accomplishing the above is to design streaming algorithms that use linear sketches. Suppose we are interested in a problem, which can be modeled in the following turnstile model. We have a vector $x \in R^n$ that receives a stream of coordinate-wise updates of the form $x_i \leftarrow x_i + v$ (v may be positive or negative). We then at the end of the stream want to approximate $f(x)$ for some function f. For example in the distinct elements problem, v is always 1 and $f(x) = |\{i : x_i \neq 0\}|$. A streaming algorithm using a linear sketch is then one whose memory contents can be viewed as Ax for some (possibly random) matrix A. Unfortunately the algorithms discussed above do not operate via linear sketches, but now we will see examples where this is the case.

Join size estimation. When querying a relational database there can be multiple ways of executing the query to obtain the result, for example by taking advantage of associativity. Database query optimizers try to cheaply estimate a plan to use to answer the query so as to minimize the time required. For queries involving joins or self-joins, such optimizers make use of size estimates of these joins to estimate intermediate table sizes. Ideally we would like to obtain these estimates from a short sketch that can fit in cache and thus be updated quickly as data is inserted into the database.

Let us formally define this problem. We have an attribute A and domain D, and for $i \in D$ we let x_i denote the frequency of i in A. We will assume that $D = [n]$. The self-join size on this attri-

Mastering Real-time Big Data with Stream Processing Chains

DOI: 10.1145/2331042.2331050

Pervasive applications rely on increasingly complex streams of sensor data continuously captured from the physical world. Such data is crucial to enable applications to "understand" the current context and to infer the right actions to perform, be they fully automatic or involving some user decisions. The continuous nature of such streams, the progressive increase of monitored features, the relatively high throughput at which data is generated and the number of sensors usually deployed in the environment, impose stricter requirements on monitoring networks and software, by requiring both single-event granularity and aggregate measures computation. The former ensures fine analysis of anomalous conditions while the latter grants constant human-addressable monitoring of relevant features. We introduce an open source stream-processing framework, named spChains (http://elite.polito.it/spchains), based upon state-of-the-art stream processing engines, which enables declarative and modular composition of stream processing chains built atop of a set of stream processing blocks. Blocks are predefined in an extensible library and designed to be application-independent; the library components cover a relevant set of elaborations emerging from typical energy monitoring applications. They can readily be reused in almost any processing task. On the converse, chains (i.e., connected sets of blocks) must be designed according to the specific data-processing needs, composing together the available blocks and extending the base block library when needed. We demonstrate the flexibility and effectiveness of the spChains framework using a two-phase experimentation process. In the first phase, performance characterization is carried, showing that the current spChains implementation can easily handle from 20k to 130k events per second, depending on the required processing. Secondly, a real-world trial on some commercial applications, is analyzed and results confirm the flexibility of the approach and its applicability in typical enterprise-level settings. (You can read the full article online at http://xrds.acm.org/article.cfm?aid=2331050) —*Dario Bonino and Luigi De Russis*

Large ISPs like AT&T want to monitor IP traffic being routed through their network to understand, for example, which servers are receiving the largest amounts of traffic.

bute is then $\| x \|_2^2 = \Sigma_i x_i^2$, and thus we simply want to estimate the squared ℓ_2-norm of a vector being updated in a data stream. In fact this general problem has a wider range of applicability. For example, noting that $\| x \|_2^2$ is sensitive to heavy coordinates, AT&T used ℓ_2-norm estimation to detect traffic anomalies where servers were receiving too much traffic, signaling potential Denial of Service attacks (in this case x_i is the number of packets sent to IP address i) [23, 24].

The "AMS sketch" of Alon, Matias, and Szegedy [20, 25] provides a low-memory streaming algorithm for estimating $\| x \|_2^2$. Suppose we had a random hash function $h:[n] \rightarrow \{-1, 1\}$. We initialize a counter X to 0, and when some value v is added to x_i we increment X by $v \cdot h(i)$. Thus at the end of the stream, $X = \Sigma_i \ x_i \cdot h_i$. It can be shown that $EX^2 = \| x \|_2^2$ and that the variance satisfies $E(X^2 - EX^2)^2 \leq 2\| x \|_2^4$. By keeping track of k such counters $X_1, X_2, \ldots, X_k$ each using independent random hash functions h_i and averaging the X_i^2, we obtain an unbiased estimator with smaller variance. Standard tools like Chebyshev's inequality then imply that if $k = O(1/\varepsilon^2)$ then the average of the X_i^2 will be within $\varepsilon \| x \|_2^2$ of $\| x \|_2^2$ with large constant probability. Note that this is a linear sketch using a $k \times n$ matrix A, where $A_{i,j} = h_i(j)/\sqrt{k}$ and our estimate of $\| x \|_2^2$ is $\| Ax \|_2^2$.

PSEUDORANDOMNESS

One caveat in many of the algorithms presented above is our assumption that the hash functions used be random. There are t^n functions mapping $[n]$ to $[t]$, and thus a random such function requires at least $n \log_2 t$ bits to store. In applications where we care about small-memory streaming algorithms, n is large, and thus even if we find an algorithm using sublinear space it would then not be acceptable to use an additional n bits of space or more to store the hash function needed by the algorithm.

The above consideration thus pushes streaming algorithm designers to look for hash functions, which are not actually fully random, but only "random enough" to ensure that the algorithms in which they are being used will still perform correctly. At the highest level there are two directions one can then take to find such hash functions. One is to make some complexity theoretic assumptions, for example to assume that no efficient algorithm exists for some concrete computational problem, then to construct hash functions that can be proven sufficient based on the assumption made. The other direction is to construct hash functions that are provably sufficiently random without making any such assumptions. This latter direction is for obvious reasons typically harder to implement but is possible in certain applications, such as for all the problems mentioned above. This area of constructing objects (such as functions) that look "random enough" for various computational tasks is known as pseudorandomness, and the interested readers may wish to read further [26].

References

[1] Arge, L. External memory data structures. In *Handbook of Massive Data Sets*. Kluwer, 2002.

[2] Vitter, J.S. Algorithms and data structures for external memory. *Foundations and Trends in Theoretical Computer Science* 2, 4 (2006), 305–474.

[3] Frigo, M. Leiserson, C.E., Prokop, H., and Ramachandran, S. Cache-oblivious algorithms. *ACM Transactions on Algorithms* 8, 1 (2012), 4.

[4] Prokop, H. Cache-oblivious algorithms. Master's thesis. Massachusetts Institute of Technology, 1999.

[5] Goldreich, O. Combinatorial property testing (a survey). *Electronic Colloquium on Computational Complexity (ECCC)* 4, 56 (1997).

[6] Borthakur, D. The Hadoop distributed file system: Architecture and design. http://hadoop.apache.org/common/docs/r0.17.2/hdfs_design.html (last accessed June 25, 2012).

[7] Dean, J. and Ghemawat, J. MapReduce: Simplified data processing on large clusters. *Communications of the ACM* 51, 1 (2008), 107–113.

[8] Cormode, G. Sketch techniques for massive data. In *Synposes for Massive Data: Samples, Histograms, Wavelets and Sketches*, eds.Graham Cormode, Minos Garofalakis, Peter Haas, and Chris Jermaine, Foundations and Trends in Databases, vol. 4, no. 1-3. NOW publishers, 2011.

[9] Muthukrishnan, S. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science* 1, 2 (2005), 117–236.

[10] Chakrabarti, A. Data stream algorithms. Lecture Notes. http://www.cs.dartmouth.edu/~ac/Teach/CS49-Fall11/

[11] Indyk, P. Sketching, streaming and sub-linear space algorithms. Lecture Notes. http://stellar.mit.edu/S/course/6/fa07/6.895/

[12] McGregor, A. More advanced algorithms. Lecture Notes. http://people.cs.umass.edu/~mcgregor/courses/CS711S12/index.html

[13] Morris, R. Counting large numbers of events in small registers. *Communications of the ACM* 21, 10 (1978), 840-842.

[14] Flajolet, P. Approximate counting: A detailed analysis. *BIT* 25, 1 (1985), 113–134.

[15] Boyer, R. S. and Moore, J. S. MJRTY—A fast majority vote algorithm. In *Automated Reasoning: Essays in Honor of Woody Bledsoe,* ed. R.S. Boyer. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991, 105-117

[16] Misra, J. and Gries, D. Finding repeated elements. *Science of Computer Programming* 2, 2 (1982), 143–152.

[17] Demaine, E. D., L´opez-Ortiz, A., and Munro, J. I. Frequency estimation of Internet packet streams with limited space. In *Algorithms - ESA 2002*, Lecture Notes in Computer Science, eds. Rolf Möhring and Rajeev Raman. Springer Berlin/Heidelberg, 2002, 348–360.

[18] Karp, R. M., Shenker, S., and Papadimitriou, C. H. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems* 28 (2003), 51–55.

[19] Flajolet, P. and Martin, G.N. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences* 31, 2 (1985), 182–209.

[20] Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences* 58, 1 (1999), 137–147.

[21] Bar-Yossef, Z., Jayram, T. S., Kumar, R., Sivakumar, D., and Trevisan, L. Counting distinct elements in a data stream. In *Proceedings of the Sixth International Workshop on Randomization and Approximation Techniques, (RANDOM)*. Springer-Verlag, London, 2002, 1–10.

[22] Beyer, K. S., Gemulla, R., Haas, P. J., Reinwald, B., and Sismanis, Y. Distinct-value synopses for multiset operations. *Communications of the ACM* 52, 10 (2009), 87–95.

[23] Krishnamurthy, B., Sen, S., Zhang, Y., and Chen, Y. Sketch-based change detection: Methods, evaluation, and applications. In *Proceedings of the Third ACM SIGCOMM Conference on Internet Measurement* (Miami Beach, Oct. 27-29, 2003), 234–247.

[24] Thorup, M. and Zhang, Y. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal of Computing* 41,2 (2102), 293–331.

[25] Alon, N., Gibbons, P. B., Matias, Y., and Szegedy, M. Tracking join and self-join sizes in limited storage. *Journal of Computer and System Sciences* 64, 3 (2002),719–747.

[26] Vadhan, S. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, to appear.

Biography

Jelani Nelson recently finished his Ph.D. in computer science at the Massachusetts Institute of Technology in 2011. He is currently a postdoctoral researcher in theoretical computer science at Princeton University and the Institute for Advanced study and will begin as an assistant professor of computer science at Harvard University in 2013. His research interests are primarily in algorithms for processing massive data.

Big Privacy: Protecting Confidentiality in Big Data

Approaches from computer science and statistical science for assessing and protecting privacy in large, public data sets.

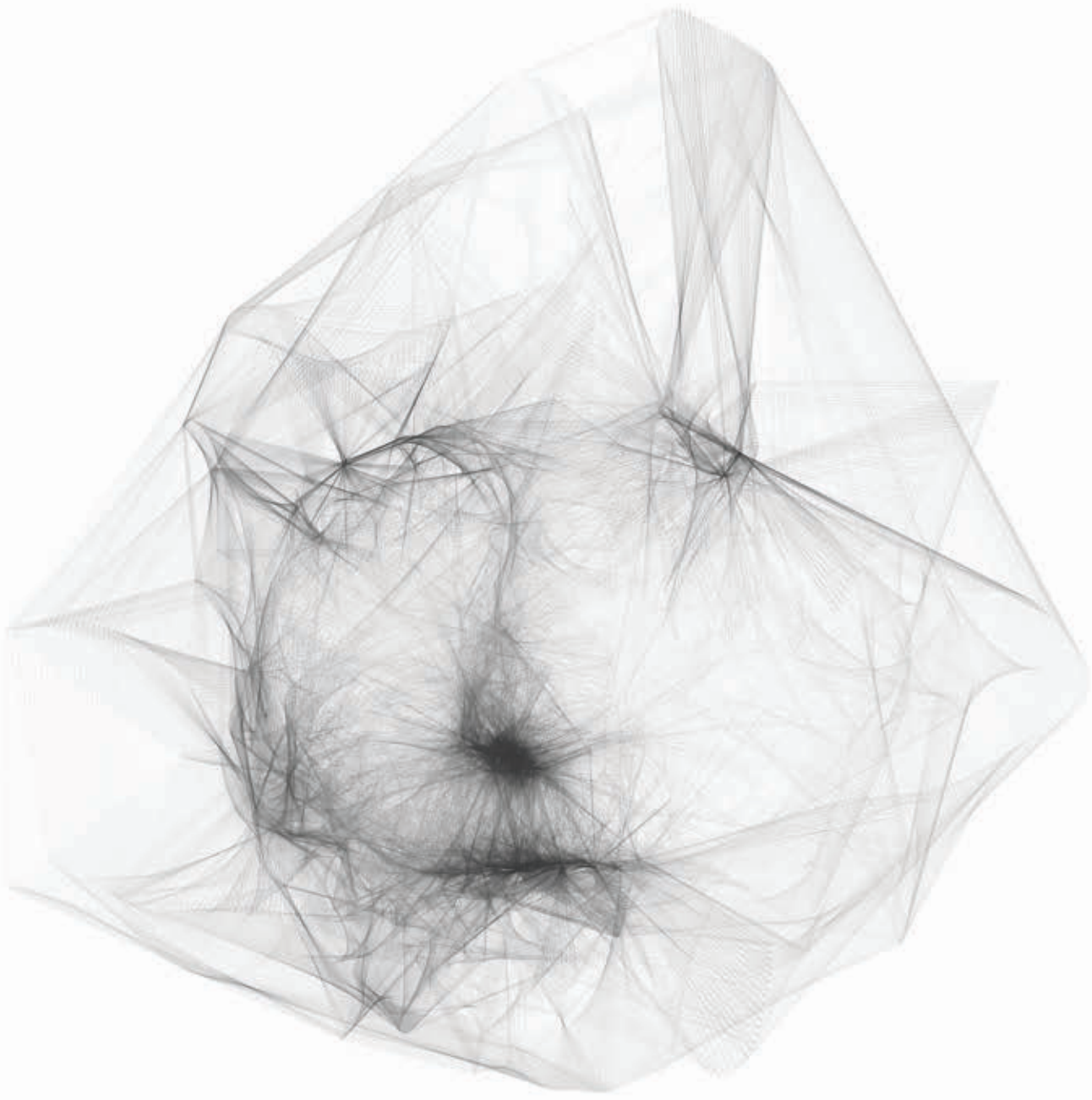
By Ashwin Machanavajjhala and Jerome P. Reiter

DOI: 10.1145/2331042.2331051

A tremendous amount of data about individuals—demographic information, Internet activity, energy usage, communication patterns, and social interactions, to mention a few—are being collected by national statistical agencies, survey organizations, medical centers, and Web and social networking companies. Wide dissemination of such microdata (data at the granularity of individuals) facilitates advances in science and public policy, helps citizens to learn about their societies, and enables students to develop data analysis skills. Often, however, data producers cannot release microdata as collected, because doing so could reveal data subjects' identities or values of sensitive attributes. Failing to protect confidentiality (when promised) is unethical and can cause harm to data subjects and the data provider. Failure to protect individuals' privacy may even be illegal, especially in government and research settings. For example, if one reveals confidential data covered by the U. S. Confidential Information Protection and Statistical Efficiency Act, one is subject to a maximum of $250,000 in fines and a five-year prison term.

At first glance, sharing safe microdata seems a straightforward task: Simply strip unique identifiers like names, addresses, and tax identification numbers before releasing the data. However, anonymizing actions alone may not suffice when other readily available variables, such as aggregated geographic or demographic data, remain on the file. These quasi-identifiers can be used to match units in the released data to other databases. For example, computer scientist Latanya Sweeney showed as part of her Ph.D. thesis that 97 percent of the records in publicly available voter registration lists for Cambridge, MA, could be uniquely identified using birth date and a nine-digit zip code. By matching the information in these lists, she identified Governor William Weld in an anonymized medical database. More recently, the company Netflix released supposedly de-identified data describing more than 480,000 customers' movie viewing habits. However, computer scientists Arvind Narayanan and Vitaly Shmatikov identified several customers by linking to an online movie ratings website, thereby uncovering apparent political preferences and other potentially sensitive information.

Arguably the most sensational privacy breach occurred in 2006 when AOL released 20 million search queries posed by users over a three-month period in order to facilitate research on information retrieval. They knew the information in Web searches contained potentially identifying and sensitive information (including social security and credit card numbers) and hence attempted to anonymize the data by



Generative artwork by Sergio Albiac. Created in Processing.

replacing user identifiers with random numbers. Within a couple of hours of releasing the anonymized data, two reporters from the *New York Times* uncovered the identity of user No. 4417749 based on just her search history: "landscapers in Lilburn, GA," several people with last name Arnold, and "numb fingers." This breach had far reaching consequences. Not only were several high-ranking officials at AOL fired, search companies are now reluctant to release search logs and other personal information. Even researchers are wary of using the now publicly available AOL data. Similar re-identification is possible from social network data, location traces, and power usage patterns.

Although these re-identification exercises were done to illustrate concerns over privacy, one can easily conceive of re-identification attacks for nefarious purposes, especially using large databases on individuals. A nosy neighbor or relative might search through a public database in an attempt to learn sensitive information about someone who they knew participated in a survey or administrative database. A journalist might try to identify politicians or celebrities. Marketers or creditors could mine large databases to identify good, or poor, potential customers. Even more frightening, disgruntled hackers might try to discredit organizations by identifying individuals in public use data.

The threat of breaches, whether perceived or imminent, has serious implications for the practice and scope of data sharing, especially with the availability of massive and richly detailed data. These threats have created a fascinating area of research for aspiring computer scientists, mathematical and statistical scientists, and social scientists. This area of research is sometimes referred to as "privacy preserving methods" (computer science) or "statistical disclosure limitation" (statistical science). It is an area where the research challenges are grand and interdisciplinary, the opportunities for high profile publications and external funding are strong, and the potential to impact the practice of data sharing is genuine and significant.

In this article, we describe some general research themes in this area with the aim of pointing out opportunities for students. Keeping with the area's interdisciplinary nature, we present perspectives from both computer science and statistical science, which are our two home departments. There are many more topics in big privacy that we do not cover for lack of space. These include, for example, systems for collecting data privately, access control in Web and social networking applications, data security and cryptography, and protocols for secure computation. These are equally rich and complementary areas for research that are important for secure and confidential use of big data.

DEFINING AND MEASURING CONFIDENTIALITY RISKS

Both the computer science and statistical science communities have developed a variety of criteria and methods for quantifying confidentiality risks. Indeed, a major thrust of research funded by the U.S National Science Foundation is to integrate these two perspectives, taking the best of what both have to offer. In reviewing some of the risk metrics, we do not attempt to cover all approaches. Rather, we cover a few important ones that we are most familiar with.

In statistical science measures used in practice tend to be informal and heuristic in nature. For example, a common risk heuristic for publishing tabular magnitude data for business establishments (e.g., tables of total payroll within employee size groupings) is that no one establishment should contribute in excess of 20 percent of the cell total, and no cell should comprise less than three establishments. Cells that do not meet these criteria are either suppressed or perturbed. The most general and mathematically formal method of disclosure risk assessment is based on Bayesian probabilities of re-identification, by which we mean posterior probabilities that intruders could learn information about data subjects given the released data and a set of assumptions about the intruder's knowledge and behavior. Data disseminators can compute these measures across a variety of intruder knowledge scenarios to identify particularly risky records and make informed decisions about data release policies in the face of uncertainty (the goal of statistical science in general). Computing these probabilities is computationally demanding and requires innovative methodology, especially for big data.

In computer science early efforts to quantify confidentiality risk were targeted to thwart re-identification attacks by ensuring that no individual's record is unique in the data. This motivated a popular notion of privacy called *K*-anonymity, which requires that no individual's record be distinguishable from at least *K*-1 other records. While this seemingly avoids confidentiality breaches, intruders (especially ones with prior knowledge) still can infer sensitive properties. For instance, suppose a hospital releases *K*-anonymous microdata about patients. You know your neighbor Bob is in the data. If individuals in the anonymous group containing Bob all have either cancer or the flu, and you know that Bob does not have the flu, then you deduce that Bob has cancer. *K*-anonymity has been extended in a number of ways to handle this shortcoming. One example is *L*-diversity, which requires each group of individuals who are indistinguishable via quasi-identifiers (like age, gender, zip code, etc.) not share the same value for sensitive attributes (like disease), but rather have *L* distinct, well-represented (of roughly same proportion) values.

The current state-of-the-art disclosure metric is differential privacy. It eliminates—to a large extent—the confidentiality issues in *K*-anonymity, *L*-diversity, and their extensions. Differential privacy can be best explained using the following opt-in/opt-out analogy. Suppose an agency wants to release microdata, a data subject has two options: Opt out of the microdata so that her privacy is ensured, or opt in and hope that an informed attacker can't infer sensitive information using the released microdata. A mechanism for microdata release is said to guarantee ε-differential privacy if: (i) for every pair of inputs $D1$ and $D2$ that differ in one individual's record, e.g., $D1$ contains record t and $D2$ does not contain t, and (ii) for every microdata release M, then the probability that the mechanism outputs M with input $D1$ should be close to (within an $\exp(\varepsilon)$ factor of) the probability that the mechanism outputs M with input $D2$. In this way, the release mechanism is insensitive to any single individual's presence (opt-in) or absence (opt-out) in the data.

Differential privacy satisfies an important property called composability. If $M1$ and $M2$ are two mechanisms that satisfy differential privacy with parameters $\varepsilon1$ and $\varepsilon2$, releasing the outputs of *M1* and *M2* together satisfies differential privacy with parameter $\varepsilon1+\varepsilon2$. Other known privacy measures (like *K*-anonymity and *L*-diversity) do not satisfy composability. Hence, two privacy preserving releases using other definitions could result in privacy breaches.

Failing to protect confidentiality (when promised) is unethical and can cause harm to data subjects and the data provider. Failure to protect individuals' privacy may even be illegal.

METHODS FOR PROTECTING PUBLIC RELEASE DATA

Like risk measures, both computer scientists and statistical scientists have developed methods of altering or perturbing data before release. Indeed, sometimes very similar methods are developed independently in both communities. Apart from whether a privacy protection method results in low disclosure risk, there are two important considerations when designing a privacy protection method. First, the method should result in outputs that retain useful information about the input. Every privacy protection results in some loss in utility—after all, we are trying to hide individual-specific properties. Hence, prudent data disseminators assess the quality of candidate data releases on representative analyses and

choose policies that offer acceptable trade offs on risk and quality. Second, a privacy protection method should be simulatable: An attacker must be assumed to know the privacy protection method. For instance, a method that reports the age of an individual (x) as [x–10, x+10] is not simulatable, since an attacker who knows this algorithm can deduce the age of the individual to be x.

We next present a few important types of privacy protection methods. We focus here on dissemination of microdata. An alternative undergoing significant research is to design systems that release perturbed answers to statistical queries.

Aggregation. Aggregation reduces disclosure risks by turning atypical records, which generally are most at risk, into typical records. For example, there may be only one person with a particular combination of demographic characteristics in a city, but many people with those characteristics in a state. Releasing data for this person with geography at the city level might have a high disclosure risk, whereas releasing the data at the state level might not. Unfortunately, aggregation makes analysis at finer levels difficult and often impossible, and it creates problems of ecological inferences (relationships seen at aggregated levels do not apply at disaggregated levels).

Suppression. Agencies can delete sensitive values from the released data. They might suppress entire variables or just at-risk data values. Suppression of particular values generally creates data that are missing because of their actual values, which are difficult to analyze properly. For example, if incomes are deleted because they are large, estimates of the income distribution based on the released data is biased low.

Data swapping. Agencies can swap data values for selected records—such as switching values of age, race, and sex for at-risk records with those for other records—to discourage users from matching, since matches could be based on incorrect data. Swapping is used extensively by government agencies. It is generally presumed that swapping fractions are low—agencies do not reveal the rates to the public, so that these algorithms are not simulatable—because swapping at high levels destroys relationships involving the swapped and unswapped variables.

The threat of breaches, whether perceived or imminent, has serious implications for the practice and scope of data sharing, especially with the availability of massive and richly detailed data.

Adding random noise. Agencies can protect numerical data by adding some randomly selected amount—e.g., a random draw from a normal distribution with mean equal to zero—to the observed values (or to answers to statistical queries). Adding noise can reduce the possibilities of accurate matching on the perturbed data and distort the values of sensitive variables. The degree of confidentiality protection depends on the nature of the noise distribution; e.g., a large variance provides greater protection. However, adding noise with large variance introduces measurement error that stretches marginal distributions and attenuates regression coefficients. When perturbing query answers, adding noise from a heavy tailed distribution, like a Laplace distribution, can satisfy differential privacy.

Synthetic data. The basic idea of synthetic data is to replace original data values at high risk of disclosure with values simulated from probability distributions. These distributions are specified to reproduce as many of the relationships in the original data as possible. Synthetic data approaches come in two flavors: partial and full synthesis. Partially synthetic data comprise the units originally surveyed with some subset of collected values replaced with simulated values. For example, the agency might simulate sensitive or identifying variables for units in the sample with rare combinations of demographic characteristics; or, the agency might replace all data for sensitive variables. Fully synthetic data comprise an entirely simulated data set; the originally sampled units are not on the file. Synthetic data can offer provable privacy with high quality. For example, the U.S. Census Bureau releases statistics about individuals' commute patterns (http://onthemap.ces.census.gov/) using a synthetic data generation technique that guarantees a variant of differential privacy.

RESEARCH CHALLENGES

While recent research has shed much light on formal disclosure metrics and provably private methods that provide useful statistical information, there are many intriguing research challenges in this area. For instance, most work on privacy considers data in which each record corresponds to a unique individual, and records are typically considered independent. One important problem is ensuring the privacy of linked and relational data, e.g. social networks in which people are linked to other people. Reasoning about privacy in such data is tricky since information about individuals may be leaked through links to other individuals. Another interesting problem is releasing sequential releases of data over time. Attackers may link individuals across releases and infer additional sensitive information that they could not have from a single release. Finally, as data becomes extremely highly dimensional, we need techniques that protect privacy while guaranteeing utility. Understanding theoretical trade-offs between privacy and utility is another important open area for research.

Biographies

Ashwin Machanavajjhala is an assistant professor of computer science at Duke University. His primary research interests lie in data privacy, systems for massive data analytics, and statistical methods for data integration. Previously, he was a senior research scientist at Yahoo! Research, received an M.S. and Ph.D. from Cornell University, and a B.Tech. from IIT-Madras.

Jerome P. Reiter is the Mrs. Alexander Hehmeyer Associate Professor of Statistical Science at Duke University. His primary research interests include methods for protecting confidentiality of data, handling missing values in statistical analyses, and making inferences in complex surveys. He is the recipient of the Alumni Distinguished Undergraduate Teaching Award from Duke University. He received a Ph.D. from Harvard University and an B.S. from Duke University.

Taming Big Probability Distributions

New algorithms for estimating parameters of distributions over big domains need significantly fewer samples.

By Ronitt Rubinfeld

DOI: 10.1145/2331042.2331052

These days, it seems that we are constantly bombarded by discussions of big data and our lack of tools for processing such vast quantities of information. An important class of big data is most naturally viewed as samples from a probability distribution over a very large domain. Such data occurs in almost every setting imaginable—samples from financial transactions, seismic measurements, neurobiological data, sensor nets, and network traffic records.

In many cases, there is no explicit description of the distribution—just samples. In order to effectively make use of such data, one must estimate natural parameters and understand basic properties of the underlying probability distribution. Typical questions include: How many distinct elements have non-zero probability in the distribution? Is the distribution uniform, normal, or Zipfian? Is a joint distribution independent? What is the entropy of the distribution? All of these questions can be answered fairly well using classical techniques in a relatively straightforward manner.

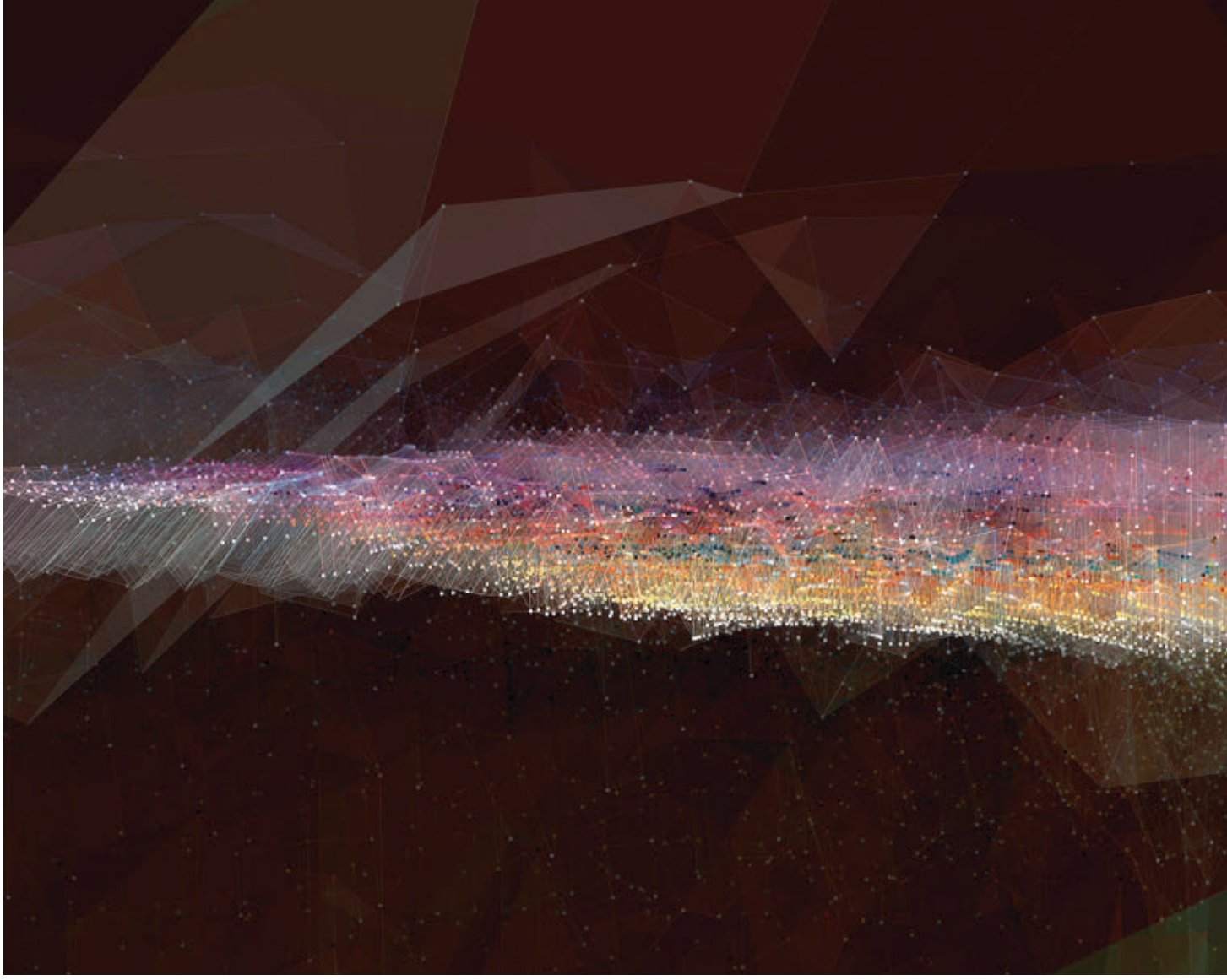
However, unless assumptions are made on the distribution—such as the distribution is Gaussian or has certain smoothness properties—such techniques use a number of samples that scales at least linearly with the size of the domain of the distributions. Unfortunately, the challenge of big data is that the sizes of the domains of the distributions are immense, resulting in a very large number of samples. Thus, we are left with an unacceptably slow algorithm.

The good news is that there has been exciting progress in the development of sublinear, sample algorithmic tools for such problems. In this article we describe two recent results that highlight the main ideas contributing to this progress: The first on testing the similarity of distributions, and the second on estimating the entropy of a distribution. We assume that all of our probability distributions are over a finite domain D of size n, but (unless otherwise noted) we do not assume anything else about the distribution.

CLOSENESS TO ANOTHER DISTRIBUTION

How can we tell whether two distributions are the same? There are many variants of this question that have been considered, but let's begin with a simpler question, motivated by the following: How many years of lottery results would it take for us to believe in its fairness? In our setting, given samples of a single distribution p, how many samples do we need to determine whether p is the uniform distribution?

To properly formalize this problem, we need to allow some form of approximation, since p could be arbitrarily close to uniform, though not exactly uniform, and no algorithm that takes finite samples would have enough information to detect this. We will use the property testing framework: What we ask of our testing algorithm is to "pass" distributions that are uniform and to "fail" distributions that are far from uniform. We next need to decide what we mean by "far"—many distance



measures are in common use, but for this article we will use the L_1 distance between two probability distributions p and q, defined as:

$$||p, q||_1 = \sum_{x \in D} |p(x) - q(x)|$$

For $0 < \epsilon < 1$, we say that p and q are ϵ-close with respect to the L_1 distance if $||p, q||_1 \leq \epsilon$. Denote by U_D the uniform distribution on D. Then, on input parameter $0 < \epsilon < 1$, the goal of the testing algorithm will be to pass p if it is uniform and fail if $||p, U_D||_1 \geq \epsilon$. If p is in the middle—not uniform, but not far from uniform—then either "pass" or "fail" is an allowable, and not unreasonable, answer.

One natural way to solve this problem, which we will refer to as the "naive algorithm," is to take enough samples of p so that one can get a good estimate of the probability $p(x)$ for each domain element x. It is easy to see that there are distributions for which such a scheme would require at least linear in $|D| = n$ samples.

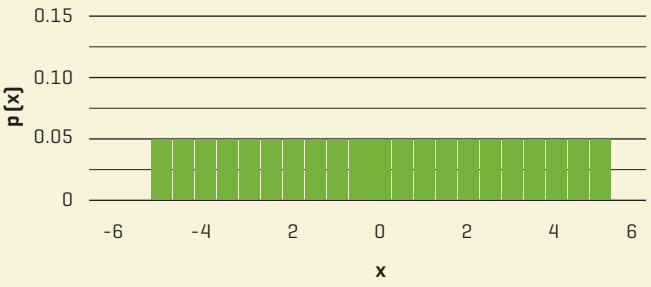
However, there is a much more efficient $O(\sqrt{n}/\epsilon^4)$ sample algorithm, based on an idea of Goldreich and Ron [1]. (See also Paninski for a more recent algorithm, which requires only $O(\sqrt{n}/\epsilon^2)$ samples [2]). This algorithm does not attempt to learn any of the probabilities of specific domain elements according to the distribution p. Instead, the algorithm counts collisions—the number of times that samples coincidentally fall on the same domain element.

Slightly more specifically, for a set of k samples $x_1, \ldots, x_k$, let $i, j \in [1..k]$ be two indices of samples. Then we say that i and j collide if they output the same domain element, i.e., $x_i = x_j$. Note that the probability that i and j collide does not depend on i, j, and is an important parameter of the distribution p, which we will refer to as the collision probability C_p. Consider the fraction of pairs i, j in the actual sample set that collide, it is easy to see that its expectation is exactly C_p. A simple calculation shows that C_p is minimized when p is the uniform distribution, in which case $C_{U_D} = 1/n$. One can show that when p is far from the uniform distribution, then C_p is very different than $1/n$. So now it should be clear that the collision probability C_p is a useful statistic to estimate. The especially convenient property of C_p is that it is a statistic that one can estimate with surprisingly few samples, since k samples yield $\binom{k}{2}$ pairs from which to estimate the collision probability.

Although these pairs are not independent, Goldreich and Ron have shown that they have nice properties [1], yielding an algorithm with sample complexity $O(\sqrt{n} \log n/\epsilon^4)$, which estimates the collision probability. This solves our uniform distribution, testing problem. In fact, one cannot do much better in terms of n. It is easy to see that generalized collision sta-

Generative artwork by Gwen Vanhee. Created in HTML5 and Javascript.

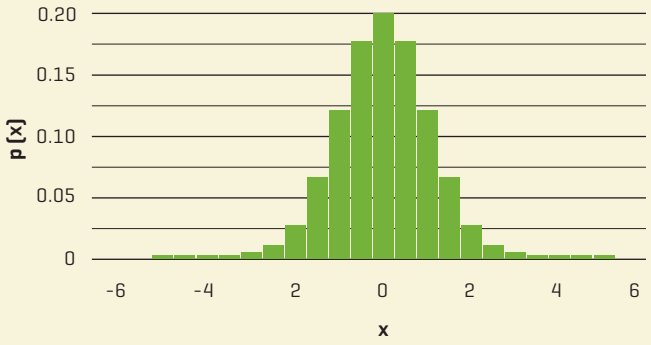
Figure 1. Distributions



Uniform Distribution

In the uniform distribution, n equally spaced domain values are equally likely to be observed.

$$p(x) = \frac{1}{n}$$

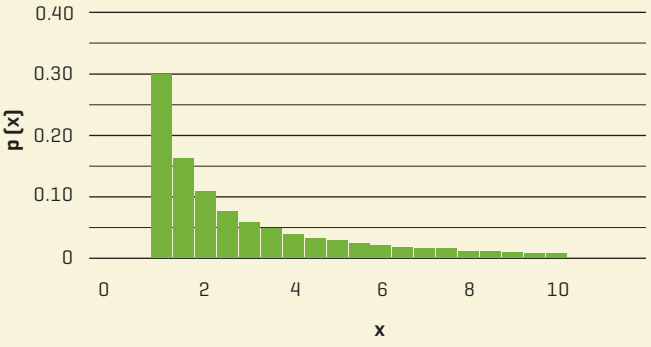


Gaussian Distribution

The Gaussian (or normal) distribution is often described as a bell-curve due to the shape of $p(x)$, defined as

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

where μ is the mean and σ is the standard deviation. This distribution is one of the most commonly used in statistics.



Zipf Distribution

The Zipf distribution has practical applications in linguistics, finance, and modeling rare phenomena. The probability function is

$$p(x) = \frac{x^{-(p+1)}}{\zeta(p+1)}$$

where p is a positive constant and ζ denotes Riemann's zeta function.

tistics (including l-way collisions for all l) are the only information that an algorithm can use to determine whether a distribution is uniform. More than that, generalized collision statistics are the only information that an algorithm can use for determining whether p has any of a large class of properties—namely those properties that are independent of the names of the domain elements, the so called "symmetric properties." Using this observation, one only needs to find a distribution which has no collisions at all until at least $\Omega(\sqrt{n})$ samples are taken, but on the other hand is very far from uniform. Such a distribution can be constructed by taking the uniform distribution over a random subset S of half of the domain [3].

What if we want to know whether p is the standard normal distribution? More generally, what if we want to know whether p is the same as another distribution q, where q is known explicitly by the algorithm—that is, $q(i)$ for any domain element i can be determined essentially for free? For example, this would be the case if q is a Gaussian, Zipfian, or exponential distribution with known parameters of expectation and variance. Batu et. al. give an algorithm which solves this problem for any q using $O(\sqrt{n}\log n)$ samples from p to perform $O(\log n)$ collision probability estimations over carefully chosen subdomains of D [4].

Finally, what if both p and q are unknown and the only way we can find out anything about them is to view samples? Up until now, though the analyses are nontrivial, the sample complexities are not terribly surprising to those that have studied "birthday paradox"-type analyses of collisions and hashing.

Here the story takes a fascinating turn—in this case, the complexity of the problem is quite different from $n^{1/2}$. Why? The reason is that there may be elements, which are quite "heavy" on which p, q are identical, but whose collision statistics are so big that they hide what is happening on the rest of the domain. Formalizing this lower bound reasoning was quite challenging and eluded researchers for several years, but in 2008, Paul Valiant was able to prove that $\Omega(n(2/3))$ samples are required for this task [5, 6]. The $O(n^{2/3}\log n\ \mathrm{poly}(1/\epsilon))$ algorithm, proposed in 2000 by Batu et. al. [3, 7], distinguishes pairs of distributions p and q that are identical from those pairs p, q which are ϵ-far as follows:

1. First, find the "heavy" domain elements, or those that have probability at least $1/n^{2/3}$. Using this definition of heavy, this set will contain at most $n^{2/3}$ domain elements, since the sum of the probabilities over all domain elements must be 1. The naive algorithm, which takes $O(n^{2/3}\log n\ \mathrm{poly}(1/\epsilon))$ samples of p and q and estimates their probabilities on each of the heavy elements, is likely to give very good estimates of their probabilities.

2. If p and q seem similar, then check that they are also similar on the rest of the domain by sieving out the heavy elements and using a test that is based on estimating collision probabilities—this time, not just collision probabilities of p and q, but also collisions between samples of p and q. Since none of the remaining domain elements are heavy, one can show that $O(n^{2/3}\log n\ \mathrm{poly}(1/\epsilon))$ samples suffice for this latter task as well.

Such ideas have had further applications; they have been used to design algorithms for testing whether a distribution is monotone increasing or bimodal over the domain [8], or whether a joint distribution is independent [4]. The sample complexity of many of these problems have been

investigated over distance norms other than L_1 [1, 9, 10], but in many cases the same collision-based ideas apply. There is much further work on testing similarity [9, 11, 12].

A tolerant test, given parameters $\epsilon_1 < \epsilon_2$, passes distributions p that are ϵ_1-close to q and fails distributions p that are not even ϵ_2-close to q. Unfortunately, even for the problem of testing whether p is the uniform distribution, Valiant has shown for large enough values of ϵ_1, the task becomes much harder, requiring at least $n^{1-o(1)}$ samples [5, 6] (it is known that $n/(\epsilon^2 \log n)$ samples are sufficient [13]). Still, some tiny amount of tolerance can be squeezed out of the more efficient algorithms. It is an interesting direction to see how much more can be achieved.

One method for overcoming the lower bound is to find specialized algorithms for distributions that have convenient properties, such as that of being continuous, monotone, or normal.

ESTIMATING THE ENTROPY OF A DISTRIBUTION

The entropy of a distribution is an important measure of the randomness of the distribution and the compressibility of the data produced by that distribution. Thus, entropy plays a central role in statistics, information theory, data compression and machine learning. The entropy of distribution p over a discrete domain D is defined as:

$$H(p) \equiv \sum_{x \in D} -p(x) \log p(x)$$

The problems of estimating the entropy of a distribution and the closely related measures of KL-divergence and mutual information have received much attention because of their usefulness in analyzing data in machine learning and the natural sciences [14, 15]. How many samples of a distribution are required in order to get a good estimate of the entropy?

First, we must determine what we mean by a good estimate. Let's begin with the setting in which one would like an additive estimate of the entropy—that is, the algorithm should output a number y such that,

$$H(p) - \epsilon < y < H(p) + \epsilon$$

for given input parameter ϵ. In such a case, one very common estimator for the entropy, sometimes referred to as the "plug-in estimate," is based on a best guess of what the entire distribution p looks like. That is, if $\hat{p}(x)$ is the fraction of times that a domain element x is seen in a sample, then the estimate of the entropy is given by the entropy of $\hat{p}$, namely,

$$\hat{H} = \sum_{x \in D} -\hat{p}(x) \log \hat{p}(x)$$

It is easy to see that for this estimate to have good quality, one must take enough samples to get a good estimate the value of $p(x)$ for most x, which in general is only guaranteed to happen when the number of samples is at least linear in n.

Other commonly used estimators, such as the Miller-Madow corrected estimator and the jackknifed naive estimator, are similar in that they require a linear number of samples; this is because they do not adequately account for the contribution to the entropy from unseen domain elements.

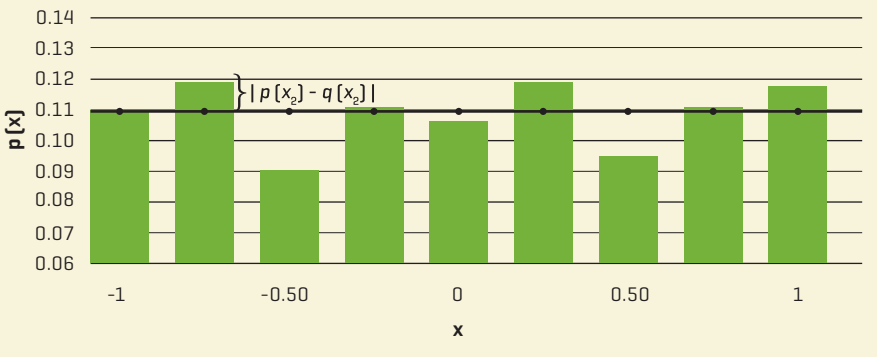
A major barrier was broken when Paninski gave a non-constructive proof of the existence of an algorithm for estimating the entropy with a number of samples that is sublinear in the domain size [16]. Recently, very exciting results of Gregory and Paul Valiant settled the longstanding open question of the complexity of this problem [13, 17, 18, 19]. On one hand, they give an $O(n/(\epsilon^2 \log n))$-sample algorithm for estimating the entropy of a distribution over a domain of size n to within an additive error of ϵ. On the other hand, they show that this task is not doable with fewer than $\Omega(n/(\epsilon \log n))$ samples (improving on previous results [6, 20]). In order to show the first result, they show linear programming can be used to find a distribution that is expected to have a similar set of collision statistics to p. Then, this new distribution, though likely to be very different from p in L_1-distance, is similar to p in at least one important way—it is likely to have similar entropy. To show the second result, their lower bound constructs two families of distributions that have very similar collision behaviors, yet are very different in terms of support size or entropy.

Let us now turn to the setting in which a much weaker multiplicative estimate of the entropy is sufficient. That is, on input parameter $\gamma > 1$, the algorithm should output a number y such that,

$$H(p)/\gamma < y < \gamma H(p)$$

In this case, it suffices to have a number of samples that is dramati-

Figure 2. The L_1 distance between p and q is the sum of this quantity for each x_i in the domain



cally smaller that the domain size—algorithms that use only $O(n^{(1+o(1))/\gamma^2})$ samples can be achieved [21]. (This statement is a minor simplification, in fact, it only holds for distributions that have at least constant minimum entropy.) Furthermore, it is known that $\Omega(n^{(1/\gamma^2)})$ samples are necessary for this task [5, 6]. Just to be concrete, this means that one can approximate the entropy to within a multiplicative factor of two using only slightly worse than $O(n^{1/4})$ samples, which is significantly less than what is required for the additive error case. The description of the algorithm is very simple—it uses the plug-in estimate for any domain elements that have high probability, and assumes that the distribution is uniform over the rest of the domain.

Similar results can be achieved for another closely related and well-studied task—estimating the support size of a distribution. The question of estimating the support size of a distribution has been considered since the early 1940s, beginning with Fisher and Corbet in their estimations of the number of butterflies in a region (for a large number of other reasons to consider this problem, see this archived bibliography [22]). In recent research, $\theta(n/\log n)$ samples are shown to be both necessary and sufficient to achieve an additive estimate of the support size [13].

SUMMARY AND FINAL WORDS

The study of big data has led the computer science community to make very exciting progress on classical statistical problems. Still, in some settings, the minimum number of data points required to obtain an acceptable answer is too large to be practical.

One method for overcoming the lower bound is to find specialized algorithms for distributions that have convenient properties, such as that of being continuous, monotone, or normal. Such assumptions often lead to dramatically better sample complexities.

A second method for overcoming the lower bound is to note that in some settings it is natural to assume the algorithm has access to other information in addition to random samples, such as the ability to quickly determine $p(x)$ for any domain element x. For example, when determining distributional properties of data that is stored in sorted order, one can still take a random sample of the data; but it is also easy to determine the number of collisions, or the number of times a given element appears in the data set. This information can be used to create algorithms that are sublinear in the number required samples, significantly reducing the computational complexity.

The entropy of a distribution is an important measure of the randomness of the distribution and the compressibility of the data produced by that distribution.

Such new approaches in statistical modeling can lead to significantly faster algorithms for handling distributions on increasingly large domains. Moving forward, it is crucial to take advantage of these more powerful statistical models and algorithm—these are the tools we need in order to tame big data.

Acknowledgments

The author would like to thank Reut Levi and Ning Xie for their very helpful comments on this article.

References

[1] Goldreich, O. and Ron, D. On testing expansion in bounded-degree graphs. *Electronic Colloqium on Computational Complexity* 7, 20 (2000).

[2] Paninski, L. Testing for uniformity given very sparsely-sampled discrete data. *IEEE Transactions on Information Theory* 54, 10 (2008), 4750–4755.

[3] Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., and White, P. Testing that distributions are close. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science* (Redondo Beach, Nov. 12-14). IEEE Computer Society, Los Alamitos, CA, 2000, 259–269.

[4] Batu, T., Fortnow, L., Fischer, E., Kumar, R., Rubinfeld, R., and White, P. Testing random variables for independence and identity. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science* (Las Vegas, Oct. 14-17). IEEE Computer Society, Los Alamitos, CA, 2001, 442–451.

[5] Valiant, P. Testing symmetric properties of distributions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing* (Victoria, Canada, May 17-20). ACM Press, New York, 2008, 383–392.

[6] Valiant, P. Testing symmetric properties of distributions. *SIAM Journal on Computing* 40, 6 (2011), 1927–1968.

[7] Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., and White, P. Testing closeness of discrete distributions. CoRR, abs/1009.5397, 2010. (This is a long version of [3].)

[8] Batu, T., Kumar, R., and Rubinfeld, R. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing* (Chicago, June 13-15). ACM Press, New York, 2004, 381–390.

[9] Guha, S., McGregor, A., and Venkatasubramanian, S. Sub-linear estimation of entropy and information distances. *ACM Transactions on Algorithms* 5, 4 (Oct. 2009).

[10] Ba, K. D., Nguyen, H. L. , Nguyen, H. N. , and Rubinfeld, R. Sublinear time algorithms for earth mover's distance. *Theory of Computing Systems* 48, 2 (2011), 428–442.

[11] Acharya, J., Das, H., Jafarpour, A., Orlitsky, A., and Pan, S. Competitive closeness testing. *Journal of Machine Learning Research, Proceedings Track* 19 (2011), 47–68.

[12] Levi,R., Ron, D., and Rubinfeld, R. Testing properties of collections of distributions. In *Proceedings of Second Symposium on Innovations in Computer Science* (Beijing, Jan. 7-9). Tsinghua University Press, 179–194, 2011. See also ECCC TR10-157.

[13] Valiant, G. and Valiant, P. The power of linear estimators. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science* (Palm Springs, Oct. 23-25). IEEE Computer Society, Los Alamitos, 2011, 403–412.

[14] Ma, S. Calculation of entropy from data of motion. *Journal of Statistical Physics* 26, 2 (1981), 221-240.

[15] Strong, S. P., Koberle, R., de Ruyter van Steveninck, R. R. , and Bialek, W. Entropy and information in neural spike trains. *Physical Review Letters* 80, 1 (1998), 197–200.

[16] Paninski, L. Estimating entropy on m bins given fewer than m samples. *IEEE Transactions on Information Theory* 50, 9 (2004), 2200–2203.

[17] Valiant, G. and Valiant, P. A CLT and tight lower bounds for estimating entropy. *Technical Report TR10-179*. Electronic Colloquium on Computational Complexity (ECCC), 2010.

[18] Valiant, G. and Valiant, P. Estimating the unseen: A sublinear-sample canonical estimator of distributions. *Technical Report TR10-180*. Electronic Colloquium on Computational Complexity (ECCC), 2010.

[19] Valiant, G. and Valiant, P. Estimating the unseen: An n= log(n)-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing* (San Jose, June 6-8). ACM Press, New York, 2011, 685–694.

[20] Raskhodnikova, S., Ron, D., Shpilka, A., and Smith, A. Strong lower bonds for approximating distributions support size and the distinct elements problem. *SIAM Journal on Computing* 39, 3 (2009), 813–842.

[21] Batu, T., Dasgupta, S., Kumar, R., and Rubinfeld, R. The complexity of approximating the entropy. *SIAM Journal on Computing* 35, 1 (2005), 132–150.

[22] Bunge, J. Estimating the Number of Classes in a Population. Archive Bibliography. Cornell University. http://www.stat.cornell.edu/~bunge/bibliography .html

Biography

Ronitt Rubinfeld is a professor of electrical engineering and computer science at MIT, as well as a professor of computer science at Tel Aviv University. Her research for the past 20 years has focused on sublinear time algorithms for all kinds of big data sets.

Computing Reviews is on the move

Our new URL is

ComputingReviews.com

CR

COMPUTING REVIEWS

A daily snapshot of what is new and hot in computing

Designing Good MapReduce Algorithms

An introduction to designing algorithms for the MapReduce framework for parallel processing of big data.

By Jeffrey D. Ullman

DOI: 10.1145/2331042.2331053

If you are familiar with "big data," you are probably familiar with the MapReduce approach to implementing parallelism on computing clusters [1]. A cluster consists of many compute nodes, which are processors with their associated memory and disks. The compute nodes are connected by Ethernet or switches so they can pass data from node to node.

Like any other programming model, MapReduce needs an algorithm-design theory. The theory is not just the theory of parallel algorithms—MapReduce requires we coordinate parallel processes in a very specific way. A MapReduce job consists of two functions written by the programmer, plus some magic that happens in the middle:

1. The *Map function* turns each input element into zero or more key-value pairs. A "key" in this sense is not unique, and it is in fact important that many pairs with a given key are generated as the Map function is applied to all the input elements.

2. The system sorts the key-value pairs by key, and for each key creates a pair consisting of the key itself and a list of all the values associated with that key.

3. The *Reduce function* is applied, for each key, to its associated list of values. The result of that application is a pair consisting of the key and whatever is produced by the Reduce function applied to the list of values. The output of the entire MapReduce job is what results from the application of the Reduce function to each key and its list.

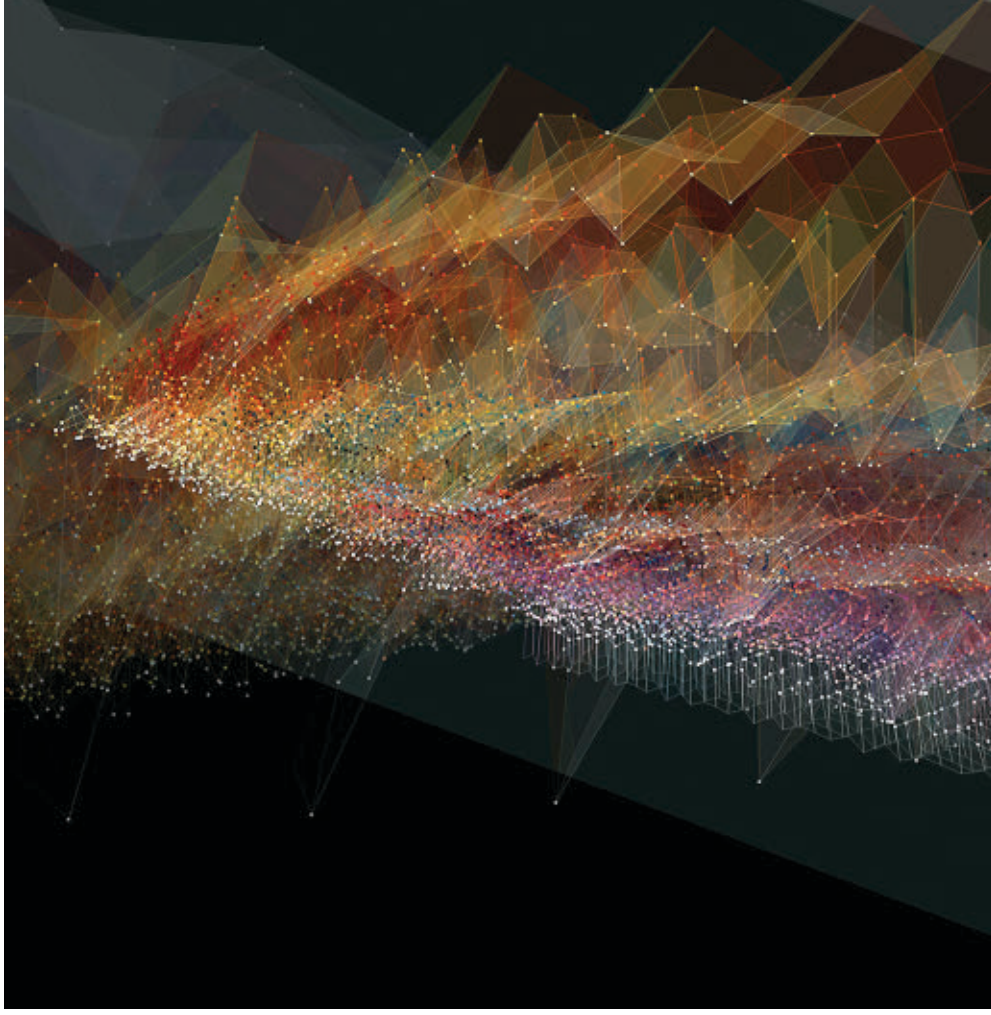
When we execute a MapReduce job on a system like Hadoop [2], some number of Map tasks and some number of Reduce tasks are created. Each Map task is responsible for applying the Map function to some subset of the input elements, and each Reduce task is responsible for applying the Reduce function to some number of keys and their associated lists of values. The arrangement of tasks and the key-value pairs that communicate between them is suggested in Figure. 1. Since the Map tasks can be executed in parallel and the Reduce tasks can be executed in parallel, we can obtain an almost unlimited degree of parallelism—provided there are many compute nodes for executing the tasks, there are many keys, and no one key has an unusually long list of values

A very important feature of the MapReduce form of parallelism is that tasks have the blocking property [3]; that is, no Map or Reduce task delivers any output until it has finished all its work. As a result, if a hardware or software failure occurs in the middle of a MapReduce job, the system has only to restart the Map or Reduce tasks that were located at the failed compute node. The blocking property of tasks is essential to avoid restart of a job whenever there is a failure of any kind. Since MapReduce is often used for jobs that require hours on thousands of compute nodes, the probability of at least one failure is high, and without the blocking property large jobs would never finish.

There is much more to the technology of MapReduce. You may wish to consult, a free online text that covers MapReduce and a number of its applications [4].

EFFICIENT MAPREDUCE ALGORITHMS

A given problem often can be solved by many different MapReduce algorithms. We shall start with a real example of what can go wrong and then examine a model that lets us talk about the important tradeoff between the communication (from Map to Re-



duce tasks) and computation (at the Reduce tasks).

Reducers. It is convenient to have a term to refer to the application of the Reduce function to a single key and its list. We call this application a reducer. The input size for a reducer is the length of the list. Notice that reducers are not exactly the same as Reduce tasks. Typically a Reduce task is given many keys and their lists, and thus executes the work of many "reducers." However, there could be one Reduce task per reducer, and in fact, there could even be one compute node per reducer if we wanted to squeeze the absolute maximum degree of parallelism out of an algorithm.

Analogously, we can think of a mapper as the application of the Map function to a single input element. Normally, mappers are grouped into Map tasks, and each Map task is responsible for many mappers. It is more common for us to be able to gain efficiency by redesigning the nature of the reducers than by redesigning the mappers, so we shall be concentrating on the reducers in this article.

Communication and computation costs. There are three principal sources of cost when you run a MapReduce job:

1. There is a map cost of executing the mappers. Normally, the input is a file distributed over many compute nodes, and each Map task executes at the same compute node that holds the input elements to which it is applied. This cost is essentially fixed, and consists of the computation cost of executing each mapper.

2. Each key-value pair must be transmitted to the location of the Reduce task that will execute the reducer for that key. While by coincidence this Reduce task may be located at the same compute node as the Map task that generated that key-value pair, we shall assume for convenience each key-value pair is shipped across the network that connects the compute nodes. The communication cost, or cost of moving the data from Map tasks to Reduce tasks, is thus proportional to the total number of key-value pairs generated by the mappers.

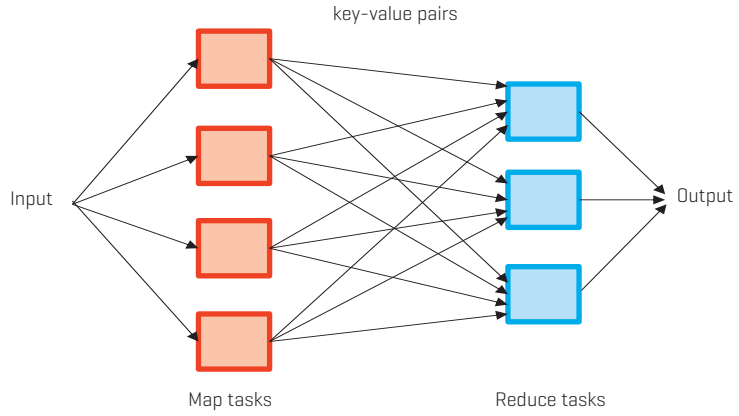
3. Each reducer must execute at the compute node to which its key is assigned. The computation cost for an algorithm is the sum of the time needed to execute each reducer.

This distinction between communication cost and computation cost appears to ignore the computation needed to execute the mappers. However, commonly, this cost is proportional to the number of key-value pairs generated, and thus can be included in the communication cost. We shall therefore not discuss the cost of executing the mappers further.

It may not be obvious, but communication cost often dominates the computation cost. Typically, compute nodes are connected by gigabit Ethernet. That seems fast if you are downloading a song, but when you have to move a terabyte, it will take at least three hours across a gigabit Ethernet connection.

Skew and wall-clock time. We focus on communication and computation cost because in a public cloud, like

Figure 1: The structure of a MapReduce job.



Generative artwork by Gwen Vanhee. Created in HTML5 and Javascript.

Amazon's EC2, that is what you pay for [5]. You pay by the gigabyte for moving data across the network, and you rent compute nodes by the hour. However, in addition to wanting to minimize what we pay, we also want our job to finish soon. Thus, the total elapsed time before finishing the MapReduce job is also important.

As long as no mapper or reducer has too large an input size, we can divide them among as many compute nodes as we have access to, and thus have a wall-clock finishing time that is roughly the total time of the computation and communication, divided by the number of compute nodes. However, if we are not careful, or the data has a bad form, then we are limited in how fast we can finish by the phenomenon of skew.

The most common form of skew is when the data causes one key K to be produced a significant fraction of the time. If, say, half the key-value pairs generated by the mappers have key K, then the reducer for key K has half of all the data communicated. The computation time of the reducer for K will be at least half of the total computation time; it could be more if the running time of the Reduce function grows faster than linearly in the size of the list. In such a situation, the wall-clock time for finishing cannot be less than half the total computation cost, no matter how many compute nodes we use. From this point onward, we shall assume that skew is not a problem, although there is much evidence that skew does affect the wall-clock time significantly in many cases; see Kwon et al. for example [6].

The grand compromise. For many problems, there is a tradeoff between the input size for the reducers and the replication rate, or number of key-value pairs generated per input element. The smaller the input size, the more parallelism we can have, which leads to a lower wall-clock time. But for problems that are not "embarrassingly parallel," lowering the input size for the reducers means increasing the replication rate and therefore increasing the communication. The more communication, the slower the job runs and the more it costs. Thus, we must find just the right input size to compromise between our desire for low cost and low wall-clock time.

The study of optimal MapReduce algorithms can thus be viewed as the study of the function that gives the least possible replication rate for a given reducer input size. We need to do two things: Prove lower bounds on the replication rate as a function of input size; and discover algorithms whose replication rate matches the lower bound.

AN EXAMPLE OF THE TRADEOFF

To see how the grand compromise works in practice, I am going to tell a story about a real project. At Stanford, I coached several teams in the data-mining project course. One of the teams was looking at medical records for about a million patients, and was trying to discover unknown drug interactions. They were indeed successful not only in verifying known interactions, but in discovering several very suspicious, heretofore unknown, combinations of drugs that have significant side effects.

To find pairs of drugs that had particular side effects, they created a record for each of the 6,500 drugs in the study. That record contained information about the medical history of patients who had taken the drug; these records averaged about a megabyte in length. The records for each pair of drugs needed to be compared in order to determine whether a particular side effect was more common among patients using both drugs than those using only one or neither.

Their initial plan was to use MapReduce with one reducer for each pair of drugs. That is, keys would be ordered lists of two drugs $[i, j]$ with $i < j$, and the associated values would be the records for the two drugs. The Map function would take a drug i with record R and turn it into many key-value pairs. Each of these had a value (i, R), meaning that R was the record for drug i. But the keys were all the lists consisting of i and any other drug j. For each of the 6,500 drugs they therefore created 6,499 key-value pairs—each about a megabyte in size—for a total communication cost of about 20 terabytes. It was no surprise that they were unable to do this MapReduce job, even given the generous allocation of free EC2 service that Amazon had provided for the class to use.

So they needed to make a compromise between their desire to run as many reducers as possible in parallel and their need to keep the communication within bounds. They grouped the drugs into 65 groups, numbered 1 to 65, of 100 drugs each. Instead of keys being sets of two drugs, they used keys sets of two group numbers. The mapper for drug i and record R created 64 key-value pairs. In each, the value was (i, R), as before. The keys were all pairs of two groups, one of which is the group of drug i and the other of which is any other group.

A reducer in the new scheme received a key that is a set of two groups, and a list of 200 elements (i, R), where i is a drug in one of the two groups and R is the patient record for that drug. The reducer compared each element (i_1, R_1) and (i_2, R_2) on its list, provided i_1 and i_2 were drugs in different groups. A small trick that I won't go into was necessary to make sure that drugs in the same group were also compared by exactly one of the reducers.

As a result, every pair of drugs had their records compared exactly once, just as in the original scheme, so the computation cost was essentially the same as before. The input size to a reducer grew by a factor of 100, so the minimum wall-clock time was much greater under the new scheme. However, the replication rate shrunk by a factor of over 100, so the communication was around 200 gigabytes instead of 20 terabytes. Using the new scheme, the various costs balanced well, and the job was able to complete easily.

SOME CONCRETE TRADEOFFS

Now, we are going to see some facts about particular problems and the way reducer input size and replication rate are related for these problems. We shall look at the problem of finding bit strings at Hamming distance 1, and then at the problem of finding triangles in a graph. However, we begin by looking at the tradeoff implied by the previous discussion.

Tradeoff for the medical example. We can generalize the two different strategies we considered as follows. Suppose there are d drugs, and we want to group them into g groups. The record for each drug is then replicated

$g-1$ times, which we'll approximate as g times to simplify the formulas. The input size for each reducer is $2d/g$ records. Conventionally, we use q for the maximum allowable input size for a reducer and r for the replication rate. In this case, we have $r = g$ and $q = 2d/g$, so r as a function of q is

$$r = 2d/q$$

That is, the replication rate is proportional to the number of drugs and inversely proportional to the reducer input size.

As long as g divides d evenly, we can choose any g we like and have an algorithm that solves the problem. We discussed only two cases: $d = g = 6{,}500$ (the original attempt) and $d = 6{,}500$, $g = 65$, which worked. However, if the communication were still too costly at $g = 65$, we could have lowered it further to decrease the replication rate yet again. At some point, the communication cost would cease to be the dominant cost, and we could extract what parallelism remains to keep the wall-clock time as low as possible.

Strings at Hamming distance 1. We are now going to take up a problem that was analyzed in a recent paper on understanding the limits of map-reduce computation [7]. Two bit strings of the same length b are at Hamming distance d if they differ in exactly d corresponding bits. For example, 0011 and 1011 are at Hamming distance 1 because they differ only in the first bit.

For $d = 1$ there is an interesting lower bound on replication rate as a function of q, the maximum number of strings that can be sent to any reducer. For an algorithm to find all pairs of strings at Hamming distance 1 in some input set of bit strings of length b, every pair of bit strings at distance 1 must be covered by some reducer; in the sense that if they exist in the input set, then both strings will be sent to that reducer (perhaps among other reducers). The number of possible inputs is 2^b, and the number of possible outputs—pairs at distance 1— is $(b/2)2^b$. To see why the latter count is correct, notice that each of the 2^b bit strings of length b is at distance 1 from b other strings; those are the strings constructed by flipping exactly one of the b bits. So we would expect $b2^b$ pairs, but that counts each pair twice, once from the point of view of each of the two strings. Thus the correct count of possible outputs is $(b/2)2^b$.

There is a theorem that says among any collection of q bit strings, there are at most $(q/2)\log_2 q$ pairs at distance 1 [7]. We're not going to prove it here, but we'll use it to get an exact lower bound on the replication rate r as a function of q. First, suppose we use p reducers, and the i^{th} reducer has $q_i \leq q$ bit strings that it will receive if they are present in the input. Since all the $(b/2)2^b$ pairs of strings at distance 1 must be covered by some reducer, we know that

$$\sum_{i=1}^{p} (q_i/2)\log_2 q_i \geq (b/2)2^b$$

That is, the sum of the maximum number of outputs that each reducer can cover must be at least the number of outputs.

We are going to replace $\log_2 q_i$ by $\log_2 q$ in the above inequality. Since q is an upper bound on q_i, the inequality must continue to hold; that is

$$\sum_{i=1}^{p} (q_i/2)\log_2 q \geq (b/2)2^b$$

Notice we chose not to replace the factor q_i by q.

The replication rate r is the sum of the number of inputs q_i to each reducer divided by the total number of possible inputs, 2^b, that is, $\Sigma_{(i=1)}^{p} q_i/2^b$. We can manipulate the inequality above so that exactly $\Sigma_{(i=1)}^{p} q_i/2^b$ appears on the left, and everything else is on the right. That gives us

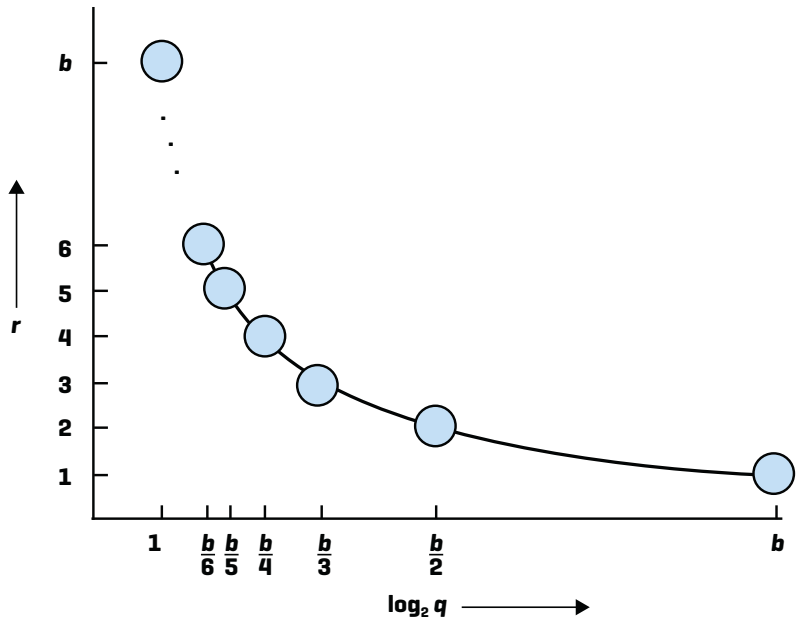
$$r = \sum_{i=1}^{p} q_i/2^b \quad \text{b}/\log_2 q$$

This inequality says for the problem of finding strings at Hamming distance 1, the replication rate is proportional to b, the string length, and inversely proportional to the logarithm of the maximum number of inputs that can be assigned to one reducer. Figure 2 shows the form of the lower bound on r and also shows points where we have algorithms that match the lower bound.

The algorithms at the endpoints are easy to see. If $\log_2 q = b$, then $q = 2^b$, which means that one reducer can get all the possible inputs. In that case, there is no need for any replication; that is, if $\log_2 q = b$, then $r = 1$ suffices.

At the other extreme, if $\log_2 q = 1$, that is, $q = 2$, then we need one reducer for each possible pair of strings at distance 1. Each string s must be sent to the b different reducers that correspond to pairs $\{s, t\}$ where t is one of the b strings at Hamming distance 1 from s. In terms of key-value pairs, the keys are pairs of strings at distance 1. The Map function generates from an input

Figure 2: Known algorithms matching the lower bound on replication rate.



string s the b key-value pairs with value s and key $\{s, t\}$, where t is one of the bit strings at distance 1 from s. Then the reducer for key $\{s, t\}$ looks at the list of values associated with this key, and if both s and t are present outputs that pair. Otherwise, it outputs nothing. (In fact, unless at least one of s and t is present on the input, this reducer will not even exist.)

The other points shown in Figure 1 represent variants of the "splitting" algorithm [8]. For any integer $k \geq 2$ that divides b, we can split the positions of b-bit strings into k equal parts. Let a reducer correspond to one of these k segments and a particular bit string of length $2^{(k-1)b/k}$ that can appear in all but that segment. A bit string s is sent to k different reducers. Start by deleting the first of the k segments from s and send s to the reducer corresponding to segment number 1 and the bits of s in all but segment 1. Then, starting from s again, drop the second segment and send s to the reducer corresponding to segment 2 and the bits of s that remain. Continue in this way for each of the k segments. For example, if $b = 6$, $k = 3$, and $s = 011100$, then send s to the three reducers:

1. Segment = 1 and string = 1100.
2. Segment = 2 and string = 0100.
3. Segment = 3 and string = 0111.

The replication rate is clearly $r = k$, and the number of bit strings that can be assigned to any reducer is the number of possible strings in any one segment, that is, $q = 2^{b/k}$. If we take logarithms, we get $\log_2 q = b/k$. Since $r = k$, we find $r = b/\log_2 q$ is an upper bound as well as a lower bound.

Triangle Finding. Another problem for which we can obtain closely matching upper and lower bounds on the replication rate as a function of the maximum input size for a reducer is finding the number of triangles in a large graph, such as the graph of a social network. We shall not go into the applications of triangle-finding, but intuitively, we expect that closely knit communities of friends would have many triangles. That is, whenever A is friends with both B and C, we would expect it is likely that B and C are also friends with each other. The most efficient MapReduce algorithm for finding triangles is from a technical report published last year [9]. On a graph with m edges, it uses total computation time $O(m^{(3/2)})$, which is the best possible according to Alon [10]. This MapReduce algorithm makes use of a serial algorithm for finding all triangles in time $O(m^{3/2})$, due to Schank's Ph.D. work [11], and the conversion of that algorithm to a MapReduce algorithm using the same total computation is from Suri and Vassilvitskii [12].

Suppose the m edges of a graph on n nodes are chosen so that each possible edge is equally likely to be chosen. If we run the algorithm using enough reducers so that the expected number of edges at any reducer is q, then the replication rate is $O(\sqrt{m/q})$. That is, each edge will be sent as the value of a key-value pair to that number of different reducers. We shall not give the argument here, but it is shown that $\Omega(\sqrt{m/q})$ is also a lower bound on the replication rate [7]; i.e., the algorithm mentioned gives, to within a constant factor, the lowest possible replication rate.

SUMMARY

We have tried to motivate the need to study MapReduce algorithms from the point of view of how they trade parallelism for communication cost. We represent the degree of parallelism by the upper limit on the number of inputs that one reducer may receive; the smaller this limit, the more potential parallelism. We represent communication cost by the replication rate, that is, the number of key-value pairs produced for each input. Depending on your computational resources and your network, you may prefer one of many different points along the curve that represents this tradeoff. As a result, it is interesting to discover lower bounds on the replication rate as a function of the reducer input size.

For two problems, finding strings at Hamming distance 1 and finding triangles in a graph, we gave lower bounds on replication rate r as a function of input size q that are tight. That is, there are algorithms for a wide variety of q values whose replication rate is, to within a constant factor, that given by the lower bound.

However, there are problems in a variety of domains for which optimal MapReduce algorithms have not been studied. Analyzing these problems requires deriving new lower bounds, designing algorithms that attain them, and choosing parameters to balance the tradeoff between communication and computation costs on modern computer architectures. By understanding such tradeoffs, we can design MapReduce algorithms that are efficient both in terms of wall-clock time and in terms of data movement.

References

[1] Dean, J. and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Conference on Symposium on Opearting Systems Design & Implementation* (San Francisco, Dec. 6-8, 2004). 137–150.

[2] White, T. *Hadoop: The Definitive Guide. Storage and Analysis at Internet Scale, Second Edition.* O'Reilly Media, Sebastopol, CA, 2011.

[3] Afrati, F.N., Borkar, V. R., Carey, M. J., Polyzotis, N., and Ullman, J. D. MapReduce extensions and recursive queries. In *Proceedings of the 14th International Conference on Extending Database Technolog* (Uppsala, Sweden, March 21-24). ACM Press, New York, 2011, 1–8.

[4] Rajaraman, A., and Ullman, J. D. *Mining of Massive Datasets*. Cambridge University Press, Cambridge, UK, 2011. Also available on-line at http://infolab.stanford.edu/~ullman/mmds.html.

[5] Amazon Elastic Compute Cloud (Amazon EC2). Amazon Inc., 2008.

[6] Kwon, Y., Balazinska, M., Howe, B., and Rolia, J. A. Skewtune: mitigating skew in MapReduce applications. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (Scottsdale, May 20-24). ACM Press, New York, 2012, 25–36.

[7] Afrati, F. N., Das Sarma, A., Salihoglu, S., and Ullman, J. D. Vision paper: Towards an understanding of the limits of MapReduce computation. CoRR, abs/1204.1754, 2012.

[8] Afrati, F. N., Das Sarma, A., Menestrina, D., Parameswaran, A. and Ullman, J. D. Fuzzy joins using MapReduce. In *Proceedings of the International Conference on Data Engineering* (Washington, D.C., April 1-5, 2012).

[9] Afrati, F. N., Fotakis, D., and Ullman, J. D. Enumerating subgraph instances using MapReduce. Technical report. Stanford University, December 2011. http://ilpubs.stanford.edu:8090/1020/.

[10] Alon, N. On the number of subgraphs of prescribed type of graphs with a given number of edges. *Israel Journal of Mathematics* 38, 1-2 (1981), 116–130.

[11] Schank, T. Algorithmic Aspects of Triangle-Based Network Analysis. Ph.D. Thesis. Universitat Karlsruhe (TH), 2007.

[12] Suri, S. and Vassilvitskii, S. Counting triangles and the curse of the last eeducer. In *Proceedings of the 20th International Conference on World Wide Web* (Hyderabad, India, Mar. 28-April 1). ACM Press, New York, 2011, 607–614.

Biography

Jeff Ullman is a retired professor of computer science at Stanford University. He has written textbooks covering automata theory, compilers, data structures and algorithms, database systems, and data mining. During his teaching career at Princeton and Stanford, he graduated 53 Ph.D. students, many of whom have become leaders in academia and industrial startups.

Big Data and Internships at Cloudera

Students working in the big data space get uniquely valuable experiences and perspectives by taking industrial internships, which can help further their research agendas.

By Yanpei Chen, Andrew Ferguson, Brian Martin, Andrew Wang, and Patrick Wendell

DOI: 10.1145/2331042.2331054

The world has caught fire with "big data," which is a collection of tools and strategies for processing massive datasets at low cost. At the epicenter of the big data movement is Apache Hadoop [1], an open-source distributed data processing framework similar to Google's MapReduce [2]. Hadoop was originally developed at Yahoo to ingest and analyze Web-scale datasets and has quickly been adopted by other tech companies and industries.

Hadoop's rapid ascent is due in part to the burgeoning ecosystem of private sector companies. One of these companies is Cloudera, which was founded in 2008 by early Hadoop evangelists. Today, Cloudera is a leading force in Hadoop development, contributing code to the core storage and processing layers of the open-source Hadoop stack and providing consulting and support services for Hadoop. Cloudera's software enjoys broad adoption; supporting 80 percent of online travels booked and helping half of the Fortune 50 make sense of big data [3]. Collaboration with hundreds of industrial partners places Cloudera in a unique position to provide forward-looking research problem statements and incorporate cutting edge research into its products.

When we were invited to contribute to *XRDS*, we decided it would be meaningful for us to talk about why we have come to believe industry-academia collaboration is essential to big data. This is followed by several stories that discuss our individual internship experiences on big data projects, including reflections on how these experiences helped refine our research agendas and accelerate progress on our dissertation work.

BIG DATA IN THE BIG WORLD

Based on joint observations as students and Cloudera interns, we believe the diversity and scale of big data systems set them apart from other computer systems and necessitate industry-academia collaboration.

Enterprises understand the importance of big data; it is crucial to their bottom line. However working across multiple industry sectors creates significant and sometimes mutually exclusive variations in big data sources and system requirements. A priority for one customer may be secondary or even harmful for another. A series of overly specific solutions would result in an unwieldy and inefficient product portfolio. Additionally, knowledge extracted from big data brings about rapid innovations in business, scientific, and consumer activities. This, in turn, causes rapid evolution in the data generated by such activities, both in data volume and in the inherent nature of the data. Managing this diversity requires applying scientific principles to distill the common technology requirements and specifying the dimensions of customization that allow different needs to be systematically expressed. Conversely, a cross-industry perspective offers academic researchers empirical guidance with regard to research priorities and complementary research agendas.

Big data systems often involve multi-layered and distributed com-

ponents, while big data itself often involves multiple data sources in different formats. The recent experiences of large technology companies have indicated the sheer scale of such systems and data sources can introduce many design and evaluation challenges. Realistic system and data scale exceeds the scale that can be afforded by purely academic systems or research prototypes. This creates the need for researchers to appreciate and address real-life problems at enterprise scale. At the same time, the scale of the systems and the data demand principled architectures combined with rigorous implementation and evaluation processes.

This experience put me right where the "rubber meets the road" in large-scale data management and has led to several insights about problems faced in day-to-day big data deployments.

BIG DATA INTERNSHIPS AT CLOUDERA

The following is a series of stories about each co-author's individual internship experience at Cloudera. These internships concretely illustrate how industry-academia collaboration helped big data research at our respective universities. We hope these stories will encourage students to consider whether an industrial internship makes sense for their own work.

Yanpei Chen, 2011 Intern

My internship project was to collect and analyze Cloudera Distribution of Hadoop (CDH) traces from Cloudera customers. We observed that several MapReduce "benchmarks," popular even now, were not representative at all of real-life use cases. As Cloudera's customers increased their MapReduce expertise, they voiced similar concerns. Therefore, the lack of empirical, real-life cluster traces created huge barriers for quality assurance and performance testing of the core CDH product and technology certification for Cloudera's partner vendors.

The actual process of collecting customer cluster traces proved necessarily troublesome. Customers were rightly concerned about leaking proprietary information. With help from Cloudera's engineering, support, marketing, and executive teams, we collected an unprecedented set of real-life MapReduce cluster traces from both technology and traditional enterprises. Insights from this data set have led to key publications of my dissertation, which characterized a new class of MapReduce workloads for interactive analysis (see Figure 1).

Andrew Wang, 2012 Intern

As a part of the Algorithms, Machines, and People Laboratory at Berkeley, my research revolves around big data and the components of the Hadoop software ecosystem. More specifically, I am interested in providing high-level service-level objectives (SLOs) for distributed storage systems.

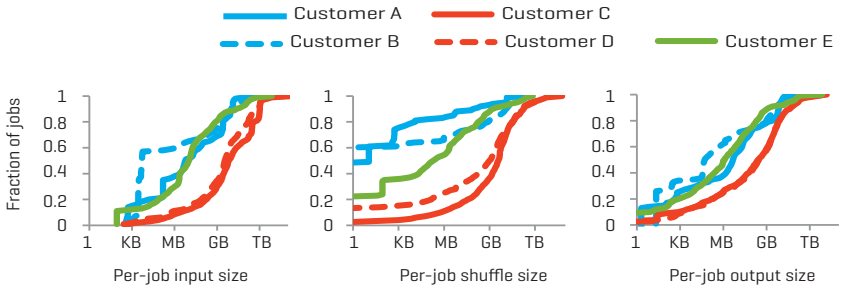
Working at Cloudera has been an eye-opening experience in many ways. Foremost is the incredible perspective it provides on how storage systems like HDFS and HBase are used in practice. Getting to talk directly with customers and developers has helped me refine my research agenda in terms of problem selection and approach. Another valuable experience has been learning to work with the open-source community. Research code is often left to bit-rot after the associated paper is published, but open-source is an opportunity for additional impact and a way of publicizing your research. While at Cloudera, I have been able to disseminate my research ideas within the open-source community, as well as work on directly applying them to a product that will ultimately be used by hundreds of companies.

Brian Martin, 2012 Intern

As a student in machine learning and natural language processing, my research involves parallel inference and learning in graphical models for large-scale information extraction. During my time at Cloudera I worked directly with Josh Wills, Director of Data Science, and developed several new statistics and machine learning tools for advanced analytics on Hadoop.

My initial project was to develop a tool for calculating distance correlation over giant tables of data (e.g. all Chicago crimes and building permits in the last decade grouped by location or all purchase histories grouped by various demographic variables). Distance correlation is a recent statistical measure of dependence, linear and nonlinear, between variables (see Figure 2). This tool will soon be available open-source. Additionally, I implemented a recently proposed solver for large-scale linear regression problems using Apache Hadoop Next-Gen MapReduce (YARN). YARN allows for running non-MapReduce applications on a Hadoop cluster, while playing well with the resource manager.

Figure 1. Per-job data sizes for a new class of MapReduce workloads for interactive analysis, which contains many jobs that manipulate data sizes less than a terabyte.



Andrew Ferguson, 2012 Intern

My Ph.D. work is on software defined networks (SDNs) and platforms for big data processing. In both of these areas, the core technologies were developed in academic labs and Internet-based companies. However, both technologies are now reaching new markets with more traditional companies. As new users adapt technologies, new use cases and new problems arise—opening fresh avenues for systems research. This exposure to new types of Hadoop customers drew me to Cloudera for the summer.

I encourage students to consider interning at a start-up, even if they are set on joining the academy after graduation. Numerous faculty members start companies during their careers, as it can be an effective way to change the world through research. And even for those who do not start companies, the experience will help when advising future students on career options and selecting their own internships. As students, we often have the twin luxuries of unstructured time and an ease of moving, so pick a city and an interesting company and explore a new side of your life and research.

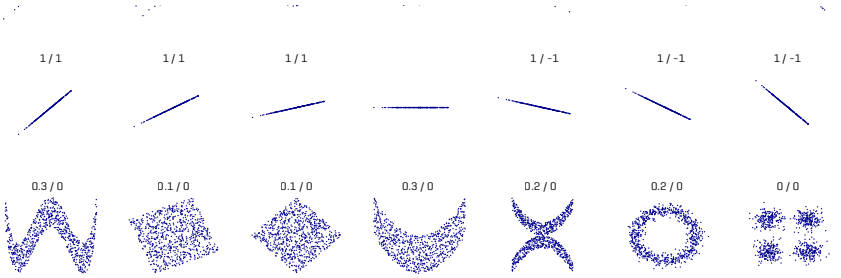
Patrick Wendell, 2012 Intern

My thesis work is on resource management for large-scale data processing systems. However, my summer internship at Cloudera was off the beaten path for most academic engineers. I spent three months travelling in the field with Cloudera's engineers and worked directly with customers as they assessed, prototyped and deployed Hadoop in live environments. This experience put me right where the "rubber meets the road" in large-scale data management and has led to several insights about problems faced in day-to-day big data deployments. Additionally, it has provided me with a perspective on which types of engineering solutions are most successful in the wild, which I will take back with me as I continue my research degree.

I have been able to disseminate my research ideas within the open-source community, as well as work on directly applying them to a product that will ultimately be used by hundreds of companies.

The most salient lesson I've taken from the summer is that when companies are evaluating a new technology, performance, with respect to alternative solutions, is but one of many criteria considered. Factors like deployment complexity, interoperability with existing systems, cost, fitness for a particular business problem, and ease-of-use combine to influence adoption of new technologies. As researchers, we tend to focus on innovativeness and differentiation, rather than simplicity and interoperability. I increasingly believe that the latter two traits are necessary requirements for any viable technology.

Figure 2. Examples of Pearson correlation and distance correlation for various relationships. Note that Pearson correlation only picks up linear dependence, giving no dependence for any examples in the bottom two rows.



Source: http://en.wikipedia.org/wiki/File:Distance_Correlation_Examples.svg

SUMMARY

Students benefit from experiencing real-life, big data problems and having access to a broad spectrum of industrial engineers, partners, and customers. Conversely, Cloudera benefits from the industry-academia cross-pollination of ideas and the methodical approach to problem solving brought by interns. We hope our stories can encourage big data companies to continue reaching out to academia, while helping student readers consider whether an industrial internship makes sense for their dissertation work.

References

[1] Apache Hadoop; http://hadoop.apache.org/

[2] Dean, J. and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating Systems Design and Implementation* (San Francisco, Dec. 6–8). USENIX Association, Berkeley, CA, 2004.

[3] King, R. Chevron explores open source using Hadoop. *Wall Street Journal*. June 5, 2012.

Biographies

Yanpei Chen just finished his Ph.D. at UC Berkeley, where he was advised by Prof. Randy Katz. His dissertation was on workload-driven design and evaluation of big data systems. He is currently on the performance team at Cloudera.

Andrew Ferguson is a fourth-year Ph.D. student at Brown University, advised by Prof. Rodrigo Fonseca. He works to improve computational and network infrastructure through his research, and enjoys critiquing coffee and crafting cocktails.

Brian Martin is an M.Sc. student in the Information Extraction and Synthesis Lab at UMass Amherst where his research interests include natural language processing, graphical models, and probabilistic inference. He is advised by Prof. Andrew McCallum.

Andrew Wang is a third-year Ph.D. student in the Algorithms, Machines and People Lab at UC Berkeley, where he is advised by Prof. Ion Stoica. His research interests are in distributed systems and storage. In his copious spare time, he likes to host dinner parties and go on bike rides.

Patrick Wendell is a second-year Ph.D. student in the AMP lab at U.C. Berkeley, where he is advised by Prof. Ion Stoica. His current research focuses on low-latency resource scheduling in large clusters. He enjoys contributing code to several Hadoop projects and drinking good beer, sometimes concurrently.

An Interview with Surajit Chaudhuri

Surajit Chaudhuri, Distinguished Scientist and head of the Extreme Computing Group (XCG) at Microsoft Research, Redmond provides valuable insights for revisiting data analytics in the context of big data.

By Aditya Parameswaran

DOI: 10.1145/2331042.2331055

ADITYA PARAMESWARAN: In your opinion, what does "big data" stand for? What are the most important challenges in dealing with big data?

SURAJIT CHAUDHURI: "Big data" has become a catch phrase and like many other catch phrases has multiple interpretations. As a researcher interested in the data analytics space, I view the underlying challenges as that of building platforms and tools that enable businesses to derive actionable insights from raw data dramatically faster and finer-grained than is possible today. This aspiration maps into a set of concrete technical challenges. I described them in a short article I prepared for my keynote at ACM Principles of Database Systems conference (PODS) earlier this year [1].

AP: The database community has been working on "very large data bases" for a while now. What is new now? Are we simply marketing big data as old wine in new bottles?

SC: It is true that the database community has always paid close attention to building scalable data platforms, but there have been big changes quantitatively as well as qualitatively over the last few decades. Let me give you a few examples: As more data gets digitally born, we have seen a sharp increase in the volume of raw data.

At the same time, cost of raw storage has plummeted. Together, this means that data acquisition cost today is really low and that leads to a growing increase in data available to enterprises. Another example of change is increasing interest and opportunities in deriving value from customer interactions (e.g., query logs) and text data. There is also the desire to take business action on fresh data rapidly, and the rise of cloud services. These are example of changes that are big. Thus, our community has a lot of new opportunities to impact the technology. However, I think we have not acted on these trends boldly yet. If I were to look at recent research publications, a disproportionately large fraction of them are focused on solving for MapReduce platforms the same problems we addressed for parallel database systems. We can and should do so much more.

AP: An argument against big data is that while we are now able to collect data more cheaply, it is not necessarily "useful" data.

SC: Of course, turning raw data into actionable insights is really hard. Indeed, the end goal for enterprises is not storing and managing lots of raw data—it is to get to *newer actionable* business insights *faster*. Our community can do a lot more to get to such insights "faster" by giving platform support and tools for quicker and novel exploration of data.

AP: Are there examples of how big data analysis has made a significant impact on consumer or organization value?

SC: There are many examples of using data analytics for business value. Using transactional and query logs to do finer segmentation of customers is a classic example that has been used for direct mailing campaign as well as deciding differentiated pricing structures by retail as well as TELCO.

AP: Machine learning is thought to be a key component of big data analytics. How

do you envision machine learning to be used within a database or in commercial database systems?

SC: Microsoft SQL Server (and other commercial relational databases) developed extensible APIs that integrate mining with querying paradigm and also provided a set of core algorithms for machine learning as far back as 2000. That line of work led to only limited success because for an average developer, applying machine learning isn't easy. I am not sure that the situation has yet changed dramatically in that regard today. It should also be noted that identifying the relevant subset of data and preparing data so that it is ready for application of machine learning is a critical part and is often as hard if not harder than the phase of actual application of machine learning algorithms for deriving insights from data. Therefore, not surprisingly, vertically integrated use of machine learning in the context of high-value analytic applications (e.g., credit card fraud) delivered as either packaged application or as SaaS (software as service) has seen the best successes. Supporting the workflow of applying machine learning for mere mortals is still a big challenge and it is in this facet that we need to push if we aspire to see broader usage of machine learning.

AP: How do you visualize the progress of query optimization in the last two decades? Is it worth revisiting query optimization in the context of big data?

SC: We have seen steady incremental progress in query optimization in the last two decades, but I cannot say that I have witnessed a big breakthrough. Almost all of us who have worked on query optimization find the current state of the art unsatisfactory with known big gaps in the technology. I absolutely think it is worth revisiting query optimization in the context of big data. Like many others, I too believe that declarative queries (SQL or other flavors of query languages) are essential for productivity of application developers in the context of big data. There is a lot of exploration of newer data platforms in the context of big data. This disruption provides a great opening to piggyback fresh ideas and revisit classical assumptions and constraints we have associated with query optimization. It is not an easy problem to crack but a breakthrough in this technology will get everyone's attention. So, perhaps a perfect thesis topic for a courageous Ph.D. student?

The end goal for enterprises is not storing and managing lots of raw data—it is to get to newer actionable business insights faster.

AP: What are your views on the rise of NoSQL systems and on dropping "ACID" guarantees?

SC: Exploration of data platforms that question the principles and design of known solutions (such as relational databases) is healthy. On the other hand, taking a religious position against all declarative query languages or against ACID is not helpful. As we all know, for a specific problem, a specialized solution can often be built that will be significantly more efficient. So, the real success for such NoSQL platforms should be judged by their usage in a broad application area as well as their ability to balance programmer productivity with opportunities for high performance. But, there is no question that exploration of NoSQL system has been great to stir the pot to revitalize discussions on architectures and algorithms in today's database systems.

AP: What is the future of big data analytics in your view?

SC: The ability to quickly explore data indeed critically depends on crafting the right set of tools. The challenge is really to identify the right abstractions for such explorations and not just exciting visual interfaces. This remains a wide-open challenge.

AP: What is the current emphasis of your group and how has it been impacted by big data?

SC: My group has worked on three core areas in the last decade at a considerable depth: self-tuning database systems, data cleaning, and identifying paradigms for flexible exploration of structured and text data. In recent years, our agenda has been influenced by both big data and the advent of cloud services. We are trying to identify operators that can help connect data that are seemingly different through statistical techniques—this line of work really grew out of our experience in working with our Bing team using data cleaning technology. We are applying machine learning and other analytics tools to better understand our operational data from cloud services to derive actionable insights. We are also interested in revisiting our work on approximate query processing in the context of big data. Finally, we are working on the problem of providing performance isolation in the context of multi-tenant systems.

AP: In this fast-paced and diverse area, how would you advise an undergraduate or graduate student to best prepare for a research or industry career?

SC: In general, learning outside of one's core research area is increasingly more important. That is much easier to do at school than when you are working. For database researchers, understanding of statistics, distributed systems, and hardware is more important today than a couple of decades ago. I also recommend seizing internship opportunities at cutting-edge technology companies, where there is a lot of exciting exploration of newer data platforms. More importantly, you also gain an understanding of their customer data, which is hard to gain access to at school.

References

[1] Chaudhuri, S. What next?: A half-dozen data management research goals for big data and the cloud. In *Proceedings of the 31st Symposium on Principles of Database Systems* (Scottsdale, May 20-24). ACM Press, New York, 2012, 1-4.

Biography

Aditya Parameswaran is a PhD student at Stanford University. He is broadly interested in information management. He is a recipient of the Key Scientific Challenges award (2010) from Yahoo! Research, selection for the Best Papers of VLDB 2010 and KDD 2012, as well as the Terry Groswith fellowship at Stanford. He graduated at the top of his undergraduate class from IIT Bombay in 2007.

Peregrine: Low-Latency Queries On Hive Warehouse Data

How Facebook is analyzing big data.

By Raghotham Murthy and Rajat Goel

DOI: 10.1145/2331042.2331056

Big data is only as valuable as the useful analyses it supports. However, current database systems either slow down or become very expensive as the size of the data increases. This means that the full value of the data being collected is not being exploited. At Facebook, the size of our data has grown as the number of users and their interactions with new and existing Facebook products increases. Given this growth, we needed a way to empower our analysts and engineers to effectively query these large data sets with the same ease as with smaller data sets on existing database systems.

Back in 2008, the Facebook data infrastructure team built and open sourced Hive [1], a warehousing solution built on top of Hadoop [2]. Hive supports queries expressed in an SQL-like declarative language named HiveQL [3], which are compiled into MapReduce jobs executed on Hadoop. In addition, HiveQL supports custom MapReduce scripts that can be plugged into queries. The language includes a type system with support for tables containing primitive types; collections like arrays and maps; and nested composition of these types to create new types. The underlying IO libraries can be extended to query data in custom formats. Hive also includes a system catalog, Hive-Metastore, containing schemas and statistics, which is useful in data exploration and query optimization.

Over the years, the Hive warehouse has been the primary way that Facebook analysts and engineers store and analyze large amounts of data. Hive is used in data analysis Facebook in a variety of ways:

- The Newsfeed team analyzes the degree of connection between users to rank stories in the Facebook Newsfeed.
- Activity logs are analyzed to gain insights on how Facebook services are being used. These analyses drive internal business intelligence applications as well as externally available tools that provide insights to Facebook advertisers, application developers, and page administrators.
- The Ads team runs complex data mining algorithms to optimize the kind of advertisements shown to Facebook users.
- The Security team mines usage logs to identify spam and other abuse.

We have made it easier for even non-engineers to use Hive by building Web-based tools for authoring and executing Hive queries (Hipal); for authoring, debugging, and scheduling complex data pipelines (Dagger); and for generating reports based on data stored in Hive and other relational databases like MySQL and Oracle (Argus). At the same time, we have scaled the Hive warehouse stack to manage more than 150,000 tables occupying more than 100PB of storage while supporting user queries that scan 5PB of compressed data every day.

That said, one of the most common complaints from Hive users is that its latency is too high for interactive analyses. Even the simplest of queries take several seconds or minutes. This latency significantly affects the productivity of analysts. The current workaround involves the creation of data pipelines that load aggregate data from Hive into other RDBMS like MySQL and Oracle, and then perform ad-hoc analysis and build reports using these RDBMS. Over the years, our engineers and analysts have written hundreds of such pipelines, which then have to reliably run

Detail of Paul Butler's visualization of Facebook global friend connections found in the Hive data warehouse.

on a regular basis. However, the overhead of this workaround for getting low-latency access to data has proven to be very cumbersome and inconvenient for users. To work on decreasing the latency of queries in Hive generally, in late 2010 we started Peregrine. This independent project focuses on supporting low latency queries.

Peregrine is a distributed query engine for ad hoc analysis built on top of the Hadoop Distributed File System (HDFS) [4] and the Hive-Metastore. It is fully compatible with Hive's data formats and metadata and supports a subset of HiveQL. At the same time, it is able to achieve much lower query latencies compared to Hive by combining an in-memory, serving-tree based computation framework with approximate query processing.

Peregrine has been running in production at Facebook for more than a year now. We have found, in many cases, Facebook data analysts and engineers are using Peregrine instead of Hive for discovering patterns in Hive data. They are able to do quick analyses and rapid prototyping of queries that eventually need to run in Hive data pipelines. In addition, Peregrine is being used to directly serve aggregate data stored in Hive for our reporting systems. Finally, the physical operator library in Peregrine is modular enough that we were able to reuse it to build a distributed streaming query processing system called PStream that is capable of handling several million rows of input data per second using a handful of commodity machines.

PEREGRINE QUERY LANGUAGE

Peregrine's query language is a subset of HiveQL. Similar to Hive, it supports simple filters, aggregates, top-k, percentiles, sub-queries in FROM clauses, UNION ALL, and a rich set of user-defined functions. The TABLESAMPLE clause can be used to explicitly limit the amount of input data that is scanned and the WITH clause allows users to write more readable queries by declaring variables for complex expressions that are used multiple times in the same query. The WITH clause also provides a way for the user to specify hints to the optimizer such that it evaluates common sub-expressions only once during run time.

When users are just exploring data, even partial answers are good enough. So, in Peregrine, syntactically correct queries never fail. Run-time errors like node failures, corrupted input data, or even users killing their queries are just treated as situations where all input data was not scanned. In these cases, Peregrine returns partial answers and clearly indicates what percentage of input was scanned. If users want to get exact answers, they need to specify WITH true as mode.exact, which causes run time errors to fail queries.

Given that Peregrine only uses one-pass algorithms for aggregations and stores all intermediate results in memory, the sizes of the intermediate results and the final outputs are forced to be small. This limitation implies Peregrine returns approximate answers for some types of queries. For example, queries containing ORDER BY on aggregates (SUM, MIN, COUNT) always return approximate answers. Also, some aggregate functions like distinct counts and percentiles are approximate and are named accordingly (e.g., APPROX_COUNT_DISTINCT and APPROX_PERCENTILE). Figure 1 shows a sample query in a Peregrine session where a user has terminated the query after some time. Note that, in addition to the result of the query itself (not shown), there is a lot of information provided to the user about the query processing and the result itself. For example, it shows the query result is partial because of the failures in the query. Similarly, the same query run with mode.exact (see Figure 2) takes close to five times as long to run since it has to deal with stragglers. We will describe what stragglers are and how we deal with them.

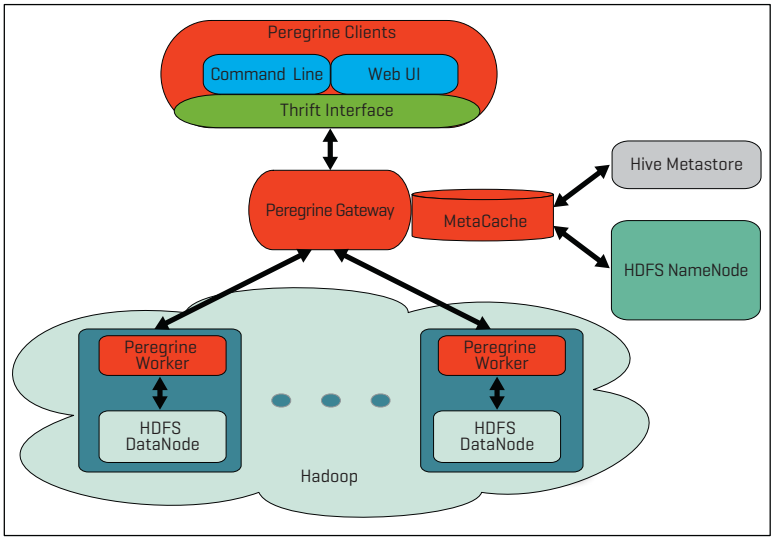
Figure 1. Query terminated by user using Ctrl-C.

```
SELECT country, COUNT(*) FROM T1 GROUP BY country;
^C

[Result is partial] [Took 22.2s] [Threads used 3300]
[Rows scanned 4269913519] [251 rows out]
[Read 275.66 GB / 359.17 GB (76.7484%, 265 failures)]
```

Figure 2. Query run in exact mode.

```
WITH TRUE AS mode.exact
SELECT country, COUNT(*) FROM T1 GROUP BY country;

[Result is exact] [Took 99.4s] [Threads used 3400]
[Rows scanned 5057153726] [251 rows out]
[Read 359.75 GB / 359.17 GB (100.0%, 0 failures)]
```

Figure 3. Peregrine architecture.



PEREGRINE ARCHITECTURE

Peregrine's architecture is shown in Figure 3. Before describing it, we will briefly introduce the architecture of HDFS, the distributed file system that stores all Hive table data. HDFS consists of a central NameNode that manages the file system namespace. In addition, there are several DataNodes, usually one per node in the cluster. HDFS files are split into one or more blocks that are stored and served from a set of DataNodes. Peregrine has a single-level serving-tree architecture that was inspired by Dremel [5], but is simpler.

Peregrine has a central gateway much like Dremel's root server. The actual query processing is done in a cluster of worker nodes. This cluster is overlaid on the HDFS cluster that stores Hive table data, and as such, worker nodes are collocated on the same hosts as HDFS data nodes to facilitate low-latency node-local reads. Additionally, all servers have been implemented using Apache Thrift [6], an RPC framework built and open sourced by Facebook. Thrift makes it very easy to write highly scalable and high performance servers quickly. Its communication protocols are also optimized for low latency.

Query Execution. Queries submitted through the client are sent to the gateway. The gateway parses the query and retrieves column metadata and HDFS-file locations from the Hive metastore. It then retrieves the locations of the corresponding HDFS blocks from the NameNode. Each block becomes a separate unit of processing called a "split." Each split is then sent for local processing in a worker that is collocated with the HDFS datanode that hosts the split. The workers periodically return status updates, called "heartbeats," back to the gateway, indicating progress. If for any split, there are no heartbeats or there is no progress, the gateway deems it to have failed and reschedules it. At the end of processing (or on demand when the user kills the query), the workers send partial results back to the gateway. The gateway then combines the partial values and returns the result to the client. All query execution is done in memory and partial results are transferred to the gateway

over the network via Thrift calls.

Optimizations. There are a many optimizations that we have done in Peregrine with the explicit goal of decreasing query latency. In this section, we describe a couple.

MetaCache. The gateway caches both the table-level and the file-level metadata in the MetaCache. If multiple queries are run on the same Hive table partitions in quick succession, which is typically when users are interactively analyzing a table, Peregrine reuses the metadata from the MetaCache instead of fetching it repeatedly from the Hive metastore and the HDFS NameNode. Such an optimization is fine since most data in Hive is written once and read many times. The MetaCache is mostly consistent with the Hive metastore—it uses a real-time feed of the Hive audit logs to invalidate entries for partitions that may have been changed by Hive queries. As a fail-safe, and to prevent arbitrary growth in memory usage, the MetaCache automatically purges entries that have not been queried for a predefined time (typically one hour).

Histograms for stragglers and adaptive resizing of failed splits. Once 90 percent of the data has been processed, "stragglers" are identified as splits that have been processed for over a fixed number of seconds and with a rate of progress below the fifth percentile. The gateway kills these stragglers. Then it further breaks down such splits into smaller splits. It then sends out these smaller splits for workers to reprocess. This additional parallelism speeds up the retries and thus decreases the query latency caused by stragglers.

USAGE AND PERFORMANCE

Peregrine's architecture and the optimizations demonstrate latencies of queries on data stored in HDFS can be dramatically decreased as compared to Hive. At Facebook, Peregrine has gained wide usage over the past year. In addition to use for ad hoc queries, Facebook engineers and analysts came up with novel ways of using Peregrine. For example, our Web-based reporting tool, Argus, uses Peregrine to fetch data directly from the Hive warehouse. Before Peregrine, Argus relied on users to explicitly load data from Hive into MySQL or Oracle databases and then fetch data from these databases to drive reports. Another example is our HiveQL query authoring tool, Hipal, which uses Peregrine to fetch data samples to show users a preview of the data in input tables. In addition, Facebook engineers have reused Peregrine's physical operator library to run SQL transformations on data in their customized stores. We have also built a flexible SQL transformation binary that has a plug-in architecture for both sources and sinks.

Table 1. Major sources of queries in Peregrine.

		Time Taken		Input scan size	
Query Source	**#queries**	**p50 (sec)**	**p90 (sec)**	**p50**	**p90**
Hipal Preview	350k	1	4	1.2KB	1MB
Argus Reports	650k	1	6	88KB	459
Ad-hoc queries	200k	2	70	2.4MB	4.1GB

Peregrine currently drives more than 30 different internal Web-based querying and reporting tools. From a small cluster of 10 nodes, it has grown to more than 1,000 nodes overlaid on top of our ad-hoc Hive cluster. Table 1 shows usage and performance stats of the major sources of queries for a single Peregrine cluster in production for a period of six months. Given that the workload on Peregrine is varied, we provide 50th and 90th percentile numbers for time taken and input scan size.

These numbers give an idea of both the scale of usage and the query latencies. It turns out most users doing ad-hoc analysis do not need the full power of Hive. While Hive provides a complex query language, and returns exact answers all the time, it cannot support the rapid interactions our analysts and engineers require. They can, however, get by with approximate and partial answers as long as the queries return quickly. Here's a quote from a Facebook analyst:

"Peregrine sped up my workflows considerably when it came out. No more waiting for Hive on simple data searching. If it slows down and I have to go back to Hive, I'll slow down as well."

CONCLUSION

In this paper, we have shown how a low-latency query execution engine can be built on top of an existing cluster used for batch processing. Given that Peregrine is fully compatible with existing Hive data and metadata, it was very easy for our users to try it out without having to explicitly load data into the new query engine. In addition, given that Peregrine is driving reports there is no need for pipelines that load data into MySQL or Oracle for ad-hoc analysis and reporting. As part of future work, we will build support for indexing and pinning popular data sets in memory to improve query latencies even further.

Acknowledgments

We are thankful to all the Facebook engineers and analysts for using Peregrine and providing valuable feedback.

References

[1] Apache Hive; http://hive.apache.org.

[2] Apache Hadoop; http://wiki.apache.org/hadoop.

[3] Hive Query Language; https://cwiki.apache.org/confluence/display/Hive/LanguageManual.

[4] Apache Hadoop HDFS; http://hadoop.apache.org/common/docs/stable/hdfs_design.html.

[5] Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., and Vassilakis, T. Dremel: Interactive analysis of web-scale datasets. In *Proceedings of the 36th International Conference on Very Large Data Bases* (Singapore , Sept. 13-17, 2010). 330–339.

[6] Apache Thrift; http://thrift.apache.org.

Biographies

Raghotham Murthy received his M.S. in computer science from Stanford University and was in the computer science Ph.D. program at Stanford University. As an engineering manager at Facebook, he is part of the Data Infrastructure team. He is interested in distributed data management systems. Murthy has worked on large-scale data processing systems at Yahoo.

Rajat Goel received a B. Tech. degree in computer science from IIIT Hyderabad. He was a software engineer in Facebook's Data Infrastructure team; currently he works with the Traffic Infrastructure team. He is interested in developing high performance services.

Big Data Platforms: What's Next?

Three computer scientists from UC Irvine address the question "What's next for big data?" by summarizing the current state of the big data platform space and then describing ASTERIX, their next-generation big data management system.

By Vinayak R. Borkar, Michael J. Carey, and Chen Li

DOI: 10.1145/2331042.2331057

A wealth of digital information is being generated daily, information has great potential value for many purposes if captured and aggregated effectively. In the past, data warehouses were largely an enterprise phenomenon, with large companies being unique in recording their day-to-day operations in databases, and warehousing and analyzing historical data to improve their businesses. Today, organizations and researchers in a wide variety of areas are seeing tremendous potential value and insight to be gained by warehousing the emerging wealth of digital information, popularly referred to as "big data," and making it available for analysis, querying, and other purposes [1].

Online businesses of all shapes and sizes are tracking customers' purchases, product searches, website interactions, and other information to increase the effectiveness of their marketing and customer service efforts. Governments and businesses are tracking the content of blogs and tweets to perform sentiment analysis. Public health organizations are monitoring news articles, tweets, and Web search trends to track the progress of epidemics. Social scientists are studying tweets and social networks to understand how information of various kinds spreads and/or how it can be more effectively utilized for the public good. It is no surprise that support for data-intensive computing, search, and information storage—a.k.a. big data analytics and management—is being touted as a critical challenge in today's computing landscape.

In this article we take a look at the current big data landscape, including its database origins, its more recent distributed systems rebirth, and current industrial practices and related trends in the research world. We then ask the question "What's next?" and provide a very brief tour of what one particular project, namely what our ASTERIX project, is doing in terms of exploring potential answers to this question (and why).

A BIT OF BIG DATA HISTORY

It is fair to say that the IT world has been facing big data challenges for over four decades—it's just that the definition of "big" has been changing. In the 1970s, big meant megabytes; over time, big grew to gigabytes and then to terabytes. Today, the IT notion of big has reached the petabyte range for conventional, high-end data warehouses, and exabytes are presumably waiting in the wings.

In the world of relational database systems, the need to scale databases to data volumes beyond the storage and/or processing capabilities of a single large computer system gave birth to a class of parallel database management systems known as "shared-nothing" parallel database systems [2]. As the name suggests, these systems run on networked clusters of computers, each with their own processors, memories, and disks. Data is spread over the cluster based on a partitioning strategy—usually hash partitioning, but sometimes range partitioning or random partitioning—and queries are processed by employing parallel, hash-based divide-and-conquer techniques.

A first generation of systems appeared in the 1980s, with pioneering prototypes from the University of Wisconsin and the University of Tokyo and the first commercial offering coming

Generative artwork by Gwen Vanhee using Craig Reynolds' algorithms for flocking behaviour.

from Teradata Corporation. The past decade has seen the emergence of a second major wave of these systems, with a number of startups delivering new parallel database systems that were then swallowed up through acquisitions by the industry's major hardware and software vendors. Because high-level, declarative language (SQL) front relational databases, users of parallel database systems have been shielded from the complexities of parallel programming. As a result, until quite recently, these systems have arguably been the most successful utilization of parallel computing.

During the latter 1990s, while the database world was admiring its "finished" work on parallel databases and major database software vendors were busy commercializing the results, the distributed systems world began facing its own set of big data challenges. The rapid growth of the World Wide Web and the resulting need to index and query its mushrooming content created big data challenges for search companies such as Inktomi, Yahoo, and Google. The processing needs in the search world were quite different, however, and SQL was not the answer, though shared-nothing clusters once again emerged as the hardware platform of choice. Google responded to these challenges by developing the Google File System (GFS), offering a familiar byte-stream-based file view of data that is randomly partitioned over hundreds or even thousands of nodes in a cluster [3]. GFS was then coupled with a programming model, MapReduce, to enable programmers to process big data by writing two user-defined functions, map and reduce [4]. The MapReduce framework applied these functions in parallel to individual data instances (Map) in GFS files and to sorted groups of instances that share a common key (Reduce)—similar to the partitioned parallelism used in shared-nothing parallel database systems. Yahoo and other big Web companies such as Facebook created an Apache open-source version of Google's big data stack, yielding the now highly popular Hadoop platform with its associated HDFS storage layer.

Similar to the big data back-end storage and analysis dichotomy, the historical record for big data also has a front-end (i.e., user-facing) story worth noting. As enterprises in the 1980s and 1990s began automating more and more of their day-to-day operations using databases, the database world had to scale up its online transaction processing (OLTP) systems as well as its data warehouses. Companies such as Tandem Computers responded with fault-tolerant, cluster-based SQL systems. Similarly,

but later in the distributed systems world, large Web companies were driven by very expansive user bases (up to tens or even hundreds of millions of Web users) to find solutions to achieve very fast simple lookups and updates to large, keyed data sets such as collections of user profiles. Monolithic SQL databases built for OLTP were rejected as being too expensive, too complex, and/or not fast enough, and today's "NoSQL movement" was born [5]. Again, companies such as Google and Amazon developed their own answers (BigTable and Dynamo, respectively) to meet this set of needs, and again, the Apache open-source community created corresponding clones (HBase and Cassandra, two of today's most popular and scalable key-value stores).

TODAY'S BIG DATA PLATFORM(S)

Hadoop and HDFS have grown to become the dominant platform for Big Data analytics at large Web companies as well as less traditional corners of traditional enterprises (e.g., for clickstream and log analyses). At the same time, data analysts have grown tired of the low-level MapReduce programming model, now choosing instead from among a handful of high-level declarative languages and frameworks that allow data analyses to be expressed much more easily and written and debugged much more quickly. These languages include Hive from Facebook (a variant of SQL) and Pig from Yahoo (a functional variant of the relational algebra, roughly). Tasks expressed in these languages are compiled down into a series of MapReduce jobs for execution on Hadoop clusters. Looking at workloads on real clusters, it has been reported that well over 60 percent of Yahoo's Hadoop jobs and more than 90 percent of Facebook's jobs now come from these higher-level languages rather than hand-written MapReduce jobs. MapReduce is essentially being relegated to the role of a big data runtime for higher-level, declarative data languages (which are not so very different than SQL).

Given this fact, it is interesting to analyze the pros and cons of MapReduce in this role as compared to more traditional parallel SQL runtime systems [6]. Important pros of Hadoop compared with parallel SQL systems include:

1. Open source availability versus expensive software licenses.
2. Multiple non-monolithic layers and components versus having only a top-level query API through which to access the data.
3. Support for access to file-based external data versus having to first design, load, and then index tables before being able to proceed.
4. Support for automatic and incremental forward recovery of jobs with failed tasks versus rolling long jobs back to their beginning to start all over again on failure.
5. Automatic data placement and rebalancing as data grows and machines come and go versus manual, DBA-driven data placement.
6. Support for replication and machine fail-over without operator intervention versus pager-carrying DBAs having to guide data recovery activities.

Some of the cons are:

1. Similar to early observations on why database systems' needs were not met by traditional OSs and their file systems [7], layering a record-based abstraction on top of a very large byte-sequential file abstraction leads to an impedance mismatch.
2. There is no imaginable reason, other than "because it is already there," to layer a high-level data language on top of a two-unary-operator runtime like MapReduce, as it can be quite unnatural (e.g., for joins) and can lead to suboptimal performance.
3. With random data block partitioning, the only available parallel query processing strategy is to "spray-and-pray" every query to all blocks of the relevant data files.
4. A flexible, semi-structured [8], schema-less data model (based on keys and values) means that important information about the data being operated on is known only to the programs operating on it (so program maintenance troubles await).
5. Coupling front- and back-end big data platforms to cover the full big data lifecycle requires significant use of bubble gum, baling wire, and hand-written ETL-like scripts.
6. While Hadoop definitely scales, its computational model is quite heavy (e.g., always sorting the data flowing between Map and Reduce,

Figure 1. ASTERIX Architecture

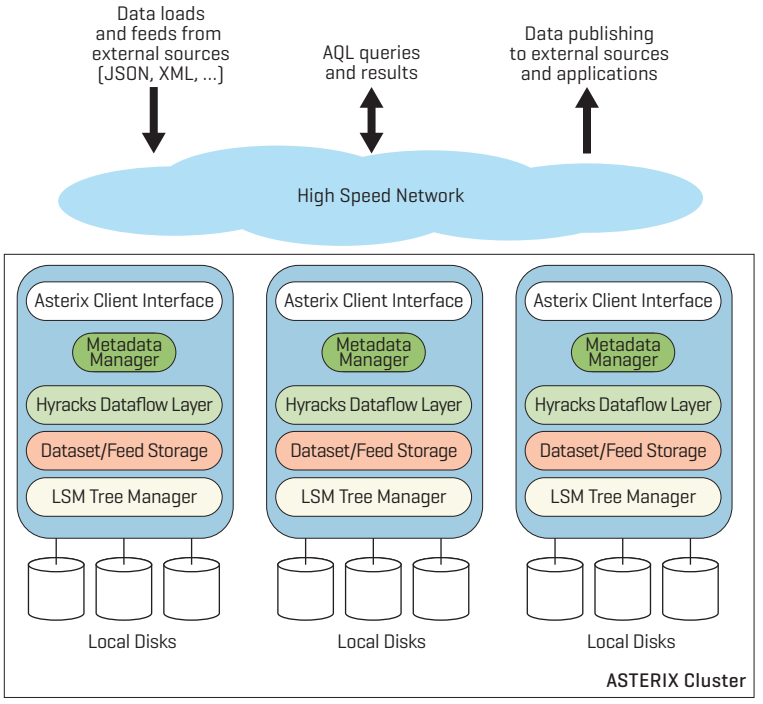


Figure 2. Example AQL schemas, queries, and results.

```
create type TweetMessageType as open {
  tweet_id: string,
  user:{
     screen-name: string,
     lang: string,
     friends_count: int32,
     statuses_count: int32,
     name: string,
     folllowers_count: int32
  },
  sender-location:_point ?
  send-time: datetime,
  referred_topics: {{string}}
  message-text: string
};

create dataset TweetMessges(TweetMessageType)
    partitioned by key tweet_id;
```

(a) ADM type and dataset for tweets

```
for $tweet in dataset('TweetMessages')
where some $topic in $tweet.referred-topics
      satisfies contains ($topic,'verizon')
for $topic in $tweet.referred-topics
group by $topic with $tweet
return {
  "topic": $topic,
  "count": count($tweet)
}
```

(b) An example query over Tweet messages to filter and aggregate tweets

```
[
    { "topic": "verizon", "count": 3 },
    { "topic": "commercials", "count": 1 },
    { "topic": "att", "count": 1 },
    { "topic": "at&t", "count": 1 }
]
```

(c) Result of example query on example data.

```
{{ {
    "tweetid": "1023",
    "user": {
        "screen-name": "dflynn24",
        "lang": "en",
        "friends_count": 46,
        "statuses_count": 987,
        "name": "danielle flynn",
        "followers_count": 47
    },
    "sender-location": "40.904177,-72.958996",
    "send-time": "2010-02-21T11:56:02-05:00",
    "referred-topics": {{ "verizon" }},
    "message-text": "i need a #verizon phone :("
  },
  {
    "tweetid": "1024",
    "user": {
        "screen-name": "miriamorous",
        "lang": "en",
        "friends_count": 69,
        "statuses_count": 1068,
        "name": "Miriam Songco",
        "followers_count": 78
    },
    "send-time": "2010-02-21T11:11:43-08:00",
    "referred-topics": {{ "commercials", "verizon", "att" }},
    "message-text": "#verizon & #att #commercials,
        so competitive"
  },
  {
    "tweetid": "1025",
    "user": {
        "screen-name": "dj33",
        "lang": "en",
        "friends_count": 96,
        "statuses_count": 1696,
        "name": "Don Jango",
        "followers_count": 22
    },
    "send-time": "2010-02-21T12:38:44-05:00",
    "referred-topics": {{ "verizon", "at&t", "iphone" }},
    "message-text": "I think I may switch from
        #verizon to #at&t"
  } }}
```

(d) Example tweet data

```
for $tweet1 in dataset('TweetMessages')
for $tweet2 in dataset('TweetMessages')
where $tweet1.tweetid != $tweet2.tweetid
    and $tweet1.message-text~=$tweet2.message-text
return {
    "tweet1-text": $tweet1.message-text,
    "tweet2-text": $tweet2.message-text
}
```

(e) An example fuzzy query over Tweet messages to find similar tweets

always persisting temporary data to HDFS between jobs in a multi-job query plan, etc. [9]).

WHAT'S NEXT?

Given the largely accidental nature of the current open-source Hadoop stack, and a need to store and manage as well as simply analyze data, we set out three years ago to design and implement a highly scalable platform for next-generation information storage, search, and analytics. We call the result a Big Data Management System (or BDMS). By combining and extending ideas drawn from semi-structured data management, parallel databases, and first-generation data-intensive computing platforms (notably Hadoop/HDFS), ASTERIX aims to be able to access, ingest, store, index, query, analyze, and publish very large quantities of semi-structured data. The design of the ASTERIX BDMS is well-suited to handling use cases that range all the way from rigid, relation-like data collections—whose structures are well understood and largely invariant—to flexible and more complex data, where little is planned ahead of time and the data instances are highly variant and self-describing.

Figure 1 provides an overview of a shared-nothing ASTERIX cluster and how its various software components map to cluster nodes. The bot-

ASTERIX aims to be able to access, ingest, store, index, query, analyze, and publish very large quantities of semi-structured data.

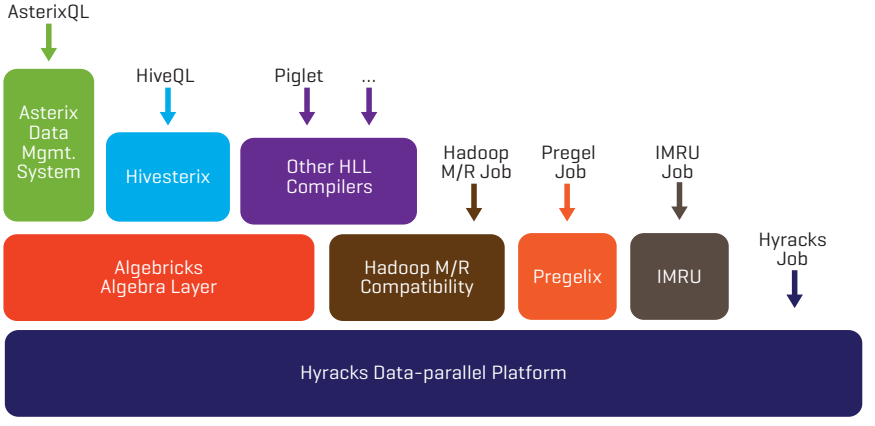
tom-most layer of ASTERIX provides storage capabilities for ASTERIX-managed datasets based on LSM-tree indexing (chosen in order to support high data-ingestion rates). Further up the stack is our data-parallel runtime, Hyracks [9], which sits at roughly the same level as Hadoop in implementations of Hive and Pig but supports a much more flexible computational model. The topmost layer of ASTERIX is a full parallel BDMS, complete with its own flexible data model (ADM) and query language (AQL) for describing, querying, and analyzing data. AQL is comparable to languages such as Pig or Hive, however ASTERIX supports native storage and indexing of data as well as having the ability to operate on externally resident data (e.g., data in HDFS).

The ASTERIX data model (ADM) borrowed data concepts from JSON and added more primitive types as well as type constructors from semi-structured and object databases. Figure 2(a) illustrates ADM by showing how it might be used to define a record type for modeling Twitter messages. The record type shown is an open type, meaning that its instances should conform to its specification but will be allowed to contain arbitrary additional fields that can vary from one instance to another. The example also shows how ADM includes features such as optional fields with known types (e.g., "sender-location"), nested collections of primitive values ("referred-topics"), and nested records ("user"). More information about ADM can be found in a recent paper that provides an overview of the ASTERIX project [10].

Figure 2(d) shows an example of how a set of TweetMessageType instances would look. Data storage in ASTERIX is based on the concept of a dataset, a declared collection of instances of a given type. ASTERIX supports both system-managed datasets—such as the TweetMessages dataset declared at the bottom of Figure 2(a)—which are stored and managed by ASTERIX as partitioned, LSM-based B+ trees with optional secondary indexes, and external datasets, whose data can reside in existing HDFS files or collections of files in the cluster nodes' local file systems.

The ASTERIX query language is called AQL, a declarative query language designed by borrowing the essence of the XQuery language, most notably its FLWOR expression constructs and composability, and then simplifying and adapting it to the

Figure 3. ASTERIX software stack.



types and data modeling constructs of ADM. Figure 2(b) illustrates AQL by example. This AQL query runs over the TweetMessages dataset to compute, for those tweets mentioning "verizon," the number of tweets that refer to each topic appearing in those tweets. Figure 2(c) shows the results of this example query when run against the sample data of Figure 2(d).

One of the primary application areas envisioned for ASTERIX is warehouse-based Web data integration [11]. As such, ASTERIX comes "out of the box" with a set of interesting capabilities that we feel are critical for such use cases. One is built-in support for a notion of data feeds to continuously ingest, pre-process, and persist data from external data sources such as Twitter. Another is support for fuzzy selection and fuzzy (a.k.a. set-similarity) joins, as Web data and searches are frequently ridden with typos and/or involve sets (e.g., of interests) that should be similar but not identical. Figure 2(e) illustrates a fuzzy join query in AQL. Yet another built-in capability is basic support for spatial data (e.g., locations of mobile users) and for queries whose predicates include spatial criteria.

Figure 3 shows the nature of the open-source ASTERIX software stack, which supports the ASTERIX system but also aims to address other big data requirements. To process queries such as the example from Figure 2(b), ASTERIX compiles each AQL query into an Algebricks algebraic program. This program is then optimized via algebraic rewrite rules that reorder the Algebricks operators as well as introduce partitioned parallelism for scalable execution, after which code generation translates the resulting physical query plan into a corresponding Hyracks job that uses Hyracks to compute the desired query result. The left-hand side of Figure 3 shows this layering. As also indicated in the figure, the Algebricks algebra layer is data-model-neutral and is therefore also able to support other high-level data languages (such as a Hive port that we have built).

The ASTERIX open-source stack also offers a compatibility layer for users with Hadoop jobs who wish to run them using our software as well as offering several other experimental big data programming packages (including Pregelix, a Pregel-like layer that runs on Hyracks, and IMRU, an iterative map/reduce/update layer that targets large-scale machine learning applications [12]).

ASTERIX GOING FORWARD

Currently the ADM/AQL layer of ASTERIX can run parallel queries including lookups, large scans, parallel joins (both regular and fuzzy), and parallel aggregates. Data is stored natively in partitioned B+ trees and can be indexed via secondary indexes such as B+ trees, R-trees, or inverted indexes. The system's external data access and data feed features are also operational. We plan to offer a first open-source release of ASTERIX in late 2012, and we are seeking a few early partners who would like to run ASTERIX on their favorite big data problems. Our ongoing work includes preparing the code base for an initial public release, completing our initial transaction story, adding additional indexing support for fuzzy queries, and providing a key-value API for applications that prefer a "NoSQL" style API over a more general query-based API. More information about the project and its current code base can be found on our project website (http://asterix.ics.uci.edu/).

It is worth pointing out ASTERIX is a counter-cultural project in several ways. First, rather than "tweaking" Hadoop or other existing packages, we set out to explore the big data platform space from the ground up. We are learning a great deal from doing so, as it is surprising just how many interesting engineering and research problems are still lurking in places related to "things that have already been done before." Second, rather than building a highly-specialized system to later be glued into a patchwork of such systems, we are exploring the feasibility of a "one size fits a bunch" system that addresses a broader set of needs (e.g., by offering data storage and indexing as well as support for external data analysis, short- and medium-sized query support as well as large batch jobs, and a key-value API as well as a query-based one).

Acknowledgments

The ASTERIX project has been supported so far by NSF IIS awards 0910989, 0910859, 0910820, and 0844574, a grant from the UC Discovery program, a matching donation from eBay, a Facebook Fellowship, and a donation of cluster access time from Yahoo! Research.

References

[1] Lohr, S. The age of big data. *The New York Times*. February 12, 2012.

[2] DeWitt, D. and Gray, J. Parallel database systems: The future of high performance database systems. *Communications of the ACM* 35, 6 (June 1992).

[3] Ghemawat, S., Gobioff, H., and Leung, S. The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems* (Lake George, Oct. 2003), 29-43.

[4] Dean, J. and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth USENIX Symposium on Operating Systems Design and Implementation* (San Francisco, December 2004), 137-150.

[5] Cattell, R. Scalable SQL and NoSQL data stores. *SIGMOD Record* 39, 4 (May 2011).

[6] Borkar, V., Carey, M., and Li, C. Inside "big data management": Ogres, onions, or parfaits. In *Proceedings of the 15th International Conference on Extending Database Technology (EDBT)* (Berlin, March 2012), 3-14.

[7] Stonebraker, M. Operating system support for database management. *Communications of the ACM* 24, 7 (July 1981).

[8] Abiteboul, S., Buneman, P., and Suciu, D. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 1999.

[9] Borkar, V., Carey, M., Grover, R., Onose, N., and Vernica, R. Hyracks: A flexible and extensible foundation for data-intensive computing. In *Proceedings of the 2011 IEEE International Conference on Data Engineering (ICDE)* (Hannover, Germany, April 2011), 1151-1162.

[10] A. Behm, A., Borkar, V., Carey, M., Grover, R., Li, C., Onose, N., Vernica, R., Deutsch, A., Papakonstantinou, Y., and Tsotras, V. ASTERIX: Towards a scalable, semistructured data platform for evolving-world models. *Distributed and Parallel Databases* 29, 3 (2011).

[11] Alsubaiee, S., Behm, A., Grover, R., Vernica, R., Borkar, V., Carey, M., and Li, C. ASTERIX: Scalable Warehouse-Style Web Data Integration. In *Proceedings of the Ninth International Workshop on Information Integration on the Web (IIWeb)* (Phoenix, May 2012).

[12] Malewicz, G., Austern, M., Bik, A., Dehnert, J., Horn, I., Leiser, N., and G. Czajkowski, G. Pregel: A system for large-scale graph processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Indianapolis, June 2010), 135-146.

Biographies

Vinayak R. Borkar is an assistant project scientist and part-time computer science Ph.D. student in the Bren School of Information and Computer Sciences at UC Irvine. He has an M.S. in computer science and engineering from IIT Bombay.

Michael J. Carey is a professor in the Bren School of Information and Computer Sciences at UC Irvine. He has a Ph.D. in computer science from UC Berkeley.

Chen Li is an associate professor in the Bren School of Information and Computer Sciences at UC Irvine. He has a Ph.D. in computer science from Stanford University.

Interactive Analysis of Big Data

New user interfaces can transform how we work with big data, and raise exciting research problems that span human-computer interaction, machine learning, and distributed systems.

By Jeffrey Heer and Sean Kandel

DOI: 10.1145/2331042.2331058

Big data is all the rage. Computer scientists in databases, distributed systems, machine learning and visualization have all trumpeted the challenge and opportunities of our unprecedented—and exponentially increasing—access to data. Across academia, many have heralded the dawn of a "fourth paradigm" of data-driven scientific research [1]. Industrial observers see a growing demand for "data scientists" skilled in making sense of everything from sensor data to health records to copious logs of social and financial transactions. Recent reports indicate that in the next decade the demand for skilled analysts will far outstrip the supply [2].

But what exactly constitutes "big data"? Petabytes? Exabytes? Yottabytes?! (Yes, yottabyte is an actual word for 1,024 bytes.) To characterize big data, we must consider multiple dimensions. Data may be tall: A database table or log file might contain billions or even trillions of records. Or, data can be wide: A single data set might contain hundreds or thousands of variables to consider. Moreover, data are often diverse: Many analyses require integrating multiple data sources with varied data types.

Each of these dimensions introduces challenges for effective analysis. Processing tall data requires scalable distributed systems and may suffer from long-running queries that stymie rapid exploration. Analysis of wide data may involve a combinatorial set of relationships among variables, complicating data quality assessment and model design. Transforming and blending diverse data (e.g., improving predictions of internal sales by incorporating public weather and population demographics data) often entails significant manual effort that is both difficult and time-consuming.

Another notion of big data with particular end-user relevance is data that is too large to manipulate on an interactive time-scale. In the face of a data deluge, what remains relatively constant is our own cognitive ability to make sense of the data and reach reliable, informed decisions. Big data is of little help when decoupled from sound judgment. Interactive analysis tools can help quell "big data" by augmenting our ability to manipulate and reason about it. For example, well-designed visualizations can leverage visual perception to help us identify patterns and form new hypotheses. Novel interfaces can enable us to iteratively transform and model subsets of data, rapidly assess initial results, and translate the resulting procedures to run on scalable backends. Enabling such interactive analysis requires research that combines systems, algorithms, and human-computer interaction in new ways.

WHY INTERACTIVITY?

The goal of interactive analysis tools is to empower data analysts to formulate and assess hypotheses in a rapid, iterative manner—thereby supporting exploration at the rate of human thought. In a recent interview study of 35 data analysts at 25 different companies [3], we observed a general pattern of work shared by most analysts. This workflow consists of data discovery and acquisition; wrangling data through reformatting, cleaning,

The HIPerWall (Highly Interactive Parallelized Display Wall) system located at the University of California, Irvine. Falko Kuester, California Institute for Telecommunications and Information Technology (Calit2). University of California, San Diego. Photo courtesy of the NSF.

and integration; profiling data to explore its contents, identify salient features, and assess data quality issues; modeling data to explain or predict phenomena; and reporting results to disseminate findings. Most of these analyses are highly iterative in nature, with analysts moving back and forth among these different tasks. For example, errors uncovered during profiling may reveal the need to acquire additional data, while feedback from readers of a report may uncover flawed assumptions or suggest improved modeling approaches.

Interactive tools for data analysis should make technically proficient users more productive while also empowering users with limited programming skills. In our interviews we found that the programming skills of professional data analysts vary widely. Some primarily work within a graphical application like Excel or SAS/JMP. Others work with scripting languages in analytic environments such as R and MATLAB. Meanwhile, proficient "hackers" use a diversity of tools and languages, including distributed computation models such as MapReduce.

For application users and scripters, the lack of interactive tools for tasks such as data reformatting and integration leaves them dependent on corporate IT departments and induces significant delays in analysis workflows. On the other hand, the overhead of writing programs (in multiple languages) for routine tasks leaves data scientists spending much of their time performing tedious data "munging"—time that could otherwise be spent gaining insights from the data.

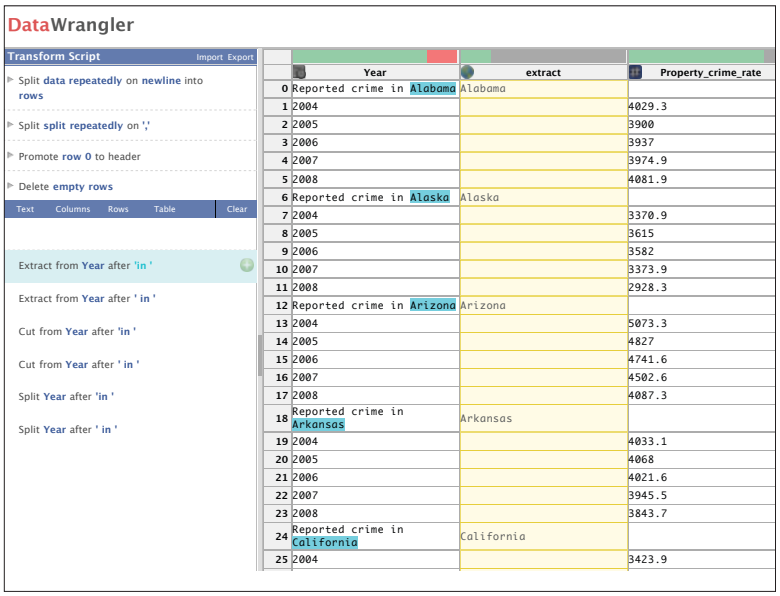
In addition, significant delays or unnecessarily complex interfaces may impede not only the pace of analysis, but also its breadth and quality. For instance, the latency of an interactive system can exert surprising effects on user activity. A study by Google engineers found that adding just 200ms of latency to search results measurably decreased the number of searches conducted by users. Even more surprisingly, this effect can persist for weeks after full performance is restored [4]. These and related results suggest unresponsive tools can significantly impact our search strategies and task performance [5]. Accordingly, interactive systems for big data must effectively orchestrate responsive client-side interfaces with slower but scalable backend processing.

The goal of facilitating interactive analysis raises exciting research questions that span systems, statistics, machine learning and human-computer interaction. How might we enable users to transform, integrate, and model data while minimizing the need for programming? How might we build scalable systems that can query and visualize data at interactive rates? How might we enable domain experts to help guide machine learning methods to produce better models? In the remainder of this article, we examine a few research projects that attempt to address some of these questions.

WRANGLING BIG DATA

One precursor to analysis—particularly with diverse data—is the tedious process of reformatting data values or layout, correcting erroneous or miss-

Figure 1. End-user programming in Data Wrangler. An analyst selects state names in a data table, indicating her desire to extract them to a new column. In response, an inference engine recommends possible operations (bottom left). Highlights in the table visually preview the results of a selected extraction rule (right).



ing values, and integrating multiple data sources. Analysts must regularly restructure data to make it palatable to databases, statistics packages and visualization tools. For example, one analyst we interviewed noted that:

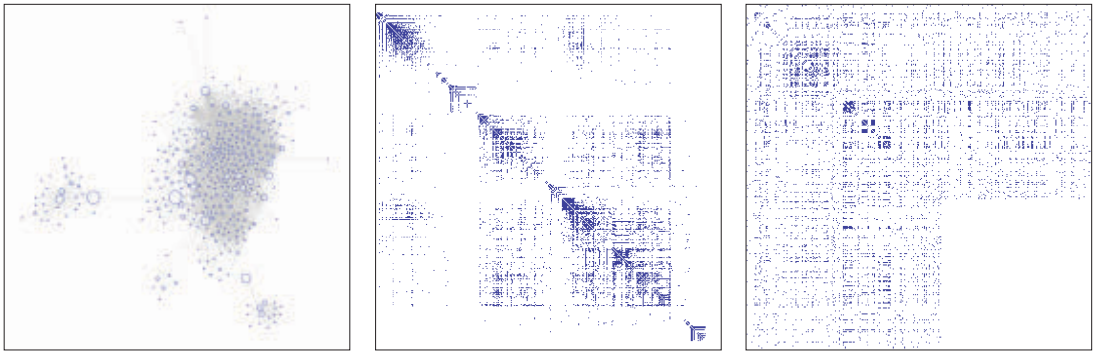
"I spend more than half of my time integrating, cleansing, and transforming data without doing any actual analysis. Most of the time I'm lucky if I get to do any 'analysis' at all!"

Others estimate that data cleaning is responsible for up to 80 percent of the development time and cost in data warehousing projects [6]. Such wrangling often requires writing idiosyncratic scripts in programming languages such as Python and Perl, or extensive manual editing using tools such as Excel. This hurdle can also discourage many people from working with data in the first place.

To assist this process, researchers have developed a number of novel interactive tools. Potters Wheel [7] and Google Refine (http://code.google.com/p/google-refine/) are menu-driven interfaces that provide access to common data transforms. Other researchers have contributed relevant algorithms for programming-by-demonstration [8]. With these methods, users first demonstrate desired actions in a user interface, for example selecting text such as addresses or phone numbers from larger strings. The system then attempts to generalize from these examples to produce robust programs, such as for address or phone number extraction [9].

Our work on Wrangler builds on these prior efforts to help analysts author expressive transformations [10]. To do so, Wrangler couples a mixed-initiative user interface with a declarative language for data transformation. Mixed-initiative systems combine automated services with direct user manipulation: As a user performs a task, the system may offer various forms of support, including automatic corrections or recommended actions [11]. Declarative programming languages express the desired result of a computation (high-level operations or properties of an output) without describing its control flow (e.g., if statements or for loops). By decoupling specification from execution, a declarative language can succinctly model a domain while freeing language designers to unobtrusively optimize processing. With Wrangler, user selections on a data

Figure 2. Assessing social network data with three different views. The choice of representation impacts the perception of data quality issues. (a) A node-link diagram does not not reveal any irregularities. (b) A matrix view sorted to emphasize connectivity shows more substructure, but no errors pop out. (c) Sorting the matrix by raw data order reveals a significant segment of missing data.



(a) (b) (c)

table trigger suggestions of possible operations, each of which is actually a statement in an underlying declarative language. As a result, the user and system work together to author scalable data transformation scripts.

Analysts using Wrangler specify transformations by building up a sequence of basic operations (see Figure 1). As users select data within a table display, Wrangler suggests applicable operations based on the current context of interaction. Meanwhile, programming-by-demonstration techniques help analysts specify complex criteria such as regular expressions. To ensure relevance, Wrangler enumerates and rank-orders possible operations using a model that incorporates user input with the observed frequency, diversity, and specification difficulty of applicable transform types. Visual previews of transformation results help analysts rapidly navigate and assess the space of viable operations.

To support rapid interaction, Wrangler works with a sample of a data set within its Web-based user interface. The result of this wrangling process is not just transformed data, but a reusable program for data transformation. The resulting program is specified in a high-level declarative language that can be cross-compiled to a variety of runtime environments, including JavaScript (for processing in the browser) as well as Python, SQL and MapReduce (for server-side processing). By interacting with a sample of data in the browser, users can generate programs that can process much larger data sets on the backend.

As an initial evaluation, we conducted a controlled user study comparing Wrangler and Excel across a set of data cleaning tasks. We found that Wrangler significantly reduced specification time: Even with small data sets (< 30 rows), median completion time with Wrangler was still twice as fast for all tasks. By producing not just data but an executable program, Wrangler also enables a level of scalability simply not possible with other graphical tools.

Of course, reformatting data is just one of many wrangling problems. Other tasks that can benefit from interactive solutions include entity resolution (for correctly matching similar but non-identical records) [12], schema mapping (for integrating disparate data sources)[13], and anomaly detection and correction (for assessing data quality issues)[14]. More research is needed into systems that leverage user interaction to solve problems resistant to automation, and which provide procedures that can be executed at scale.

The goal of interactive analysis tools is to empower data analysts to formulate and assess hypotheses in a rapid, iterative manner.

VISUALIZING BIG DATA

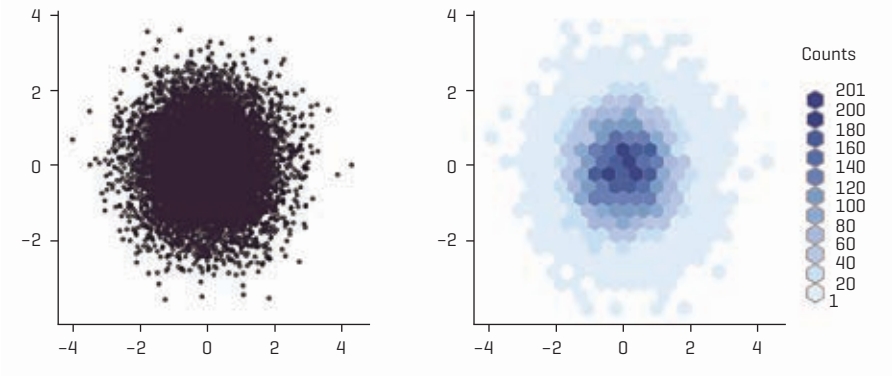
Once data has been suitably transformed, analysis can begin in earnest. Exploratory analysis through visualization is often a critical component for assessing data quality and developing hypotheses.

For an example of data quality assessment, consider the social network diagrams in Figure 2. The data consist of a social network of friends, extracted from Facebook using their Web API. Figure 2(a) visualizes the data as a node-link diagram with nodes placed via force-directed layout. We can see that the data contains multiple clusters, but not much else. Figure 2(b) shows the same data as a matrix diagram; the rows and columns represent people and filled cells represent a connection between them. Following best practices, we automatically permute (or "seriate") the rows and columns of the matrix to minimize the distance between highly-connected people. One can see clusters of friendship communities along the diagonal, revealing more substructure than is apparent in the node-link view.

However, for the purposes of data cleaning, the "raw" visualization in Figure 2(c) is the most revealing. The rows and columns are sorted in the order provided by the Facebook API. We now see a striking pattern: The bottom-right corner of the matrix is completely empty. Indeed, this is a missing data problem that arose because Facebook enforced a 5,000 item result limit per query. In this case, the maximum was reached, the query failed silently, and the mistake went unnoticed until visualized. As this example indicates, choices of representation (e.g., matrix-diagram) and interactive parameterization (e.g., default sort order) can be critical to unearthing data quality issues that can otherwise undermine accurate analysis.

The challenges of effective visualization become more acute as the data grow larger. For tall data, a multitude of records can lead to crowded, uninformative displays. Consider the scatterplot in Figure 3; with only thousands of points, the display becomes cluttered and difficult to interpret. A scalable alter-

Figure 3. Normal (left) and binned (right) scatter plots. Adapted from [14].



native is a binned scatterplot, which can faithfully convey the underlying distribution while preserving observation of outliers through a careful color encoding. In this case, hexagonal bins are chosen because they provide a (slightly) more efficient approximation of density than rectangular bins [15].

This example illustrates a more general design principle: The perceptual scalability of a data display should be limited by the chosen resolution of the data, not the number of records. In the example, binning is used to limit the resolution of the data. A different approach that also adheres to our principle would be to show a representative sample with a bounded number of points. These two strategies might also be combined: Aggregate views of appropriately chosen samples may provide a close approximation for the full data. To support these methods, pre-processing is necessary to prepare the data for visualization. Additional challenges accrue when attempting to supporting rapid interaction, such as dynamic filtering and linked selection across visualization views.

Wide data with many variables pose another set of difficulties. As a first step, visualization techniques such as scatterplot matrices or parallel coordinates can help reveal multidimensional patterns [16]. However, these methods also have scalability limits, visualizing at most a few dozen variables at once. An alternative is to use mixed-initiative methods to recommend subsets of related dimensions. For example, an analyst might select a small set of variables that she is interested in. In response, the system analyzes the degree to which other attributes in the data predict the chosen variables (e.g., via mutual information or other measures of correlation) and produces visualizations for just the subset of highly explanatory attributes. Similar techniques have been used to automatically construct multiview displays for assessing anomalies such as missing values or extreme outliers [14]. An important component of such intelligent interfaces is to keep the user in control, enabling them to modify or override algorithmic recommendations. Corresponding research challenges include the development of accurate and performant recommendation algorithms coupled with the design of usable interaction and visualization methods.

Interactive tools for data analysis should make technically proficient users more productive while also empowering users with limited programming skills.

GOING FORWARD

The previous examples only begin to scratch the surface, touching on issues that primarily stem from wrangling and profiling activities. Additional research problems abound throughout the lifecycle of data analysis. How might improved data indexing, metadata, and search methods facilitate data discovery? How might we design effective interactive systems not only for wrangling individual tables, but for performing data integration? Or for manipulating text, image, or video data? Or creating, assessing, and actively guiding machine learning models for classification or prediction? And how might we best record and represent the analysis process to aid auditing, sharing and reuse? As the diversity, size, and availability of relevant data continues to increase, the design of novel interactive tools to aid analysis will remain an exciting and important topic for computer science research.

Acknowledgments

We thank Joe Hellerstein, Andreas Paepcke, Pat Hanrahan, Jock Mackinlay, Zhicheng Liu, Philip Guo, and Ravi Parikh for ideas and feedback that informed this article.

References

[1] Hey, T., Tansley, S., and Tolle, K.ed. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, 2009.

[2] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. *Big data: The next frontier for innovation, competition, and productivity.* McKinsey Global Institute, May 2011.

[3] Kandel, S., Paepcke, A., Hellerstein, J. M., and Heer, J. Enterprise data analysis and visualization: An interview study. In *Proc. IEEE Visual Analytics Science & Techology* (VAST), 2012.

[4] Brutlag ,J. Speed Matters. Google. Research Blog. June 23, 2009; http://googleresearch.blogspot.com/2009/06/speed-matters.html

[5] Gray, W. D., and Boehm-Davis, D. A. Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied* 6, 4 (2000),322–335.

[6] Dasu, T., and Johnson, T. *Exploratory Data Mining and Data Cleaning.* John Wiley & Sons, Inc., New York, 2003.

[7] Raman, V., and Hellerstein, J. M. Potter's wheel: An interactive data cleaning system. In *Proceedings of the 27th International Conference on Very Large Data Bases* (Rome, Sept. 11-14). Morgan Kaufmann, San Francisco, 2001, 381–390.

[8] Cypher, A. *Watch What I Do: Programming by Demonstration.* MIT Press, Cambridge, MA, 1993.

[9] Gulwani, S. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the 38th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Austin, Jan. 26-28). ACM Press, New York, 2011, 317–330.

[10] Kandel, S., Paepcke, S., Hellerstein, J. M., and Heer, J. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the 2011Annual Conference of Human Factors in Computing Systems* (Vancouver, May 7-12). ACM Press, New York, 2011, 3363–3372.

[11] Horvitz, E. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Pittsburgh, May 15-20). ACM Press, New York, 1999, 159–166.

[12] Kang, H., Getoor, L., Shneiderman, B., Bilgic, M., and Licamele, L. Interactive entity resolution in relational data: A visual analytic tool and its evaluation. *IEEE Transactions on Visualization & Computer Graphics* 14, 5 (2008),999–1014.

[13] Robertson, G. G., Czerwinski, M. P., and Churchill, J. E. Visualization of mappings between schemas. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Portland, April 2-7). ACM Press, New York, 2005, 431–439.

[14] Kandel, S., Parikh, R., Paepcke, A., Hellerstein, J. M., and Heer, J. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (Capri Island, Italy, May 22-25). ACM Press, New York, 2012, 547–554.

[15] Carr, D. B., Littlefield, R. J., Nicholson, W. L., and Littlefield, J. S. Scatterplot matrix techniques for large *N. Journal of the American Statistical Association*, 82, 398 (1987),424–436.

[16] Heer, J., Bostock, M., and Ogievetsky, V. A tour through the visualization zoo. *Communications of the ACM* 53, 6 (2010), 59–67.

Biographies

Jeffrey Heer is an assistant professor of computer science at Stanford University, where he works on human-computer interaction, visualization, and social computing. The visualization tools developed by his lab (Prefuse, Flare, Protovis and D3) are used by researchers, corporations and thousands of data enthusiasts around the world. Heer holds B.S., M.S., and Ph.D. degrees in computer science from the University of California, Berkeley.

Sean Kandel is a Ph.D. candidate in the Stanford University Computer Science Department. His research combines human-computer interaction and database systems, resulting in new interactive systems for data management and exploration.

Visit the new Website for

ACM Inroads

The magazine for computing educators worldwide

http://inroads.acm.org

Paving the way toward excellence in computing education

acm Inroads
PAVING THE WAY TOWARD EXCELLENCE IN COMPUTING EDUCATION

Association for Computing Machinery
Advancing Computing as a Science & Profession

Propagation and Immunization in Large Networks

Many interesting research questions can be explored by studying processes running over networks.

By B. Aditya Prakash

DOI: 10.1145/2331042.2331059

How do contagions spread in populations? Who are the best people to vaccinate? Which group should we market to for maximizing product penetration? Will a given YouTube video, meme, or link go viral? And what happens when two products compete?

One feature these questions have in common is they are all important research problems (see the titles listed at the end of this article for further reading). The other is that they all can be characterized as problems based on giant graphs (networks) of nodes (people) and edges (relationships). Such networks are ubiquitous, from online social networks and gene-regulatory networks to router graphs. Networks effectively model a wide range of phenomena by exposing local-dependencies while simultaneously capturing large-scale structure. Questions such as how blackouts can spread on a nationwide scale, how social systems evolve on the basis of individual interactions, or how efficiently we can search data on large networks of blogs or websites, are all related to phenomena on networks. Clearly progress here holds great scientific as well as commercial value.

Big data is a natural and necessary part of research in this sphere. Although the actions of a particular individual or component may be too difficult to model, data mining and machine learning can be applied to large groups or ensembles, in turn yielding effective models with the ability to predict future events. For instance, modeling the response of every individual to a particular marketing strategy might be too difficult, but modeling the behavior of large groups of people based on demographics and geography is feasible. Models are useful as they allow us to abstract out the process and simulate it on our machines, and we can then try to explore even more complex issues using these models. For example, how should we distribute resources to control an epidemic? How should we manage communities to make them more productive? And how can we design these policies so that they can be implemented on an extremely large-scale?

Invariably, solving such problems involves working with huge amounts of data—millions of users, billions of tweets, and trillions of network connections—as well as designing algorithms and experiments using generated models, which can themselves be run on large and complex data. Two trends have emerged to allow such an approach: The increasing ability to collect more and more data, and the increasing ability to run more and more large-scale and complex models. In the past, when the amount of data available was small and computing power was limited, researchers used markedly different approaches. Sociologists, for example, used to collect small samples and then extrapolate to develop very sophisticated models. We are now in a position to do the opposite. Through the emergence of big data, we can develop and test increasingly abstracted models on larger and larger sample sizes.

Dynamical processes over net-

works can give rise to astonishing macroscopic behavior, leading to challenging and exciting research problems. How stable is a predator-prey ecosystem, given intricate food webs? How do rumors spread on Twitter/Facebook? How should we administer software patches optimally? Herein, we will try to illustrate some big-data challenges using two problems related to dynamical phenomena (like propagation) on large networks: thresholds and immunization. Their applications are broad, and these problems are central to surprisingly diverse areas including cyber security, epidemiology, and public health, through to product marketing and information dissemination.

TIPPING POINTS AND THRESHOLDS

Consider the following problem: Given a network of who-contacts-whom, will a contagious virus "take-over" (cause an epidemic) or die-out quickly? What will change if nodes have partial, temporary, or permanent immunity? What if the underlying network changes over time, e.g., if people have different connections during the day at work, and during the night at home? An important concept in answering these questions is the "epidemic threshold," which is the minimum level of virulence required to prevent a viral contagion from dying out quickly. Determining the epidemic threshold is a fundamental question in epidemiology and related areas.

Apart from the fundamental nature of this problem, it turns out that it is also very helpful in running large-scale epidemiological simulations. While it is very difficult to model each and every person's response to a disease, it is much more feasible to run epidemic simulations on huge populations (city or nationwide) to understand which sections get infected, which should be quarantined, and so on. One problem here is that running big simulations—potentially involving hundreds of machines—is very expensive. How can we speed up these simulations to enable more useful and more frequent runs? The epidemic threshold problem we described above comes to our need: We don't need to run simulations when the disease or virus in question is "below-threshold," thus speeding-up the simulations.

Surprisingly, it can be shown that when the underlying contact-network does not change over time [1], the threshold condition is,

$$\lambda_1 C < 1$$

Where λ_1 is the first eigenvalue of the connectivity matrix, and C is a virus-model dependent constant. This holds true for (a) any graph; and (b) all propagation models in standard literature, including the AIDS virus HIV and more than 25 others from canonical texts [2]. So, the result we achieve decouples the effect of the topology and the virus model. What makes the result practical is the eigenvalue computation on graphs is linear-time in the size of the graph, and also can be efficiently parallelized on Hadoop [3].

What exactly is λ_1? Algebraically, it is simply the eigenvalue of the underlying adjacency matrix with the largest magnitude. Intuitively though, it captures how vulnerable the graph is for an epidemic (a concept which will prove useful later too). Roughly, it describes the number of paths between pairs of nodes in a graph, discounting for longer paths, effectively controlling the number of ways the virus can spread. Hence, the larger is λ_1, the better the graph's connectivity for the virus (see Figure 1).

Figure 2 demonstrates the result of computer simulation experiments on a large public dataset representing a synthetic population of the city of Portland, OR [4]. The dataset is based

Figure 1. Why λ_1 matters more than number of edges E. Changing connectivity and vulnerability of graphs with changing λ_1. The clique (largest λ_1) is the most vulnerable. Note that E is not enough. Star and chain have the same number of edges (E = 4), but the star is intuitively more vulnerable (it also has a higher λ_1).

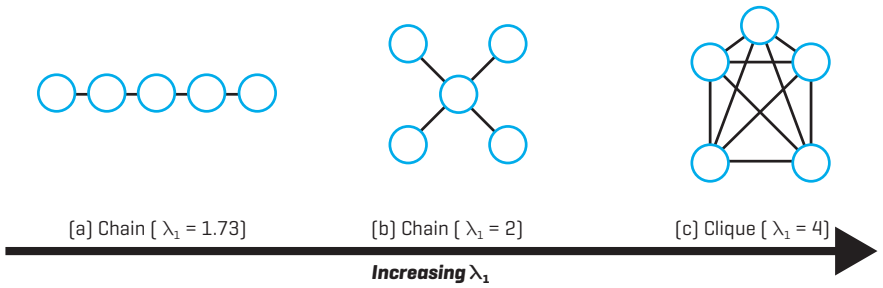
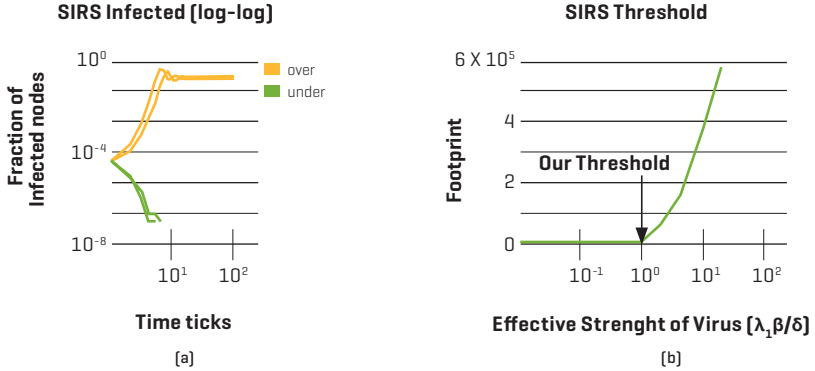


Figure 2. The tipping-point. Simulation results on a massive social-contact graph PORTLAND (31 mil. edges, 1.5 mil. nodes) and the SIRS model (temporary immunity like pertussis). (a) Plot of Infected Fraction of Population versus Time (log-log). Note the qualitative difference in behavior under (green) the threshold and above (red) the threshold. (b) Footprint (expected final epidemic size) versus Effective Strength (lin-log). Notice the prediction is exactly at the take-off point.



on detailed microscopic simulation-based modeling and integration techniques, and has been used in modeling studies on smallpox outbreaks. It is a social-contact graph containing 31,204,286 links (interactions) among 1,588,212 nodes (people). The simulations were conducted using the so-called "SIRS" model, which models diseases to which we can become temporarily immune (like *pertussis*, more commonly known as whooping cough). In such cases, an infected person develops immunity, which he or she ultimately loses, thereby becoming susceptible to the disease again. Figure 2(a) plots infected population versus time, showing clear broad differences between the curves when the disease strength is under and above the epidemic threshold (according to Equation 1). In particular, as Figure 2(b) shows, the final epidemic size changes abruptly exactly at our predicted tipping-point (i.e. when $\lambda_1 C = 1$).

Through the emergence of big data, we can develop and test increasingly abstracted models on larger and larger sample sizes.

FAST IMMUNIZATION

Consider the problem of prevention of hospital-to-hospital transfer of drug resistant bacteria. Critically ill patients are frequently and routinely transferred between hospitals in order to provide necessary specialized care. While such inter-hospital transfers are an essential part of routine patient care, they also enable the transfer from hospital to hospital of highly virulent microorganisms resistant to many or all antibiotics. So, given a fixed amount of medicines with partial impact, like bottles of disinfectant, how should they be distributed among hospitals?

Due to the scale of this problem, any method linear in the size of the graph is better. However, an easily parallelizable algorithm would be even better. Since doctors may have different resources, each with different effectiveness, clinicians want to get good allocations quickly so that a coherent policy can be constructed and deployed. At the same time, the policy should not sacrifice accuracy. The current practice in allocating varying amounts of antidote across a network is essentially uniform, with hospitals independently tackling infection control. However, this makes no use of the connected network we are given. Another obvious method is to estimate the effect of medicines through computer simulations. However, such simulations are computationally expensive and can often take weeks to run. Given these constraints, how can we get a practical and effective algorithm?

Collaborating with domain experts, we studied this problem and developed a fast and effective linear-time algorithm titled SMART-ALLOC [5]. Recall from our tipping-point discussion the connectivity of the network (in the form of λ_1) controls the vulnerability of a graph to an infection. Hence, we just need to decrease this value as fast as possible. It turns out that this problem is NP-hard. So, under the hood, SMART-ALLOC tries to drop the eigenvalue in a smart way. In particular, the special form of the impact function of a resource allowed us to get a provably near-optimal greedy solution. SMART-ALLOC runs in seconds on commodity hardware, as opposed to weeks required for other approaches. Figure 3 demonstrates the algorithm on the network of US-MEDICARE patient transfers.

Crucially, these results show significant benefits can be achieved by concentrating infection-control resources at a carefully chosen subset of nodes, rather than doing so in a network-agnostic fashion or using ad-hoc heuristics. The current practice has been largely focused within individual hospitals. Hence, current public-health policy is missing an opportunity to significantly reduce infection rates

Figure 3. SMART-ALLOC method has six times fewer infections (red circles). (a) The US-MEDICARE network of hospitals overlaid on a map. (b) Infected hospitals after a year (365 days) under current practice. (c) Similarly, under SMART-ALLOC. The current practice allocates equal amounts of resource to each hospital.

(a) US-MEDICARE Inter-hospital Connections

(b) Infected Hospitals (in red) under current practice

(c) Infected Hospitals (in red) under SMART-ALLOC

with an infection prevention strategy that accounts for the potential transfer of bacteria along the network of inter-hospital patient transfers.

This approach can also be extended to other scenarios, like when we can completely remove a node (i.e. vaccinate it). For example, given a large network, such as a computer communication network, which k nodes should we remove (or monitor, or immunize), to make the network as robust as possible against a computer virus attack? Making careful approximations, NETSHIELD [6] exploits the submodular structure of the set of possible solutions, getting a simple provably near-optimal algorithm.

Further, the inner-loops of both these algorithms use eigenvalue computation on graphs, which, as we have already seen earlier in this article, are very efficient to compute.

CONCLUSION

Graphs—also known as networks—are powerful tools for modeling processes and situations of interest in real-life, including social-systems, cyber-security, epidemiology, and biology. In this article we reviewed two recent developments in studying propagation-like processes on large networks: The importance of eigenvalue in understanding the tipping-point of epidemics, and subsequently leveraging that to design fast and scalable immunization policies. There are several other extensions, like having competing viruses [7] or networks that change over time [8], which we did not have space to describe here.

Really, we have given just a glimpse of the types of big-data questions we encounter after we have already built models. How can we use these models for our benefit, to actually manipulate something we care about? For example, after building models of both diseases and the underlying population, how can we study the interactions between them? How can we design policies to do effective immunization? All of these questions have to be answered in the context that we are trying to both understand and manage real-life processes on a societal-scale. These are pretty exciting times for research in networks.

References

[1] Prakash, B. A., Chakrabarti, D. , Faloutsos, M. , Valler, N., and Faloutsos, C. Threshold conditions for arbitrary cascade models on arbitrary networks. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining* (Vancouver, Dec. 11-14). IEEE Computer Society, Washington, DC, 2011, 537-546.

[2] Anderson, R. M., and May, R. M. *Infectious Diseases of Humans*. Oxford University Press, Oxford, 1991.

[3] Kang, U. Mining Tera-Scale Graphs: Theory, engineering and discoveries. Ph.D. thesis, Carnegie Mellon University, 2012.

[4] Eubank, S., Guclu, H. , Anil Kumar, V. S. , Marathe, M. V., Srinivasan, A., Toroczkai, Z., and Wang, N. Modelling disease outbreaks in realistic urban social networks. *Nature* 429 ,6988 (2004), 180–184.

[5] Prakash, B. A., Adamic, L., Iwashnya, T., Tong, H., and Faloutsos, C. Fractional immunization in networks. Under review, 2011.

[6] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *Proceedings of the 2010 IEEE International Conference on Data Mining*. IEEE Computer Society, Washington, DC, 2010, 1091-1096.

[7] Prakash, B. A., Beutel, A. , Rosenfeld, R. , and Faloutsos, C. Winner takes all: Competing viruses or ideas on fair-play networks. In *Proceedings of the 21st International Conference on World Wide Web* (Lyon, France, April 16-20). ACM Press, New York, 2012, 1037-1046.

[8] Prakash, B. A. , H. Tong, N. Valler, M. Faloutsos, and C. Faloutsos. Virus propagation on time-varying networks: Theory and immunization algorithms. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III* (Barcelona, Sept. 20-24). Springer-Verlag, Berlin, 2010, 99-114.

Further Reading

E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st International Conference on World Wide Web* (Lyon, France, April 16-20). ACM Press, New York, 2012, 1037-1046.

D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Washington, DC, Aug. 24-27). ACM Press, New York, 2003, 137-146 .

J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, June 28-July 1). ACM Press, New York, 2009, 497-506.

M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Canada, July 23-25). ACM Press, New York, 2002, 61-70.

Biography

B. Aditya Prakash is a Ph. D. student in the Computer Science Department at Carnegie Mellon University. He received his B. Tech. in computer science from the Indian Institute of Technology (IIT) Bombay. His research interests include data mining, applied machine learning and databases, with an emphasis on large real-world networks and time-series. He will soon be joining the Computer Science Department at Virginia Tech as an assistant professor.

Parallel Machine Learning on Big Data

On algorithms for parallel machine learning, and why they need to be more efficient.

By John Langford

DOI: 10.1145/2331042.2331060

Parallel machine learning is a very exciting topic. One can potentially use massive quantities of data about the real world to predict useful things. For example, can a computer tell whether or not a particular email is unwanted spam? Can a computer determine which search result is best? Can a computer recognize what is pictured? Parallel machine learning on big data provides a means to explore and potentially answer these questions.

Nevertheless, serious caution is in order. Efforts to create effective parallel machine learning algorithms have existed since at least the 1980s, but with a low success rate. Reasons include:

1. **The parallel algorithm is compared with a slow but easily parallelized sequential algorithm.** This means that, in practice, people just use the sequential algorithm. A canonical example of this is batch gradient descent (slow and easily parallelized) rather than stochastic gradient descent.

2. **There is an investment decision about where time should be spent.** The choice is either create a faster (or better) sequential learning algorithm or a parallel learning algorithm. In the past, it was almost always better to spend time on the faster sequential learning algorithm.

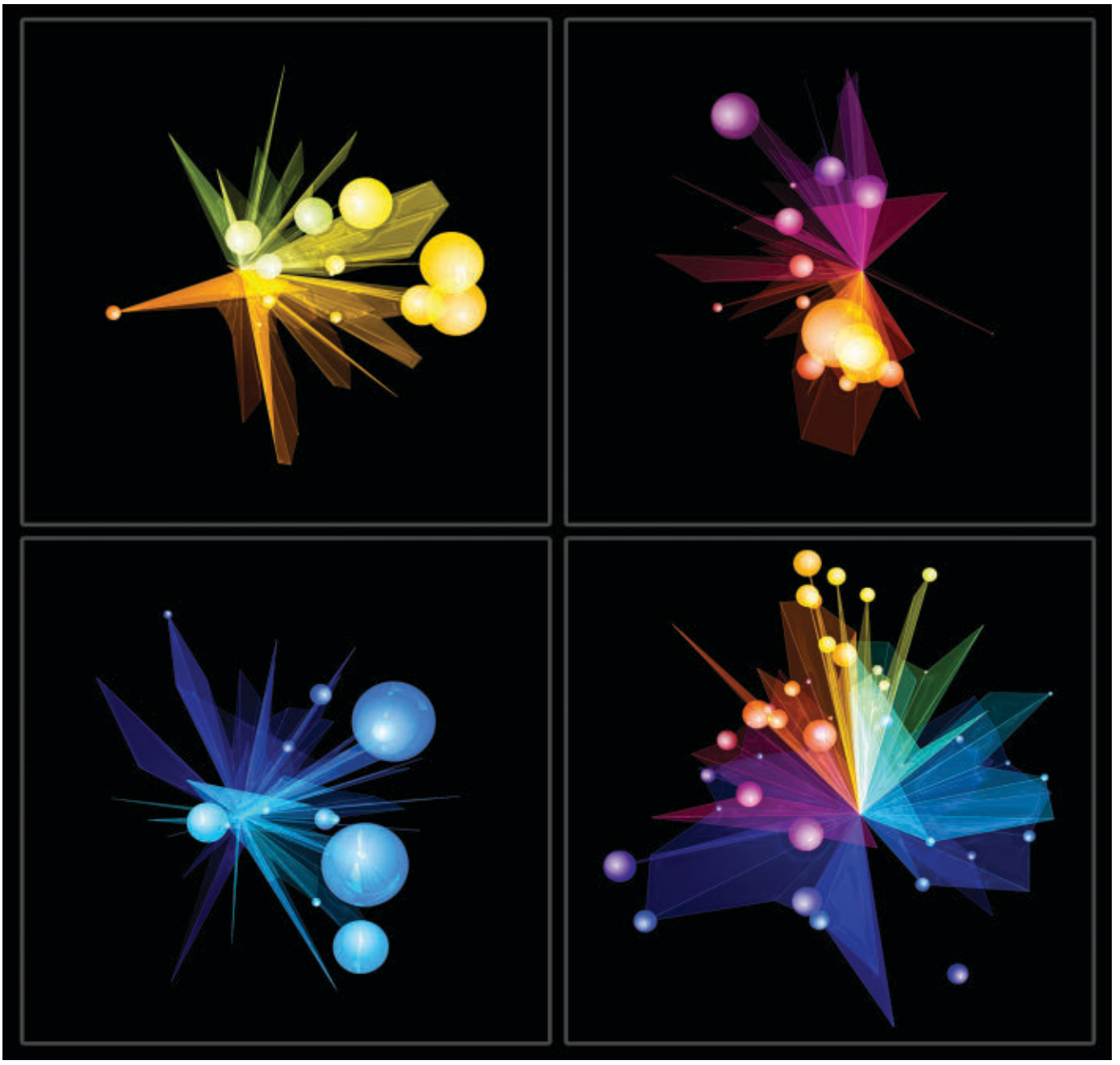
3. **Parallel algorithms are often hard to use, with extra libraries and additional CPUs required to see significant benefits.** Not many people have access to these components, and the overhead of setup can be significant, even for those who do have access.

4. **The actual need for parallel machine learning is fairly small.** Most machine learning problems use small amounts of data that are easily represented and manipulated on a single machine. If a learning algorithm is excessively slow given even a modest amount of data, then it's probably the wrong learning algorithm for the dataset.

5. **Even when the amount of data is not small, an arsenal of simple tricks can make sequential algorithms tractable.** Such tricks are easy but are not systematically taught, leading to chronic overlook. The canonical example of a simple trick is downsampling—when you want to learn a linear predictor on 100 parameters and have access to 1 billion samples, you should discard all but 0.01 percent of the data and employ a simple single-machine learning algorithm.

The generally high barrier to entry makes it difficult to develop new parallel learning algorithms, so it is critical that attempts to do so are done with the proper support.

How can we overcome all of these difficulties? The good news is that they are not without solution. Nowadays, one core difference is the systematic digitization and storage of many forms of data, providing us with



Generative artwork by Gwen Vanhee. Created in Actionscript 3.0.

extremely large datasets. However, there has been a failure to scale up processor speeds. The response on the computer architecture side has been to provide more, rather than faster, CPUs, slowly addressing difficulty No. 3. Parallelism is now routine, and software techniques for using it are improving in viability.

Difficulty No. 4 is addressed by noting that some of the big datasets really matter. Measured in dollars, good solutions to the "ad display problem" (and high frequency trading in general) are easily worth many billions. Measured in time, a good spam filter or optimized search engine can save millennia of time per day.

Difficulty No. 5 is addressed by noting that many of these problems are inherently complex. What is the inherent complexity of the function that always returns the best answer given any question by anyone anywhere? If a significant portion of that function is parameterized and the parameters are learned, we want significant quantities of data directly informing those parameter choices.

This leaves difficulties No. 1 and No. 2, both of which can be dealt with by researchers.

TOWARD MORE EFFICIENT ALGORITHMS

Recently, my colleagues and I edited a book surveying the state of the art in parallel machine learning [1]. Based on this, we created a survey tutorial, which provides a high-level view of the state of public research alongside a summary of the book's contents [2]. Of particular interest to me are the quantifications of gross computational performance in Part 3, where we quantified the computational performance of each algorithm while neglecting the predictive performance. The core unit of interest was a feature (i.e., a nonzero

entry in a data matrix), and we measured algorithms according to their features/second.

Looking through the results, we saw two things. First, the notion of "large scale" is broadly varying over many orders of magnitude. This is important because, in any discussion about parallel learning, it is critical to nail down exactly what is meant by large scale—use of the term tends to vary radically by area and background within machine learning. Second, with the exception of one system (discussed later in this article), the most efficient public learning systems top out at 107 features/second. (There are several private machine-learning systems, but we could not evaluate them without access).

Our system, Vowpal Wabbit, managed $5 * 10^8$ features/second on a 2.1 terafeature dataset using 1,000 nodes [3]. Many tricks were used to achieve this result. The principal ones included an efficient online learning algorithm, which reached a near optimal solution in a single pass over the data, and an efficient batch-learning algorithm (L-BFGS), which transformed the near optimal solution to an optimal solution. We also applied hashing to features to reduce the parameter dimensionality, and made use of three other techniques: An efficient implementation with dataset caching to minimize network use; a system for moving computation to data rather than vice-versa; and a system for efficient synchronization of learning algorithm state (see Agarwal et al. for more detail on these tricks [4]). This "many-things-right structure" appears necessary for a high performing system, complicating research into parallel learning algorithms.

In addition to the "many-things-right" aspect of the learning algorithm, access to interesting data and sufficient hardware and software support to use the algorithms are required. With respect to software, Hadoop (http://hadoop.apache.org/) is perhaps the most common adequate open data processing platform, and the most commonly utilized hardware is the x86 processor. Working with these minimizes the difficulty in starting up and will make your work more widely useful. The generally high barrier to entry makes it difficult to develop new parallel learning algorithms, so it is critical that attempts to do so are done with the proper support.

In any discussion about parallel learning, it is critical to nail down exactly what is meant by large scale—use of the term tends to vary radically by area and background within machine learning.

It is also important to note that the first four tricks utilized in Vowpal Wabbit (were developed individually and without reference to parallel learning. Restated, improvements in core learning algorithms, representation, and learning systems contribute substantially to the effectiveness of a parallel learning algorithm, and these are much easier to work on individually, particularly after their role in the design of a high performance system is understood.

CONDUCTING PARALLEL ALGORITHMS RESEARCH

What else is required for successful research in this area? When working on a parallel machine-learning algorithm, it is very important that one is able to perceive and be dissatisfied with defects. For instance, online learning is a good algorithm, but it can be slow to converge when working on poorly normalized datasets. L-BFGS is a good batch-learning algorithm, but it can be particularly slow to converge initially. Standard dictionary building approaches are perfect at not losing information, but are also very RAM intensive.

Furthermore, the standard text formats that learning algorithms use as inputs are relatively readable but are also relatively difficult to parse. Moving data to the program is standard, but can often be inefficient by a factor of 1,000. And, although MapReduce is an effective mechanism for aggregating data, we really need to pair it with broadcast of that aggregation (an All-Reduce operation) and the operation can be a factor of 1,000 or so more efficient at hardware limits. [For more on MapReduce see Jeff Ullman's article on page 30.]

In a spirit of dissatisfaction, it seems important to realize and confront other radical inefficiencies that exist in machine learning algorithms today. Let us round off this article with two examples, both of which provide useful directions for future research.

First, when predicting one of k choices, conventional approaches require $\Omega(k)$ computation (or worse) while the lower bound is $\Omega(\log k)$. Although we have various logarithmic time approaches, they are relatively nonstandard and have caveats that make them difficult to apply in various settings. Can these caveats be removed?

Second, learning on nonlinear representations tends to be radically less efficient than learning on a linear representation. Is it possible to learn an effective nonlinear representation quickly enough to effectively use large quantities of data? The goal here would be a learning algorithm only a constant factor slower than linear learning while still providing the representational power of a boosted decision tree, for example.

References

[1] Bekkerman, R., Bilenko, M., and Langford, J., *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, Cambridge, UK, 2011.

[2] Scaling up Machine Learning, The Tutorial. http://hunch.net/~large_scale_survey/

[3] Vowpal Rabbit (Fast Learning). http://hunch.net/~vw

[4] Agarwal, A., Chapelle, O., Dudik, M., and Langford, J., 2012. A Reliable Effective Terascale Linear Learning System. Available at: http://arxiv.org/abs/1110.4198

Biography

John Langford is a machine learning research scientist. He is the author of the weblog hunch.net and the principal developer of Vowpal Wabbit. Langford works at Microsoft Research and was previously affiliated with Yahoo Research, Toyota Technological Institute, and IBM's Watson Research Center. He studied physics and computer science at the California Institute of Technology, earning a double bachelor's degree in 1997. He received his Ph.D. in computer science from Carnegie Mellon University in 2002.

interactions

EXPERIENCES | PEOPLE | TECHNOLOGY

interactions magazine is pleased to introduce its new website—interactions.acm.org—designed to capture the influential voice of its print component in covering the fields that envelop the study of people and computers.

The site offers a rich history of the conversations, collaborations, and discoveries from issues past, present, and future.

Check out the current issue, look up a past prototype, or discuss an upcoming trend in the communities of design and human-computer interaction.

FEATURES

BLOGS

FORUMS

DOWNLOADS

Welcome to
interactions.acm.org

Association for Computing Machinery

Big Data in Computational Biology

An invitation to the digital science of life.

By Cliburn Chan

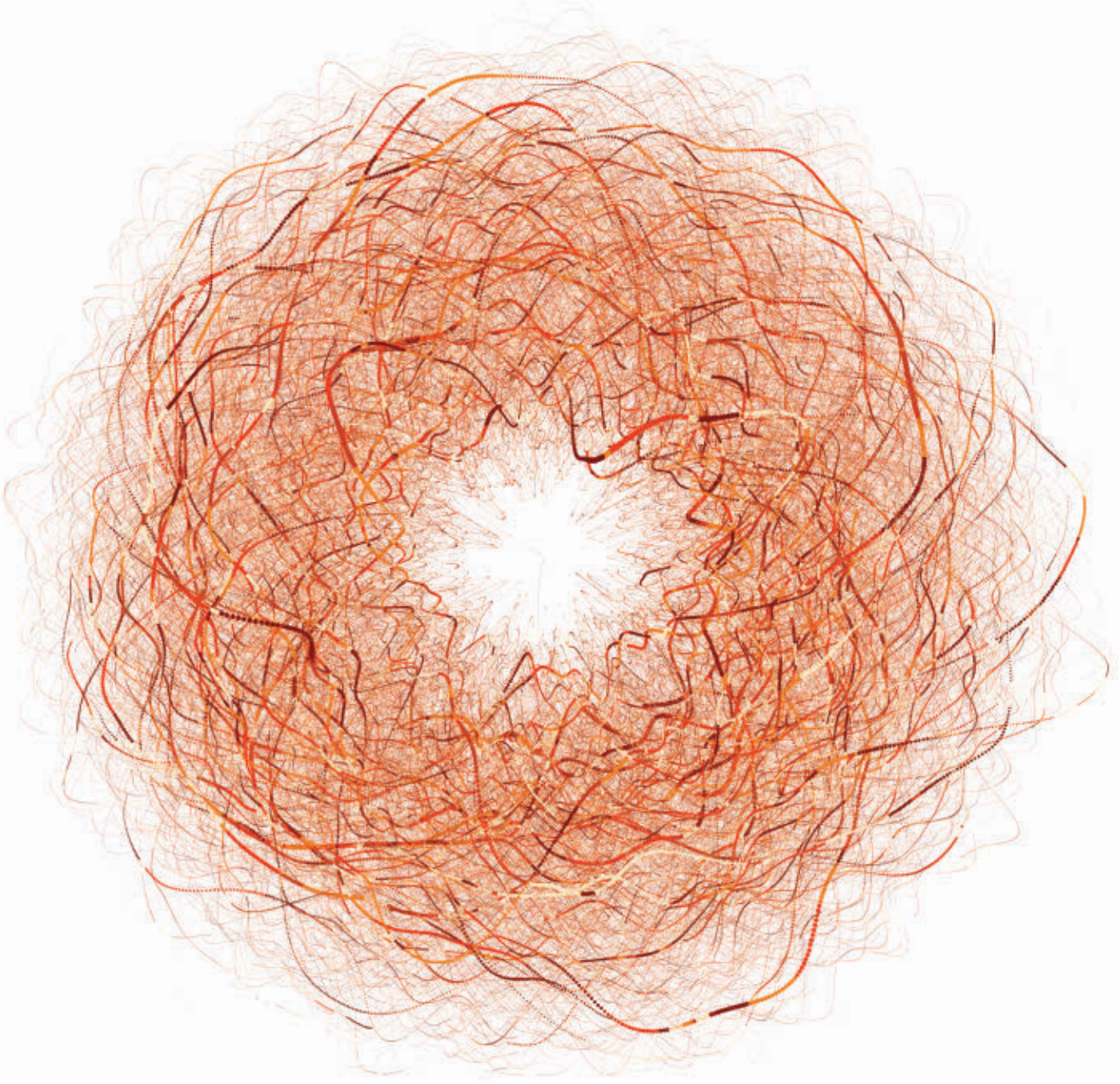
DOI: 10.1145/2331042.2331061

The amount of data generated by next-gen sequencing (NGS) machines is now doubling every five months and the trend is expected to continue for the next few years [1]. In contrast, the number of transistors on a chip only doubles every two years (Moore's law), with chip performance doubling at a slightly faster rate of 18 months (attributed by Intel executive David House). Equivalently, the doubling time for drive capacity is also about 18 months. Hence, the growth rate of sequence data generation is outpacing that of hardware capabilities by a factor of approximately four every year. Without human ingenuity, it is apparent that not only will we be restricted to analyzing an ever-smaller fraction of the data generated, we may not even have the capacity to store all the data generated.

While the explosion in data generated by sequencing machines has generated the most attention, parallel developments are occurring in all fields of biomedicine. Epigenomics, transcriptomics, proteomics, metabolomics, functional genomics, structural biology, single cell analysis, and biomedical imaging have similar explosive growth in data generation. With the transition to electronic health records, clinical data analysis will also be joining the rich data party. Interestingly, increasingly large data sets are also being generated by computer simulations, which often have to be stored and analyzed in the same way as biological assay data. For example, agent-based simulations, which are increasingly popular in the study of complex adaptive systems such as the brain or immune system, simulate individual entities (cells, people) as autonomous entities and track the properties of these agents over time. Said simulations can generate massive amounts of data. In order to meet the challenges of big data in biology and medicine, fundamental innovations in data structures and algorithms will be critical. The same can be said about breakthroughs in database technologies, bioinformatics, machine learning, and systems biology. This is a great time for students of computer science with an interest in biology and medicine to become involved with opportunities that are as vast as the challenges presented.

USES OF BIG DATA

How can we use big data in biomedicine? Grossly oversimplifying, big data is currently used for understanding disease risk in individuals, and to a lesser extent, for providing insight into disease mechanisms. An example of how big data is used for linking risk of disease to personal biomedical data are genome-wide association studies (GWAS) that make use of single-nucleotide polymorphisms (SNP) arrays to probe for hundreds of thousands to millions of genetic variants. In typical case-control studies, variations in the frequencies of SNPs are then used to find SNPs associated with the disease being studied. Similar association studies are widely used for data from other genomic assays such as full sequence reads, expression arrays, proteomics, and metabolomics. The ultimate goal of this research is to create a database of disease signatures that can be used to predict the risk of disease in an individual, and then customize appropriate prevention or therapeutic efforts for personalized medicine. One caveat with the possibility of such massive data mining is a high risk of



Visualization by Jer Thorp of data associated with the UK's National DNA Database. Created in Processing for Wired UK.

false positive results. Fortunately, well-established statistical methods that limit such false positives are available (e.g. permutation resampling methods to control the family-wise Type 1 error rate), but the lessons learned by biostatisticians may not have fully filtered down to all research communities. A notorious poster reports on the use of standard functional brain imaging analysis methods to demonstrate "a dead salmon perceiving humans can tell their emotional state" [2].

The use of big data for providing insight into disease mechanisms is less mature; this is a challenging problem for which the appropriate mathematical and statistical framework for analysis is less defined. Understanding disease mechanisms from big data requires tight feedback loops between experimental research and computational analysis. Cohesive inter-disciplinary teams that can perform such work are rare. Finally, the nature of the current data being generated is often highly homogeneous (e.g. DNA strings) and not ideal for mechanism discovery that may require linking multiple types of data over several time points. Although mechanistic models based on rich data may be used in the future, the analysis of big data has already revealed several surprising challenges to our biological knowledge. One surprise was the discovery that non-coding DNA (accounting for more than 90 percent of our DNA and sometimes derogatively labeled "junk" DNA) is highly evolutionarily conserved, suggesting essential, albeit unknown functionality [3]. Borrowing terminology from cosmology, such DNA is often known as "dark matter," after the missing matter hypothesized to be necessary for the observed large-scale dynamics and structure of the universe. Another surprise was that the more than 1,200 genetic variants discovered in GWAS studies account for only a small fraction of the total heritability. Presently, it remains unknown if the "missing heritability" is due to rare variants not detected by GWAS studies or an artifact of our current statistical models for estimating heritability [4].

BOTTLENECKS IN BIG DATA ANALYSIS

The first bottleneck in big data analysis is data storage and retrieval. Given that

the rate of growth for storage capacity is not likely to suddenly increase, attention has focused on more efficient ways of data compression. An interesting direction is in the use of probabilistic data structures and algorithms that can store data with dramatic increases in compression efficiency in exchange for only a small loss in certainty. For example, Bloom filters that guarantee a specified level of false positives and zero false negatives can be constructed to store sequence data. A variety of ingenious proposals for compressing NGS data were submitted to the Sequence Squeeze competition sponsored by the Pistoia Alliance (http://www.sequencesqueeze.org).

Simply storing big data is suboptimal—given the cost of generating it, ideally, big data should be freely shared and reused by different investigators. Funding agencies and top journals require big data be deposited in online repositories before publication, making the data publicly accessible in principle. However, the data in public repositories may be poorly annotated and linking information from distinct databases might be impossible because of different data schemas and lack of unifying metadata. To address this issue, data standards in the form of minimal information requirements have been published for several data types (e.g. MIAME; minimal information about microarray experiments) and there is a drive to create standard vocabularies in the form of biomedical ontologies to allow data sharing across databases and machine processing.

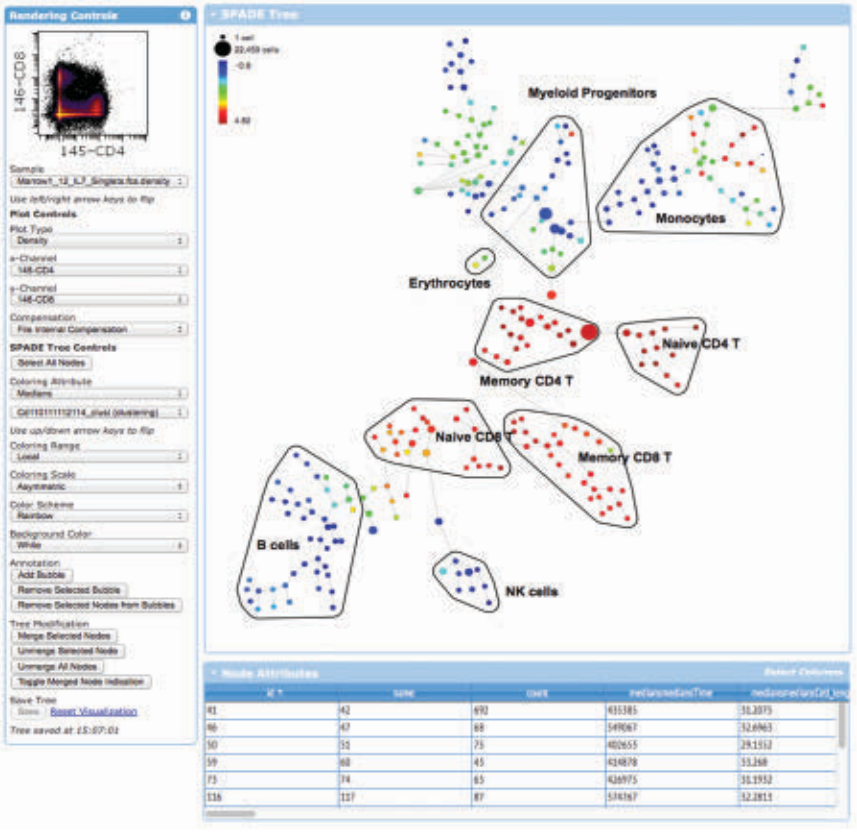
Without human ingenuity, it is apparent that not only will we be restricted to analyzing an ever-smaller fraction of the data generated, we may not even have the capacity to store all the data generated.

Even if the data can be stored and retrieved efficiently, big data is often too large to fit into available RAM, and languages that support generators are likely to be increasingly popular for the analysis of big data. This explains why there is a critical need for online algorithms that can efficiently process an input piece-by-piece. In the fields of statistical and machine learning, Bayesian models with conjugate priors are great examples of an online algorithm. Since the prior family is closed under Bayesian updating, and as data streams in, we can recursively apply Bayes' theorem to update the posterior distribution.

Machine learning is a field central to the analysis of big data. The information-processing rate of the human brain is severely constrained, necessitating the use of algorithms that can summarize the data and reduce the number of interesting features to a manageable level. Probabilistic graphical models with a foundation in Bayesian statistics play an increasing role in big data machine learning algorithms due to their ability to learn structure as well as parameters, ease of constructing hierarchical ("mixed effects") models, and natural fit to online processing requirements. Another advantage of Bayesian probabilistic models is their declarative nature, allowing algorithms developed for applications such as text mining by Yahoo or social network modeling by Facebook to be easily adaptable to biomedical data (or vice versa).

Finally, the ability to visualize or summarize big data is crucial to scientific discovery and insight, since the optic cortex takes up a larger share of our brain than any other sensory modality. Most investigators still rely on variations of pie and bar charts or scatter and line plots to visualize their

Figure 1. A spanning-tree progression analysis of density-normalized events (SPADE) visualized using Cytobank [5].



Courtesy of Nikesh Kotecha, http://blog.cytobank.org/

data, but these classical techniques are woefully inadequate to reveal the complexities of massive high-dimensional data sets. Innovative methods for scientific visualization and teaching are required, perhaps building on the availability of open source visualization and animation libraries such as Processing (http://processing.org).

BIG DATA AND SINGLE CELL ANALYSIS

Innovations in data generation have affected the field of single cell analysis and in particular, flow cytometry assays—a topic discussed less in the context of big data, but which is believed to become increasingly prominent. Essentially all of the "-omics" assays previously described are indiscriminately applied to tissue samples with thousands or millions of cells. As such, they report features of cells summed or averaged over many different cell types. However, the process of averaging loses information about individual cell differences and obscures the complexity of a biological response coordinated among heterogeneous cell types. For example, the immune response to a pathogen or tumor involves the orchestration of a large variety of immune cells often driven by low frequency but potent antigen-specific T lymphocytes. As such, there is a need for assays that report on single cells rather than aggregate properties of cell populations. Similarly, the malignant cells in a tumor are typically extremely heterogeneous due to genomic instability and reside in a complex microenvironment comprising of stromal cells and infiltrating anti-tumor and subverted protumor immune cells.

In order to exploit these weak signals for association analysis, or to capture the rich network of cell types involved in a physiological or pathological process, we need to measure properties of individual cells. Flow cytometry is perhaps the exemplar of multiplexed, single-cell analysis and its importance is amplified by its use in fluorescent activated cell sorting (FACS) that enables the application of derivative assays such as single cell polymerase chain reaction (PCR). Flow cytometry measures the emissions ("colors") of multiple fluorescent reporters bound to different cell surfaces and intracellular proteins of single cells in solution as they stream past multiple interrogating lasers. This technology has been in research and clinical use for several decades now. However, there has been a dramatic increase in the power of flow cytometry in the past decade—nearly all flow cytometers were restricted to three or four colors; the current generation of flow cytometers is capable of measuring 17 colors. A recently developed variant (mass cytometry) based on mass spectrometric rather than optical detection has pushed the number of measurable parameters to 40-plus. Cytometers that combine fluorescent labels with single cell imaging (image cytometers) can measure hundreds of parameters per cell if imaging features are included. Thus, the trend toward increasing the number of measurable parameters is certain to continue. Critically, current cytometers can capture single cell fea-

Figure 2. Cell subset identification with a hierarchical Dirichlet Process Gaussian mixture model with a consensus modal clustering strategy for Gaussian component merging. The model was fitted to data from 21 flow cytometry data samples (only four are shown for illustration), comprising a total of 7,255,967 events in nine dimensions.

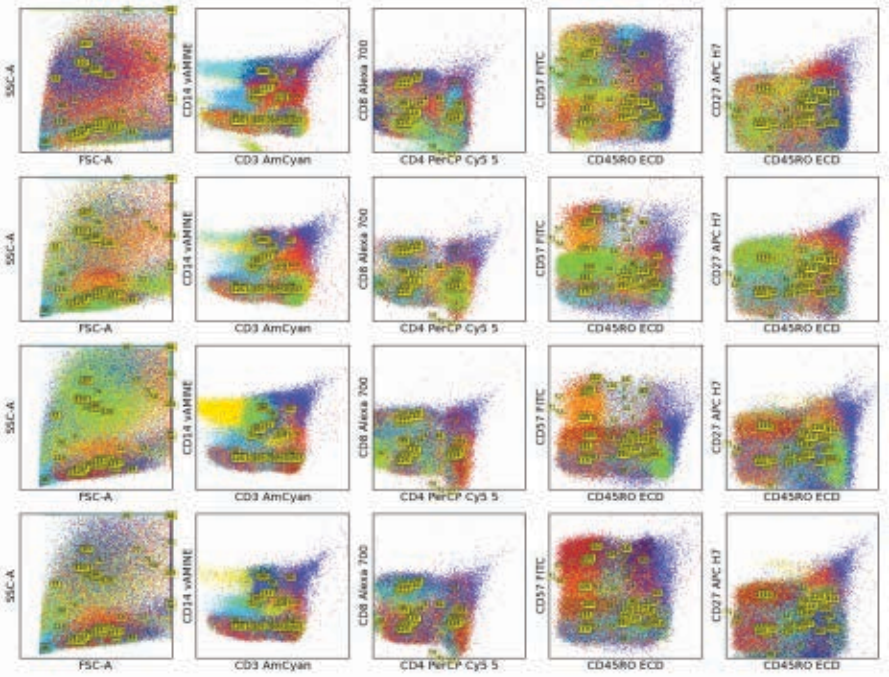
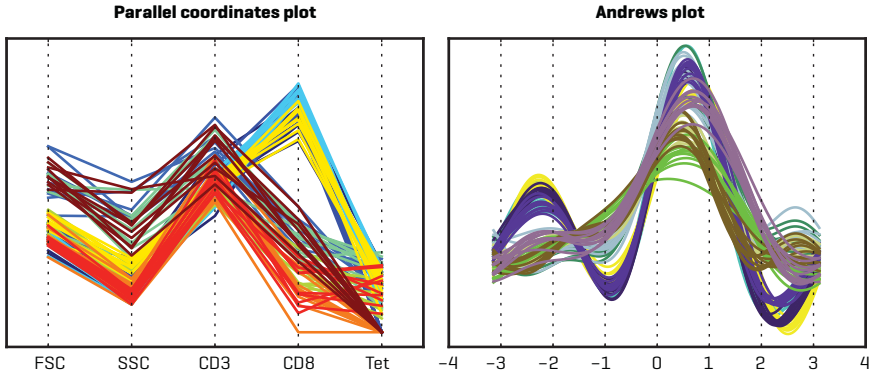


Figure 3. Sampled events from 10 largest clusters from the model fit shown in Figure 2 are illustrated using traditional multivariate statistical graphics, namely the parallel coordinates and Andrews plots.



tures on millions of cells in minutes, providing information as to the phenotype, activation, functional, regulatory, and cell-cycle status of each cell while assuring robust statistics due to the large total population size assayed.

Used with laboratory robotics, flow cytometers now routinely measure samples from 96- or 384-well plates, with each sample potentially providing data on tens or hundreds of parameters on thousands to millions of cells. In particular, imaging cytometers can record photographic images of each cell from six different perspectives and generate up to 1GB of data per sample. Such data has traditionally been stored locally on a disk in individual laboratories, either in an ad-hoc fashion or using a laboratory information management system (LIMS). More recently, sharing data via centralized (http://flowrepository.org/) or cloud based repositories (http://www.cytobank.org/) is gaining in popularity. Standards for describing flow cytometry experiments (e.g. Minimal information about Flow cytometry experiments or MiFlowcyte) and Minimal Information About T cell Assays (MIATA)) have been published, although adoption of these standards is still sporadic. Recent efforts at developing ontologies for flow cytometry are also being addressed by the National Center for Biomedical Ontology, which has recently co-organized a workshop on ontology-based research in immunology and infectious disease (http://tinyurl.com/84p3vdq). Innovative uses of visualization for data interpretation are also being developed (see Figure 1). It is clear that the field of flow cytometry is recapitulating the informatics developments in genomics.

Traditionally, flow cytometry assays are analyzed using gating in which cell subsets are visually identified using serial 2-dimensional projections and isolated using polygonal or elliptical boundaries called gates. However, statistical/machine learning methods have also been increasing in popularity in flow cytometry due to the unwieldiness and complexity of manually evaluating new data sets with tens to hundreds of dimensions. Our research focuses on the use of non-parametric Bayesian mixture models to represent the distribution of cells in the multivariate feature space and ways to summarize relevant features of the density that map to biological cell subsets. Figures 2 and 3 illustrate recent work in the development of hierarchical mixture models that "borrow strength" across multiple samples to sensitively find consistent cell subset assignments across data samples. Specifically, we fit Hierarchical Dirichlet process Gaussian mixture models (HDPGMM) to flow cytometry data using Markov Chain Monte Carlo (MCMC) methods coded in Python, exploiting large-scale parallelism with message passing interface (MPI) and fine scale parallelism with GPU programming via the Compute Unified Device Architecture (CUDA) API. The advent of GPU computing has transformed our research, converting a research method applicable to toy data sets into a practical approach for automating flow cytometry data analysis that we are actively developing for several large projects in cancer vaccination, HIV/AIDS biomarker discovery, and solid organ transplantation immune monitoring. We owe a debt of gratitude to video gamers around the world for spurring the development of commodity massively parallel processing, a testament to the unlikely origins of advances in scientific computing.

The advent of GPU computing has transformed our research, converting a research method applicable to toy data sets into a practical approach for automating flow cytometry data analysis.

THE FUTURE OF COMPUTATIONAL BIOLOGY

In closing, I would like to reiterate that it is a great time for students of computer science with an interest in biology and medicine to join the big data party. While the big-data deluge in biology seems overwhelming, this is just the beginning. To a large extent, today's big data in biology and medicine is merely about building a parts list of the sequences, proteins, metabolites, and cells involved in various pathophysiological processes of staggering complexity. We have barely begun to investigate the dynamics of how the parts change over time or the combinatorics of how the parts interact with each other. Essentially, we have little knowledge on how to link the molecular to cellular to tissue to individual levels, which will doubtlessly require new developments in multi-scale modeling. On the related computer-engineering front, the fields of DNA computing, natural algorithms (algorithms inspired by biological processes), and synthetic biology (construction of programmable genetic circuits) are wide open for innovative research. Digital and computational biology is changing how we understand life itself and there is much work to be done and much fun to be had.

References

[1] Stein, L. D. The case for cloud computing in genome informatics. *Genome Biol* 11, 207 (2010); doi:10.1186/gb-2010-11-5-207.

[2] Bennett, C. M., Miller, M.B., and Wolford, G. Neural correlates of interspecies perspective taking in the post-mortem Atlantic Salmon: An argument for multiple comparisons correction. *NeuroImage* 47, Supplment 1 (July 2009), S39-S41.

[3] Lander, E. S. Initial impact of the sequencing of the human genome. *Nature* 470, 7333 (Feb. 2011), 187-197, doi:10.1038/nature09792.

[4] Zuk, O., Hechter, E., Sunyaev, S. R. and Lander, E. S. The mystery of missing heritability: Genetic interactions create phantom heritability. *Proceedings of the National Academy of Sciences of the United States of America* 109, 4 (2012), 1193-1198.

[5] Kotecha, N., Krutzik P.O., and Irish, J.M. Web-based analysis and publication of flow cytometry experiments. *Current Protocols in Cytometry* 2010 Jul, Chapter 10, Unit10.17. PMID: 20578106.

Biography

Cliburn Chan is an assistant professor of Biostatistics and Bioinformatics at Duke University. Original trained as a medical doctor, he was bitten by the math bug and went on to pursue a Ph.D. in nonlinear dynamics at University College London. His research is focused on the modeling of human immune responses and the analysis of data from single cell immunological assays.

Jeff Dean
Big Data at Google

BY EDWARD Z. YANG

DOI: 10.1145/2331042.2331062

True fact: As a high school student, Jeff Dean wrote a statistics package that, on certain functions, was 26 times faster than equivalent commercial packages. These days, Dean works at Google, helping architect and optimize some of the biggest data-crunching systems Google employs on a day-to-day basis. These include the well-known MapReduce (a programming model for parallelizing large computations) and BigTable (a system which stores almost all of Google's data). Dean's current project is infrastructure for deep learning via neural networks, a system with applications for speech/image recognition and natural language processing.

Photo courtesy of Google

While Dean has become a public face attached to much of Google's internal infrastructure projects, he stresses the fact that these projects require a mix of areas of expertise. Any given project might have team members with backgrounds in networking, machine learning, and distributed systems. Collectively, a project can achieve more than any person individually. The downsides? With all of the different backgrounds, you really need to know when to say: "Hold on, I don't understand this machine learning term." However, he explains, working on these teams is lots of fun; you get to learn about a sub-domain you might not have known very much about.

Along with a different style of solving problems, Google also has different research goals than academia. Dean gave a particular example of this: When an academic is working on a system, they don't have to worry about what happens if some really rare hardware failure occurs—they simply have to demo the idea. But Google has to worry about these corner cases; it is what happens when one of your priorities is building a production system. There is also a tension with releasing results to the general public. Before the publication of the MapReduce paper, there was an internal discussion about whether or not to publish. Some were concerned the paper could benefit Google's competitors. In the end, though, Google decided to release the paper, and you can now get any number of open source implementations of MapReduce.

While Dean has been at Google for more than a decade, the start of his career looked rather different. He recounts how he ended up getting his first job. "I moved around a lot as a kid; I went to 11 schools in 12 years in lots of different places in the world. We moved to Atlanta after my sophomore year in high school, and in this school, I had to do an internship before we could graduate. I knew I was interested in developing software. So the guidance counselor of the school said, 'Oh, great, I'll set up something?' and she set up this boring sounding internship. I went to meet with them before I was going to start, and they essentially wanted me to load tapes into tape drives at this insurance company. I thought, 'That doesn't sound much like developing software to me.' So, I scrambled around a bit, and ended up getting an internship at the Center for Disease Control [CDC] instead."

This "scrambled" together internship marked the beginning of many years of work for the CDC and the World Health Organization (WHO). First working in Atlanta, and then in Geneva, Dean spent a lot of time working on what progressively grew into a larger and larger system for tracking the spread of infectious disease. These experiences—including a year working full-time between his graduation from undergraduate and his arrival at graduate school—helped fuel his eventual choice of a thesis topic. When Dean took an optimizing compilers course, he wondered if he could teach compilers to do the optimizations he had done at the WHO. He ended up working with Craig Chambers, a new faculty member who had started the same year Dean started as a grad student. "It was great, a small research group of three or four students and him. We wrote this optimizing compiler from scratch, and had fun and interesting optimization work." When he finished his Ph.D. thesis, he went to work at Digital Equipment Corporation and worked on low-level profiling tools for applications.

Dean likes doing something different every few years. After working on something for a while, he'll pick an adjacent field and then learn about that next. But Dean was careful to emphasize that while this strategy worked for him, he also thinks it is important to have different types of researchers; to have people who are willing to work on the same problem for decades, or an entire career—these people have a lot of in-depth knowledge in this area. "There's room in the world for both kinds of people," he explains. But, as he has moved from topic to topic, it turns out Dean has come back around again. His current project at Google on parallel training of neural networks was his undergraduate senior thesis topic. "Ironic," says Dean.

Edward Z. Yang will be a first year Ph.D. student at Stanford University this fall. In his spare time, he enjoys playing the oboe and British change ringing.

"London Population Density." A simple choropleth map, made more iconic and relatable by the inclusion of OpenStreetMap building and road data.

LABZ

The Centre for Advanced Spatial Analysis at University College London

London, UK

Editor's Note: *In line with this issue's theme of "big data," our featured lab is the Centre for Advanced Spatial Analysis (CASA) at University College London. The research center focuses on digital technologies in geography, space, and the built environment. With the recent London Olympics, CASA is sure to have plenty of opportunity to collect and analyze big data. Martin Dittus, shares his experience at CASA.*

—Jeff Koh

Currently I am part of a group of postgraduate students at the Centre for Advanced Spatial Analysis (CASA), a research center at University College London. CASA offers a Master of Research program that is led by an interdisciplinary team of researchers and practitioners who are grounded in advanced spatial analysis. At CASA I am part of an international group of people, where our academic and professional backgrounds are spread across multiple disciplines including computer science and software engineering, architecture and the built environment, cartography and geography, physics and mathematics, public policy, and foreign relations. This diverse group represents quite a wealth of problem domains that CASA attempts to bridge.

CASA was founded in 1995 and forms part of the UCL's Bartlett Faculty of the Built Environment, with an overall focus on cities. It is active in a large spectrum of research topics, including the analysis of urban transport flows using complexity science, agent-based models, and other techniques; the development of public participation mapping projects that allow scientists and hobbyists alike to easily gather survey data and present it as beautiful online maps; the analysis of surnames as descriptor of place; the use of QR codes to track the history of any object; an attempt to model the global dynamics of trade, migration, security, and development aid; a spatial flow analysis of the London riots; and much more. The data for such studies comes from a variety of sources. Base maps are equally likely to be provided by OpenStreetMap, the UK Ordnance Survey, or Google; and data may come from government bodies, industry partners, the Twitter fire hose, and many other public and private sources.

So far as an MRes student, I have been afforded an enchanting combination of rigorous science and playful use of new technologies, with plenty of scope to make my own things happen. Some of my work entails aspects of GIScience, cartography, spatial modeling, and urban planning as well as data visualization fundamentals, collaborative data gathering problems, a fair amount of playful and explorative project work, and beyond. As a new student I am amazed at the proposition to "just follow your interests" as a recommended approach to selecting my dissertation topic; this is not something you would often be told as a professional software

Photo by Jack Harrison

15PB

The amount of data generated by the Large Hadron Collider each year.

6:00EST

The time that Twitter users are happiest, according to an analysis of 300 million Tweets.

developer (my previous job), and it is this kind of freedom that inspires me to work harder.

Data visualization has always been an important aspect of CASA's research output. With this in mind, I spend much time and effort on the visual representation of my findings, and this is reflected well in the structure of our course. Building on an introduction to Geographic Information Science, an applied class in digital visualization techniques, a wide range of software packages and techniques, and an optional introduction to the Processing language for people without previous programming experience, I am frequently encouraged to produce visual representations to illustrate and explore a particular scenario or data set. Professors and peers in class then review my work. I learned a lot just from watching others, and from having my own work critiqued.

One commonality is the heavy reliance on computational methods, be it in the acquisition of data, the analysis, or the presentation of insights. I get to use the modern classics like R, ArcGIS, 3D Studio Max, and Processing just as much as experimenting with more recent tools like CityEngine, Unity, and Lumion. In more than one group project my colleagues and I were making use of online collaboration softwares to produce and refine documents while we were meeting. This works well when everyone is in the same room, but this has also helped me bridge physical distances. After having experienced the benefits of document editors with live collaboration modes (like Google Docs, Piratepad, or Prezi), single-person editors seem almost obsolete now. Such multi-modal teamwork takes a bit of practice, but once it flows it feels magical.

Biography

Martin Dittus studied computing in Berlin. In 2006 he moved to London to work at the Internet startup Last.fm as a software developer and in 2011 he joined CASA as an MRes graduate student.

BACK

Automated DNA Sequencers

Big data is a broad term that comprises a number of challenges including the analysis, search, storage, and capture of very large and/or high-dimensional data sets. Relatively recent cultural trends along with many technological advances have led to dramatic increases in the area of capture. One specific domain that has seen fantastic technological advances is the capture of DNA sequence data. The process of sequencing DNA involves translating an organism's DNA into the sequence of nucleobases (bases) of which it is composed. While the methods for sequencing DNA have changed over the years, almost all have involved some process of breaking the DNA into small fragments and identifying the sequence of bases for each fragment using chemical markers. These small fragments are then reassembled into one large sequence. Early methods required extraordinary amounts of manual labor and were very expensive, but in 1987 the first automated DNA sequencer was introduced. These machines continued to get faster and cheaper at an incredible rate. In fact, the task of sequencing the human genome, a sequence of more than 3 billion base pairs, began in 1990 and a rough draft was completed in 2001. Using more modern sequencers, this task could be completed in under a week and at a tiny fraction of the cost. With the help of automated DNA sequencers, increasingly massive quantities of genetic data are becoming available for all kinds of biological and medical research with each passing year.

—Finn Kuusisto

A DECADE OF DNA SEQUENCING

YEAR	2001	2011
Cost per Mega base of DNA	$5,292.39	$0.12
Cost per Human-sized Genome	$95,263,072.00	$10,497.00
GenBank Sequence Database Size (bases)	12 billion	127 billion
Completely Sequenced Genomes in GenBank	~50	~1,900
Notable Event	Draft of human genome published	Orangutan genome published

HELLO WORLD

Finding Yourself Using Geolocation and the Google Maps API

BY COLIN J. IHRIG

Today, many websites are able to access a visitor's physical location and generate pages specifically tailored to their surroundings. Some examples might include providing driving directions or locating nearby landmarks. The process by which a site determines a user's physical location is called geolocation. The World Wide Web Consortium (W3C)—a standards organization for the Web—has created a geolocation API, which defines a JavaScript interface that can be utilized in modern HTML5 capable browsers [1].

In this tutorial we will create an example Web page that uses geolocation, in conjunction with Google's Maps API [2], to generate and display driving directions on a map and in textual format. When the page is loaded, the user's current physical location is displayed on a map. When the user clicks on the map, driving directions are generated starting from the current location and ending at the clicked location on the map.

Geolocation Prerequisites

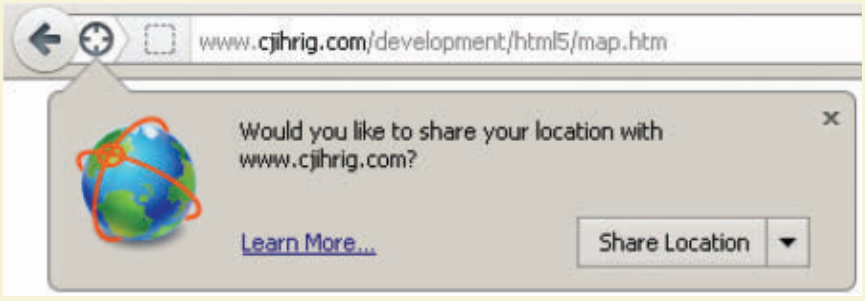
The geolocation API is not yet supported by all browsers. In order to run the example, you will need a browser that supports geolocation. Our example uses Firefox 12.0, which can be downloaded for free from Mozilla. Another thing to note is the issue of privacy arising from sharing a user's physical location. When the example page is loaded, you will notice that the browser must first get explicit consent from the user before sharing their location. Figure 1 shows the dialog box used by Firefox to get the user's consent.

Web Page Setup

First, we need to create the HTML file that will contain the map and directions. Listing 1 shows the HTML source for our example page. The `DOCTYPE` on line 1 instructs the browser to render the page according to HTML5 standards [3]. In the document's head, two external JavaScript files are included. The first file, included on lines 6-8, is the Google Maps API. This is the file that provides the mapping functionality. Luckily, Google has already done most of the hard work with this file. Notice the source URL takes a parameter named `sensor`. For our purposes this is set to `false`. Line 9 includes a second JavaScript file, `helloWorld.js`, which we will create later. This is where our application's functionality is going to be implemented.

Two `<div>` elements are defined in the page's body on lines 12-15. The element named `map_canvas` is where the map is going to be displayed, while the `dir_panel` element will show the corresponding text directions. The `style` attributes of the `<html>`, `<body>`, and `<div>` elements ensure the document is sized and displayed properly.

Figure 1. Firefox requesting permission to share the user's location.



Geolocation Lookup

The next step is to actually create the `helloWorld.js` file. First, we need to add a function that will perform a geolocation lookup when the page is loaded. Listing 2 shows the JavaScript code that implements this functionality. On line 1, an anonymous function is attached to the page's load event. This ensures that our code is called each time the page is loaded. The `if` statement on line 2 determines whether or not the browser supports geolocation by checking for the existence of the `navigator.geolocation object`. If geolocation is not supported then the error message on line 8 is shown.

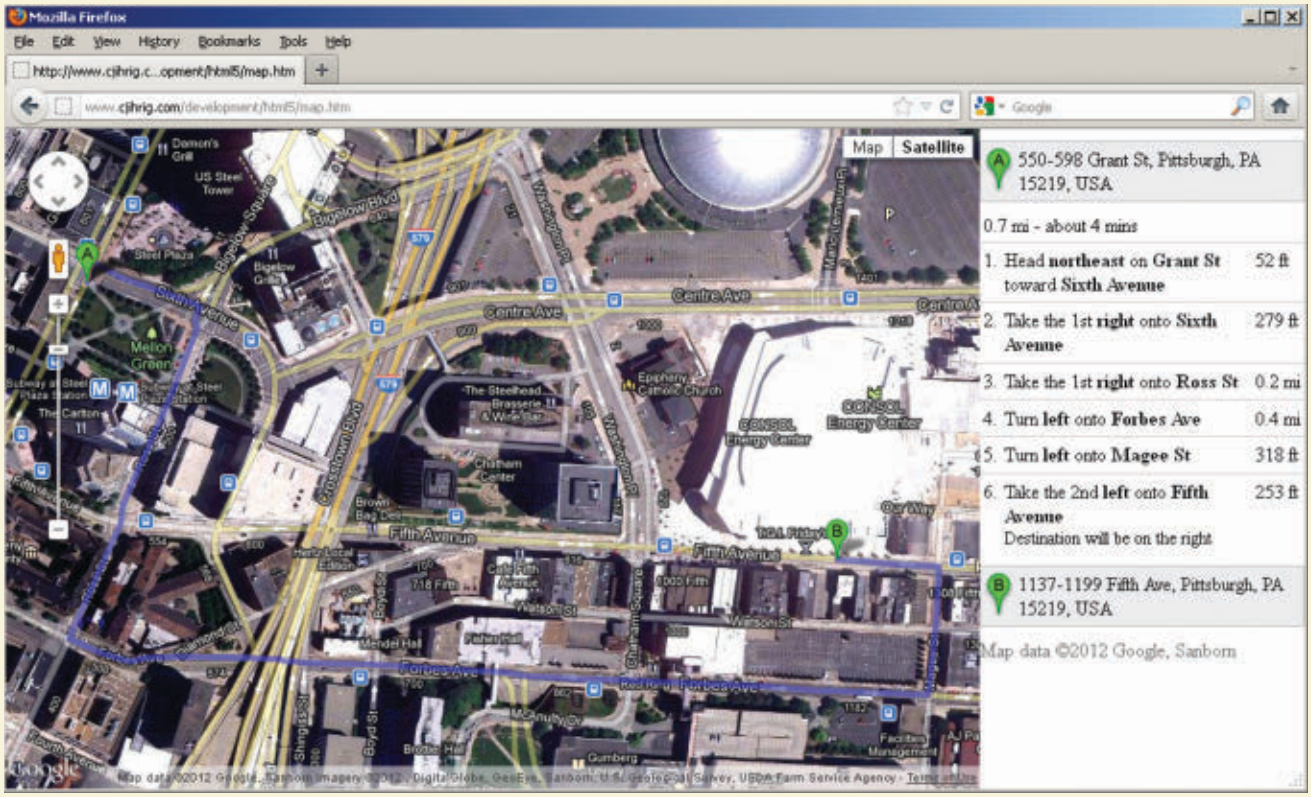
If the browser does support geolocation then the `getCurrentPosition()` function is called on line 3. As the name implies, this function attempts to determine the user's physical location. The `getCurrentPosition()` function takes two callback functions as arguments. The first argument is a function that is executed if the user's location is determined successfully. In this example, the success callback function is named `showMap()`. We'll return to `showMap()` shortly.

During a geolocation lookup, any number of things can go wrong, resulting in a lookup failure. The second callback function passed to `getCurrentPosition()` is used to handle lookup failures. Lines 4-6 show the error callback function for the example page. For this example, a simple error message is shown if the user cannot be located.

Displaying the Map

If geolocation succeeds, the `showMap()` function is called. The code for `showMap()` is shown in Listing 3. The first thing to point out is the `position` argument. This is an object that contains the user's physical location defined in terms of latitude and longitude. The variable assignments

Figure 2. Example of the finished Web page.



on lines 2 and 3 are used to store the these coordinates. The `latlng` variable, declared on line 4, is an object in the Google Maps API that represents the user's location.

The variable `mapOpts`, declared on lines 5 and 6, is a JavaScript object literal. The Maps API makes extensive use of JavaScript objects for passing arguments due to their simplicity and flexibility. In our example, `mapOpts` is used to create a hybrid, style map (street names overlaid on satellite imagery), which is centered on the user's coordinates and zoomed in by a factor of 15. The `canvas` variable on line 7 corresponds to the `<div>` element where the map will be displayed. The map is created on line 8. Lines 9 and 10 create a marker on the map at the user's location. Since the map is centered on the user's location, the marker appears in the middle of the map.

Next, we will allow the user to generate driving directions by clicking on the map. First, we need to create two Maps API objects, a `DirectionsService` and a `DirectionsRenderer`.

Listing 1. HTML source for the example page.

```
1: <!DOCTYPE html>
2: <html style="height:100%; width:100%;">
3:   <head>
4:     <title>Geolocation and Google Maps Example</title>
5:     <meta charset="UTF-8" />
6:     <script
7:       src="http://maps.googleapis.com/maps/api/js?sensor=false">
8:     </script>
9:     <script src="helloWorld.js"></script>
10:  </head>
11:  <body style="height:100%; width:100%; margin:0; padding:0;">
12:    <div id="map_canvas"
13:         style="float:left; width:75%; height:100%;"></div>
14:    <div id="dir_panel"
15:         style="float:right; width:25%; height:100%;"></div>
16:  </body>
17: </html>
```

Listing 2. JavaScript function to perform geolocation lookup.

```
1:  window.addEventListener('load', function() {
2:    if (navigator.geolocation)
3:      navigator.geolocation.getCurrentPosition(showMap,
4:        function(error) {
5:          alert('Cannot determine your location!');
6:        });
7:    else
8:      alert('Your browser does not support geolocation!');
9:  });
```

Listing 3. JavaScript function to display a map and driving directions.

```
1: function showMap(position) {
2:   var latitude  = position.coords.latitude;
3:   var longitude = position.coords.longitude;
4:   var latlng    = new google.maps.LatLng(latitude, longitude);
5:   var mapOpts   = {zoom: 15, center: latlng,
6:                    mapTypeId: google.maps.MapTypeId.HYBRID};
7:   var canvas    = document.getElementById('map_canvas');
8:   var map       = new google.maps.Map(canvas, mapOpts);
9:   var marker    = new google.maps.Marker({position: latlng,
10:                                         map: map});
11:   var dirServ   = new google.maps.DirectionsService();
12:   var dirDisp   = new google.maps.DirectionsRenderer();
13:   var panel     = document.getElementById('dir_panel');
14:
15:   dirDisp.setMap(map);
16:   dirDisp.setPanel(panel);
17:   google.maps.event.addListener(map, 'click', function(event) {
18:     dirServ.route({origin: latlng, destination: event.latLng,
19:                    travelMode: google.maps.TravelMode.DRIVING},
20:       function(result, status) {
21:         if (status === google.maps.DirectionsStatus.OK)
22:           dirDisp.setDirections(result);
23:       });
24:     });
25: }
```

The `DirectionsService` declared on line 11 provides routing directions, while the `DirectionsRenderer` on line 12 displays the directions on the map and in the directions panel. On line 13, a reference to the `dir _ panel` element is stored in the panel variable. Lines 15 and 16 tell the `DirectionsRenderer` where to display the graphical and textual directions, respectively.

The final step in the example is to add an event listener to the `showMap()` function. This will allow the directions to update automatically when the user clicks on the map. To do this, the `addListener()` function is called on line 17. The arguments passed to `addListener()` are the map object, the type of event (`click` in this case), and a function that processes the event. On line 18, the event handler calls the `route()` function of the `DirectionsService`. The first argument passed to `route()` is an object literal containing the user's current location, the travel destination stored in the `latLng` field of the `event` argument, and the travel mode, which is driving in our example. The second argument passed to `route()` is a function that handles the routing information returned by the `DirectionsService`. On lines 21 and 22, this function checks that the directions were properly created and then tells the `DirectionsRenderer` to display them.

Conclusion

By following the steps outlined in this tutorial you have just created a non-trivial, location aware Web page. By utilizing Google's Maps API, we were able to create the page in approximately 50 lines of HTML and JavaScript code. Figure 2 shows the example page used to navigate downtown Pittsburgh, PA. If you would like to use the example page while on the go, it is available online at http://www.cjihrig.com/development/html5/map.htm.

References

[1] Geolocation API Specification; http://www.w3.org/TR/geolocation-API.

[2] Google Maps Javascript API; https://developers.google.com/maps/documentation/javascript.

[3] The DOCTYPE - Your First Step to HTML5; http://www.cjihrig.com/blog/the-html5-doctype.

Biography

Colin J. Ihrig is currently a Ph.D. student at the University of Pittsburgh, studying computer engineering. Ihrig also received his M.S. and B.S. in computer engineering from the University of Pittsburgh. His current research focuses on CAD tools for the emulation of many-core chip multiprocessors.

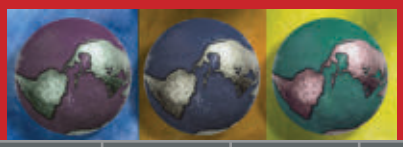
THE ACM A. M. TURING AWARD

by the community◆ from the community◆ for the community

ACM, Intel, and Google congratulate

JUDEA PEARL

for fundamental contributions to artificial intelligence through the development of a calculus for probabilistic and causal reasoning.



"Dr. Pearl's work provided the original paradigm case for how to do statistical AI. By placing structured knowledge representations at the heart of his work, and emphasizing how these representations enabled efficient inference and learning, he showed the field of AI how to build statistical reasoning systems that were actually telling us something about intelligence, not just statistics."

Limor Fix
Director, University Collaborative Research Group
Intel Labs

For more information see www.intel.com/research.

"Judea Pearl is the most prominent advocate for probabilistic models in artificial intelligence. He developed mathematical tools to tackle complex problems that handle uncertainty. Before Pearl, AI systems had more success in black and white domains like chess. But robotics, self-driving cars, and speech recognition deal with uncertainty. Pearl enabled these applications to flourish, and convinced the field to adopt these techniques."

Alfred Spector
Vice President, Research and Special Initiatives
Google Inc.

For more information, see http://www.google.com/corporate/index.html and http://research.google.com/.

Financial support for the ACM A. M. Turing Award is provided by Intel Corporation and Google Inc.

EVENTS

CONFERENCES

Fifth Balkan Conference in Informatics (BCI)
University of Novi Sad
Novi Sad, Serbia
September 16-20, 2012
http://bci2012.bci-conferences.org

International Conference on Emerging Intelligent Data and Web Technologies (EIDWT)
University Politehnica of Bucharest
Bucharest, Romania
September 19-21, 2012
http://voyager.ce.fit.ac.jp/~eidwt2012/index.html

International Conference on Information Technology, E-Government and Applications (ICITEA)
Hotel BurJuman Arjaana
Abu Dhabi, UAE
September 20-21, 2012
http://www.icitea.com/callforpapers12.php

Research In The Large: App Stores, Wide Distribution, and Big Data
Westin St.Francis Hotel
San Francisco, CA
September 21, 2012
http://large.mobilelifecentre.org/2012

Applied Statistics 2012 (AS)
Hotel Ribno
Bled, Slovenia
September 23 -26, 2012
http://conferences.nib.si/AS2012

OSS, BSS World Summit
Sheraton, The Park Lane Hotel
London, UK
September 25-26, 2012
http://www.ossbssworld.com/programme.html

11th International Conference on Information Systems and Industrial Management (CISIM)
Palazzo Ca'Dolfin, Dorsoduro
Venice, Italy
September 26-28, 2012
http://www.dsi.unive.it/CISIM

Eight IEEE International Conference on eScience (eScience 2012)
Hyatt Regency Chicago
Chicago, IL
October 8-12, 2012
http://www.ci.uchicago.edu/escience2012/index.php

Big Data Europe
Holiday Inn Vienna-South
Vienna, Austria
October 9-10, 2012
http://www.big-data-europe.com

International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)
Sanya, China
October 10-12, 2012
http://www.cyberc.org

First International Workshop on Dependability Issues in Cloud Computing (DISCCO)
Irvine, CA
October 11, 2012
https://sites.google.com/site/discco2012/home

IEEE Symposium on Large, Scale Data Analysis and Visualization (LDAV)
Seattle, WA
October 14-15, 2012
http://www.ldav.org

IEEE Symposium on Biological Data Visualization (BIOVIS)
Seattle, WA
October 14-15, 2012
http://www.biovis.net/about

ACM Symposium on Cloud Computing (SOCC)
San Jose Marriot
San Jose, CA
October 14-17, 2012
http://www.socc2012.org/home

Second International eConference on Computer and Knowledge Engineering (ICCKE)
Mashhad, Iran
October 18-19, 2012
http://iccke2012.um.ac.ir

IEEE International Conference on Advanced Computational Intelligence (ICACI)
Jintailong International Hotel
Nanjing, Jiangsu, China
October 18-20, 2012
http://www.iwaci.org

2012 IEEE Symposium on e-Learning, e-Management, and e-Services (IS3E)
Grand Seasons Hotel
Kuala Lumpur, Malaysia
October 21-24, 2012
http://computer.ieeemy.org/is3e

Sixth International Conference on New Trends in Information Science, Service Science, and Data Mining (NISS, ICMIA and NASNIT)
Taipei, Taiwan
October 23-25, 2012
http://www.aicit.org/issdm/home/index.html

Conference on Intelligent Data Understanding (CIDU)
National Center for Atmospheric Research
Boulder, CO
October 24-26, 2012
https://c3.nasa.gov/dashlink/events/1

Fifth Romania Tier 2 Federation Conference: Grid, Cloud, and High Performance Computing in Science (RO-LCG 2012)
National Institute for Research and Development of Isotopic and Molecular Technologies
Cluj Napoca, Romania
October 25-27, 2012
http://www.itim-cj.ro/rolcg2012

The Eleventh International Symposium on Intelligent Data Analysis (IDA)
Finlandia Hall
Helsinki, Finland
October 25-27, 2012
http://ida2012.org

2012 Third International Conference on E-business, Management and Economics (ICEME)
Hong Kong
October 27-28, 2012
Cost: Students $300
http://www.iceme.org/index.htm

2012 International Symposium on Information Theory and its Applications (ISITA)
Hawaii Convention Center
Honolulu, HI
October 28-31, 2012
http://www.isita.ieice.org/2012

ACM International Conference on Information and Knowledge Management (CIKM)
Sheraton Maui Resort & Spa
Maui, HI
October 29-November 2, 2012
http://www.cikm2012.org

ACM Multimedia 2012 (ACMMM)
Nara Prefectural New Public Hall
Nara, Japan
October 29- November 2, 2012
http://www.acmmm12.org

2012 Second IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)
Dragon Hotel
Hangzhou, China
October 30-November 1, 2012
Cost: Students $450
http://conference.bupt.edu.cn/ccis2012

Big Data Europe
Paris, France
November 6-7, 2012
http://www.big-data-europe.com

2012 International Conference on Knowledge, Information, and Creativity Support Systems (KICSS)
Monash Conference Centre
Melbourne, Australia
November 8-10, 2012
http://gnn.infotech.monash.edu.au/kicss2012/DropBox/site.content

The International Conference for High Performance Computing, Networking, Storage and Analysis (SC 12)
Salt Lake City, UT
November 10-16, 2012
Cost: Students $150 (advanced), $200 (late), $225 (onsite)
http://sc12.supercomputing.org

IEEE Asia Pacific Cloud Computing Congress (APCloudCC 2012)
Shenzen, China
November 14-17, 2012
Cost: Students $360 (before Sept 16), $425 (after)
http://www.apcloudcc.org

Big Data Europe
Frankfurt, Germany
November 20-21, 2012
http://www.big-data-europe.com

2012 Third International Conference on Emerging Applications of Information Technology (EAIT)
Indian Statistical Institute
Kolkata, India
November 29-December 1, 2012
https://sites.google.com/site/csieait2012

IEEE International Conference on High Performance Computing (HiPC)
Le Meridien Hotel
Pune, India
December 18-21, 2012
http://www.hipc.org/hipc2012/index.php

International Symposium on High Performance Computer Architecture (HPCA)
Shenzen, China
February 23-27, 2013
http://carch.ict.ac.cn/~hpca19/index.html

28th ACM Symposium on Applied Computing (SAC)
Institute of Engineering of the Polytechnic Institute of Coimbra (ISEC-IPC)
Coimbra, Portugal
March 18-22, 2013
http://www.acm.org/conferences/sac/sac2013

CONTESTS & EVENTS

VisWeek 2012
With the world of information being flooded by high-dimensional, specialized data, there is an everlasting void in the space of information visualization. VisWeek 2012 is a unique event to feed the need. The weeklong event brings together researchers and practitioners from academia, industry, and

FEATURED EVENT

Big Data Innovation
Boston, MA
September 13-14, 2012

Have you ever thought, how big organizations and multinationals cope with huge volumes of data, which they have to store, analyze, and process every single day? How are the researchers trying to develop trend-setting platforms to exceed the limits of exabytes and zettabytes?

Well, we are in the world of high performance computing—the world of large and complex data sets—often known as "big data." If you are interested in delving into and learning about business data analytics, stock market predictions and analysis of business trends, and the current research going on in this field, then the Big Data Innovation Conference in Boston is your ideal destination.

As promised by the organizers, the conference will be an ideal platform for persons involved in the field of big data with more than 35 keynote presentations from big multinationals, interactive breakout sessions, and open discussions ranging from analytics to the architecture of future data systems.

For more details visit, http://analytics.theiegroup.com/bigdata-boston.

—Arka Bhattacharya

Photo by Sean Pavone

ACRONYMS

L-BFGS algorithm Limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm: A quasi-Newton optimization methods that uses a limited memory variation of the BFGS update.

BDMS Big Data Management System: Systems that can manage data sets so large and complex, which are awkward to work with using on-hand database management tools.

AQL Annotation Query Language: A language for building extractors that extract structured information from unstructured or semistructured text.

GFS Google File System: A proprietary distributed file system developed by Google Inc. for its own use.

EC2 Amazon Elastic Cloud: A central part of Amazon.com's cloud computing platform that allows users to rent virtual computers on which to run their own computer applications.

HDFS Hadoop Distributed File System: A distributed, scalable, and portable filesystem written in Java for the Hadoop framework, designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications.

SDN Software Defined Networking: An emerging architecture for computer networking that separates the control plane from the data plane in network switches and routers.

government. VisWeek 2012 will host three major events: IEEE visualization (23nd IEEE SciVis), IEEE Information visualization (18th IEEE InfoVis) and IEEE visual analytics science and technology (7th IEEE VAST).

VisWeek 2012 will be held October 14-19, 2012 in Seattle.
http://visweek.org/

MASSIVE 2012
The Fourth Workshop on Massive Data Algorithmics 2012 (MASSIVE 2012) will be held as a part of ALGO 2012. The University of Ljubljana, Slovenia will host the workshop on September 13, 2012. The workshop aims to integrate interested practitioners from academia and industries. Scope of the workshop includes fundamental as well as specialized problems graphics, database, statistics, bioinformatics etc.
http://analytics.theiegroup.com/bigdata-london

GRANTS, SCHOLARSHIPS & FELLOWSHIPS

NSF Innovation Corps Program
Website: *http://www.nsf.gov/pubs/2011/nsf11560/nsf11560.htm*
Deadline: September 15, 2012
Benefits: Up to $50,000; $5 million in total awards
Eligibility: Must be a recipient of an NSF award within the last five years.
Explanation: The Innovation Corps program seeks to give support to transform research into applied products and services. This support includes both a monetary grant and mentoring, to the end that the project can attract funding from outside investors. Innovation Corps will determine whether a viable product or service can be produced with the research, create a transition plan, and develop a demonstration of the product for investors.

Hertz Fellowship
Website: *http://www.hertzfoundation.org/dx/fellowships/application.aspx*
Deadline: Fall 2012
Benefits: Tuition is covered, includes a $31,000-$35,000 stipend for up to five years.
Eligibility: Citizens or permanent residents of the U.S. who are "willing to morally commit to make their skills available to the United States in time of national emergency."
Explanation: The Hertz Foundation awards 15-20 fellowships to students pursuing a Ph.D. in the applied physical, biological, and engineering sciences. The foundation is interested in funding candidates who will be able to make an impact by applying science to real-world human problems.

AAUW International Fellowship
Website: *http://www.aauw.org/learn/fellows_directory/index.cfm*
Benefits: 49 fellowships totaling $978,000 were awarded last year.
Deadline: Applications available August 1
Eligibility: Women who are not U.S. citizens or permanent residents.
Explanation: The fellowship can support both graduate and postgraduate study, there are fellowships available for study outside the U.S. Past recipients of AAUW fellowships in computer science have done academic work on human-computer interaction; other grants by the organization have spanned the gamut of computer science topics.

POINTERS

BIG DATA RESOURCES

Every day, we end up creating enormous volumes of digital data. Such data is generated by various means, e.g., weather sensors, social media posts, digital pictures and videos, emails, financial transactions, amongst many others. It is estimated that 90 percent of the data available to us now has been created during the last two years. This is loosely characterized as "big data."

We define big data as datasets that are so large and complex traditional data analysis and management tools no longer work. They are usually characterised by: (1) an extremely large volume; (2) a very high speed of update; and (3) a wide variety of sources, which

contribute to generating the data. For that reason, big data has to deal with engineering challenges and research opportunities in storage, privacy, and analysis issues.

Big Data Now: Current Perspectives from O'Reilly Radar
O'Reilly Radar Team, O'Reilly Media, Kindle edition (2011)
Description:
This is a collection of data related work published by O'Reilly Media in 2010-2011, mainly in the form of interviews with experts in the field. The four core areas covered in this book include: (1) data issues, (2) products and processes in the application of data, (3) data science and data tools, and (4) the business of data. This book is available for Amazon Kindle devices and reader applications.

Taming The Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics
Bill Franks, Wiley (2012)
From the back cover:
"In case you ever wondered why big data is providing business value in many industries, this book gives you perspectives and answers from many angles—from the tech side, to data science, to business users and processes. In my entire career of researching and lecturing on analytics, I have never encountered a book that combines the knowledge of both information technology and business managers in such a succinct way. I would recommend it to anyone whose career intersects with big data."

—Diego Klabjan, Professor at Northwestern University, Director, Master's of Science in Analytics

"Bill Franks provides an entertaining and consumable take on a complex and intricate topic. The mix of insights applicable to practitioners and novices alike make this a critical read for someone new to the analytics space or to anyone in the space wanting to ensure they can learn from an accomplished leader. Franks' view across multiple industries and uses of big data have positioned him well to deliver this entry into the emergence of the space."

—Richard Maltsbarger, Senior Vice President of Strategy, Lowe's Companies, Inc.

USEFUL WEBSITES

IBM Big Data
http://www-01.ibm.com/software/data/bigdata

Oracle Big Data
http://www.oracle.com/us/technologies/big-data/index.html

McKinsey Global Institute, "Big data: The next frontier for innovation, competition, and productivity"
http://tinyurl.com/74tdfdv/

Nature Special: Big Data
http://www.nature.com/news/specials/bigdata/index.html

Google Public Data Explorer
http://www.google.com/publicdata/di

The CASA Blog Network
http://blogs.casa.ucl.ac.uk

GRADUATE PROGRAMS

The Bartlett, University College London
http://www.bartlett.ucl.ac.uk/

McCormick School of Engineering, Northwestern
http://www.analytics.northwestern.edu

Advanced Analytics at North Carolina State University
http://analytics.ncsu.edu

School of Information Systems and Management, Carnegie Mellon University
http://www.heinz.cmu.edu/

School of Information Studies, Syracuse University
http://ischool.syr.edu//

Kelley School of Business, Indian University Bloomington
http://kelley.iu.edu/

FEATURED GRANT

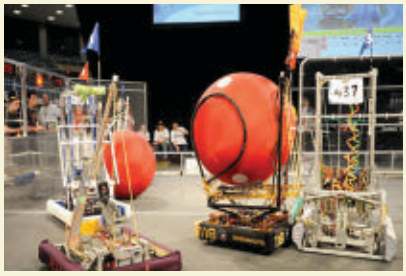


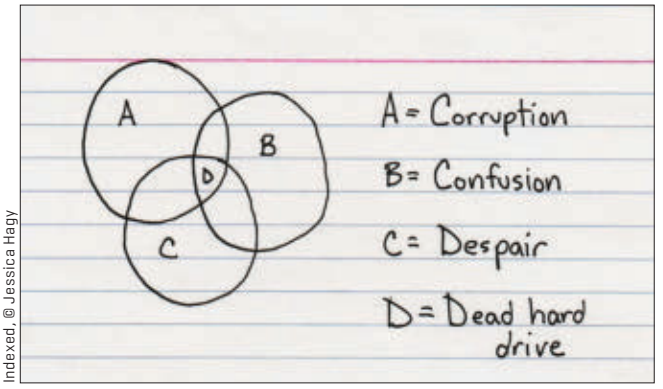
Photo Courtesy of NASA

National Robotics Initiative
Website: *http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503641&org=CISE*

Deadline: Letters of Intent due Oct. 1 for small proposals, Dec.15 for large proposals; Full proposals due Nov. 3 for small proposals, Jan. 18, 2013 for large proposals.
Benefits: 60-75 grants allotted from a $40-50 million fund.
Eligibility: Similar to other NSF projects.
Explanation: The National Robotics Initiative is a U.S. program put forward by many government agencies, including NASA and the NSF. The focus of the initiative is on "co-robots," robots that work cooperatively with people, taking the roles of co-workers, co-protectors, and co-inhabitants alongside humans. In order for these robots to be of the greatest possible use, they will have to be cheap, usable, and available anywhere. The initiative aims both to support the technical research that will make this possible, as well as the interdisciplinary insights that can be gained by working with researchers in areas like linguistics, cognition, and developmental science. Another part of the program is integrating robotics into education, both through curriculum and through research to determine the possible long-terms effects of humans living with co-robots. For more information about how this new generation of robotics might transform in space flight, health care, and food production see the official program solicitation; *http://www.nsf.gov/pubs/2011/nsf11553/nsf11553.htm/.*

BEMUSEMENT

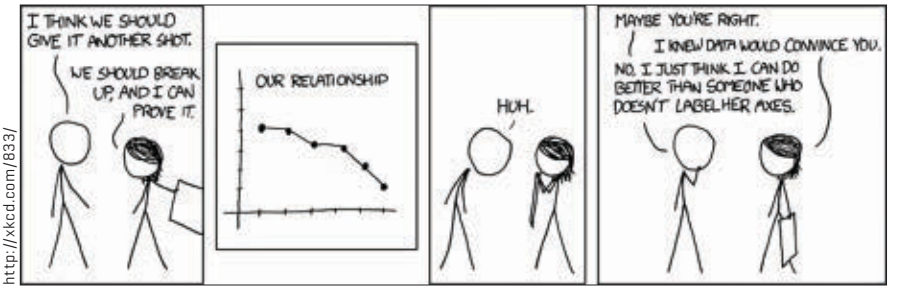
Back that Thing Up



Indexed, © Jessica Hagy

Source: http://thisisindexed.com/2006/08/back-that-thing-up/

Convincing



http://xkcd.com/833/

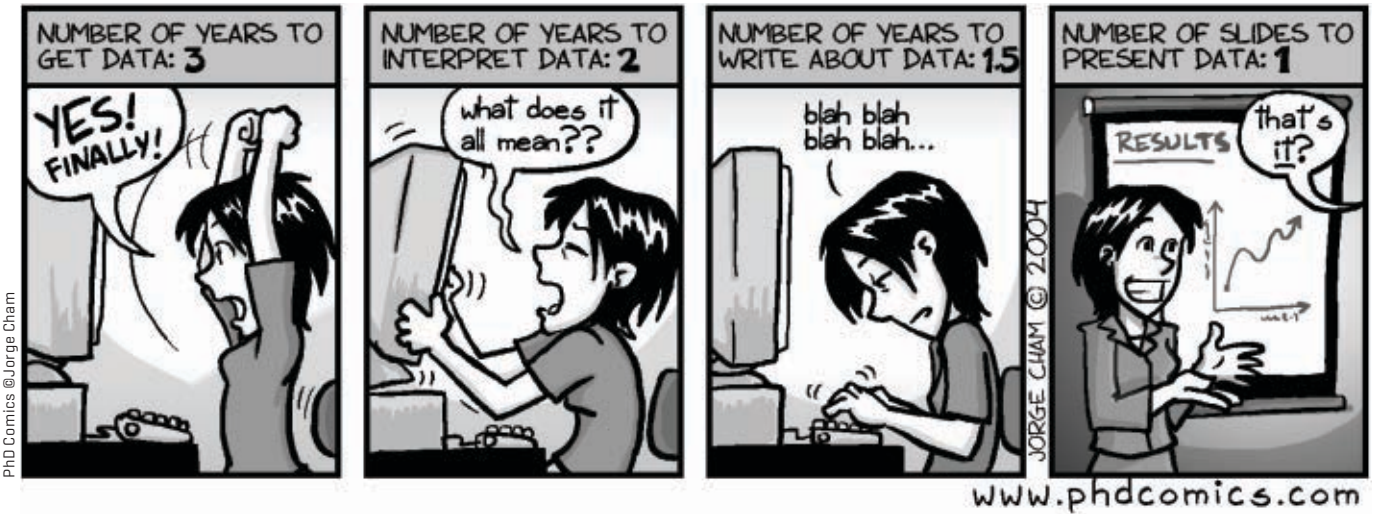
Puzzle: Grandma's Famous Cake

One day, grandma baked a cake with a square top and dimensions 30cm × 30cm × 10cm. What is a simple strategy for cutting the cake into nine equal pieces? The next day, grandma baked another cake with the same dimensions. This time, she put a thin layer of icing on top and on all four sides but not on the bottom. What is a simple strategy for cutting such a cake into nine pieces such that all pieces have the same amount of cake by volume and the same amount of icing by surface area?

Find the solution at: http://xrds.acm.org/bemusement/2012.cfm

Source: MathOverflow; http://mathoverflow.net/questions/29323/math-puzzles-for-dinner

Data: By the numbers



PhD Comics ©Jorge Cham

SUBMIT A PUZZLE

Can you can do better? Bemusements would like your puzzles and mathematical games (but not Sudoku). Contact **xrds@acm.org** to submit yours!

STUDENT MEMBERSHIP APPLICATION

CODE: CRSRDS

Join ACM online: www.acm.org/joinacm

Name *Please print clearly*

Address

City State/Province Postal code/Zip

Country E-mail address

Area code & Daytime phone Mobile phone Member number, *if applicable*

MEMBERSHIP BENEFITS AND OPTIONS

- Electronic subscription to *Communications of the ACM* and print & electronic subscription *XRDS* magazine
- Free software and courseware through the ACM Student Academic Initiative
- Online courses, Virtual labs and Online books
- ACM's *CareerNews* (twice monthly)
- ACM e-news digest *TechNews* (tri-weekly)
- Free e-mentoring service from MentorNet
- ACM online newsletter *MemberNet* (twice monthly)
- *Student Quick Takes*, student e-newsletter (quarterly)
- Free "acm.org" email forwarding address plus filtering through Postini
- Option to subscribe to the full ACM Digital Library
- Discounts on ACM publications and conferences, valuable products and services, and more

PLEASE CHOOSE ONE:

❑ **Student Membership: $19 (USD)**
❑ **Student Membership PLUS Digital Library: $42 (USD)**
❑ **Student Membership PLUS Print *CACM* Magazine: $42 (USD)**
❑ **Student Membership w/Digital Library PLUS Print *CACM* Magazine: $62 (USD)**

PUBLICATIONS

Please check

Check the appropriate box and calculate amount due on reverse.	Issues per year	Code	Member Rate	Air Rate*
• ACM Inroads	4	178	$14 ❒	$54 ❒
• Communications of the ACM	12	101	$43 ❒	$54 ❒
• ACM Queue	N/A	N/A	N/A	N/A
• Computers in Entertainment (online only)	4	247	$45 ❒	N/A
Computing Reviews	12	104	$53 ❒	$35 ❒
• Computing Surveys	4	103	$34 ❒	$28 ❒
Evolutionary Computing (MIT Press)	4	177	$30 ❒	$26 ❒
• interactions, new visions of human-computer interaction (included in SIGCHI membership)	6	123	$20 ❒	$31 ❒
Int'l Journal of Network Management (Wiley)	6	136	$90 ❒	$26 ❒
Int'l Journal on Very Large Databases	4	148	$81 ❒	$26 ❒
• Journal of Educational Resources in Computing (see TOCE)	N/A	N/A	N/A	N/A
• Journal of Experimental Algorithmics (online only)	12	129	$30 ❒	N/A
• Journal of Personal and Ubiquitous Computing	6	144	$63 ❒	$26 ❒
• Journal of the ACM	6	102	$53 ❒	$54 ❒
• Journal on Computing and Cultural Heritage	4	173	$47 ❒	$19 ❒
• Journal on Data and Information Quality	4	171	$47 ❒	$21 ❒
• Journal on Emerging Technologies in Computing Systems	4	154	$40 ❒	$19 ❒
• Linux Journal (SSC)	12	137	$29 ❒	$29 ❒
• Mobile Networks and Applications	6	130	$70 ❒	$25 ❒
• Wireless Networks	4	125	$70 ❒	$25 ❒
• XRDS: Crossroads	4	XRoads	$37 ❒	N/A
Transactions on:				
• Accessible Computing	4	174	$47 ❒	$20 ❒
• Algorithms	4	151	$50 ❒	$19 ❒
• Applied Perception	4	145	$41 ❒	$19 ❒
• Architecture & Code Optimization	4	146	$41 ❒	$19 ❒
• Asian Language Information Processing	4	138	$37 ❒	$19 ❒
• Autonomous and Adaptive Systems	4	158	$39 ❒	$19 ❒
• Computational Biology and Bio Informatics	4	149	$18 ❒	$45 ❒
• Computer-Human Interaction	4	119	$41 ❒	$21 ❒
• Computational Logic	4	135	$42 ❒	$21 ❒
• Computation Theory	8	176	$47 ❒	$28 ❒
• Computer Systems	4	114	$45 ❒	$21 ❒
• Computing Education (formerly JERIC)		277	$39 ❒	N/A
• Database Systems	4	109	$44 ❒	$21 ❒
• Design Automation of Electronic Systems	4	128	$41 ❒	$21 ❒
• Embedded Computing Systems	4	142	$42 ❒	$19 ❒
• Graphics	4	112	$49 ❒	$21 ❒
• Information and System Security	4	134	$42 ❒	$19 ❒
• Information Systems	4	113	$45 ❒	$21 ❒
• Intelligent Systems and Technology	4	179	$44 ❒	$68 ❒
• Interactive Intelligent Systems	4	191	$48 ❒	$74 ❒
• Internet Technology	4	140	$40 ❒	$19 ❒
• Knowledge Discovery From Data	4	170	$48 ❒	$19 ❒
• Management Information Systems	4	190	$45 ❒	$18 ❒
• Mathematical Software	4	108	$45 ❒	$21 ❒
• Modeling and Computer Simulation	4	116	$49 ❒	$21 ❒
• Multimedia Computing, Communications, and Applications	4	156	$40 ❒	$19 ❒
• Networking	6	118	$28 ❒	$48 ❒
• Programming Languages & Systems	6	110	$57 ❒	$28 ❒
• Reconfigurable Technology & Systems	4	172	$47 ❒	$20 ❒
• Sensor Networks	4	155	$40 ❒	$19 ❒
• Software Engineering and Methodology	4	115	$41 ❒	$21 ❒
• Speech and Language Processing (1 print / 4 online)	4	153	$40 ❒	N/A
• Storage	4	157	$40 ❒	$19 ❒
• Web	4	159	$39 ❒	$19 ❒

Marked • are available in the ACM Digital Library
** Check here to have publications delivered via Expedited Air Service. For residents outside North America only.*

PUBLICATION SUBTOTAL: _____

INSTRUCTIONS

Carefully complete this application and return with payment by mail or fax to ACM. You must be a **full-time** student to qualify for student rates.

CONTACT ACM

phone: 800-342-6626 (US & Canada)
+1-212-626-0500 (Global)
hours: 8:30am–4:30pm US Eastern Time
fax: +1-212-944-1318
email: acmhelp@acm.org
mail: Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

For immediate processing, FAX this application to +1-212-944-1318.

PAYMENT INFORMATION

Payment must accompany application

Member dues ($19, $42, or $62) $ _____
To have *Communications of the ACM* sent to you via Expedited Air Service, add $54 here (for residents outside of North America only). $ _____

Publications $ _____

Total amount due $ _____

Check or money order (make payable to ACM, Inc. in U.S. dollars or equivalent in foreign currency)

❑ **Visa/Mastercard** ❑ **American Express**

Card number Exp. date

Signature

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

EDUCATION

Name of School

Please check one: ❒ **High School** (Pre-college, Secondary School) **College:** ❒ Freshman/1st yr. ❒ Sophomore/2nd yr. ❒ Junior/3rd yr. ❒ Senior/4th yr. **Graduate Student:** ❒ Masters Program ❒ Doctorate Program ❒ Postdoctoral Program ❒ **Non-Traditional Student**

Major Expected mo./yr. of grad.

Age Range: ❒ 17 & under ❒ 18-21 ❒ 22-25 ❒ 26-30 ❒ 31-35 ❒ 36-40 ❒ 41-45 ❒ 46-50 ❒ 51-55 ❒ 56-59 ❒ 60+

Do you belong to an ACM Student Chapter? ❒ Yes ❒ No

I attest that the information given is correct and that I will abide by the ACM Code of Ethics. I understand that my membership is non transferable.

Signature

COMPUTER SCIENCE AT THE NATIONAL SECURITY AGENCY

Make a critical difference with what you know.

You already know that intelligence is vital to national security. But here's something you may not know.

The National Security Agency is the only Intelligence Community agency that generates intelligence from foreign signals and protects U.S. systems from prying eyes.

If you like using scientific methods and systematic thinking to solve complex problems, then explore NSA. At NSA you can experience a variety of opportunities throughout your career as you work on real-world challenges with the latest technology. You'll also enjoy a collaborative work environment with flexible hours that will enable you to strike a balance between work and life.

You won't find this kind of experience anywhere else.

KNOWINGMATTERS

NSA
NSA.gov/Careers
APPLY TODAY

WHERE INTELLIGENCE GOES TO WORK®

Excellent Career Opportunities in the Following Fields:

- Computer/Electrical Engineering
- Computer Science
- Cybersecurity
- Information Assurance
- Mathematics
- Foreign Language
- Intelligence Analysis
- Cryptanalysis
- Signals Analysis
- Business & Contracting
- Finance & Accounting
- Paid Internships, Scholarships and Co-op

>> **Plus** other opportunities

Watch the Video.
Get the free App for your camera phone at **gettag.mobi** and then launch the App and aim it at this tag.

Search NSACareers

Available on the App Store

Search NSA to Download

U.S. citizenship is required. NSA is an Equal Opportunity Employer. All applicants for employment are considered without regard to race, color, religion, sex, national origin, age, marital status, disability, sexual orientation, or status as a parent.